Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка методу генерації масивів даних для навчання глибинних нейронних мереж
(тема)

Виконав:
студент 2 курсу, групи СШІм-18-2
Норцова А.В.
(прізвище, ініціали)

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

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва спеціалізації)

Керівник проф. Бодянський Є.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова
(освітньо-проіфесійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Норцовій Анні Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи роботи Розробка методу генерації масивів даних для навчання глибинних нейронних мереж
затверджена наказом університету від 30 березня 20 20 р. № 480Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19 травня 20 20 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки та дослідження методів для генерації зображень, дані відомих наукових проектів щодо розробки та дослідження алгоритмів генерації складних розподілів даних, нейромережеві методи для ефективної генерації складних розподілів даних та чотири основні набори даних: реальні фотографії, зібрані з автореєстраторів, синтезовані забраження транспортних маршрутів "SYNTHIA", реальні фотографії собак та згенеровані зображення собак, з використанням GAN.

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі та постановка задачі, 3-D методи генерації, методи на основі нейронних мереж, гіпотеза про статистичні відмінності в розподілах даних, перевірка гіпотези про несхожість даних, побудова репрезентативного прихованого простору, доведення сили варіаційних автоенкодерів, технічні деталі реалізації.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Рисунок 1 – Тришарова нейронна мережа, Рисунок 2 – Архітектура LeNet, Рисунок 3 - Одночасне навчання GAN, Рисунок 4 – Стандартна модель VAE, Рисунок 5 – Стратегія VAE

для створення довільних розподілів, Рисунок 6 – Дані у піксельному просторі, Рисунок 7,8 – Приклади наборів даних, Рисунок 9 – Перша ітерація: перевірка гіпотези про розподіли реальних та синтетичних даних, Рисунок 10 – Друга ітерація: перевірка гіпотези про призований простір і відображення, Рисунок 11 – Архітектура мережі-класифікатора для зображень 128x128 пікселів, Рисунок 12 – ROC-криві для тестових даних, Рисунок 13 – Результати тренування класифікаторів на тестових даних, Рисунок 14 – Карти активації після згорткових шарів, Рисунок 15 – Активаційні карти повнозв'язного шару, Рисунок 16 – Візуалізація прихованого простору, Рисунок 17 – Розподіли ознак, Рисунок 18 – Архітектура класичного варіаційного автоенкодера, Рисунок 19 – Модифікований автоенкодер, Рисунок 20 – Графіки збіжності функцій втрат для модифікації VGG16 BN, Рисунок 21 – Графіки збіжності функцій втрат для модифікації ResNet152, Рисунок 22 – Результати навчання, Рисунок 23 – Схеми репараметризації, Рисунок 24 – Результати для мереж з ріним розміром residual блоків, Рисунок 25 – Згенеровані зображення, Рисунок 26 – Візуалізація прихованого простору VAE

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Бодянський Є.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	30.03.2020	виконано
2	Аналіз предметної області і постановка завдання	30.03-05.04	виконано
3	Дослідження методів навчання	06.04-12.04	виконано
4	Створення імітаційної моделі	13.04-19.04	виконано
5	Тестування і відладка імітаційної моделі	20.04-03.05	виконано
6	Обробка і оформлення результатів	04.04-06.05	виконано
7	Оформлення пояснювальної записки	06.05-11.05	виконано
8	Нормоконтроль	12.05.2020	виконано
9	Попередній захист	13.05.2020	виконано
10	Захист перед ЕК	19.05.2020	

Дата видачі завдання 30 березня 20 20 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Бодянський Є.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснительная записка: 79 страниц, 26 рисунков, 5 таблиц, 11 формул, 2 приложения, 35 источников.

ВАРИАЦИОННЫЙ АВТОЕНКОДЕР, ГЕНЕРАТИВНЫЕ СОРЕВНОВАТЕЛЬНЫЕ СЕТИ, ГЕНЕРАТИВНЫЕ МОДЕЛИ, ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ, СКРЫТОЕ ПРОСТРАНСТВО, РЕПРЕЗЕНТАЦИИ, СЛОЖНОЕ РАСПРЕДЕЛЕНИЕ ДАННЫХ

Объектами исследования являются нейросетевые методы для эффективной генерации сложных распределений данных и четыре основные набора данных: реальные фотографии, собранные с авторегистраторов, синтезированные изображения транспортных маршрутов «SYNTHIA», реальные фотографии собак и сгенерированные изображения собак, с использованием GAN.

Предметом исследования является построение архитектуры глубокой нейронной сети для генерации сложных распределений данных.

Целью работы является выявление скрытых различий между реальными и синтетическими данными для их высококачественной генерации.

Методы исследования включают в себя анализ существующих подходов и моделей для генерации синтетических данных. Анализ литературных источников и последних исследований в области компьютерного зрения и глубинных нейросетевых архитектур

Предполагается, что идентификация отличительных особенностей в распределении реальных и синтетических данных и их использование поможет избежать трудностей перехода модели машинного обучения между ними и генерации качественных синтетических данных для обучения глубоких нейронных сетей.

ABSTRACT

Explanatory note: 79 pages, 26 figures, 5 tables, 11 formulas, 35 sources, 2 annex.

COMPLEX DATA DISTRIBUTIONS, DEEP NEURAL NETWORKS, GENERATIVE ADVERSARIAL NETWORKS, GENERATIVE MODELS, HIDDEN SPACE, REPRESENTATIONS, VARIATIONAL AUTOENCODER

Objects of the study are neural network methods for efficient generation of complex data distributions and four major datasets: real photos collected from autorecorders, generated pictures «SYNTHIA» transport routes, real photos of dogs and generated images of dogs using GANs .

The subject of the study is the construction of a deep neural network architecture to generate complex data distributions.

The aim of the work is to identify hidden differences between real and synthetic data for their high-quality generation.

Research methods include the analysis of existing approaches and models for the generation of synthetic data. Analysis of literature sources and recent research in the field of computer vision and deep neural network architectures.

It is assumed that the identification of distinctive features in the distribution of real and synthetic data and their usege will help to avoid the difficulties of the transition of the machine learning model between them and the generation of quality synthetic data for training deep neural networks.

РЕФЕРАТ

Записка пояснювальна: 79 сторінок, 26 рисунків, 5 таблиць, 11 формул, 2 додатки, 35 джерел.

ВАРІАЦІЙНИЙ АВТОЕНКОДЕР, ГЕНЕРАТИВНІ ЗМАГАЛЬНІ МЕРЕЖІ, ГЕНЕРАТИВНІ МОДЕЛІ, ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, ПРИХОВАНИЙ ПРОСТІР, РЕПРЕЗЕНТАЦІЇ, СКЛАДНІ РОЗПОДІЛИ ДАНИХ

Об'єктами дослідження є нейромережеві методи для ефективної генерації складних розподілів даних та чотири основні набори даних: реальні фотографії, зібрані з автореєстраторів, синтезовані зображення транспортних маршрутів «SYNTHIA», реальні фотографії собак та згенеровані зображення собак, з використанням GAN.

Предметом дослідження є побудова архітектури глибокої нейронної мережі для генерації складних розподілів даних.

Метою роботи є виявлення прихованих відмінностей між реальними та синтетичними даними для їх високоякісної генерації.

Методи дослідження включають в себе аналіз існуючих підходів і моделей для генерації синтетичних даних. Аналіз літературних джерел і останніх досліджень у області комп'ютерного зору та глибинних нейромережевих архітектур

Припускається, що ідентифікація відмітних особливостей у розподілі реальних та синтетичних даних та їх використання допоможе уникнути труднощів переходу моделі машинного навчання між ними і генерації якісних синтетичних даних для навчання глибоких нейронних мереж.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГНМ – глибинна нейронна мережа;

ШНМ – штучна нейронна мережа;

BN – Batch Normalization – пакетна нормалізація;

Conv – Convolution layer – конволюційний шар;

FC – Fully Connected – повнозв’язний шар;

GAN – Generative Adversarial Network – генеративна змагальна мережа;

LR – Learning Rate – крок навчання;

PCA – Principal Component Analysis – метод головних компонент;

ROC – Receiver Operating Characteristic;

VAE – Variational Autoencoder – варіаційний автоенкодер.

ВСТУП

Розширення галузі комп'ютерного зору та глибинного навчання відкриває можливості та підходи до вирішення багатьох задач, які раніше залишалися невирішеними.

Але багато з них, і досі, залишаються поза досяжністю сучасних технологій глибокого навчання – навіть незважаючи на те, що існує велика кількість вручну анотованих даних.

Моделі глибинного навчання не розуміють вхідних даних, так я це може людина. Люди сприймають образи на основі свого досвіду. Моделі машинного навчання не мають доступу до такого досвіду, і тому не можуть «зрозуміти» вхідні дані таким чином. Анотуючи велику кількість навчальних прикладів для моделей, ми змушуємо їх вивчити геометричну трансформацію, яка робить дані більш близькими до людського сприйняття, для конкретного набору прикладів, але це перетворення є лише спрощеною інтерпретацією оригінальної моделі об'єкту.

Моделі глибинного навчання в даний час не мають механізму вивчення абстракцій за допомогою прямого визначення об'єкта, але робота з тисячами, мільйонами, а то й мільярдами навчальних прикладів вирішує цю проблему лише частково [19].

Збір даних для таких завдань є важливим, але іноді дуже складним, особливо у випадку рідкісних класів об'єктів. Слід зазначити, що для такої кількості даних анотація вручну – це не найкраще рішення, оскільки вимагає багато ресурсів і чітко сформованих стратегій розмітки.

Один зі способів вирішити цю проблему – використовувати штучно створені дані. Однак, використовуючи синтетичні дані, ми можемо зіткнутися з проблемою великого стрибка у складності вибору архітектури та методів навчання моделі. Можна припустити, що для моделі існує принципова різниця між реальними та створеними даними.

В даний час для активного розвитку методів обробки зображень

потрібна велика кількість правильно розмічених даних. Відсутність якісних даних унеможливлює використання різних якісних алгоритмів машинного навчання.

У разі обмежених можливостей збору реальних даних використовуються методи їх синтетичної генерації. З практичної точки зору ми можемо сформулювати завдання якісної генерації синтетичних зображень як ефективної генерації складних розподілів даних, що є об’єктом дослідження цієї роботи.

Генерація високоякісних синтетичних даних – це дорогий і складний процес з точки зору існуючих методів. Ми можемо виділити два основні підходи, які використовуються для генерування синтетичних даних: генерування зображення на основі відтворених 3-D сцен та використання генеративних моделей для простих зображень.

Ці методи мають деякі недоліки, такі як вузький діапазон застосовності та недостатня складність розподілу отриманих даних. Використовуючи генеративні моделі для генерації складних розподілів, на практиці ми стикаємось із помітним збільшенням складності архітектури моделі та процедури навчання.

Глибоке розуміння реальних розподілів даних може бути використане для покращення якості синтетичної генерації. Мінімізація відмінностей у реальному та синтетичному розподілі даних може покращити не лише процес генерації, але й розробити інструменти для вирішення проблеми нестачі даних у сфері обробки зображень.

1 МЕТА РОБОТИ

Дане дослідження має на меті порівняти розподіли реальних та синтетичних даних, вивчити причини підвищення трудомісткості роботи при використанні синтетики та способи її усунення.

Основна проблема, що розглядається в цій роботі, полягає у складності генерації високоякісних синтетичних даних для подальшого їх використання у моделях глибинного навчання для обробки зображень.

Отже, головна мета – виявити приховані відмінності між реальними та синтетичними даними для їх високоякісного генерування.

Для досягнення мети виділено такі основні задачі:

- підтвердження гіпотези про наявність статистично значущої різниці в розподілах реальних та синтетичних даних;
- побудова пайплайну для перетворення зображень;
- вибір критерію якості для оцінки згенерованих даних.

Об'єктами дослідження є чотири основні набори даних: реальні фотографії, зібрані з автореєстраторів [34], синтезовані зображення транспортних маршрутів "SYNTHIA" [5], реальні фотографії собак [21] та згенеровані зображення собак, з використанням GAN (Generative Adversarial Network)[3].

Припускається, що ідентифікація відмітних особливостей у розподілі реальних та синтетичних даних допоможе уникнути труднощів переходу моделі машинного навчання між ними.

2 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

2.1 Поняття штучної нейронної мережі

В сучасному світі штучні нейронні мережі продовжують набувати широкого поширення для вирішення великого обсягу задач з різних галузей, перш за все інтелектуального управління, комп’ютерного зору, класифікації, кластеризації, прогнозування, та інших.

Штучна нейронна мережа (ШНМ) – це математична модель, яка складається зі взаємопов'язаних шарів нейронів (груп вузлів), як показано на рисунку 2.1.

Шари можуть бути різних типів, і зазвичай складають кілька різних шарів разом певним чином, щоб отримати нейронну мережу з гарною продуктивністю. Модель на рисунку 2.1 містить два повністю пов'язані приховані шари, де нейрони попарно з'єднані між двома сусідніми шарами [19]. Штучні нейронні мережі, що містять кілька прихованих шарів, називаються глибинними нейронними мережами (ГНМ).

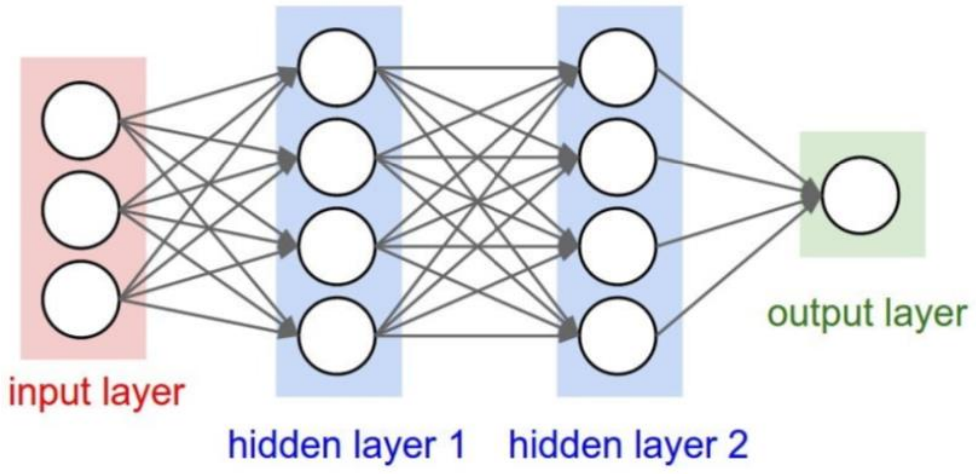


Рисунок 2.1 – Тришарова нейронна мережа

Нейронну мережу можна використовувати як для задач з учителем, так і для задач без учителя. У разі навчання з учителем (наприклад, класифікація) мережа обробляє входи та надає результати (прогнози). Потім прогнози порівнюються з правильними значеннями цільових змінних.

Вузли нейронної мережі надзвичайно взаємопов’язані і мають активаційні функції, що забезпечують нелінійність.

Тренування нейронної мережі складається з проходів вперед і назад. У прямому проході ми просуваємо вхід через шари і обчислюємо функцію помилок.

У зворотному проході ми з'ясовуємо, як кожна вага впливає на загальну помилку обчислення градієнтів за допомогою ланцюгового правила, і змінюємо ваги мережі, щоб зменшити помилку.

У ідеальних умовах ми продовжуємо навчання, поки функція помилок не перетвориться на нуль.

Мета полягає в тому, щоб після багатьох ітерацій описаної процедури отримати чітко налаштовані ваги нейронної мережі, які забезпечують задовільну продуктивність моделі.

Нейронні мережі – це ідеальний інструмент для дослідження прихованих закономірностей, особливо у просторах з великими розмірами, і вони можуть вирішити надзвичайно складні проблеми, які потребують більше, ніж виготовлені вручну функції та евристики.

Оскільки ШНМ містять безліч зв’язків, для моделювання належним чином потрібно пропустити багато даних через модель. Оскільки дані, передані в модель, зберігаються в оперативній пам’яті з випадковим доступом (ОЗУ) робочої машини під час проходження вперед і назад, ми здебільшого не можемо передати всі наявні дані відразу нашій моделі.

Ось чому дані розбиваються на невеликі порції, що називаються батчами. Повний пропуск даного набору даних через модель називається

епохою. Це одна ітерація навчання. Різна кількість епох необхідна для вивчення різних архітектур ГНМ, від 5 до сотень.

2.2 Конволюційні нейронні мережі

Конволюційні нейронні мережі – це такі нейронні мережі, які мають принаймні один шар зі згортковими операціями замість стандартного множення матриць. [14]

CNN – це підкатегорія моделей глибокого навчання, які виявилися дуже потужними в завданнях з комп'ютерним зором.

За допомогою локальних полів сприйняття нейрони можуть отримувати елементарні візуальні ознаки, такі як орієнтовані краї, кінцеві точки, кути (або подібні ознаки в інших сигналах, таких як мовленнєві спектрограми).

Ці функції потім поєднуються наступними шарами, щоб виявити ознаки вищого порядку. [14]

У відомому LeNet-5 (рисунок 2.2), класичній архітектурі CNN, було 4 основні будівельні блоки:

- згортковий шар. Обчислює точковий добуток між вагами ядра та невеликим виправленням вхідних даних, має відстежувані параметри;
- піддиагностичний шар. Здійснює зменшення тиску за просторовими розмірами, не має відстежуваних параметрів;
- нелінійність або функція активації. Застосовує нелінійну функцію активації, наприклад ReLU, Leaky ReLU, гіперболічний тангенс; не має відстежуваних параметрів;
- повнозв'язний шар. Стандартний шар у стилі багатошарового перцептрона, має відстежувані параметри.

