Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ШТУЧНА НЕЙРОМЕРЕЖА ГЛИБОКОГО НАВЧАННЯ ДЛЯ МОДЕЛЮВАННЯ 3D-ФОРМ ОБ'ЄКТІВ З РОЗРІДЖЕНИХ ХМАР 3D-ТОЧОК

Виконав: студент 2 курсу, групи 2КІ-22м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Підцерковний Є. О.
Керівник: к.т.н., доц.каф. ОТ
_____ Кожем'яко А. В.
«____» _____ 2023 р.
Опонент: к.т.н., доц. каф. ПЗ
_____ Рейда О.М.
«____» _____ 2023 р.

Допущено до захисту

Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О. Д.
«14» 12 2023 р

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

"26" вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Підцерковному Євгену Олександровичу

1 Тема роботи «Штучна нейромережа глибокого навчання для моделювання 3D-форм об'єктів з розріджених хмар 3D-точок» керівник роботи Кожем'яко Андрій Вікторович к.т.н., доцент, затверджено наказом вищого навчального закладу від **18.09.23** року **№ 247**.

2 Строк подання студентом роботи **12.12.23**.

3 Вихідні дані до роботи: призначення: точна та повна реконструкція 3D-об'єктів згідно даних, отриманих з камер, сенсорів, LiDAR, засоби — середовище програмування Visual Studio, PyCharm , мови програмування C++, Python.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз проблем та методів у Deep Learning, обґрунтування вирішення і представлення фігур та форм, аналіз наборів даних, проведення експериментів із даними.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): ілюстрації апроксимації функції відстані та їх шарів, приклад з синтетичного набору 2D, приклади зі згенерованих наборів даних.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Кожем'яко Андрій Вікторович к.т.н., доцент	19.09.2023р	12.12.2023
5	Небава Микола Іванович проф., к.е.н	1.11.2023	12.12.2023

7 Дата видачі завдання **19.09.2023** .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	**19.09.23**	
2	Аналіз предметної області	**20.09.23**	
3	Розробка структурної схеми	**25.09.23**	
5	Розрахунок аналогової частини	**1.10.23**	
6	Вибір ПЗ для розробки	**19.10.23**	
7	Розробка роботи нейромережі	**20.10.23**	
8	Розрахунок економічної частини	**10.11.23**	
9	Оформлення пояснювальної записки та ілюстративного матеріалу	**19.11.23**	
10	Виконання МКР	**26.11.23**	
11	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	**3.12.23**	
12	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	**11.12.23**	
13	Перевірка «антиплагіат»	**11.12.23**	
14	Попередній захист	**13.12.23**	

Студент Підцерковний Євген Олександрович

Керівник к.т.н., доц. Кожем'яко Андрій Вікторович

АНОТАЦІЯ

УДК 004

Підцерковний Є. О. Штучна нейромережа глибокого навчання для моделювання 3D-форм об'єктів з розріджених хмар 3D-точок. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2023 — 107 с. На укр. мові. Бібліогр.: 21 назв; рис.: 31; табл. 6.

У роботі розглянуто проблеми та методи у Deep Learning, обґрунтовано вирішення і представлення фігур та форм, проаналізовано наборів даних, проведено екпериментів із даними, запропоновано нові підходи до завершення форми в 3D-реконструкції.

Ключові слова: доповнення форми, 3D-реконструкція, генеративні моделі, варіаційний автокодер, амортизоване виведення, навчання без нагляду.

ABSTRACT

УДК 004

Pidtserkovnyi Ye.O. An artificial deep learning neural network for modeling 3D object shapes from sparse 3D point clouds. Master's qualification work in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — 107 p. In Ukrainian. Bibliography: 21 titles; Figures: 31; Table 6.

The paper considers the problems and methods in Deep Learning, substantiates the solution and representation of shapes and forms, analyzes data sets, conducts experiments with data, and proposes new approaches to shape completion in 3D reconstruction.

Keywords: shape completion, 3D reconstruction, generative models, variational autoencoder, amortized output, unsupervised learning.

ЗМІСТ

ВСТУП

Сприйняття форми є давньою і фундаментальною проблемою як для людського, так і для комп'ютерного зору. В обох випадках велика кількість робіт присвячена 3D-реконструкції, тобто відновленню об'єктів або сцен з одного або декількох ракурсів. Проблема реконструкції 3D-сцен або об'єктів за своєю суттю є погано поставленою оберненою задачею, оскільки багато конфігурацій форми, кольору, текстури і освітлення можуть призвести до отримання тих самих видів. У людському зорі однією з фундаментальних проблем є розуміння того, як зорова система людини виконує такі завдання, а у комп'ютерному зорі, навпаки, метою є розробка систем 3D-реконструкції. Обидві дисципліни пов'язані між собою через розуміння сигналів та обмежень, що використовуються людиною для сприйняття тривимірних форм. Результати досліджень людського зору свідчать про те, що ці передумови, а також здатність обробляти отримані сигнали є вродженими, а не набутими. У комп'ютерному зорі підказки та попередні знання зазвичай вбудовуються в конвеєри 3D-реконструкції за допомогою явних припущень. Однак останнім часом, використовуючи успіхи глибинного навчання, дослідники почали вивчати моделі форми з даних. Переважно використовують генеративні моделі, щоб навчитися генерувати, маніпулювати та аналізувати фігури, наприклад, для того, щоб зрозуміти, як їх створювати, маніпулювати та аналізувати.

У цьому контексті зосередимося на конкретній проблемі у сфері 3D-реконструкції, а саме на добудові фігури з хмари точок. Ця проблема виникає, коли надається лише один вид окремого об'єкта, а великі частини об'єкта не спостерігаються або закриті. Проблема також тісно пов'язана з реконструкцією поверхні і, таким чином, має відповідні застосування в комп'ютерній графіці. Натхненні успіхом навчання моделей фігур, маємо намір вирішити проблему завершення фігур за допомогою підходу, що базується на навчанні, де використовуємо попередні знання про фігури, отримані з великих наборів даних про фігури, таких як ModelNet або

ShapeNet [1]. Ця ідея, тобто заповнення фігур на основі навчання, нещодавно набула популярності.