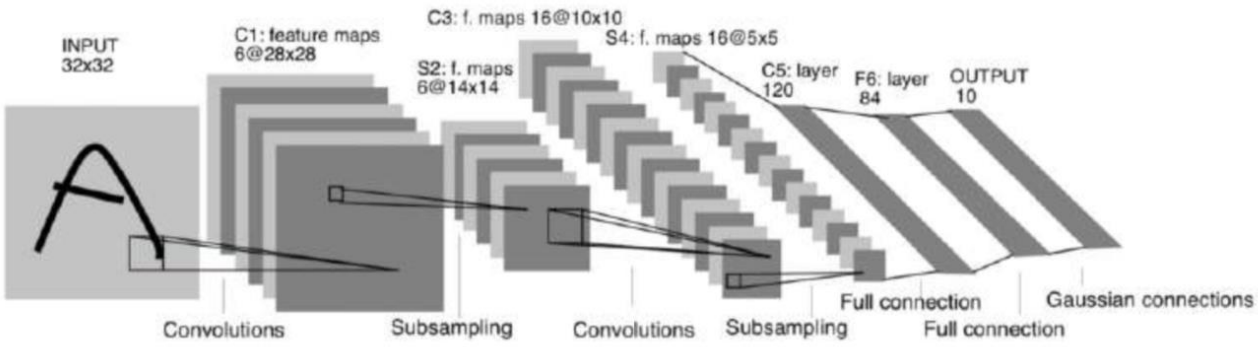


Рисунок 2.2 – Архітектура LeNet

Є кілька речей, які роблять CNN настільки ефективними для завдань з комп’ютерним зором (порівняно з багатошаровою архітектурою перцептрону):

- зображення як входи дуже великі (тисячі чи мільйони пікселів), тому подавання зображення безпосередньо на повністю підключений шар призведе до проблем із пам'яттю та відсутності навчальних даних, щоб наздогнати кількість відстежуваних параметрів мережі. Крім того, ми хотіли б, щоб мережа була надійною для локальних спотворень та перетворень даних;

- повністю пов'язані архітектури ігнорують топологію входу, тоді як зображення мають сильну локальну 2D структуру. Конволюційні нейронні мережі змушують видобувати локальні особливості, обмежуючи рецепторні поля прихованих одиниць локальними;

- після виявлення особливості релевантність її точного положення знижується, оскільки тепер має значення лише її приблизне розташування відносно інших особливостей.

Кілька конкретних архітектур, які потрібно згадати:

- AlexNet став проривом у глибокому навчанні, представляючи набагато глибшу та ширшу мережу, яка з великим відривом перемогла ILSVRC. Усі сучасні широко розповсюджені алгоритми CNN в

комп'ютерному зорі зобов'язані AlexNet своїм успіхом;

- VGGNet показав, що глибина має значення і все ще є важливою складовою для продуктивності. Її особливі карти використовуються для проектування перцептивних втрат через глибину та змістовне багатство;
- ResNet продемонстрував дуже потужну здатність до репрезентації та подолав проблему затухаючого градієнта, ввівши пропускне з'єднання, яке діє як ярлик для одного або декількох шарів.

2.3 Глибокі генеративні моделі

Глибокі генеративні моделі – це нейронні мережі, які можуть вивчати та імітувати розподіл даних, на який можна подати інформацію (в основному це розподіл вхідних даних будь-якого характеру, на яких ґрунтується завдання, від зображень до текстів). Під час навчання глибокої генеративної мережі намагаються знайти параметри, які б наблизили їх модель до реального, але невідомого розподілу даних. [10]

Існує два домінуючі та ефективні типи генеративних моделей: VAE [12] (варіаційні автоенкодери) та GANs [3] (генеративні змагальні мережі).

Під час налаштування генеративної моделі ми знаємо, що семпли приходять з різних розподілів, але важко знайти двовибірну тестову ціль у великих розмірах. Основна ідея, яка стоїть перед GAN, полягає в тому, щоб дізнатися статистику, яка максимізує відповідне поняття відстані між двома наборами вибірки [10].

Структура GAN складається з двох моделей: дискримінатора D і генератора G, а між ними відбувається гра в мінімакс.

Дискримінатор оцінює, наскільки ймовірно, що ці дані походять з реального набору даних (D також іноді називають «критичним»), і він оптимізований для розрізнення підроблених зразків від реальних. Генератор отримує шум на вхід, і генерує штучні семпли; він навчений «хитрувати» дискримінатора, фіксуючи реальний розподіл даних і роблячи

синтетичні зразки якомога складнішими.

Генератор – це спрямована, прихована змінна модель з детермінованим відображенням між вхідним шумом та генерованим виходом. Це зводить до мінімуму тест цілі на два зразки.

На відміну від варіаційного автоенкодера, що не має мережі виводу, яка може дізнатися варіації над латентними змінними вкінці. Дискримінатором є будь-яка функція; Зокрема, це може бути нейронна мережа, яка максимально збільшує двовибірну тестову ціль.

2.4 Генеративні змагальні мережі

Перший документ про GAN був опублікований у 2014 році [3]. У цій роботі Гудфеллов розповів про основну ідею моделі GAN, коли генеративна модель піддається супротивнику; з дискримінаційною моделлю, яка вчиться визначати, чи є вибірка з розподілу моделі чи розподілу даних. У цій статті автор дослідив особливий випадок, коли генеративна модель генерує вибірки, пропускаючи випадковий шум через багатошаровий перцептрон, а дискримінаційна модель також є багатошаровим перцептроном.

Цей особливий випадок називався змагальними мережами. Для того, щоб дізнатися розподіл p_q генератора над даними x, визначено попереднє значення вхідних змінних шумів $p_z(z)$, а потім представити відображення в просторі даних як $G(z;\, \theta_g)$, де G є диференційованою функцією, представленою багатошаровим перцептроном з параметрами θ_g.

Також визначений другий багатошаровий перцептрон $D\,(x;\, \theta_d)$, який виводить один скаляр. $D(x)$ представляє ймовірність того, що x походить з даних, а не з p_g. І D навчається максимально збільшувати ймовірність присвоєння правильної мітки як навчальним прикладам, так і зразкам G.

Дискримінатор та генератор навчаються на кожній ітерації тренувань

і змагаються між собою, намагаючись досягти мети згідно з наступним рівнянням:

$$\min_{G}\max_{D} V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (2.1)$$

Спочатку генератор синтезує дані, далекі від реального розподілу даних. Ці дані, здається, легко класифікуються як підроблені. Але для того, щоб дискримінатор не був «досвідченим», розрізняти реальні та підроблені дані буде непростим завданням. Під час навчального процесу і генератор, і дискримінатор навчаються: перший, як генерувати кращі дані, другий – як розрізняти реальні та підроблені дані. Тож тренувальний процес можна розглядати як гру між генератором та дискримінатором. Ключовим аспектом є синхронність процесу, оскільки якщо будь-який з них виграє, він зупинить процес. Описана стаття довела свої результати на наборі даних MNIST і є справжнім бестселером у дослідницькому світі. Багато моделей, заснованих на початковій ідеї GAN, пропонуються з 2014 року. Розглянемо найважливіші моделі.

Глибокі згорткові генеративні змагальні мережі (DCGAN), показують, як згортання шарів можна використовувати з GAN. Згідно зі статтею, запропонована архітектура може розглядатися з певними архітектурними обмеженнями як сильний кандидат на непідвладне навчання.

Модель навчалася на трьох наборах даних: Large-scale Scene Understanding, набору даних Imagenet 1k та Faces [2]. У іншій статті [35] запропоновано деякі покращені методики навчання GAN.

У ній представлено різноманітність нових архітектурних особливостей та процедур навчання, які можна застосувати до моделей GAN. Для заохочення конвергенції використовуються такі прийоми, як

відповідність характеристик, дискримінація міні-батчів, усереднення історичних даних, однобічне згладжування міток та нормалізація віртуальних батчів.

Модель BigGAN – це сучасна модель для покоління ImageNet [14]. Ця стаття поєднує в собі безліч сучасних технологій, таких як спектральна нормалізація та умовні GAN з дискримінаторами проекцій. Всі описані моделі GAN використовуються для створення зображень. На рисунку 2.3 зображено навчання всіх описаних частин GAN.

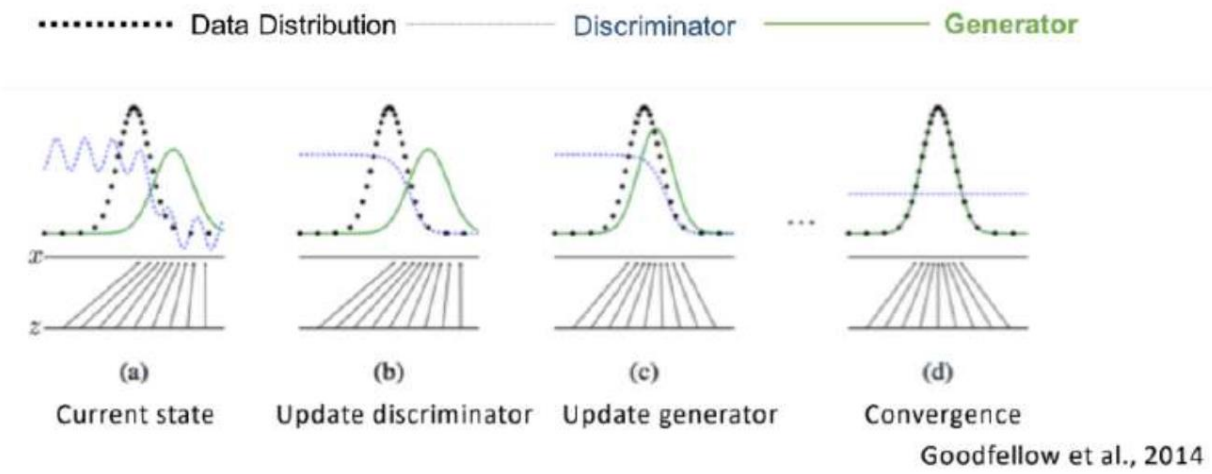


Рисунок 2.3 - Одночасне навчання GAN

2.5 Моделі прихованих змінних

Під час підготовки генеративної моделі, чим складніші залежності між просторовими розмірами, тим складніше навчати моделі. Розглянемо, проблему генерації зображень рукописних символів. Для простоти, будемо вважати, що нас хвилює лише моделювання цифр від 0 до 9. Якщо ліва половина символу містить ліву половину п'ятірки, то права половина не може містити ліву половину нуля, або символ буде чітко не схожий на реальну цифру.

Інтуїтивно це допомагає, якщо модель спочатку вирішить, який символ генерувати, перш ніж присвоїти значення якомусь конкретному пікселю. Таке рішення формально називається латентною змінною.

Тобто, перш ніж наша модель щось намалює, вона спочатку випадковим чином відбирає значення z із набору $[0, \ldots, 9]$, а потім переконує, що всі штрихи відповідають цьому символу, z називається "прихованою", оскільки, маючи на увазі лише символ, створений моделлю, ми не обов'язково знаємо, які параметри прихованих змінних генерували символ.

Нам потрібно зробити висновок, використовуючи щось на зразок комп’ютерного зору. Перш ніж ми зможемо сказати, що наша модель є представником нашого набору даних, ми повинні переконатися, що для кожної точки даних X у наборі даних є одне (або багато) налаштування прихованих змінних, що змушує модель генерувати щось дуже схоже на X.

Формально скажімо, у нас є вектор прихованих змінних z у великовимірному просторі Z, з якого ми можемо легко відбирати семпли відповідно до деякої функції щільності ймовірностей $P(z)$, визначеної на Z. Тоді, у нас є сім'я детермінованих функцій $f(z; \theta)$, параметризовані вектором θ в деякому просторі Θ, де f: $Z \times \Theta \rightarrow X$. Функція f є детермінованою, але якщо z є випадковим і θ фіксованим, то $f(z; \theta)$ – випадкова величина в просторі X.

Ми маємо на меті оптимізувати θ таким чином, щоб ми могли відібрати z з $P(z)$ і, з великою часткою ймовірності, $f(z; \theta)$ буде аналогічно значенню X у нашому наборі даних.

Щоб зробити це поняття математично точним, ми прагнемо максимально збільшити ймовірність кожного X у навчальному наборі за весь генеративний процес відповідно до рівняння:

$$\boldsymbol{P(X) = \int P(X|z;\, \theta) P(z) dz} \tag{2.2}$$

Тут $f(z;\ \theta)$ було замінено розподілом $P(X \mid z;\ \theta)$, що дозволяє зробити залежність X від z явною, використовуючи закон повної ймовірності.

Інтуїція, що стоїть за цим підходом, називається "максимальною вірогідністю", і полягає в тому, що якщо модель, ймовірно, створює зразки навчальних наборів, то вона також може створити подібні зразки, і навряд чи зможе створити різні.

У VAE (рисунок 2.4) вибір цього розподілу виходу часто є Гауссовим, тобто $P(X \mid z;\ \theta) = N(X \mid f(z;\ \theta), \sigma^2 * I)$. Тобто, він має середнє $f(z;\ \theta)$ і коваріантність, рівну матриці тотожності I в рази більше скалярного σ (що є гіперпараметром).

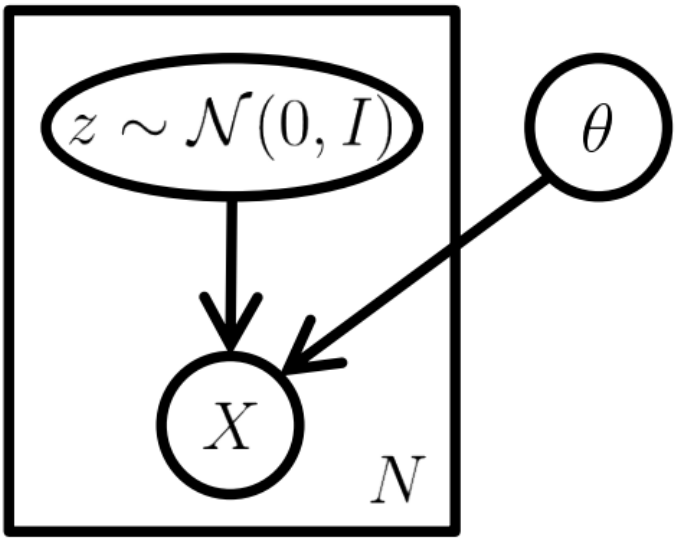


Рисунок 2.4 – Стандартна модель VAE

Ця заміна необхідна для формалізації інтуїції, що для деякого z потрібно отримати зразки, схожі лише на X. Загалом, і особливо на початку навчання, наша модель не дасть результатів, ідентичних жодному конкретному X. Маючи Гауссовий розподіл ми можемо використовувати

градієнтний спуск (або будь-яку іншу техніку оптимізації) для збільшення $P(X)$, застосовуючи $f(z;\theta)$ підхід X для деякого z, тобто, поступово роблячи більш імовірними дані тренувань за генеративною моделлю.

Слід зауважити, що розподіл вихідних даних не повинен бути Гауссовим: наприклад, якщо X є двійковим, тоді $P(X \mid z)$ може бути параметром Бернуллі, параметризованим згідно $f(z;\theta)$. Важливою властивістю є те, що $P(X \mid z)$ можна обчислити і є неперервною у θ.

2.6 Варіаційні автоенкодери

Математична основа VAE насправді має відносно мало спільного з класичними автоенкодерами, наприклад. розрідженими автокодерами , або знешумлюючими автокодерами [12, 13]. VAE приблизно максимізують рівняння у формулі 2.2 , відповідно до моделі, зображеної на рисунку 2.4. Вони називаються «автоенкодерами» лише тому, що кінцева мета тренінгу, що виходить з цього налаштування, має енкодер і декодер і нагадує традиційний автоенкодер. На відміну від розріджених автокодерів, у яких, як правило, немає параметрів налаштування, аналогічних мірі покарання.

І на відміну від розріджених та знешумлюючих автоенкодерів, ми можемо відібрати приклад безпосередньо з $P(X)$ (не виконуючи Марківський ланцюг Монте-Карло).

Для вирішення рівняння заданого формулою 2.2 є дві проблеми, з якими повинні стикнутись VAE: як визначити латентні змінні z (тобто вирішити, яку інформацію вони представляють), і як поводитися з інтегралом над z. VAE дають певну відповідь на обидва.

По-перше, поставимо питання про те як ми вибираємо латентні змінні z такі, щоб ми фіксували приховану інформацію. Розглядаючи приклад з генерації цифр, «латентні» рішення, які модель повинна прийняти, перш ніж вона починає малювати цифру, насправді досить складні. Для цього потрібно вибирати не просто цифру, а кут, на який

намальована цифра, ширину обведення, а також абстрактні стилістичні властивості. Для нас погано, що ці властивості можуть бути скорельовані: якщо піксель записується швидше, це може призвести до більш кутової цифри, що також може спричинити тонший штрих. В ідеалі ми хочемо уникати таких ситуацій і вручну вирішувати, яку інформацію кодує кожен вимір z (хоча ми можемо вказати його вручну для деяких розмірів).

Ми також хочемо уникати явного опису залежностей, тобто латентної структури між розмірами z. VAE застосовують незвичний підхід до вирішення цієї проблеми: вони припускають, що не існує простої інтерпретації розмірів z, і замість цього стверджують, що зразки z можна отримати з простого розподілу, тобто $N(0, I)$, де I – матриця ідентичності.

Ключовим є зауваження, що будь-який розподіл у d-мірному просторі можна генерувати, беручи набір d змінних, які нормально розподілені, та відображати їх через досить складну функцію.

Наприклад, скажімо, ми хотіли побудувати 2D випадкову змінну, значення якої лежать на кільці. Якщо z дорівнює 2D і нормально розподілена, $g(z) = z / 10 + z / ||z||$ має приблизно кільцеподібну форму, як показано на рисунку 2.5.

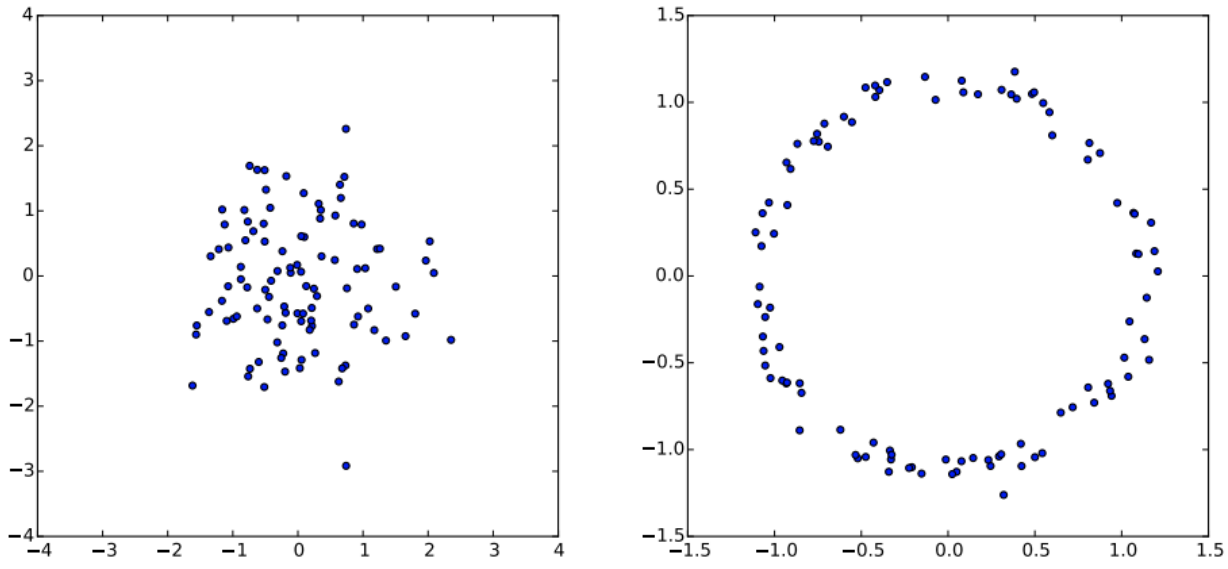


Рисунок 2.5 – Стратегія VAE для створення довільних розподілів

Отже, за умови ітеративного наближення функцій ми можемо просто вивчити функцію, яка відображає наші незалежні, нормально розподілені значення z на будь-які латентні змінні, які можуть знадобитися для моделі, а потім відобразити ці латентні змінні на X.

Якщо $f(z;\ \theta)$ є багатошаровою нейронною мережею, то ми можемо уявити мережу, використовуючи кілька перших її шарів для зіставлення нормально розподілених z до прихованих значень (наприклад, ідентифікація цифри, вага ходу, кут, тощо) зі збереженням точних статистичних властивостей. Тоді мережа може використовувати інші шари для зіставлення цих прихованих значень до повністю виведеної цифри.

Взагалі нам не потрібно турбуватися про те, щоб забезпечити існування латентної структури. Якщо така латентна структура допомагає моделі точно відтворити (тобто максимізувати ймовірність) навчального набору, то мережа вивчить цю структуру на якомусь рівні. Тепер залишається лише максимізувати рівняння зазначене у формулі 2.2, де $P(z)\ =\ N\ (z \mid 0, I)$.

Як зазвичай у машинному навчанні, якщо ми можемо знайти обчислювальну формулу для $P(X)$ і можемо взяти градієнт цієї функції, то ми можемо оптимізувати модель, використовуючи стохастичний градієнтний підйом.

Насправді концептуально просто обчислити $P(X)$ приблизно: ми спочатку вибираємо велику кількість z значень $\{z1, \ldots, zn\}$ і обчислюємо $P(X)\ \approx\ \frac{1}{n}\sum_i P(X \mid z_i)$. Проблема тут полягає в тому, що у просторах з високими розмірами n може бути надзвичайно великим, перш ніж ми отримаємо точну оцінку $P(X)$. Щоб зрозуміти чому, розглянемо наш приклад рукописних цифр. Скажімо, що наші точки даних зберігаються у

піксельному просторі, у зображеннях 28x28, як показано на рисунку 2.6.

Оскільки $P(X \mid z)$ має ізотропний Гауссовий розподіл, негативний логарифм ймовірності X пропорційний квадрату евклідової відстані між $f(z)$ та X Скажімо, що рисунок 2.6 (а) – ціль (X), для якої ми намагаємось знайти $P(X)$. Модель, яка створює зображення, зображена на рисунку 2.6 (б), ймовірно, є поганою моделлю, оскільки ця цифра не дуже схожа на двійку.

а) б) в)

Рисунок 2.6 – Дані у піксельному просторі: а) – цільова картинка, б), в) – виходи моделей, які створюють картинку.

Отже, ми повинні встановити σ гіперпараметр нашого Гауссового розподілу таким чином, щоб цей вид помилкової цифри не впливав на $P(X)$. З іншого боку, модель, яка дає рисунок 2.6 (в) (ідентична X, але зміщена вниз та вправо на пів пікселя), може бути хорошою моделлю.

Ми сподіваємось, що цей приклад впливатиме на $P(X)$. Ми можемо зробити проміжний висновок, що для того, щоб відкинути зразки, як на рисунку 2.6 (б), нам потрібно встановити параметр σ дуже малим, таким чином, що модель повинна щось генерувати щось значно більше схоже на X, ніж на рисунку 2.6 (в).

Навіть якщо наша модель є точним генератором цифр, нам,

ймовірно, доведеться відібрати багато тисяч цифр, перш ніж ми створимо двійку, достатньо схожу на ту, що зображена на рисунку 2.6 (а).

Ми можемо вирішити цю проблему за допомогою кращої метрики подібності, але на практиці їх складно розробляти в складних областях, таких як зір, і їх важко навчати без міток, які вказують, які точки даних схожі між собою. Натомість VAE змінюють процедуру вибірки, щоб зробити її швидшою, не змінюючи показника подібності.

2.5 Постановка задачі дослідження

Основна мета атестаційної роботи магістра полягає в тому, щоб побудувати модель машинного навчання на основі генеративних моделей для синтезу якісних синтетичних зображень на базі висновків з перевірки гіпотези про статистичні відмінності в розподілах реальних та синтетичних даних.

Припускається, що ідентифікація відмітних особливостей у розподілі реальних та синтетичних даних допоможе уникнути труднощів перемикання моделі машинного навчання між ними.

Відповідно до поставленої мети необхідно розв’язати такі наукові задачі:

1. Аналіз існуючих методів та підходів для генерації синтетичних зображень.
2. Перетворення зображень та їх відображення у векторний простір за допомогою нейромережевих методів.
3. Побудова простору та двох презентацій: від зображень до прихованого простору і навпаки
4. Аналіз розподілів у новому прихованому просторі та їх дослідження за допомогою статистичних методів
5. Проведення перетворень на даних у прихованому просторі для мінімізації відмінностей

6. Відображення модифікованих синтетичних даних у просторі зображення

7. Вибір формального критерію для оцінки якості штучно згенерованих даних, щоб моделі машинного навчання в галузі комп'ютерного зору, що містять синтетику в навчальному наборі даних, демонстрували високу якість роботи з тестовими та перевірочними зразками реальних даних.

Узагальнюючи мету даного дослідження, скажімо, що визначення можливості ефективної генерації синтетичних зображень для глибинних нейронних мереж у області комп’ютерного зору допоможе розв’язати складну проблему з нестачею навчальних даних.

Дана робота є актуальною як с точки зору використання синтетичних зображень у сучасних інтелектуальних системах комп’ютерного зору, так і у тому, що отримані результати можна враховувати при їх побудові. Це дозволяє значно скоротити вартість таких систем шляхом зменшення вартості збору датасетів.

3 МЕТОД ГЕНЕРАЦІЇ МАСИВІВ ДАНИХ ДЛЯ НАВЧАННЯ ГЛИБИННИХ НЕЙРОННИХ МЕРЕЖ

В даному розділі розглянуто задачу створення архітектури методу генерації масивів даних для навчання глибинних нейронних мереж. Розглянуті існуючі підходи і методи, а також описано головні гіпотези, що висуваються до складних розподілів даних.

3.1 Попередні дослідження в даній галузі

З 2010 року проводяться дослідження у галузі візуальної доменної адаптації [25], де першим запропонованим підходом до вирішення проблеми були статистичні методи. Однак з 2014 року нейромережеві методи набули значної популярності [30].

Незабаром нестача даних стала пов'язаною проблемою, що призвело до зростання методів генерації синтетики [31].

Потреба в синтетичних даних часто виникає в багатьох завданнях. Видатним представником таких завдань є автономне керування. Для виготовлення високоточних класифікаторів дорожньої розмітки потрібні знаки, автомобілі та інші великі обсяги якісно розмічених даних.

3.1.1 3D-методи

Для вирішення цієї проблеми у 2016 році була запропонована ідея створення набору даних на основі ігрового світу. У статті «Playing for Data: Ground Truth from Computer Games» [23] використано ігровий світ GTA5. Метою роботи було отримати розмітку зі скріншотів гри. Основна ідея – використовувати існуючий віртуальний 3-D світ. Однак обмеження гри не дозволили отримати повну розмітку, необхідну для вирішення проблеми автономного водіння.

У той же час, у 2016 році, використовуючи ту саму ідею віртуального світу, було створено набір даних SYNTHIA для того, щоб допомогти в семантичній сегментації та проблемах розуміння пов'язаних сцен у контексті сценаріїв руху [24]. Автори трохи змінили підхід і створили свій віртуальний світ за допомогою платформи Unity. Вони будували свої віртуальні міста на основі реальних прототипів міста.

Однією з головних переваг була можливість додавання природних подій, таких як час доби, дощ, сніг, туман та інші.

Ці методи автоматичні з точки зору генерування наборів даних, але складні на стадії проектування віртуального світу.

Ще одним чудовим прикладом використання синтетичних даних є завдання розпізнавання напрямку погляду людини. У 2016 році була опублікована стаття «Learning an appearance-based gaze estimator» [32]. Існував метод генерації очей, запропонований з урахуванням біологічних особливостей очного яблука, а також шкіри навколо нього. У роботі велика увага приділяється світловим характеристикам очної поверхні. У цій роботі використовується та ж ідея створення тривимірних моделей об'єктів, але з урахуванням їх фізичних характеристик для більшого реалізму.

3.1.2 Методи на основі нейронних мереж

Часто виникають також проблеми, в яких оригінальний набір даних містить дані різного характеру. Звідси, природно, виникають завдання доменної адаптації [29] та перенесення стилю. У 2018 році була опублікована стаття A Literature Review of Neural Style Transfer [8]. У ньому обговорюються методи перенесення стилю з одного зображення на інше за допомогою нейромережевих методів. Ідеї, засновані на принципі, що нейронні мережі виділяють особливості стилю. У перших статтях з цієї теми були використані функції, отримані за допомогою нейронних мереж

VGG [27], а також принципи автокодування [12]. Перенесення стилів здійснюється за рахунок хитрощів з проміжними виходами нейронних мереж, а також різними способами побудови функції втрат. У 2017 році Джуді Хоффман представила метод доменної адаптації під назвою CYCADA [6]. Його суть полягала у використанні складної архітектури, що складається з двох генераторів, двох дискримінаторів та чотирьох мереж прийняття рішень [10]. Метод показав хороші результати; однак для навчання необхідно мати розмічені дані семантичної сегментації [18].

У липні 2018 року команда Ming-Yu Liu, Thomas Breuel, Jan Kautz з NVIDIA запропонувала метод під назвою UNIT (Unsupervised image-to–image Translation), що поєднує ідеї VAE та GAN [16].

Ідея запропонованого способу полягає у побудові нейронних мереж на основі GAN та VAE для задач перенесення стилів.

Запропонований метод конструює відображення вихідних зображень у прихований простір та здійснює перетворення за допомогою нейронних мереж у цих просторах. Такий підхід добре переносить стилі і, як результат, добре перетворює їх синтетику в реальні дані. Однак цей метод не ставить першочергової мети генерувати високоякісну синтетику.

Останнім часом розроблено велику кількість підходів, методів та архітектури для вирішення цієї та подібних проблем. Однак, аналізуючи роботу в цій галузі, можна сказати, що недостатньо уваги було приділено проблемі розгляду генерування синтетичних даних саме з точки зору статистичних методів.

3.2 Збір даних

Для експериментів ми відбирали дані за двома критеріями: релевантність завдання, для якого вони можуть бути використані, та простота об'єктів для сприйняття людиною. Основною вимогою було існування пари "реальні дані – синтетичні дані", оскільки генерація великої

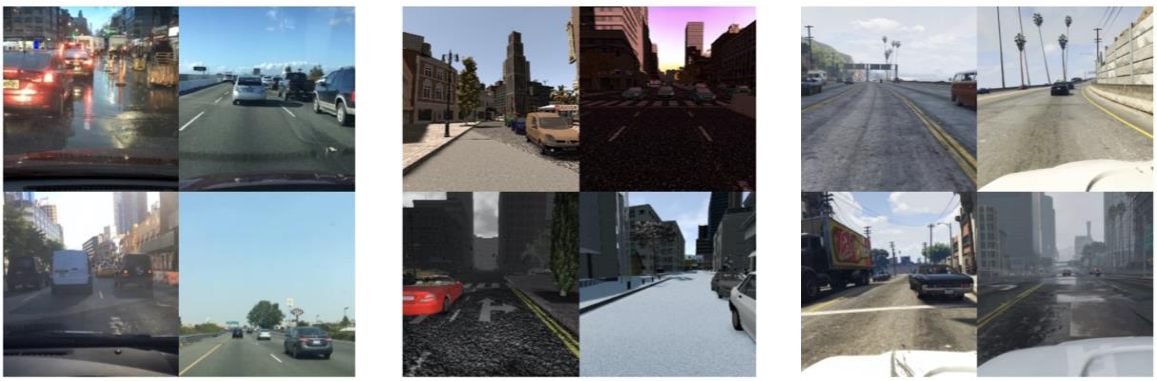
кількості синтетичних даних з нуля є дорогим і трудомістким процесом.

За першим критерієм ми обрали набір даних SYNTHIA. SYNTHIA – це набір даних, створений для сприяння семантичній сегментації та пов'язаним з цим розумінням проблем у контексті рушійних сценаріїв.

SYNTHIA складається з колекції фотореалістичних кадрів, зроблених з віртуального міста і поставляється з точними семантичними анотаціями на рівні пікселів для 13 класів: різне, небо, будівля, дорога, тротуар, огорожа, рослинність, стовп, машина, знак, пішохід, велосипедист, прокладка смуги [5].

Оскільки обсяг набору даних SYNTHIA недостатньо великий для експериментів, було вирішено додати частину набору даних, витягнуту з гри Grand Theft Auto V [20]. Набір даних містить семантичні анотації для 19 класів.

Завдання самостійного керування вимагає максимальної точності у своєму рішенні, а отже, і великих високоякісних наборів даних, що відповідає актуальності нашої роботи. У цьому випадку ми вибрали набір даних Berkeley DeepDrive як реальні дані, за якими було виконано три складні завдання для воркшопу CVPR 2018: виявлення дорожніх об'єктів, сегментація області руху та адаптація сегментації об’єктів [34]. (Рисунок. 3.1).



а) б) в)

Рисунок 3.1 – Приклади наборів даних: а) – Berkeley DeepDrive, б) –

SYNTHIA, в) – Grand Theft Auto V.

Згідно з другим критерієм, ми взяли набір зображень собак, тому що вони легкі для сприйняття людиною, але їх важко формалізувати для комп'ютера. Звідси випливає, що розподіл є складним, і це є життєво важливим аспектом, який слід врахувати в нашому дослідженні.

Як набір реальних даних ми вибрали набір даних Stanford Dogs [21], який містить зображення 120 порід собак з усього світу. Цей набір даних був створений за допомогою зображень та розміток від ImageNet для завдання детальної категоризації зображень.

В якості синтетичного аналога ми вибрали зображення собак, згенерованих за допомогою методу GAN зі змагання Kaggle Generative Dog Images [3]. Він містить 10 000 прикладів синтезованих собак без розмітки (рисунок 3.2).

а) б)

Рисунок 3.2 – Приклади наборів даних: а) – згенеровані за допомогою методу GAN, б) – Stanford Dogs

3.3 Гіпотеза про статистичні відмінності в розподілах даних

Для коректних висновків перед початком експериментів ми перетворили наші дані в зображення розміром 224x224 та з три

кольоровими каналами.

По перше для нашого дослідження принципово важливо, щоб у синтетичних і реальних даних були статистично значущі відмінності в розподілах, тому поділимо наше дослідження на дві ітерації.

У першій частині ми перевіримо гіпотезу про те, що розподіли реальних та синтетичних даних мають статистично значущі відмінності.