Аналогічно, попередні знання форми вже застосовувалися для вирішення багатьох різних завдань комп'ютерного зору, включаючи оцінку 3D-позиції, відстеження або класичну 3D-реконструкцію. Однак, у випадку завершення форми, більшість підходів на основі навчання все ще вимагають повного контролю; це означає, що спостереження або синтезуються на основі відомих моделей, або набори даних потрібно анотувати. Доповнення фігури зазвичай передбачає розв'язання складної задачі мінімізації з використанням ітераційних підходів. Підходи на основі глибокого навчання, навпаки, можуть заповнювати фігури за допомогою одного прямого проходу навченої мережі. Було виявлено, що обидві проблеми – необхідний контроль, з одного боку, і обчислювально дорогі проблеми оптимізації, з іншого, значно обмежують застосовність цих підходів до реальних даних.

Актуальність роботи полягає у вирішенні проблеми добудови 3D-фігур з хмари точок за умови обмеженого спостереження, що є актуальним завданням у галузі комп'ютерного зору та 3D-реконструкції.

Основна мета дослідження полягає у розширенні функціональних можливостей імовірнісних підходів до завершення 3D-фігур з великою кількістю невідомих параметрів на основі попередніх знань про форму, отриманих зі схожих об'єктів у великих наборах даних, застосовуючи навчання для завершення форми за допомогою глибоких нейронних мереж.

Для досягнення поставленої мети, було сформовано такі **задачі**:

— розробити ймовірнісну структуру для завершення фігури на основі попередніх знань про форму;

— вивчити, як попередні знання можуть полегшити навчання завершення фігури при слабкому контролі;

— розвинути моделі латентного простору, щоб врахувати спостереження і отримати нижню межу достовірності;

— оцінити підходи на синтетичних і реальних даних, і продемонструвати їхню застосовність і конкурентоспроможність;

— розвивати комп'ютерний 3D-зір шляхом використання попередніх даних та зменшення обчислювальної складності.

Об'єктом дослідження є процеси завершення 3D-фігур, які мають багато невідомих параметрів, а також процеси навчання мережі на основі попередніх знань про форму, які отримані з великих наборів даних..

Предметом дослідження є методи застосування імовірнісних фреймворків та глибоких генеративних моделей, таких як варіаційні автокодери, для вивчення попередніх форм для завершення фігури та використання слабкого контролю та сильних попередніх знань для ефективного завершення фігур та вивчення потенціалу навчання без контролю в цьому контексті.

Методи дослідження: методи системного та статистичного аналізу, методи схемотехнічного та алгоритмічного проектування.

Наукова новизна полягає у вдосконаленні методу формування 3D-фігур, які мають багато невідомих параметрів за рахунок введення амортизованого виводу, який дозволяє передбачати рішення з максимальною ймовірністю безпосередньо за відповідними спостереженнями, замість максимізації ймовірності незалежно для окремих спостережень.

Практичне значення отриманих результатів полягає в тому, що застосування запропонованого методу може забезпечити ефективне та економне використання даних у реальних додатках, таких як розпізнавання об'єктів, робототехніка та автономне водіння, завдяки зменшенню потреби у великих анотованих навчальних даних.

Апробація результатів роботи здійснена у доповіді на LIII Науково-технічній конференції підрозділів Вінницького національного технічного університету (2024) [2].

1 ОБҐРУНТУВАННЯ ПРОБЛЕМИ, МЕТОДІВ ТА ТЕХНІК У ГЛИБОКОМУ НАВЧАННІ

1.1 Аналіз проблеми та запропонований підхід

У сучасному глибокому навчанні існують виклики, пов'язані з нестабільністю моделей, особливо при обмеженій кількості даних. Велика складність архітектур та велика кількість параметрів можуть вести до перенавчання та обмеженої універсальності. Поєднання цього з проблемами пояснюваності та прозорості робить важкою впровадження глибокого навчання в критичних сферах, де потрібно розуміти логіку прийнятих рішень. Додатково, велика невизначеність в інтерпретації результатів та високі вимоги до обчислювальних ресурсів ускладнюють широке використання глибокого навчання в реальних умовах. Розв'язання цих викликів потребує спільних зусиль фахівців з різних галузей для розробки більш стабільних, ефективних та етичних моделей глибокого навчання.

Ми визначаємо завершення форми як реконструкцію поверхні окремого об'єкта з відомої категорії об'єктів на основі часткового спостереження за його формою — зазвичай лише з одного ракурсу. Як зазначено у вступі, дотримуємося підходу, що базується на навчанні. У найпростішій формі, тобто у випадку спостереження, задачу можна сформулювати наступним чином:

Проблема 1.1 — За заданими (частковими) спостереженнями $X = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^R$ та форми $Y^* = \{y^*_1, \ldots, y^*_N\} \subseteq \mathbb{R}^R$ певної категорії об'єктів так, що y^*_N представляє істинну форму x_N, задача полягає в тому, щоб навчитись будувати відображення $x_N \mapsto y^*_N$, яке яке здатне узагальнювати на раніше невидимі спостереження.

Тут припускаємо, що спостереження x_N та форми y^*_N є векторами у деякому простору $\mathbb{R}^R$ для стислості викладу. На практиці будемо вдаватися до до сіток зайнятості або знакових функцій відстані у трьох просторових вимірах, наприклад, $x_N \mapsto y^*_N \in \mathbb{R}^{H*W*D} \simeq \mathbb{R}^R$. Спостереження x_N відповідають спостережуваним точкам — зайнятим або незайнятим і неспостережуваним

точкам. Для того, щоб підкреслити, що це лише часткові спостереження, використовуємо $x_N \in \{0, 1, \perp\}^R$, де $\perp$ відповідає неспостережуваним точкам.

Отримання істинних форм Y^* для реальних спостережень X є трудомістким. Таким чином, версію проблеми 1.1 без спостережень можна сформулювати наступним чином:

Проблема 1.2 — За заданими (частковими) спостереженнями $X = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^R$ та форми $Y^* = \{y^*_1, \ldots, y^*_N\} \subseteq \mathbb{R}^R$ певної категорії об'єктів задача полягає в тому, щоб дізнатися відображення $x_N \mapsto y(x_N) \in \mathbb{R}^R$, де $y(x_N)$ представляє фігуру, яка якомога ближче відповідає невідомій базовій істинній фігурі $y^*_N \in \mathbb{R}^R$.

Проблему 1.2 також можна інтерпретувати як задачу зі слабким контролем, оскільки припускаємо, що знання про категорію об'єктів. Однак ці знання поки що не використовуються явно. Крім того, і в проблемі 1.1, і в проблемі 1.2 передбачається, що кожному спостереженню x_N відповідає один окремий об'єкт. Це означає, що припускаємо, що об'єктний детектор для того, щоб виділити спостереження x_N з повної хмари точок як це передбачено в реальних програмах, наприклад, на KITTI.

Рисунок 1.1 — Ілюстрація проблеми 1.3

Проблема 1.2 за своєю суттю є неоднозначною; на практиці нескінченно багато форм можуть відповідати даному спостереженню, і навіть форма істини не обов'язково повинна ідеально відповідати спостереженню через шум. пропонуємо байєсівський підхід для належного моделювання цієї невизначеності. Оскільки припускаємо, що знаємо категорію об'єкта, збираємо набір $Y = \{y_1, \ldots, y_M\} \subseteq \mathbb{R}^R$ репрезентативних фігур, що відповідають цій категорії об'єктів. На практиці це може відповідати набору моделей

автоматизованого проектування (САПР), тобто трикутних сіток [3]. Проблема 1.2 переформульована нижче:

Задача 1.3 — дано (часткові) спостереження $X = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^R$ та форми $Y = \{y_1, \ldots, y_M\} \subseteq \mathbb{R}^R$ певної категорії об'єктів, задача полягає в тому, щоб навчитись будувати відображення $x_N \mapsto y(x_N) \in \mathbb{R}^R$ таким чином, щоб $y(x_N)$ якомога ближче відповідала невідомій базовій істинній формі $y^*_N \in \mathbb{R}^R$.

Ми пропонуємо використовувати цю множину Y для отримання апріорного знання про можливі форми, наприклад, за допомогою варіаційного автокодера. Заповнення форми може бути сформульовано як задачу максимальної правдоподібності в латентному просторі нижчої розмірності $Z = \mathbb{R}^Q$, $Q \ll R$ [4]. З цією метою інтерпретуємо окремі вокселі y_i, для $y \in \mathbb{R}^R$, як випадкові величини, а x_i, для $x \in \mathbb{R}^R$, як відповідні спостереження. Потім, Y використовується для вивчення попередньої моделі $p(y, z)$, яка розкладається на генеративну модель та попередню: $p(y, z) = p(y|z)p(z)$. Використовуючи набір спостережень X, навчаємо модель виводу, яка безпосередньо вивчає відображення:

$$x \mapsto \tilde{z} = \operatorname{argmax} p(y = x|z)p(z) \tag{1.1}$$

тим самим неявно розв'язуючи проблему 2.3 інтерпретуємо цей підхід як аморалізовану максимальну правдоподібність відповідно до ідеї. Зокрема, не розглядаємо проблему максимальної правдоподібності для кожного спостереження x незалежно. Натомість, розуміємо проблему максимальної правдоподібності як втрати між спостереженнями та формами, що дозволяють вивчати вкладання $x \mapsto \acute{z}$ неконтрольованим чином. У контексті варіаційних автокодерів це зводиться до навчання нового кодера для представлення вбудовування спостережень у латентний простір [5].

Крім того, можемо вважати спостереження x_i також випадковими величинами. Для спільної ймовірності моделі, тобто $p(x, y, z)$, можемо згодом отримати нижню межу достовірності. Використовуючи спрощене припущення, що y та z є статистично незалежними від x, можемо використати

розширений варіаційний автокодер для неявного навчання наближеної моделі q(z|x), яка, знову ж таки, неявно розв'язує проблему 1.3. Тут замість прямої оптимізації максимальної правдоподібності в рівнянні (1.1) (з використанням p(y = x|z), спостереження прив'язуються до відповідних фігур за допомогою дивергенції Кульбака-Лейблера. Навчену модель q(z|x) можна інтерпретувати як імовірнісне вбудовування спостережень у простір латентних форм, за аналогією до відображення x ↦ ź, що вивчається при амортизації максимальної правдоподібності. Загалом, обидва підходи вимагають попереднього визначення латентного простору форм і подальшого навчання для вбудовування часткових спостережень у цей простір.

1.1.1 Заповнення форм

Заповнення форми тісно пов'язане з 3D-реконструкцією, а також з відновленням поверхні. Однак, проблеми, що розглядаються, зазвичай не зовсім збігаються з проблемою заповнення форми.