Для людини різниця між синтетичними зображеннями та реальними даними є інтуїтивно зрозумілою, але, як і багато подібних процесів, її важко формалізувати. На основі цього твердження наш підхід намагається формалізувати ці відмінності.

Підхід, який ми обрали для першої ітерації експерименту, передбачає використання навчених нейронних мереж для отримання інформації про зображення у векторній формі. На рисунку 3.3 показана візуалізація першої ітерації експерименту.

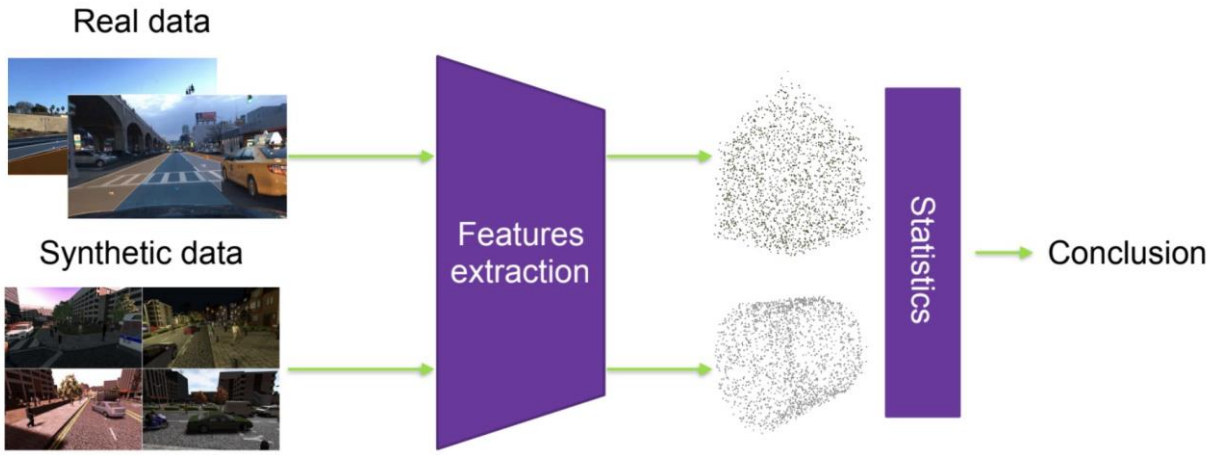


Рисунок 3.3 – Перша ітерація: перевірка гіпотези про розподіли реальних та синтетичних даних

Ми будемо використовувати декілька мереж, таких як AlexNet [14], MobileNetV2 [26], VGG13/16 з батч нормалізацією [27], MNASnet, MNASnet0.5 [35], ResNet152 [4], ResNeXt101 [33], попередньо навчені на

Imagenet [2], як екстрактори ознак [7]. Використовуючи статистичний тест, Т-критерій Стьюдента, ми можемо перевірити наше припущення про відмінність синтетичних та реальних даних з певним рівнем впевненості.

Другою ітерацією будемо вважати підготовку варіативного автоенкодера на реальних та синтетичних даних, будуючи таким чином прихований простір та два відображення: від зображень до прихованого простору та навпаки. Цей підхід фактично є нашою другою гіпотезою.

Ми припускаємо, що за допомогою нескладних перетвореннь у прихованому просторі ми зможемо вплинути на результат генерації даних близьких до оригіналу.

Буде проаналізовано та порівняно основні статистичні характеристики реальних та синтетичних даних у прихованому просторі.

Для реалізації мети даного дослідження ми будемо використовувати прості математичні операції, щоб наблизити статистичні характеристики синтетичних даних до реальних.

Потім будемо передавати перетворені приховані презентації синтетичних даних через декодер. На виході ми очікуємо отримання зображень, близьких до реальних. На рисунку 3.4 показана візуалізація другої ітерації експерименту.

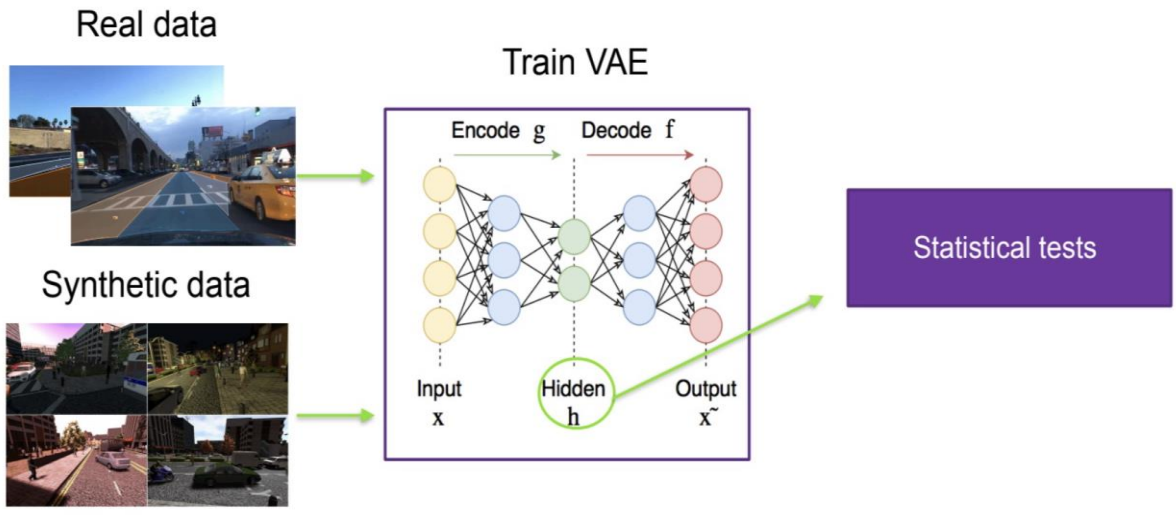


Рисунок 3.4 – Друга ітерація: перевірка гіпотези про прихований простір і

відображення

Планується два експерименти, що можуть служити перевіркою нашої основної гіпотези про існування статистично значущих відмінностей в розподілах реальних і синтетичних даних.

По-перше, ми можемо повторно передавати згенеровані дані через навчені мережі з першої ітерації. Тоді мірою якості буде статистично незначна різниця в розподілі даних.

В якості другого експерименту будемо передавати перетворені синтетичні та початкові реальні дані через просту нейронну мережу, яка вирішить задачу бінарної класифікації, тобто визначить характер зображення: синтетичне воно чи реальне.

Після цього ми використаємо валідаційний набір даних для прогнозування мітки бінарної класифікації. Якщо нейронна мережа не може точно передбачити правильну мітку, то якість конверсії синтетичних даних може вважатися високою.

Ми припускаємо, що нейронна мережа не може розрізнити мітки класів, якщо значення ROC-AUC [1] становить приблизно 0,5 на валідаційному наборі даних.

3.4 Статистичний Т-критерій Стьюдента

Т-тест Стьюдента в статистиці – це метод тестування гіпотез про середнє значення невеликого зразка, узятого з нормально розподіленої сукупності, коли стандартне відхилення невідомо.

У 1908 році Вільям Сілі Госсет, англієць, що публікується під псевдонімом «Стьюдент», розробив t-тест і t-розподіл. Розподіл t – це сімейство кривих, у яких число ступенів свободи (кількість незалежних спостережень у вибірці мінус одна) вказує на конкретну криву.

Зі збільшенням розміру вибірки (і, отже, ступенів свободи), розподіл t наближається до форми дзвоника стандартного нормального розподілу.

На практиці для тестів, що включають середнє значення для вибірки, розміру більше 30 прикладів, зазвичай застосовується нормальний розподіл.

Спочатку зазвичай формулюють нульову гіпотезу, яка стверджує, що немає значущої різниці між спостережуваним середнім рівнем вибірки та середньою гіпотезою або заявленою сукупністю, тобто, що будь-яка вимірювана різниця обумовлена лише випадковістю.

Наприклад, у даному дослідженні, нульовою гіпотезою може бути те, що неважкі математичні операції у прихованому просторі не впливають на якість згенерованих синтетичних даних, і буде проведено експеримент, щоб перевірити, чи дійсно наші маніпуляції не вплинули на результат.

Як правило, t-тест може бути або двостороннім (його також називають двохвостим), просто заявляючи, що середні значення не є рівнозначними, або одностороннім, вказуючи, чи є спостережуване середнє значення більшим чи меншим, ніж гіпотетичне середнє.

Далі обчислюється статистика тесту t. Якщо спостережувана t-статистика є більш крайньою, ніж критичне значення, визначене відповідним опорним розподілом, нульова гіпотеза відхиляється.

Відповідним опорним розподілом для t-статистики є розподіл t. Критичне значення залежить від рівня значущості тесту (ймовірність помилкового відхилення нульової гіпотези).

Формула t-статистики розраховується наступним чином:

$$\boldsymbol{t = \frac{\overline{x} - \mu}{s/\sqrt{n}}} \qquad (3.1)$$

де n – розмір вибірки;

$\overline{x}$ – середнє значення досліду;

s – стандартне відхилення;

μ – гіпотетичне середнє значення.

Наприклад, припустимо, що ми бажаємо перевірити гіпотезу про те, що вибірку розміру $n = 25$ із середнім $\overline{x} = 79$ та стандартним відхиленням $s = 10$ було отримано навмання з популяції із середнім $\mu = 75$ та невідомим стандартним відхиленням.

Використовуючи формулу для t-статистики, обчислене значення t дорівнює 2. Для двостороннього тесту на загальному рівні значущості $\alpha = 0{,}05$ критичні значення розподілу t на 24 ступені свободи становлять -2.064 та 2.064. Обчислене значення t не перевищує цих значень, отже, нульова гіпотеза не може бути відхилена з 95-відсотковою впевненістю (Рівень довіри $1 - \alpha$).

Друге застосування розподілу t перевіряє гіпотезу про те, що два незалежні випадкові вибірки мають однакове середнє значення. Розподіл t також може бути використаний для побудови довірчих інтервалів для справжньої середньої сукупності або для різниці між двома вибірковими середніми значеннями.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

4.1 Перевірка гіпотези про несхожість даних

Дані, представлені у вигляді кольорових зображень у форматі RGB, розміри зображень, з якими ми працюємо, – 224x224, 128x128, 64x64, 32x32 пікселів.

Отже, розмірність простору, з яким ми працюємо, є $[0..1]^{NxNx3}$, де інтенсивність кольору визначається інтервалом від нуля до одиниці. Вектори мають дуже високу розмірність – це значно ускладнює роботу із зображеннями та їх аналізом.

Також важливо, щоб вектори не були випадковими. Іншими словами, інтенсивність певного пікселя сильно залежить від значень пікселів, які знаходяться поруч. Тому значення інтенсивності пікселів сильно залежать від просторового положення об'єктів на зображенні, як значення пікселів у локальній області зображення.

Ці зв’язки та залежності визначають статистичний розподіл зображень. Реальні і синтетичні зображення містять в собі одну й ту ж саму суть, людина з легкістю визначає, що зображено на тих і інших зображеннях одна до точно так же легко і розрізняє їх.

Залежності пікселів що породжують наші розподіли зображень є дуже складними. Їх практично неможливо формалізувати, проте частина з них властива визначенню сенсу того що зображено на картинці, частина визначає чи є зображення реальним або синтетичним, швидше за все більша частина залежностей властива і першій і другій причині.

4.1.1 Побудова класифікатора

Нейронні мережі можуть будувати карти з простору входів до

просторів зі специфічними властивостями, наприклад, лінійною роздільністю класів.

Нейронні мережі вивчають розподіли на основі законів взаємозв'язку пікселів і знаходять правильне відображення просторів. Розглянемо нейронні мережі, що розділяють реальні та синтетичні дані. Нейронні мережі мають погану інтерпретацію.

Однак для мереж з невеликою кількістю нейронів можна візуалізувати шари і зрозуміти, на якій основі мережа вирішує, реальне чи синтетичне зображення. Загальна архітектура мережі-класифікатора, представлена на рисунку 4.1.

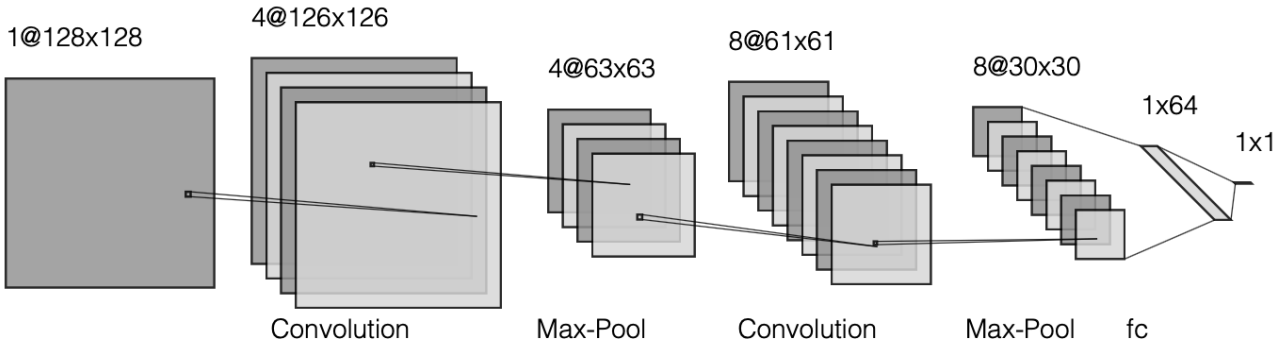


Рисунок 4.1 – Архітектура мережі-класифікатора для зображень 128x128 пікселів

Можна описати архітектуру класифікатора наступним чином:

$$[\mathbf{4} - \boldsymbol{Conv3x3}] - [\boldsymbol{MaxPool2x2}] - [\mathbf{8} - \boldsymbol{Conv3x3}] - [\boldsymbol{MaxPool2x2}] - [\boldsymbol{Dropout} - \mathbf{0.2}] - [\boldsymbol{FC} - \mathbf{64}] - [\boldsymbol{FC} - \mathbf{1}] \quad (4.1)$$

В якості функції нелінійності використовується ReLU. Остання нелінійність – сигмоїдна функція. Навчання проводилося за допомогою оптимізатора Adam [11], із використанням learning rate = 0,0001. Кінцевою функцією оптимізації є бінарна кросс-ентропія. Було проведено десять

епох навчання з розміром батча шістнадцять. Під час експериментів чотири моделі тренувались для різних розмірів зображення, зокрема, 224x224, 128x128, 64x64, 32x32. Криві ROC показують результати моделювання на тестових даних (рисунок 4.2).

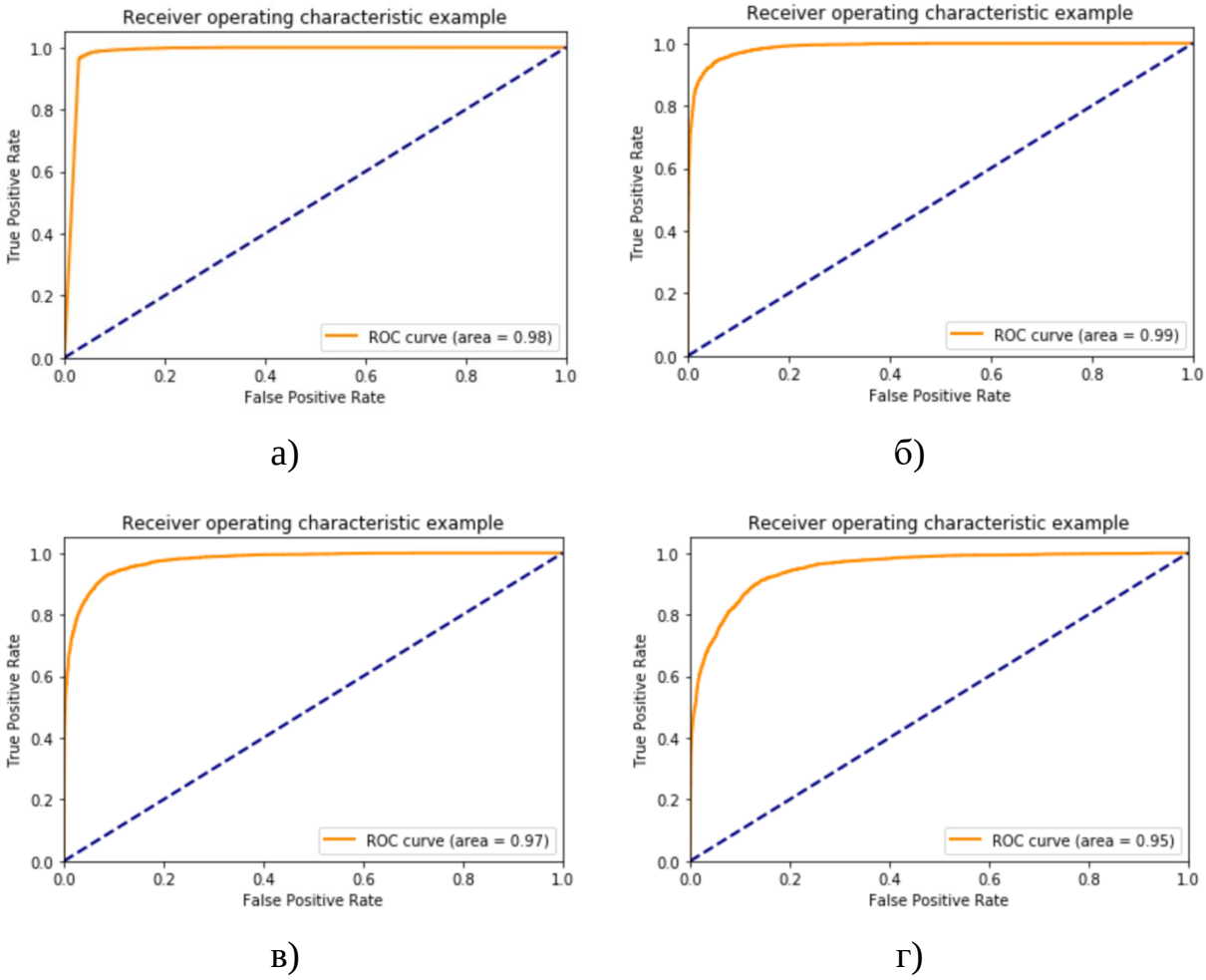


Рисунок 4.2 – ROC-криві для тестових даних: а) – 224x224, б) – 128x128 , в) – 64x64, г) – 32x32

Усі мережі дуже добре відрізняють реальні дані від синтетичних. Це означає, що мережі будують належні простори, де є лінійна роздільність. У таблиці 4.1 наведені результати показника precision-recall score для тренованих мереж з різним розміром зображення.

Таблиця 4.1 – Узагальнені показники precision-recall score для тренованих мереж з різним розміром зображення.

Розмір зображення	224x224	128x128	64x64	32x32
Average precision-recall score	0.90	0.96	0.96	0.87

На рисунку 4.3 представлені результати натренованих мереж на тестових даних.

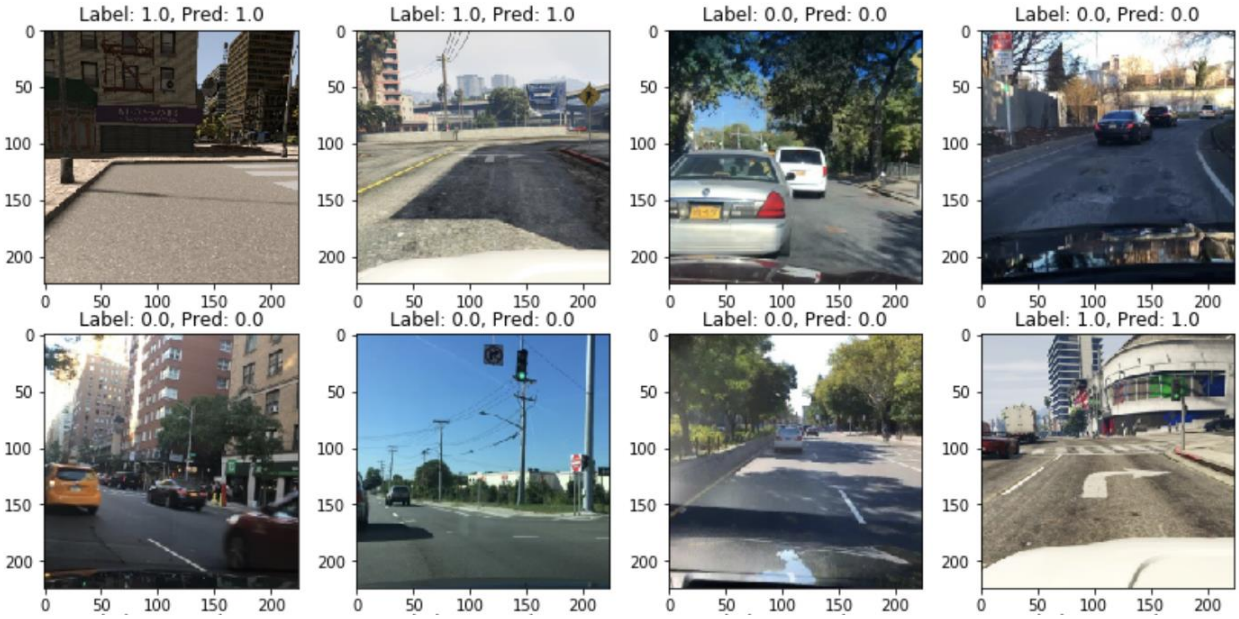
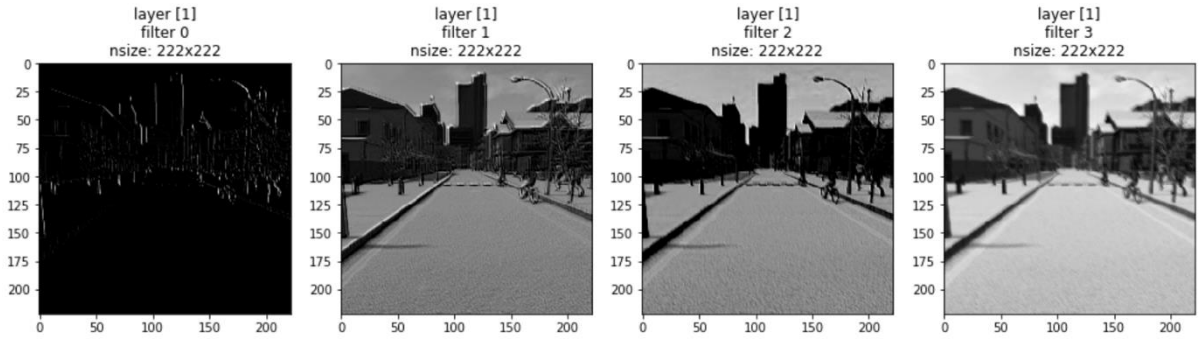


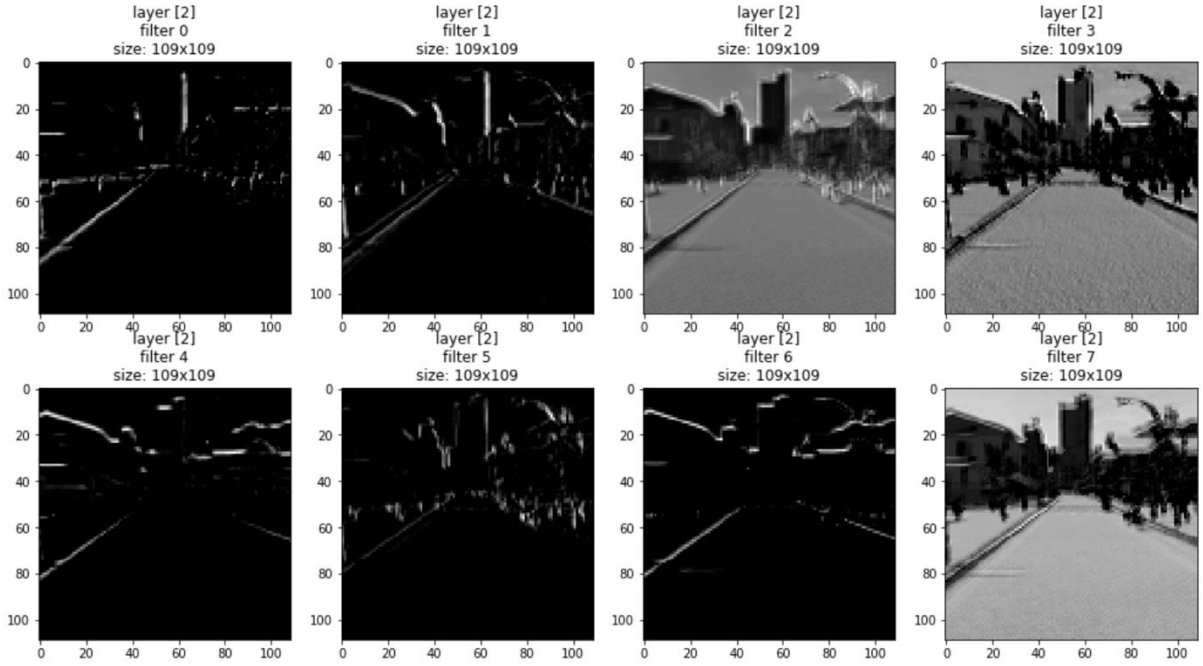
Рисунок 4.3 – Результати тренування класифікаторів на тестових даних

4.1.2 Інтерпретація моделей

Спираючись на карти активації після згорткових шарів (рисунок 4.4), можна зробити висновок, що нейронна мережа розглядає зображення в цілому, а не дивиться на конкретні її ділянки. Тобто нейронна мережа реагує на стиль малюнка, а не на конкретні об’єкти на ньому.



а)



б)

Рисунок 4.4 – Карти активації після згорткових шарів: а) –після першого згорткового шару , б) – після другого згортркового шару

Ваги першого повнозв'язного шару дивляться вичерпно на все зображення, свідоцтвом чого є рисунок 4.5, але деякі з ваг все-таки вказують на незначні артефакти, наприклад, автомобільну торпеду.

Побудовані нейронні мережі повністю розділяють реальні та синтетичні дані, орієнтуючись на стиль зображення, а не на семантичне значення об'єктів. Простір з лінійною роздільністю є важливим для нас з точки зору розуміння формальних відмінностей між реальними та синтетичними даними.

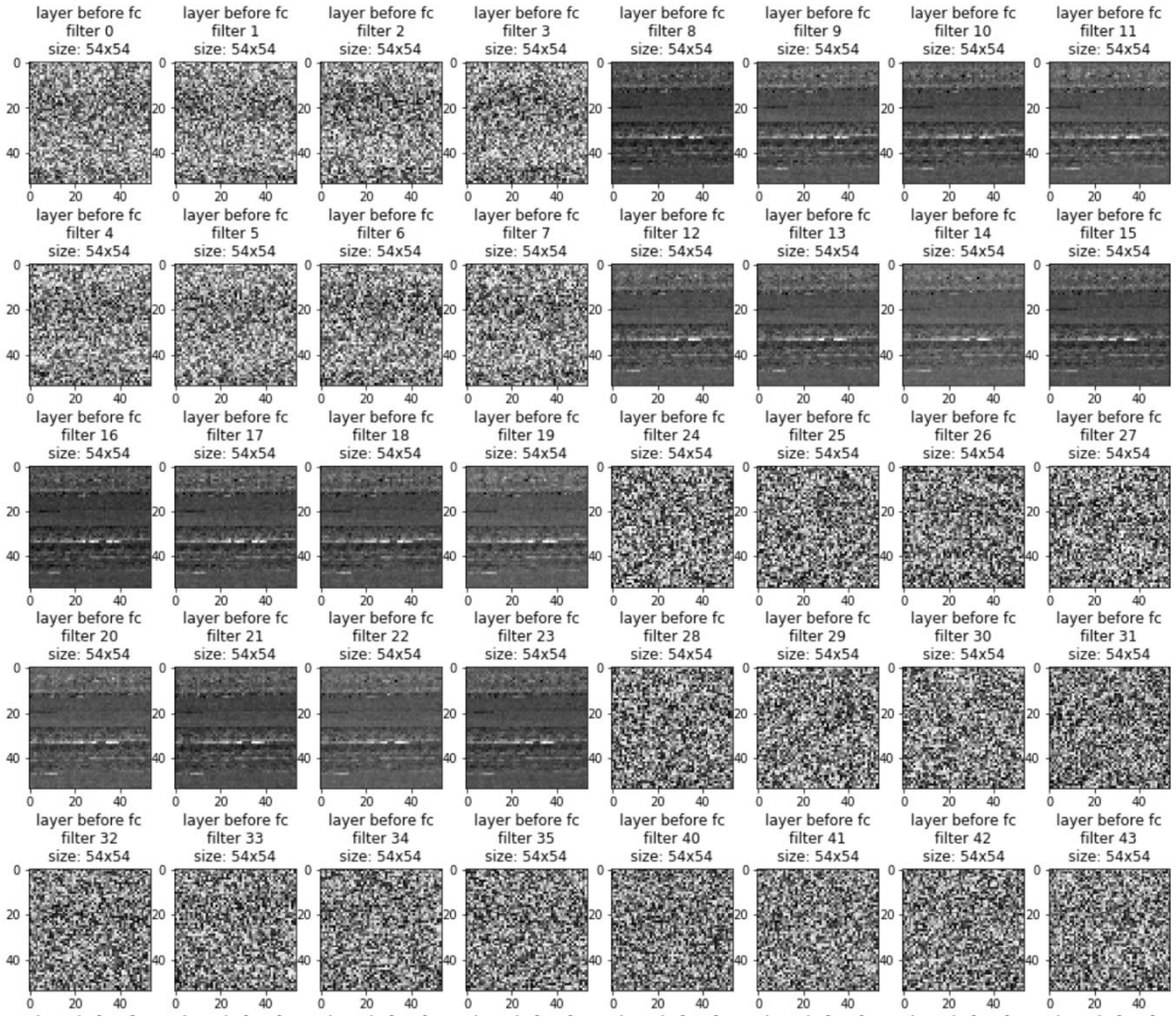


Рисунок 4.5 – Активаційні карти повнозв’язного шару

4.1.3 Аналіз простору стилів

Розглянемо передостанній шар побудованої нейронної мережі. Його розмір – 64 для усіх розмірів зображень. Далі в мережі ми використовуємо один лінійний шар із сигмоїдальною функцією активації.

Описати роботу прогнозування нашої нейронної мережі можна так: побудова карти в просторі R^{64} , побудова логістичної регресії у цьому просторі. Високі показники міри якості на тестових даних свідчать про хорошу лінійну відокремленість у проміжному просторі. Ми візуалізуємо цей простір за допомогою алгоритму PCA [9] (рисунок 4.6). Алгоритм може допомогти нам знайти таке лінійне відображення від 64-мірних

векторів до двовимірних – максимізуючи відсоток пояснень для дисперсії.

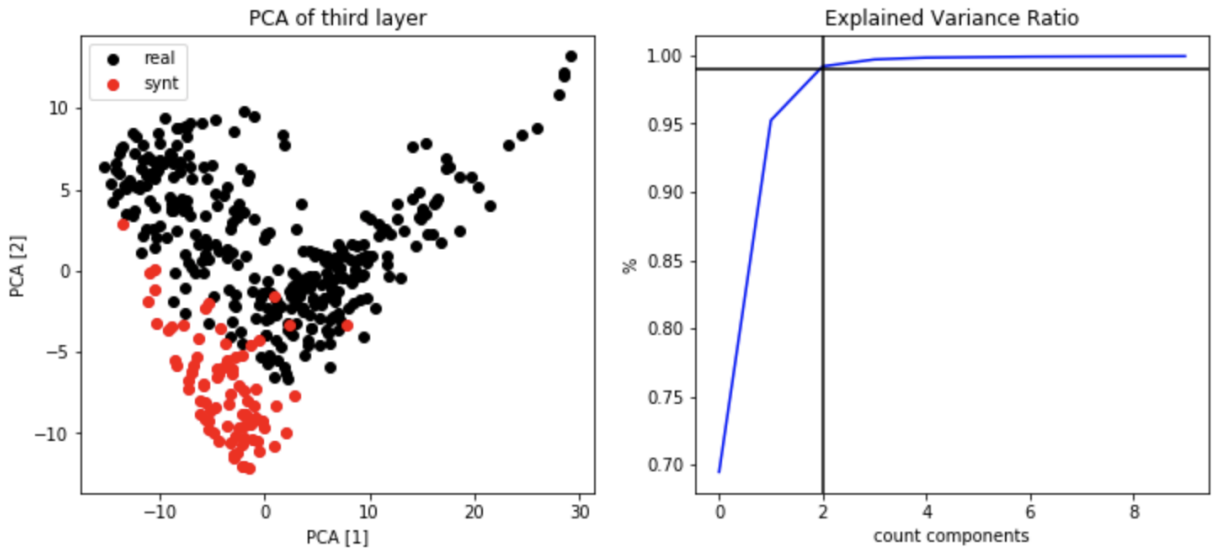


Рисунок 4.6 – Візуалізація прихованого простору

Візуалізація двох основних компонент розкладання PCA дає нам розуміння, що точки згруповані у два кластери. У той же час другий графік показує, що вже для 2 компонентів пояснюється майже вся варіативність даних у прихованому просторі. Тому візуалізація відображає реальність. Аналогічна картина спостерігається для всіх 4 нейронних мереж. Таким чином, реальні та синтетичні дані поділяються за стилем.

4.1.4 Аналіз простору змісту

Змагання ImageNet [2] сприяло появі такого напряму, як transfer learning [35]. Основна ідея полягає в тому, що, навчивши нейронну мережу на великому наборі даних, надалі її можна використовувати як екстрактор ознак високого рівня. Подальше використання означає заміну останніх шарів мережі новими та додаткове навчання для конкретно вашого набору даних. Творці концепції передачі стилів стверджують, що перші шари нейронних мереж відповідають за стиль зображення, а кінцеві – за його

зміст [8].

У нашому випадку зображення реальних та синтетичних даних, взяті з одного домену, тому вони містять однаковий вміст. Використання ознак високого рівня передбачає, що зберігається інформація про предмети на зображенні, але не про стиль.

У якості екстракторів ознак, ми використовуємо такі нейронні мережі, що навчені на ImageNet: AlexNet, [2] MobileNetV2 [26], VGG13 BN, VGG16 BN [27], MNasNet, MNasNet0.5 [35], ResNet152 [4], ResNeXt101 [33].

Ці мережі були обрані через їх невимовну популярність у сфері трансферного навчання. Отже мережі MobileNetV2, MNasNet, MNasNet0.5 невеликі та легкі, AlexNet – перший у своєму роді. VGG13 BN, VGG16 BN, ResNet152, ResNeXt101 – глибокі мережі, які дають гарні і перевірені на практиці результати.