На відміну від реконструкції поверхні та 3D-реконструкції, доповнення форми зазвичай виконується на часткових сканах окремих об'єктів. Підходи до заповнення форми можна умовно розділити на методи, що базуються на даних, і методи, що базуються на симетрії. Останні виявляють симетрію в незамкнених частинах для того, щоб завершити фігури шляхом завершення відповідної симетрії в замкнутих частинах. Ці підходи також знайшли застосування в робототехніці, наприклад, для хапання роботів. Випадок, керований даними, є більш цікавим по відношенню до запропонованих підходів; розглянемо завершення фігури як задачу пошуку та вирівнювання. Подальша робота в цьому напрямку включає, наприклад, випадки, коли локальні особливості форми, знання про об'єкт або попередні форми знайшли застосування. Припасування моделі до десятка відбувається за схемою, подібною до ітераційного алгоритму найближчої точки (ICP), або ставиться як задача мінімізації енергії [6].

Нещодавно було запропоновано декілька підходів на основі навчання, більшість з яких використовують глибоке навчання. Заповнення фігури

ставиться як задача керованого навчання, наприклад, на основі поодиноких RGB-D зображень, часткових спостережень або зашумлених фігур. У порівнянні з підходами, що базуються на даних, завершення розпізнавання фігури зазвичай передбачає один прямий прохід навченої мережі. Однак ці підходи можна застосовувати лише на синтетичних наборах даних, таких як ModelNet або ShapeNet, оскільки потрібен явний контроль. Також маємо намір використовувати глибинні нейронні мережі, однак хочемо навчитися завершувати фігури в умовах слабкого контролю, використовуючи лише дані спостережень і знання про категорію об'єкта. Це дозволить нам вивчати завершення форми на реальних даних, наприклад, на KITTI, забезпечуючи при цьому ефективний висновок через навчену мережу.

Як зазначено вище, попередні знання відіграють важливу роль у заповненні форми. Наприклад, підходи, що базуються на навчанні, можна розуміти як неявне навчання попередньої форми на основі контрольованих навчальних даних. Методи, що базуються на даних, також можуть використовувати попередні знання про форму, наприклад, використовуючи аналіз головних компонент (PCA) або моделі латентних змінних гауссівського процесу (GP-LVM) [7]. Однак, попередні знання форми також застосовуються в різних областях, наприклад, в оцінці 3D-позиції, що часто вирішується в поєднанні з такими завданнями, як сегментація зображень або потік сцени. У класичній 3D-реконструкції, попередні форми також часто використовуються для усунення неоднозначностей, дзеркальних відображень або відсутності текстури. У більшості випадків, попередні форми "вивчаються" на основі повних форм; це на відміну від попередніх, не пов'язаних з даними, таких як локальні або глобальні попередні гладкості, припущення про площинність, тощо. Будемо здебільшого дотримуватись ідеї використання пріоритету форми явно для визначення та обмеження простору можливих форм.

1.2 Глибоке навчання 3D

У комп'ютерному зорі під глибоким навчанням часто розуміють практику використання глибоких згорткових нейронних мереж для навчання

дискримінативних, а віднедавна і генеративних моделей. Хоча нейронні мережі використовуються і досліджуються вже кілька десятиліть, вони лише нещодавно привернули до себе значну увагу. Серед кількох важливих віх — значне підвищення продуктивності на задачі класифікації ImageNet у 2012 році [8]. Після цього глибокі згорткові нейронні мережі успішно застосовуються для вирішення широкого спектру завдань комп'ютерного зору.

Останнім часом значну увагу привертають глибокі генеративні моделі, наприклад, і пов'язані з ними роботи. Такі моделі дозволяють генерувати реалістичні зображення або фігури, а також вивчати сильні попередні моделі, які є цікавими для напів-, слабо- або некерованих задач. У цій тезі будемо використовувати варіаційні автокодери і подальшу роботу, оскільки вони включають в себе як висновок, так і генеративну модель. Це також мотивовано тим, що генеративні змагальні мережі, як відомо, важко піддаються навчанню — проблема, яка, як очікували, буде посилюватися на 3D-даних. Генеративні моделі також були застосовані для моделювання фігур, що дозволяє інтерполяцію, маніпуляцію та генерацію фігур.

Більшість робіт наївно узагальнюють згорткові нейронні мережі на 3D-дані, наприклад, у вигляді вокселізованих фігур. На жаль, це суттєво обмежує роздільну здатність, що використовується; зазвичай використовується роздільна здатність від 32^3 до 64^3. Деякі підходи, однак, використовують розрідженість 3D-даних за допомогою вісімок, щоб зменшити споживання пам'яті та часу і забезпечити вищу роздільну здатність, наприклад, до 256^3. Крім того, були визначені розріджені схеми згортки для прискорення навчання. Існує також напрямок роботи, що застосовує згорткові нейронні мережі безпосередньо до сіток або хмар точок. Хоча також використовуємо прості 3D-згорткові нейронні мережі для представлених експериментів, які також проводимо з роздільною здатністю 32^3, наведені вище посилання надають цікаві можливості для подальшої роботи [9].

Глибоке навчання зазвичай описує використання глибоких нейронних мереж, де глибина може стосуватися як кількості параметрів моделі, так і кількості використовуваних шарів, тобто кроків обробки. У контексті цієї

дисертації нейронні мережі використовуються в рамках варіаційних автокодерів для вивчення попередньої форми, тобто вбудовування форм у латентний простір. Зокрема, як генеративна модель p(y|z), так і наближена модель розпізнавання q(z|y) реалізуються за допомогою нейронних мереж. Це досягається шляхом вибору відповідних параметризацій та прогнозування відповідних параметрів —наприклад, середнього значення та дисперсії для гауссівських розподілів. У запропонованих підходах нейронні мережі також використовуватимуться для вивчення вбудовування спостережень x у простір латентної форми. Для амортизованої максимальної правдоподібності це буде детерміноване вбудовування $x \mapsto z(x; w)$ — тут зробили параметри w нейронної мережі явними. Для запропонованого розширеного варіаційного автокодера вбудовування представлено наближеною моделлю розпізнавання q(z|x). Взагалі, нейронні мережі можуть бути представлені з різних точок зору [10].