В усіх мереж відрізали два останні класифікаційні шари. Пропускаючи реальні та синтетичні набори зображень по мережі, ми отримуємо по дві хмари точок для кожної мережі. Статистичні характеристики хмарин точок повинні бути однаковими, як в синтетичних, так і в реальних даних для кожної групи.

Для простоти ми будемо порівнювати середні значення. Для порівняння середнього значення для двох груп ми використовуємо двосторонній t-тест. З нульовою гіпотезою, що дві незалежні вибірки мають однакові середні значення. Оскільки вектори багатовимірні, ми робимо грубе припущення, що компоненти незалежні.

Це припущення не є цілком справедливим хоча б через те, що значення компонентів, згенеровано з однієї прихованої змінної. Для отримання міри рівності засобів ми вважаємо відсоток компонентів, щодо яких ми відкидаємо нульову гіпотезу. Чим менший відсоток, тим більше ознак з однаковим середнім.

Щоб перевірити рівність середніх значень у межах реальних та

синтетичних даних, для кожної групи ми беремо дві випадкові підгрупи без перетинів та проводимо тести, порівнюючи середні значення цих двох підгруп.

Отже, ми отримуємо 2 числа – відсоток ознак з різним математичним очікуванням для реальних та синтетичних даних. Цей експеримент проводиться незалежно для всіх нейронних мереж. Результати наведені в таблиці 4.2

Таблиця 4.2 – Порівняння математичного очікування у межах однієї групи даних.

Модель	Реальні зображення	Синтетичні зображення
AlexNet	7.59 %	3.77 %
MobileNetV2	1.48 %	2.38 %
VGG13_bn	0.25 %	0.85 %
VGG16_bn	0.24 %	0.32 %
MNasNet	1.54 %	1.89 %
MNasNet0.5	2.69 %	3.57 %
ResNet152	0.0 %	0.0 %
ResNext101	0.0 %	0.0 %

Практично для всіх мереж відсоток компонентів з різним математичним очікуванням не перевищує трьох відсотків. Цей експеримент показує нам, що з високою ймовірністю дані в межах однієї групи мають однаковий характер.

Оскільки високорівневі ознаки, зібрані з останніх шарів мережі, вони містять інформацію про вміст. Тому статистичні характеристики для хмари синтетичних даних повинні відповідати статистичним характеристикам

реальної хмари.

На практиці цього не відбувається, і ми можемо спостерігати це в таблиці 4.3.

Таблиця 4.3 – Порівняння математичного очікування між реальними та синтетичними даними.

Модель	Відсоток неоднакових значень середнього для компонентів
AlexNet	96.35 %
MobileNetV2	88.58 %
VGG13_bn	91.95 %
VGG16_bn	86.17 %
MNasNet	86.48 %
MNasNet0.5	89.96 %
ResNet152	26.38 %
ResNext101	32.77 %

У більшості мереж багато ознак мають різні математичні очікування. Мережі ResNet152 та ResNext101 заслуговують на особливу увагу, якщо порівнювати рівності середніх у кожній із груп для цих 2 мереж, всі ознаки мали однакові середні значення.

У разі порівняння середніх значень для двох різних груп ці 2 мережі мають найменший відсоток різних математичних очікувань.

В якості однієї з інтерпретацій висловимо припущення, що ті ознаки, щодо яких відповідали очікування, були багатими на вміст, а ті, що не відповідали, були стилістичними, але дивлячись на типи розподілу функцій, ми можемо побачити, що для всіх мереж, крім цих двох, розподіли є нормальними, для ResNet152 та ResNext101 вони є експоненціальними.

На рисунку 4.7 представлені розподіли ознак досліджуваних мереж, де спостерігається підтвердження нашому припущенню.

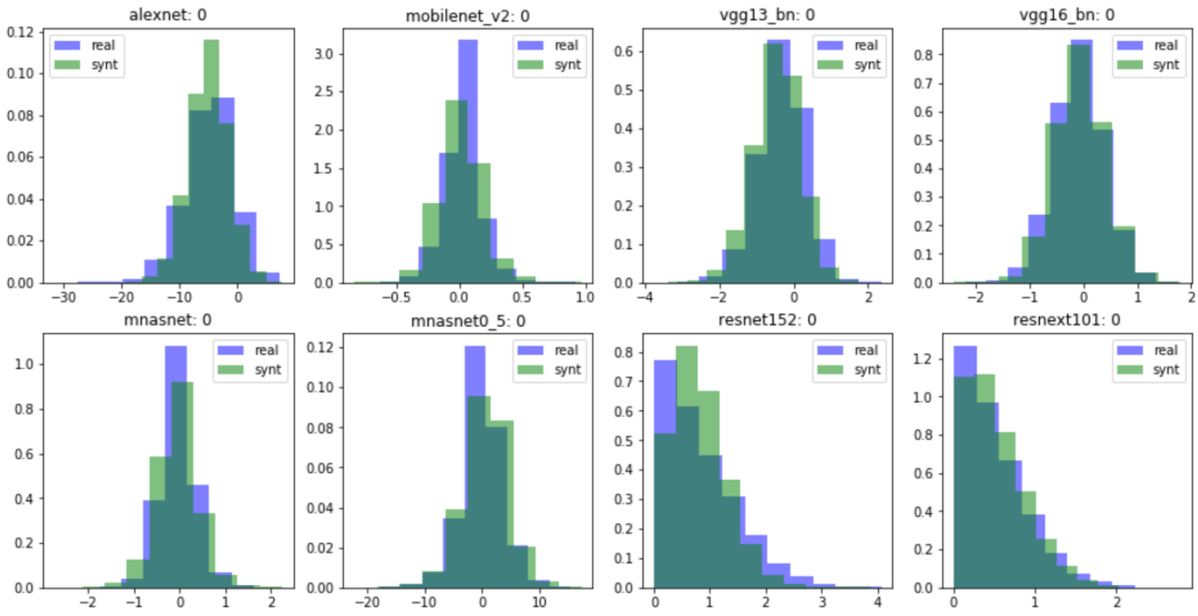


Рисунок 4.7 – Розподіли ознак

Відомо, що при великих значеннях вибірки t-тест також працює для аномальних розподілів. Швидше за все, різниця в процентному порядку для цих двох мереж пояснюється саме сімейством розподілу їх ознак.

Реальні та синтетичні дані різні, якщо: нейронна мережа ділить ці два класи за стилем зображення, а не за його змістом. Розподіл високорівневих презентацій має статистично значущі відмінності.

Ми проводили подібні експерименти на наборі даних із собаками. Невелика нейромережа не змогла відрізнити справжніх собак. Ми припускаємо, що реальні та створені задопомогою GAN зображення мали дуже схожий стиль.

Навчені мережі неефективно виділяють контент і, як наслідок, містять більше інформації про стиль, який у цьому випадку відповідний для обох груп. На жаль, ніяких статистичних відмінностей тести не виявили. Аналіз формальних відмінностей для наборів даних із собаками

більше не проводився, і подальші експерименти з ним припинилися.

4.2 Побудова репрезентативного прихованого простору

Працювати з зображеннями досить складно. Набагато зручніше працювати з числовими векторами. Побудувати функцію відображення від простору зображення до прихованого простору досить просто; однак побудувати функцію відновлення набагато складніше.

Прихований простір повинен мати властивість компактності. Компактність передбачає, що невелика зміна зображення призводить до невеликої зміни прихованого простору, і, навпаки, невеликі зміни у прихованому просторі не повинні активно змінювати зворотне відображення.

Побудова простору з такими властивостями, а також функції відображення також здійснюється шляхом побудови варіаційного автоенкодера.

4.2.1 Побудова VAE

Варіаційний автокодер складається з двох мереж – кодера і декодера. Архітектури цих двох мереж визначають складність відображень. У класичному VAE, енкодер – це тришарова нейронна мережа, яка звужує дані. Декодер також являє собою 3-шарову нейронну мережу.

Особливістю є те, що кодер генерує два вектори – середнє значення і дисперсію. Потім застосовується трюк з репараметризацією [13]. Під час репараметризації випадкове значення, що генерується від нормального розподілу з вихідними параметрами кодера, і цей вектор подається на вхід декодера. Всі шари є повнозв'язними.

На рисунку 4.8 представлена архітектура класичного варіаційного автоенкодера.

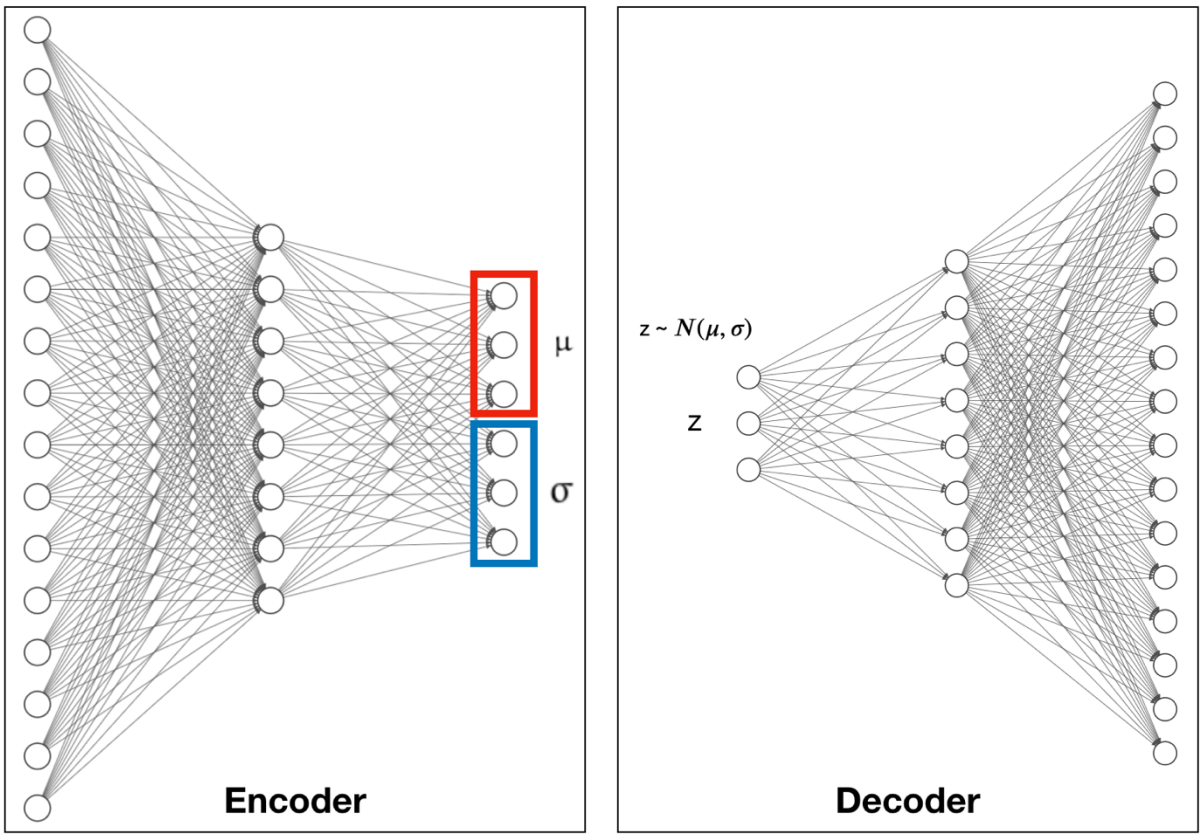


Рисунок 4.8 – Архітектура класичного варіаційного автоенкодера

Проте застосувати цю модель в чистому вигляді неможливо. Загальний розмір ваг нейронної мережі квадратично зростає від кількості нейронів. Отримана мережа буде величезною.

Нам необхідно змінити архітектуру VAE, щоб зменшити розмір мережі. Внутрішня структура та ідея з репараметризацією залишаються однаковими. Щоб зробити можливим застосування VAE прийнятного розміру на початку енкодера і в кінці декодера, ми додаємо шари згортки, класичні для комп'ютерного зору. Тільки ті функції, які відображають вміст зображення, мають сенс передаватися на вхід енкодера.

У наших експериментах ми використовували мережі ResNet152 та VGG11 BN, що навчені на ImageNet. Вибір був зосереджений на цих двох моделях, оскільки нам вдалося підібрати оптимальні параметри оптимізатора та швидкість навчання для них. Після декодера, додали три шари згортки, з батч нормалізацією, і параметри шарів обиралися таким

чином, щоб на виході ми отримували зображення. Для розміру входу зображення розтягувалося за допомогою білінійної інтерполяції.

Внутрішня архітектура VAE описана наступним чином:

$$[\boldsymbol{fc-4096}]-[\boldsymbol{fc-2048}]- \\ [\boldsymbol{2\ x\ fc-512}]-[\boldsymbol{fc-2048}]-[\boldsymbol{fc-4096}] \tag{4.2}$$

Архітектура модифікованого VAE представлена на рисунку 4.9.

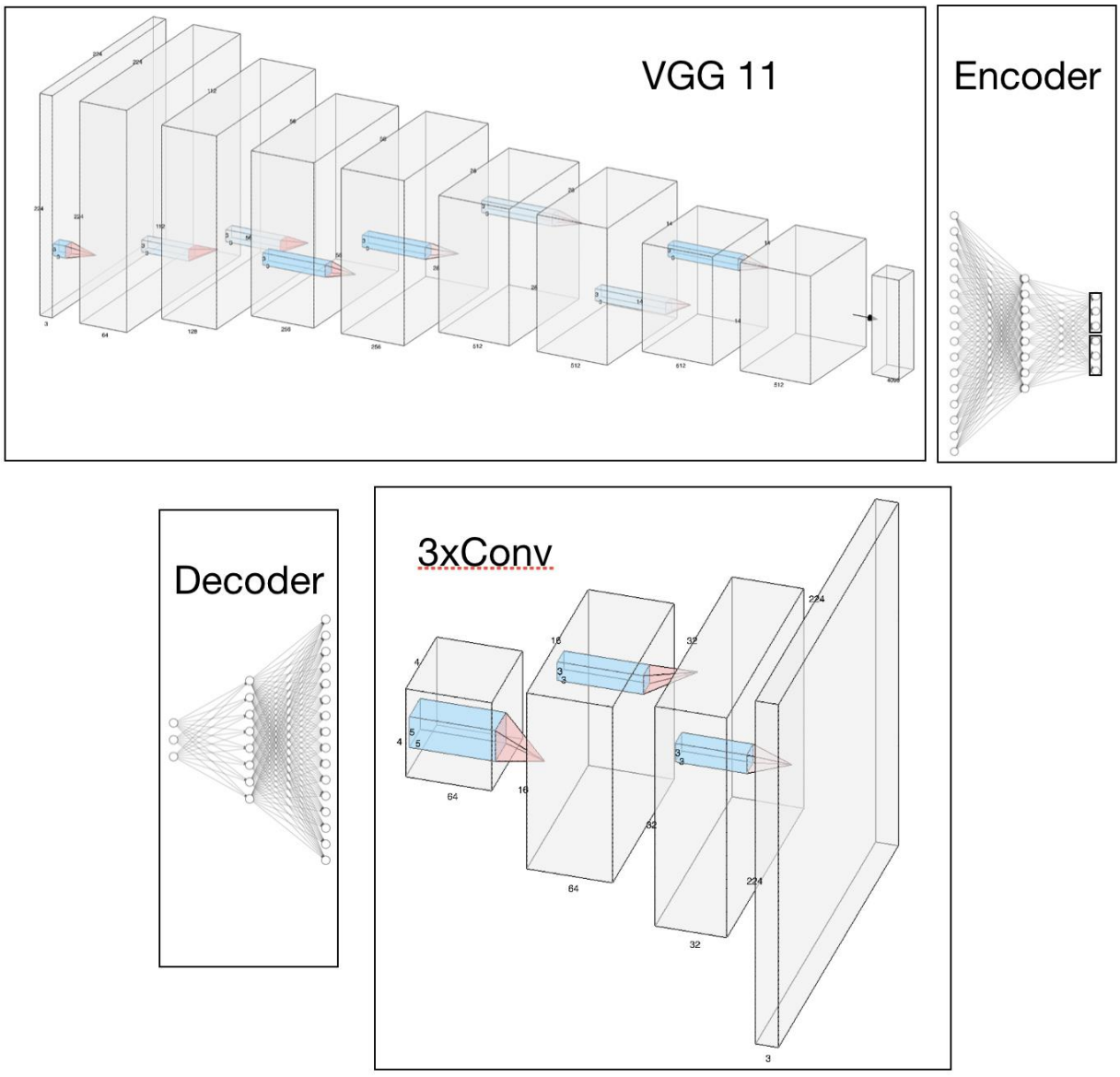


Рисунок 4.9 – Модифікований автоенкодер

4.2.2 Тренування модифікованого автоенкодера

Тренування проводилося з використанням алгоритму оптимізації адаптивного моменту (Adam). Цей алгоритм був обраний тому, що він показує найшвидші результати оптимізації, що було дуже важливим аспектом з огляду обраної архітектури.