У цьому випадку нейронні мережі розглядаються як послідовність (або ациклічний граф, у загальному випадку) тензорних операцій, які використовують набір параметрів w для того, щоб на основі вхідних даних x обчислити функцію y(x; w), яка може наближати деяку цільову функцію. Спочатку введемо поняття тензорів — структури даних, що лежить в основі всіх сучасних фреймворків глибокого навчання, а потім поступово визначимо більш складні мережі і, нарешті, обговоримо навчання мереж.

1.3 Тензори. Основні операції

У нашому контексті під дійсними тензорами зазвичай розуміють багатовимірні масиви дійсних чисел:

Визначення 1.1 — (дійсний) тензор — це елемент $t \in \mathbb{R}^{n_1 * \cdots * n_r}$, де $(n_1, \ldots, n_r) \in \mathbb{N}^r$ і r називається рангом тензора. Для індексації тензора записують $r_i := r_{(i_1,\ldots,i_r)}$; у цьому випадку $i = (i_1, \ldots, i_r)$ називається мультиіндексним [11].

На практиці ранг часто називають розмірністю. Хоча це математично неточно, будемо використовувати обидва терміни як взаємозамінні. Отже,

тензор $t \in \mathbb{R}^{n_1 * \cdots * n_r}$ має розмірність (ранг) r, а n_i називається розміром розміру i. також допускаємо так звані синглетні розміри, тобто розміри із значенням 1. Тоді термінологія може бути застосована один до одного до багатьох популярних фреймворків глибокого навчання, таких як Torch, PyTorch, Theano, Tensorflow, Caffe тощо.

Приклад 1.1 — приклади включають тривіальні випадки, такі як скаляри, тобто тензори 0-го рангу, вектори, тобто тензори 1-го рангу, та матриці, тобто тензори 2-го рангу. Приклади, специфічні для нашої задачі, включають: багатоканальне зображення, багатоканальні об’єми та пакет багатоканальних томів.

Багатоканальне зображення — це тензор рангу 3: $t \in \mathbb{R}^{C \times H \times W}$, де C — кількість каналів, H — висота зображення і W — ширина зображення. Для відтінків сірого (тобто одноканальних) зображень C = 1, тоді як кольорові зображення використовують C = 3 канали, які зазвичай відповідають червоному, зеленому та синьому. Відтінки сірого, як і колір, зазвичай кодуються по 8 біт. Багатоканальні об'єми — тензор $t \in \mathbb{R}^{C \times H \times W \times D}$ рангу 4 можна інтерпретувати як C-канальний об'єм, де H, W і D позначають висоту, ширину і глибину відповідно. Об'єми широко використовуються у медичній візуалізації (наприклад, об'єми МРТ або КТ). У нашому випадку використовуємо об'єми для представлення фігур за допомогою сіток заповнення або (знакових) функцій відстані. Пакет багатоканальних томів, тобто набір, багатоканальних томів можна інтерпретувати як тензор рангу 5:

$$t \in \mathbb{R}^{B \times C \times H \times W \times D},$$

де B — це розмір набору багатоканальних об'ємів.
При навчанні нейронних мереж B буде розміром партії.

У цій роботі переважно будемо працювати з томами та багатоканальними томами, а під час навчання — з їхніми партіями. Тому завжди розглядаємо випадок 5-го рангу і зазвичай позначаємо розмірність як $B \times C \times H \times W \times D$ [12].

Для тензорів рангу 1 або 2 можна застосовувати всі операції, відомі з лінійної алгебри (наприклад, множення матриця-вектор або матриця-матриця). Крім того, визначаємо поелементні операції, які зазвичай використовуються у глибокому навчанні:

Визначення 1.2 — нехай t, s — два тензори; нехай $i \in \mathbb{N}^5$ — мультиіндекс. Далі, x — нехай довільна бінарна арифметична операція над $\mathbb{R}$. Тоді відповідна поелементна операція визначається через $(t \times s)_i = t_i \times s_i$.

Визначення 1.3 — нехай t — тензор, а $i \in \mathbb{N}^5$ — мультиіндекс; далі, нехай $h : \mathbb{R} \mapsto \mathbb{R}$ — дійсна функція. Тоді h може діяти на t поелементно, тобто $h(t)_i = h(t_i)$.

1.4 Шаруваті нейронні мережі

Далі вводимо шаруваті нейронні мережі – моделі, що складаються з декількох шарів, тобто послідовності, тензорних операцій.

Визначення 1.4 — нейронна мережа (прямого поширення) — це спрямований ациклічний граф $G = (V, E)$, де кожна вершина $v \in V$ відповідає операції, що відображає один або декілька вхідних тензорів в один або декілька вихідних тензорів. Ребра v_i, v_j характеризують інформаційний потік, тобто один або декілька вихідних тензорів операції v_i подаються на вхід операції v_j. Операції v_i також називаються шарами і можуть мати скінченну кількість регульованих ваг [13].

Враховуючи топологічне впорядкування вершин $V = (v_i, \dots, vi_{|v|})$, мережа обчислює свій вихід шляхом послідовного обчислення виходів шарів за входом x, тобто $\nu_{i_1}(x), \nu_{i_2}(\nu_{i_1}(x), x), \nu_{i_3}\left(\nu_{i_2}\left(\nu_{i_1}(x)\right), (\nu_{i_1}(x), x)\right), \dots$ Раннім прикладом одношарової нейронної мережі є персептрон.

Приклад 1.2 — персептрон є лінійним класифікатором: $y(x; w) = xw^T$, де $x \in \mathbb{R}^{1\times C}$ — вхідний тензор, а $w \in \mathbb{R}^{1\times C}$ — ваговий вектор. Класифікація виконується шляхом розгляду ознак signy(x; w) з мітками класів у $\{-1, 1\}$. Хоча внутрішній добуток xw^T може бути неінтуїтивно зрозумілим, він

дозволяє легко застосувати функцію рішення до пачки вибірок $x_1, \ldots, x_B$, складених у $x \in \mathbb{R}^{B \times X}$.