Алгоритм Adam або Адаптивна оптимізація моментів поєднує в собі евристику як імпульсу, так і RMSProp (Root Mean Square Propagation) алгоритмів. Тут ми обчислюємо експоненційне середнє значення градієнта, а також експоненційне середнє значення квадратів градієнта для кожного параметра. Щоб обрати наш крок навчання, ми помножимо нашу швидкість навчання на середню величину градієнта і ділимо його на середньоквадратичний квадрат експоненціальної середньої площі градієнтів. Потім додаємо оновлення. Нижче у формулах 4.3 – 4.6 наведені формули для розрахунку ваг:

$$\boldsymbol{v_t = \beta_1 \cdot v_{t-1} - (1 - \beta_1) \cdot g_t} \tag{4.3}$$

$$\boldsymbol{s_t = \beta_2 \cdot s_{t-1} - (1 - \beta_2) \cdot g_t^2} \tag{4.4}$$

$$\boldsymbol{\Delta w_t = -\eta \frac{v_t}{\sqrt{s_t + \varepsilon}} \cdot g_t} \tag{4.5}$$

$$\boldsymbol{w_{t+1} = w_t + \Delta w_t} \tag{4.6}$$

де η – вихідний крок навчання;

g_t – значення градієнту в момент часу t;

v_t – експоненційне середнє значення градієнта;

s_t – експоненційне середнє значення квадратів градієнта;

$\beta_1,\ \beta_2$ – гіперпараметри.

Гіперпараметр β_1 зазвичай зберігається близько 0,9, а β_2 – 0,99. ε обрано як $1e-10$.

Кожна модель тренувалася 200 епох. Learning rate – змінювався

відповідно до CiclicLR [28] з інтервалом кроків від $1e-5$ до $1e-2$. Повний цикл зміни LR – займає 1000 кроків оптимізатора. В процесі навчання ми застосовували набір аугментацій, для того, щоб збагатити навчальну вибірку. На рисунках 4.10 – 4.11 представлені процеси навчання модифікованих VAE.

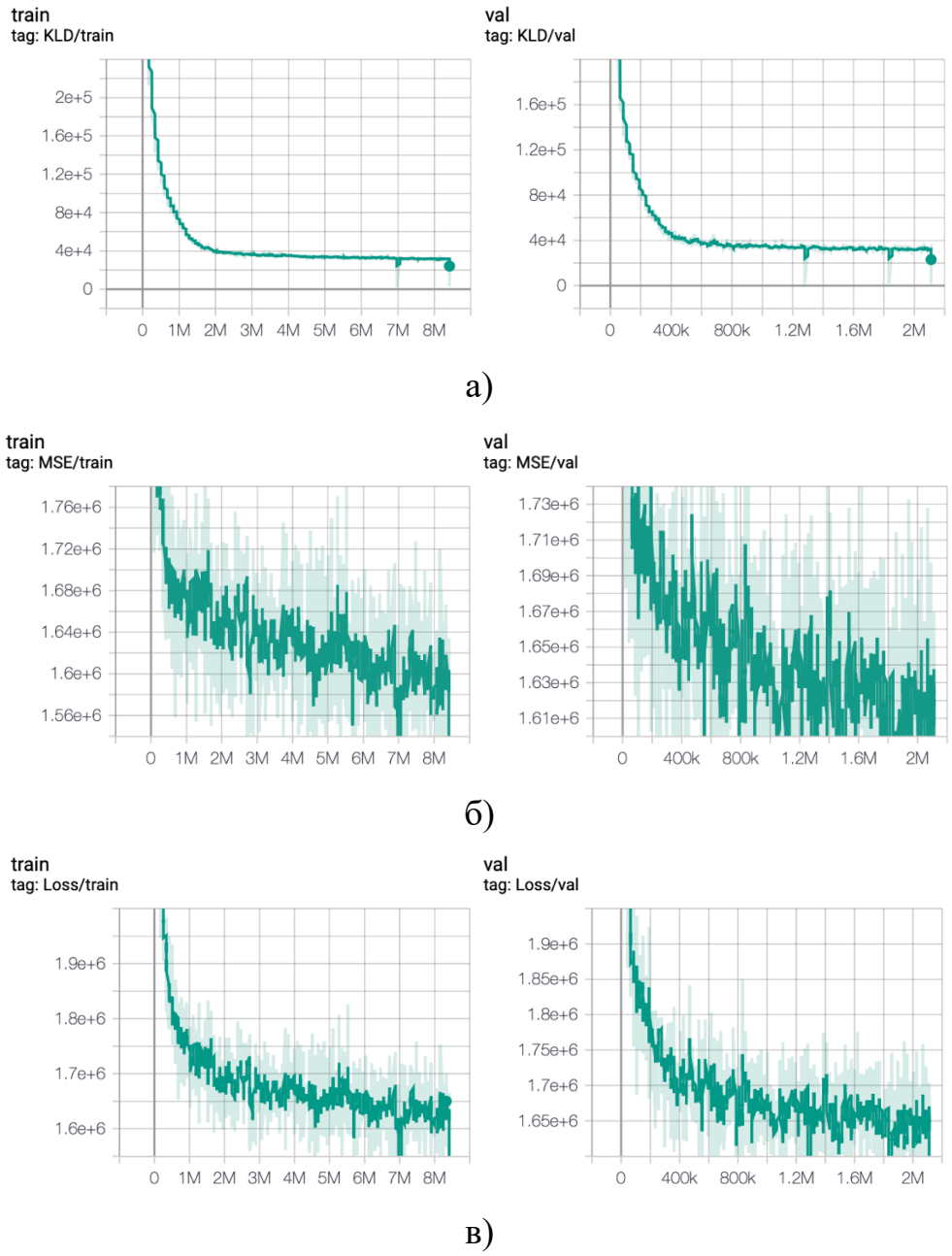
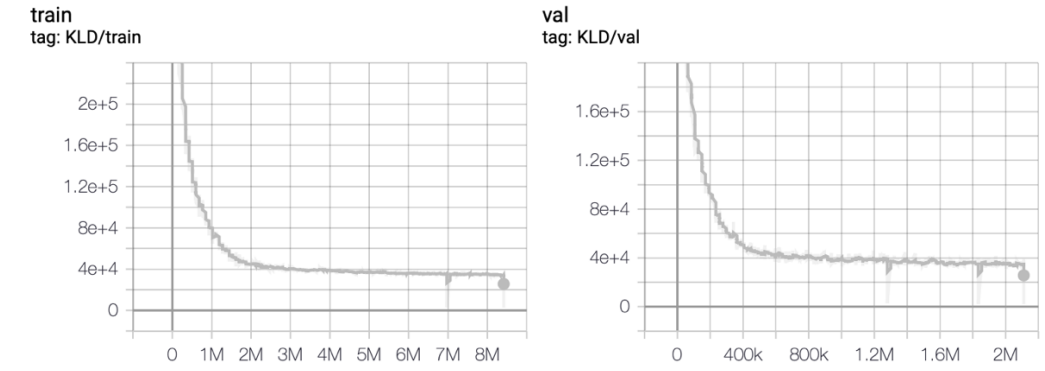
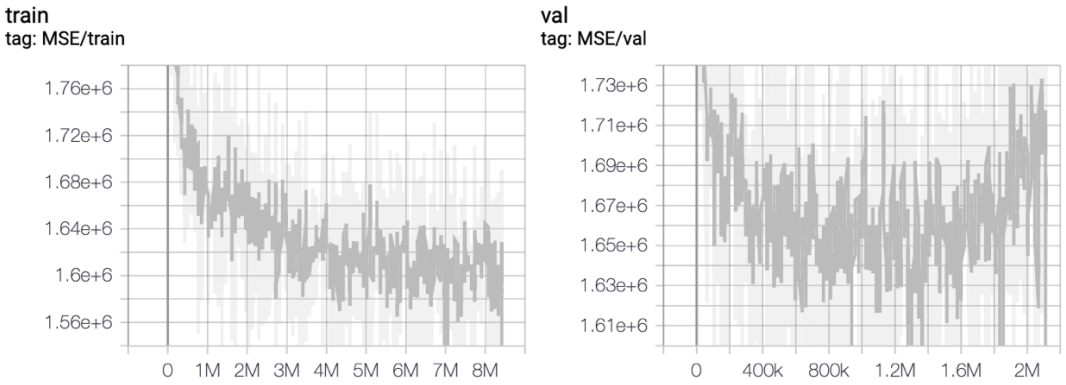


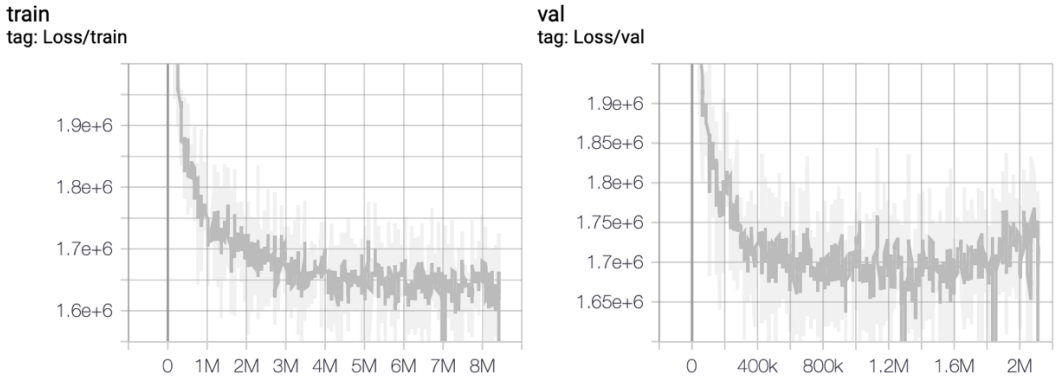
Рисунок 4.10 – Графіки збіжності функцій втрат для модифікації VGG16 BN: а) – Кульбака-Лейблера, б) – середньоквадратичного піксельного відхилення, в) – загальна функція втрат



а)



б)



в)

Рисунок 4.11 – Графіки збіжності функцій втрат для модифікації ResNet152: а) – Кульбака-Лейблера, б) – середньоквадратичного піксельного відхилення, в) – загальна функція втрат.

Навчання проходило на змішаному наборі даних реальних та синтетичних. Результати, отримані для мереж представлені на рисунку 4.12.

а)

б)

Рисунок 4.12 – Результати навчання для: а) – ResNet152 модифікації, б) – VGG16 BN модифікації

Як можемо побачити з результатів зображення дуже розмиті, і мережі добре відновлюють кольорову гаму зображення, але істотно не відновлюють деталі. Через велику глибину мережі градієнти практично не досягають початкових шарів. Як результат, початкові фільтри майже не змінюються, і шаблони, типові для наших даних, не можуть формуватися.

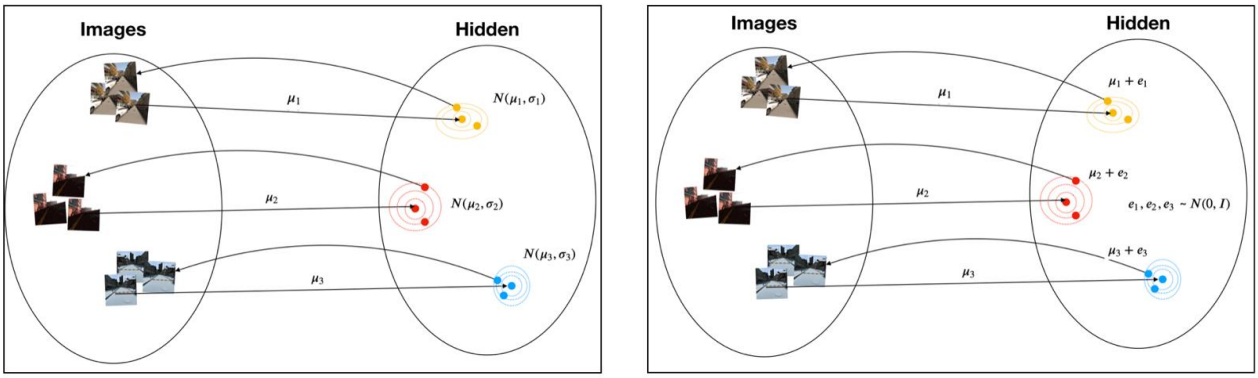
4.2.3 Побудова VAE з residual connection

Проблема загасання градієнта в мережі вирішується шляхом зміни архітектури, а саме шляхом додавання residual connection [4]. Зміни в схемі репараметризації також сприяли покращенню результатів.

У стандартній схемі репараметризації використовується така схема: $z_i N\ (\mu, \sigma)$, де z_i – вхід декодера, а μ, σ генерується енкодером. Крім того, під час оптимізації варіаційного автоенкодера функція вт включає відстань

Кульбака - Лейблера [15] між розподілами $N(\mu, \sigma)$ і $N(0, I)$.

На рисунку 4.13 представлені схеми репараметризації прихованого простору.



а) б)

Рисунок 4.13 – Схеми репараметризації: а) – класична, б) – модифікована

Використання хитрості з репараметризацією пояснюється наступною ідеєю. Енкодер створює відображення від простору зображення до прихованого простору.

Після репараметризації (рисунок 4.13 (а)) вибирається випадкова точка, взята в околицях математичного очікування, сформованого енкодером, а потім декодер тренується відновити початкове зображення з обраної точки.

Випадковий вектор, домножується на сигму, що генерується енкодером, і додається до математичного очікування. Така зміна прихованого простору призводить до незначного спотворення початкового зображення. Ця ідея забезпечує простір властивістю компактності.

Альтернативою можемо позбутися генерації вектора сигм з енкодера та відбирати нову точку з одразу розподілу $N(\mu_1, I)$ (рисунок 4.13 (б)). Це зменшує кількість мережевих параметрів, а також покращує конвергенцію. Така схема була запропонована в роботі UNIT [17], але там вона використовувалася без особливої мотивації.

Наступні формули описують архітектуру кодера і декодера для VAE із залишковими з'єднаннями на базі архітектури UNIT.

Енкодер описується формулою:

$$[\mathbf{32\ Conv7x7}] - [\mathbf{64\ Conv4x4}] - [\mathbf{128\ Conv4x4}] - \mathbf{4x}[\mathbf{64 - ResBlock}] \tag{4.7}$$

Декодер описується формулою:

$$[\mathbf{4x}[\mathbf{64 - ResBlock}] - [\mathbf{Upsample}] - [\mathbf{128\ Conv5x5}] - [\mathbf{Upsample}] - [\mathbf{64\ Conv5x5}] - [\mathbf{Upsample}] - [\mathbf{3\ Conv7x7}] \tag{4.8}$$

ResBlock – являє собою два шари згортки, включаючи residual connection. Ми натренували усього 3 мережі, і навчання проходило для мереж з різними розмірами residual блоків: 16, 32 та 64.

Реконструкції для цих мереж представлені на рисунку 4.14, зверху вниз по збільшенню розміру residual блоків.

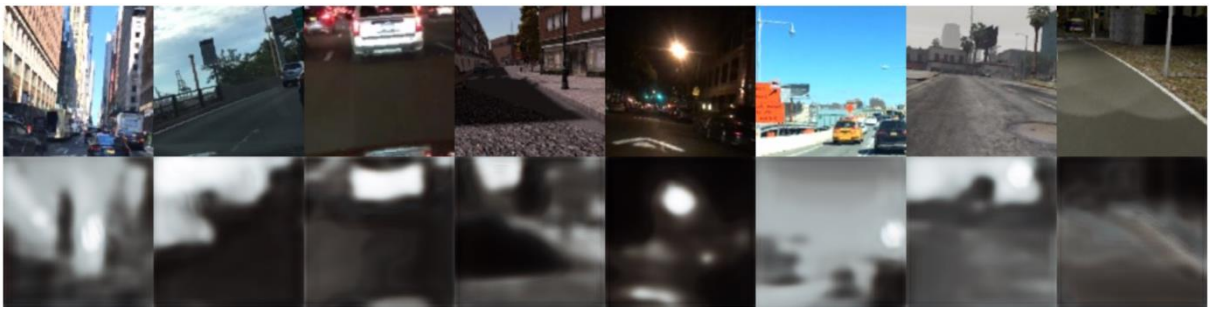
Остання мережа краще відображає кольорові характеристики зображення, швидше за все, цей ефект стався через збільшення кількості ваг мережі, а остання мережа вчилася з урахуванням більш суворих кольорових аугментацій.

Усі мережі навчилися визначати особливості рельєфу та внутрішнього середовища міста. Навчання тривало 100 епох з використанням оптимізатора Adam.

Learning rate – змінювався відповідно до CiclicLR з інтервалом кроків від $1e-7$ до $1e-4$.

Повний цикл зміни LR – займає 2000 кроків оптимізатора.

Прихований простір у всіх 3 мережах значно більший, ніж у попередніх моделей.



а)

б)

в)

Рисунок 4.14 – Результати для мереж з ріним розміром residual блоків: а) – 16, б) – 32, в) – 64

4.2.3 Ефект мультимодальності даних

Усі реконструйовані зображення показують значне зниження якості. Такий ефект притаманний практично всім варіаційним автоенкодерам. Однак ця особливість реконструйованих образів пояснюється ще одним фактом.

Під час оптимізації розподіл функцій у прихованому просторі має тенденцію до приблизно нормального, і кодер може досить добре

наблизитися до початкового розподілу, якщо він спочатку був близький до нормального.

Завдання, поставлене енкодеру – виконувати кодування з високорозмірного простору до більш низькорозмірного, зберігаючи при цьому сімейство розподілів.

Набір даних, з яким ми працюємо, має щонайменше дві моди: перша відповідає реальним даним, друга – синтетичним. Також у групі синтетичних даних можуть формуватися моди відповідно до умов, за яких відбулася генерація.

Наприклад, це може бути час доби, погода чи сезон. У частині набору даних з реальними даними ми також можемо знайти інші моди. Для простоти обмежимось двома – модою реальних даних і синтетичних. Таким чином, вихідні дані мають бімодальний розподіл.