Сьогодні персептрон є більш важливим, ніж будь-коли, оскільки він описує основні будівельні блоки нейронних мереж, повністю пов'язаний шар:

Визначення 1.5 — повністю зв'язний шар, позначений $\mathrm{fc}_{C_{in},C_{out}}$, отримує на вхід тензор $x \in \mathbb{R}^{B \times C_{out}}$ і обчислює:

$$\mathrm{fc}_{C_{in},C_{out}}(x) = xw^T \in \mathbb{R}^{B \times C_{out}},$$

де $w \in \mathbb{R}^{C_{out} \times C_{in}}$ — відповідна вагова матриця. [14]

Зазвичай, повністю зв'язаний шар комбінується з адитивним членом зсуву $y(x; w) = xw^T + \begin{bmatrix} b \\ \vdots \end{bmatrix}$ з $x \in \mathbb{R}^{B \times C_{in}}, b \in \mathbb{R}^{1 \times C_{out}}$ (який повторюється B разів, щоб можна було працювати з партіями) і, як і раніше, $w \in \mathbb{R}^{C_{out} \times C_{in}}$. Щоб зробити цей адитивний член явним, використовуємо окремий шар. Шар зсуву можна легко узагальнити на тензори вищого рангу, і зазвичай він супроводжується нелінійністю:

Визначення 1.6 — нехай $h : \mathbb{R} \mapsto \mathbb{R}$ — дійсна функція. Тоді h також позначає шар, який приймає на вхід довільний тензор x і обчислює h(x) поелементно.

Ці нелінійності також називаються активаційними або передавальними функціями і мотивуються бажанням апроксимувати складні нелінійні функції за допомогою комбінації простіших нелінійних функцій [15].

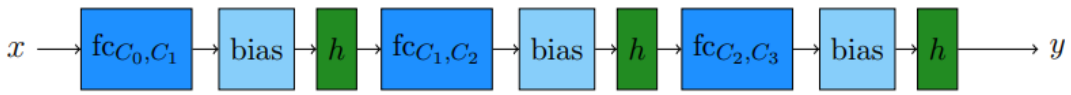

Рисунок 1.2 — Ілюстрація багатошарового персептрона з L = 3 повністю з'єднаними шарами, за якими слідують шари зміщення та нелінійності.

Приклад 1.3 — поширені нелінійності для нейронних мереж включають логістичний сигмоїд $\sigma(x) = \frac{1}{1+\exp(-x)}$ та випрямлену лінійну одиницю

$\mathrm{ReLU}(x) = \max(x, 0) = [x]_t$. Логістичний сигмоїд дозволяє моделювати ймовірності, а нелінійності ReLU, як було показано, скорочують час навчання і покращують продуктивність, будучи хорошою моделлю біологічних нейронів.

Нарешті, обговоримо нашу першу шарувату нейронну мережу, мережевий графік якої показано на рисунку 1.2: багатошаровий персептрон, який складається з послідовності повністю з'єднаних шарів, шарів зміщення та нелінійності. Назва походить від того, що його можна інтерпретувати як багатошарове розширення персептрона:

Приклад 1.4 — багатошаровий персептрон складається з декількох етапів обчислень, як показано на рисунку 1.2. Нехай $L > 0$ — кількість повністю з'єднаних шарів; $x \in \mathbb{R}^{1\times C_0}$ — вхідні дані. На першому етапі відбувається обчислення — $y^{(1)} = h\left(x\left(\omega^{(1)}\right)^T + b^{(1)}\right)$ з $\omega^{(1)} \in \mathbb{R}^{C_1\times C_0}$ та $b^{(1)} \in \mathbb{R}^{1\times C_1}$. Як показано на рисунку 4.1, це вже включає перші повністю з'єднані шари, шари зсуву та нелінійності. Далі, для $L > 1$ визначимо, що $y^{(l)} = h(y^{(l-1)}\left(\omega^{(l)}\right)^T + b^{(l)})$ з $\omega^{(l)} \in \mathbb{R}^{C_1\times C_{l-1}}$ та $b^{(l)} \in \mathbb{R}^{1\times C_l}$. Нарешті, $y^{(l)}$ — це вихід мережі. Знову ж таки, це формулювання поширюється на випадок множини $x_1, \ldots, x_B$, складених в один тензор $x \in \mathbb{R}^{B\times C}$.

1.5 Згорткові нейронні мережі

Згорткові нейронні мережі були введені і замінили повністю зв'язані шари на дискретні згортки. Перевага цього методу полягає в тому, що просторова інформація в зображеннях може бути використана явно, одночасно зменшуючи кількість параметрів. Крім того, дискретна згортка за своєю суттю є інваріантною до трансляцій, може бути ефективно реалізована і залишається лінійною операцією як по відношенню до вхідних даних, так і по відношенню до параметрів. Загалом, згорткові нейронні мережі відіграли важливу роль у нещодавньому успіху глибокого навчання в комп'ютерному зорі [16].

Дискретна згортка є добре відомою технікою в обробці сигналів. Наступний приклад представляє загальну концепцію.

Приклад 1.5 — для простоти, нехай $x \in \mathbb{R}^C$ — одновимірний вхідний сигнал. Далі, нехай $w \in \mathbb{R}^{2K+1}$ — так зване ядро. Для зручності позначимо w через w_i, $-K \leq i \leq K$. Тоді дискретна згортка визначається як:

$$(x^* w)_i := \sum_{j=-K}^{K} x_{i-j} w_j, \tag{1.2}$$

де розмір результату буде залежати від того, як визначено операцію згортки на границях сигналу.

Зокрема, для $i < K$ або $i > n - K$ рівняння 1.2 не є чітко визначеним. Зазвичай, припускаємо, що x має такий вигляд, що $x_i = 0$ для всіх $i \in \{1, \dots, n\}$. Зауважимо, що легко показати, що дискретна згортка є лінійною як за x, так і за w.

Визначення 1.7 — нехай $x \in \mathbb{R}^{n_1 \times \dots \times n_r}$ — тензор рангу r, а $w \in \mathbb{R}^{2K_1+1 \times \dots \times 2K_r+1}$ — ядро рангу r. Тоді згортка рангу r визначається:

$$(x^* w)_i := \sum_{j_1=-K_1}^{K_1} \cdots \sum_{j_r=-K_r}^{K_r} t_{i-j} s_j,$$

де w індексується за допомогою $-K_1 \leq j_1 \leq K_1, \dots, -K_r \leq j_r \leq K_r$ для простоти позначення та мультиіндексів i та j можна додавати та віднімати поелементно.

Наведене вище визначення є специфічним для нашого випадку використання, тобто для згорткових нейронних мереж, де припускаємо непарні розміри ядра для простоти. Як і в одновимірному випадку, вхідний сигнал вважається доповненим нулями; це зазвичай робиться при використанні згорток у нейронних мережах. На основі згортки, як описано вище, згортковий шар можна представити наступним чином:

Визначення 1.8 — згортковий шар $\mathrm{conv}_{C_{in}, C_{out}, K}$ отримує на вхід тензор $x \in \mathbb{R}^{B \times C_{in} \times H \times W \times D}$ і обчислює:

$$(\mathrm{conv}_{C_{in}, C_{out}, K}(x))_{b, C_{out}} = \sum_{C_{in}=1}^{C_{in}} w_{C_{out}, C_{in}} * t_{b, C_{in}},$$

де $W \in \mathbb{R}^{C_{out} \times C_{in} \times K \times K \times K}$ — відповідний ваговий тензор, де K непарний [17].

Іншими словами, згортковий шар отримує на вхід тензор, що складається з каналів C_{in}, згортає кожен з цих каналів з відповідним ядром $w_{C_{out}, C_{in}} \in \mathbb{R}^{K \times K \times K}$ і підсумовує результати. Це робиться C_{out} разів, таким чином створюючи C_{out} нових каналів. Згортковий шар має багато різних варіантів, наприклад, з використанням (дробових) кроків, як деконволюція або як розширена згортка, і це лише деякі з них; ми, однак, будемо використовувати представлений простий варіант.

Для підвибірки вхідного зображення було застосовано згортку з проміжками; як альтернатива, максимальне об'єднання обчислює максимальне значення в межах вікон, що не перекриваються. Сьогодні максимальне об'єднання популярне для забезпечення стійкості до шуму та малих перетворень:

Визначення 1.9 — максимальний шар об'єднання pool_K приймає на вхід тензор $x \in \mathbb{R}^{B \times C \times H \times W \times D}$ і обчислює тензор розміром $B \times C \times \frac{H}{K} \times \frac{W}{K} \times \frac{D}{K}$ як $(\text{pool}_K)_{b,c,i} = \max_{K(i_1 - 1) < j_1 \leq Ki_1, \ldots, K(i_3 - 1) < j_3 \leq Ki_3} x_{b,c,j}$, де $i = (i_1, i_2, i_3)$ і $j = (j_1, j_2, j_3)$ — мультиіндекси, і припускаємо, що H, W та D діляться на K. [18]

Об'єднання по суті обчислює максимальне значення над 3-вимірними кубами, що не перетинаються, з довжиною ребра K.

Замість того, щоб зменшувати розмір тензора, цікаво також просторово збільшити розмір тензора. Для цього зазвичай використовують методи білінійної інтерполяції. Інший варіант — деконволюція (або згортка з дробовим кроком) [19]. Крім того, максимальне об'єднання може бути "інвертоване. Натомість використовуємо просту вибірку найближчих сусідів, де кожен елемент дублюється у вікнах фіксованого розміру для збільшення просторового розміру:

Визначення 1.10 — шар дискретизації найближчого сусіда nnup_K отримує на вхід тензор $x \in \mathbb{R}^{B \times C \times H \times W \times D}$ і обчислює тензор розміру $B \times C \times$

$\frac{H}{K} \times \frac{W}{K} \times \frac{D}{K}$ як $(\mathrm{nnup}(x)_K)_{b,c,i} = x_{b,c,\left[\frac{i_1}{K}\right],\left[\frac{i_2}{K}\right],\left[\frac{i_3}{K}\right]}$ для $1 \le i_1 \le KH, \ldots, 1 \le i_3 \le KD$ [20].

1.6 Автокодери

Як приклади обрали популярний клас (згорткових) нейронних мереж — автокодери. Ці моделі набули більшої уваги завдяки використанню для попереднього навчання глибоких нейронних мереж:

Приклад 1.6 — основна мета автокодера — відновити вхідні дані за допомогою низьковимірного латентного представлення. Автокодери можна розділити на кодер, який обчислює відображення $z(x; w)$ від входу x до низьковимірного коду z, і декодер, який обчислює реконструкцію $\tilde{x}(z; w) \approx x$ з латентного коду z. Кодер і декодер часто "дзеркально відображають" один одного, тобто обидва можуть бути реалізовані за допомогою одного і того ж багатошарового персептрона, як показано на рисунку 1.3.