Мінімізований функціонал варіаційного автоенкодера включає міру близькості між сформованим розподілом і нормальним розподілом $N(0, I)$. Оптимальним рішенням було б зведення обох мод до нормального розподілу – $N(0, I)$.

На практиці ж розподіл, що генерується енкодером, є бімодальним, і ми не можемо досягти оптимума для міри близькості.

Тому, щоб уникнути впливу даного ефекту на якість генерації в подальшому ми вирішили зменшити вплив функції Кульбака-Лейблера на загальну функцію втрат.

4.2.4 Тести прихованого простору VAE

Проведемо тести на рівність середнього в прихованому просторі, згенерованому VAE.

У таблиці 4.4 представлений відсоток ознак з неоднаковим математичним очікуванням в межах одного типу даних.

У таблиці 4.5 наведено від неоднакових математичних очікувань

при порівнянні хмар реальних та синтетичних даних.

Таблиця 4.4 – Порівняння математичного очікування у межах однієї групи даних.

Модель	Реальні зображення	Синтетичні зображення
VGG11 VAE	2,55 %	3,76 %
ResNet152 VAE	1,87 %	2,14 %
Residual VAE Hidden-4096	3,55 %	2,61 %
Residual VAE Hidden-8192	2,71 %	2,67 %
Residual VAE Hidden-16384	3,75 %	2,50 %

Таблиця 4.5 – Порівняння математичного очікування між реальними та синтетичними даними.

Модель	Відсоток неоднакових значень середнього для компонентів
VGG11 VAE	82,75 %
ResNet152 VAE	83,93 %
Residual VAE Hidden-4096	85,81 %
Residual VAE Hidden-8192	86,27 %
Residual VAE Hidden-16384	84,65 %

На основі отриманих результатів можна сказати, що варіаційним автоенкодерам вдалося побудувати прихований простір, в якому реальні та синтетичні дані утворюють кластери.

4.3 Доведення сили варіаційних автоенкодерів

При перетворенні середнього значення синтетичних даних у прихованому просторі у реальне середнє та подальшій подачі цих точок на вхід декодера, на виході ми отримуємо зображення гарної якості, представлені на рисунку 4.15, дуже схожі на справжні.

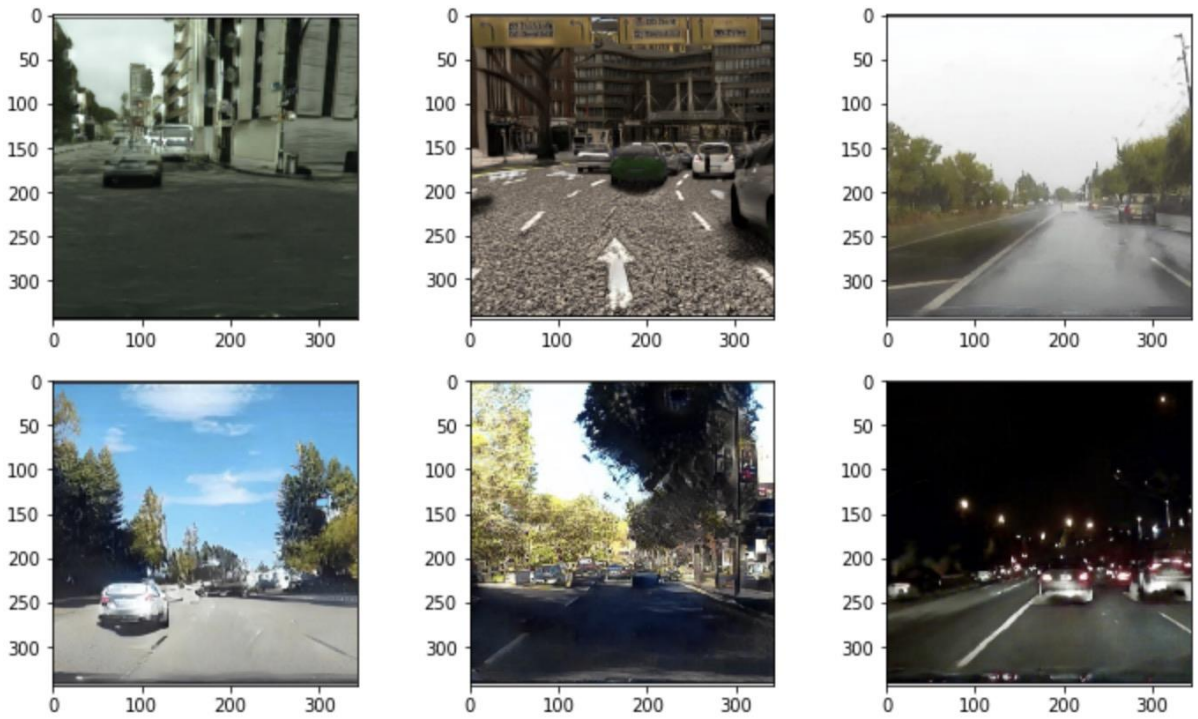
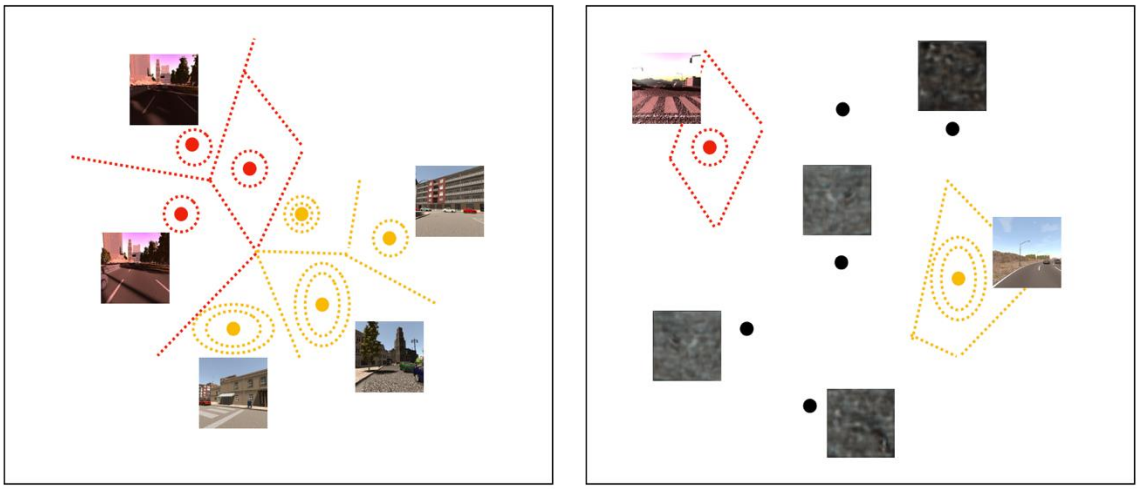


Рисунок 4.15 – Згенеровані зображення

Придивившись уважніше до точок у прихованому просторі, виявляється, що він досить насичений. Основна частина точок, з яких відновлюються зображення, перебуває у багатовимірному кубі з обмеженнями значень від -1,5 до 1,5. Кількість точок, зосереджених у цій області, становить близько вісімдесяти тисяч. Загальна щільність становить приблизно двадцять шість тисяч зображень на одну звичайну одиницю вимірювання простору.

На рисунку 4.16 зображені різні за щільністю приховані простори.

Маючи достатню щільність точок у прихованому просторі, воно має контентовмісні зображення, у кожній із точок (рис. 4.16 (а)). Однак існують випадки, коли простір буває розрідженим (рис. 4.16 (б)), і випадково прийнята точка навряд чи належить до району з належною реконструкцією. Це відбувається у випадку недостатнього обсягу тренувальних даних.



а) б)

Рисунок 4.16 – Візуалізація прихованого простору VAE: а) – з щільним простором, б) – з розрідженим простором

В нашому ж випадку ця проблема вирішується завдяки аугментації даних.

Для того, щоб генерувати якісну синтетику, принаймні необхідно, щоб простір був щільним.

Отже, маючи варіаційний автоенкодер, здатний будувати щільні приховані простори, подальше прирівнювання середнього значення синтетичних зображень до необхідного середнього реальних даних може призвести до формування якісних синтетичних даних.

4.4 Технічні деталі реалізації

Для реалізації експериментів ми використовували фреймворк глибокого навчання – PyTorch [22] з інтеграцією python 3.7.

Pytorch, побудований на основі C ++, працює на нижчому рівні і вимагає більше індивідуальних конфігурацій. Тим не менш, він найбільш відомий своєю гнучкістю, короткою тривалістю тренувань та можливостями налаштування. У нашому випадку обробка великого набору даних зображень забирає багато часу, тому цей фреймворк корисний для зменшення часу на тренування.

Більше того, PyTorch дає можливість додати деякі власні зміни до мереж, оптимізацію, розрахунок функції втрат, що забезпечує можливість налаштування архітектури для будь-якого експерименту.

PyTorch забезпечує можливість роботи з внутрішніми нейронами мережі, що дозволило провести детальний аналіз мереж, а також ефективне навчання. Навчальні моделі були взяті з бібліотеки torchvision.

Усі мережі пройшли навчання на процесорі GeForce GTX 1080 TI з 11.264 Мб пам'яті GDDR5X. Навчання варіаційного автокодера за одну епоху тривало близько 50 хвилин. Загальна підготовка мереж загалом зайняла від 4 до 8 днів. Навчання мережі класифікації протягом 20 епох тривало близько години.

ВИСНОВКИ

В атестаційній роботі магістра розглядається задача розробки методу генерації масивів даних для навчання глибинних нейронних мереж.

У цій роботі ми розглянули актуальну проблему ефективної генерації синтетичних даних. Після аналізу існуючих підходів, що оточували наше рішення, було запропоновано метод, який поєднує класичну статистику із сучасними методами нейронних мереж.

Спочатку ми поставили собі за мету довести, що існує статистично значна різниця у розподілі реальних та штучно згенерованих даних, і під час першої ітерації експериментів ми успішно підтвердили цю гіпотезу, довівши її на основі ознак, що містять зміст, отриманий з навчених нейронних мереж. Стилістичні відмінності також були підтверджені за допомогою класифікатора.

На цьому етапі нам також довелося виправити напрямок наших експериментів, виключивши з розгляду не зовсім вдало відібрані дані про собак, оскільки виявилося, що стиль синтетичних зображень занадто близький до стилю реальних.

Друге завдання полягало в роботі з автоенкодерами, щоб отримати прихований простір, і ми не знайшли там стильових відмінностей. Проаналізувавши всі отримані результати на основі двох ітерацій експериментів, ми спробували генерувати синтетичні дані відповідно до обґрунтування в пайплайні експериментів.

Запропонований метод показав, що використання варіаційних автоенкодерів для генерації синтетичних даних дає якісні результати.

Дану роботу, поглиблюючись в особливості використовуваних мережевих архітектур та збільшуючи обсяги навчальних даних та підходів алгоритмів навчання. Зі статистичної сторони було б раціонально розглянути інші статистичні характеристики.

ПЕРЕЛІК ПОСИЛАНЬ

1. Davis, Jesse and Mark Goadrich (2006). “The relationship between Precision-Recall and ROC curves”. In: ICML ’06.

2. Deng, J. et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: CVPR09.

3. Goodfellow, Ian J. et al. (2014). “Generative Adversarial Nets”. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. NIPS’14. Cambridge, MA, USA: MIT Press, pp. 2672–2680. URL: http://dl.acm. org/citation.cfm?id=2969033.2969125.

4. He, Kaiming et al. (2015). “Deep Residual Learning for Image Recognition”. In: CoRR abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

5. Hernandez-Juarez, Daniel et al. (2017). “Slanted Stixels: Representing San Francisco’s Steepest Streets”. In: British Machine Vision Conference (BMVC), 2017.

6. Hoffman, Judy et al. (2017). “CyCADA: Cycle-Consistent Adversarial Domain Adap- tation”. In: ICML.

7. Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: ArXiv abs/1502.03167.

8. Jing, Yongcheng et al. (2017). “Neural Style Transfer: A Review”. In: IEEE transactions on visualization and computer graphics.

9. Jolliffe, I.T. (1986). Principal Component Analysis. Springer Verlag.

10. Karacan, Levent et al. (2016). “Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts”. In: ArXiv abs/1612.00215.

11. Kingma, Diederik P. and Jimmy Ba (2014). Adam: A Method for Stochastic Optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd Inter- national Conference for Learning Representations, San Diego, 2015. URL: http://arxiv.org/abs/1412.6980.

12. Kingma, Diederik P. and Max Welling (2013). “Auto-Encoding Variational Bayes”. In: CoRR abs/1312.6114.

13. Kingma, Durk P, Tim Salimans, and Max Welling (2015). “Variational Dropout and the Local Reparameterization Trick”. In: Advances in Neural Information Processing Systems 28. Ed. by C. Cortes et al. Curran Associates, Inc., pp. 2575–2583. URL: http://papers.nips.cc/paper/5666-variational-dropout-and-the-local- reparameterization-trick.pdf.

14. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). “ImageNet Classi- fication with Deep Convolutional Neural Networks”. In: Advances in Neural In- formation Processing Systems 25. Ed. by F. Pereira et al., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

15. Kullback, Solomon and R. A. Leibler (1951). ON INFORMATION AND SUFFICIENCY.

16. Liu, Ming-Yu, Thomas Breuel, and Jan Kautz (2017a). “Unsupervised Image-to-Image Translation Networks”. In: ArXiv abs/1703.00848.

17. Liu, Ming-Yu, Thomas Breuel, and Jan Kautz (2017b). “Unsupervised Image-to-Image Translation Networks”. In: CoRR abs/1703.00848. arXiv: 1703.00848. URL: http: //arxiv.org/abs/1703.00848.

18. Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). “Fully convolutional networks for semantic segmentation”. In: 2015 IEEE Conference on Computer Vi sion and Pattern Recognition (CVPR), pp. 3431–3440.

19. Marcus, Gary (2018). “Deep Learning: A Critical Appraisal”. In: CoRR abs/1801.00631. arXiv: 1801.00631. URL: http://arxiv.org/abs/1801.00631.

20. Martinez, Mark et al. (2017). “Beyond Grand Theft Auto V for Training, Testing and Enhancing Deep Learning in Self Driving Cars”. In: ArXiv abs/1712.01397.

21. Parkhi, Omkar M. et al. (2012). “Cats and Dogs”. In: IEEE Conference on Computer Vision and Pattern Recognition.

22. Paszke, Adam et al. (2017). “Automatic Differentiation in PyTorch”. In: NIPS 2017 Workshop on Autodiff. Long Beach, California, USA. URL: https://openreview.net/forum?id=BJJsrmfCZ.

23. Richter, Stephan R. et al. (2016). “Playing for Data: Ground Truth from Computer Games”. In: CoRR abs/1608.02192. arXiv: 1608.02192. URL: http://arxiv.org/ abs/1608.02192.

24. Ros, Germán et al. (2016). “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes”. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3234–3243.

25. Saenko, Kate et al. (2010). “Adapting Visual Category Models to New Domains”. In: Proceedings of the 11th European Conference on Computer Vision: Part IV. ECCV’10. Berlin, Heidelberg: Springer-Verlag, pp. 213–226. ISBN: 3-642-15560-X, 978-3-642- 15560-4. URL: http://dl.acm.org/citation.cfm?id=1888089.1888106.

26. Sandler, Mark et al. (2018). “Inverted Residuals and Linear Bottlenecks: Mobile Net- works for Classification, Detection and Segmentation”. In: CoRR abs/1801.04381. arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

27. Simonyan, Karen and Andrew Zisserman (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. cite arxiv:1409.1556. URL: http://arxiv.org/ abs/1409.1556.

28. Smith, Leslie N. (2015). “No More Pesky Learning Rate Guessing Games”. In: CoRR abs/1506.01186. arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186.

29. Su, Jong-Chyi et al. (2019). “Active Adversarial Domain Adaptation”. In: CVPR Workshops.

30. Torralba, A. and A. A. Efros (2011). “Unbiased Look at Dataset Bias”. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. CVPR ’11. Washington, DC, USA: IEEE Computer

Society, pp. 1521–1528. ISBN: 978-1-4577- 0394-2. DOI: 10.1109/CVPR.2011.5995347. URL: https://doi.org/10.1109/ CVPR.2011.5995347.

31. Tzeng, Eric et al. (2014). “Deep Domain Confusion: Maximizing for Domain Invariance”. In: CoRR abs/1412.3474. arXiv: 1412.3474. URL: http://arxiv.org/abs/ 1412.3474.

32. Wood, Erroll et al. (2016). “Learning an Appearance-based Gaze Estimator from One Million Synthesised Images”. In: Proceedings of the Ninth Biennial ACM Sympo- sium on Eye Tracking Research & Applications. ETRA ’16. New York, NY, USA: ACM, pp. 131–138. ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857492. URL: http://doi.acm.org/10.1145/2857491.2857492.

33. Xie, Saining et al. (2016). “Aggregated Residual Transformations for Deep Neural Networks”. In: CoRR abs/1611.05431. arXiv: 1611.05431. URL: http://arxiv.org/abs/1611.05431.

34. Yu, Fisher et al. (2018). “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling”. In: CoRR abs/1805.04687. arXiv: 1805.04687. URL: http: //arxiv.org/abs/1805.04687.

35. Zoph, Barret et al. (2017). “Learning Transferable Architectures for Scalable Image Recognition”. In: CoRR abs/1707.07012. arXiv: 1707.07012. URL: http://arxiv. org/abs/1707.07012.