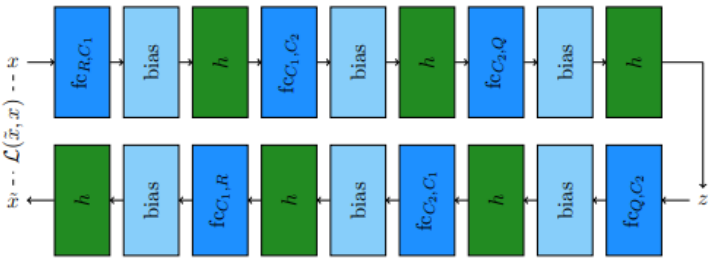

Рисунок 1.3 — Простий варіант багатошарового автокодера на основі перцептронів

Тут обидва складаються з трьох повністю з'єднаних шарів, за кожним з яких слідує шар зміщення і нелінійність. Вхідні дані $x \in \mathbb{R}^R$ перетворюються в латентний код z розмірності $Q < R$, який використовується для оцінки реконструкції $\tilde{x} \approx x$. Точні розмірності C_1, C_2 і Q, а також використовувана нелінійність h є гіперпараметрами, які можуть бути адаптовані в залежності від застосування [21]. При переведенні автокодерів у згортковий випадок зазвичай застосовуємо кілька етапів згорткових шарів (часто з наступними нелінійностями та об'єднанням). Однак для обчислення одновимірного

прихованого коду z вихідний тензор з останнього шару об'єднання потрібно переформувати.

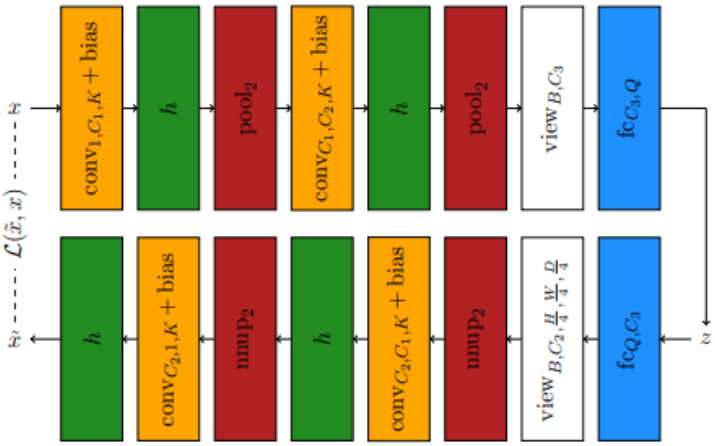

Рисунок 1.4 — Ілюстрація згорткового автокодера, що складається з кодера (вгорі) і декодера (внизу).

Визначення 1.11 — шар перегляду $\text{view}_{B',C',H'.W'.D'}$ розглядає заданий вхідний тензор розміру $B \times C \times H \times W \times D$, як тензор розміру $B' \times C' \times H' \times W' \times D'$ за умови, що $BCHWD = B'C'H'W'D'$.

На практиці тензор завжди реалізується за допомогою одновимірного масиву таким чином, щоб він представляв собою безперервний набір блоків пам'яті, наприклад, $t_{i_1,i_2,i_3,i_4i_5} := \hat{t}_{\left(((i_1C+i_2)H+i_3)W+i_4\right)D+i_5}$ для $t \in \mathbb{R}^{B\times C\times H\times W\times D}$ та $\hat{t} \in \mathbb{R}^R$ з $R = BCHWD$ — базовий одновимірний тензор вимірний тензор. Таким чином, перетворення може не передбачати фактичного переміщення даних, а лише представляти інший "view" на дані.

Приклад 1.7 — приклад загального автокодера можна поширити на випадок згортки. Замість послідовності повністю з'єднаних шарів, за якими йдуть шари зсуву та нелінійності, кодер складається з послідовності згорток, за якими йдуть шари зсуву, шари нелінійності та шари максимального об'єднання. Декодер, знову ж таки, є дзеркальним відображенням кодера, замінюючи шари максимального об'єднання на шари підвищеної вибірки. Прихований код обчислюється за допомогою повністю з'єднаного шару. Повна модель проілюстрована на рисунку 1.4.

В обох прикладах відображення $x \mapsto z(x; w)$ та $z \mapsto \tilde{x}(z; w) \approx x$ є детермінованими. Пізніше побачимо, що автокодери також можна переформулювати у ймовірнісний, недетермінований спосіб, де замість прямих відображень моделюються відповідні розподіли ймовірностей $q(z|x)$ та $p(x|z)$. Потім будемо використовувати таку модель для вивчення нашої попередньої форми.

На жаль, обсяг цієї тези не дозволяє широко обговорити останні досягнення в галузі глибокого навчання. Більшість з представлених підходів – за винятком, наприклад, пакетної нормалізації, введеної схеми ініціалізації ваг або деконволюції/дискретизації – використовуються вже кілька десятиліть, принаймні, у схожих формах. Однак багато цікавих ідей було запропоновано лише нещодавно, і, як наслідок, глибоке навчання знайшло широке застосування в комп'ютерному зорі.

Хоча представили лише базові концепції згорткових нейронних мереж, вони виявилися достатніми для вирішення складних 3D-завдань, включаючи завершення форми та генеративне моделювання форм. Крім того, обчислювальні ресурси все ще є обмежуючим фактором для згорткових нейронних мереж у 3D; більш складні мережі дуже важко навчати, особливо у високій роздільній здатності. Тому в деяких роботах також були введені розріджені варіанти обговорюваних концепцій. Представлені шари, завдяки своїй простоті, також доступні в більшості фреймворків глибокого навчання, що дозволяє легко відтворювати та оцінювати представлені приклади. Нарешті, згорточні автокодери, як показано в прикладі 1.7, можуть бути легко "модернізовані" до потужних генеративних моделей, таких як варіаційні автокодери.

2 АНАЛІЗ ПРЕДСТАВЛЕННЯ ФІГУР ТА ПОПЕРЕДНЄ ПРЕДСТАВЛЕННЯ ФОРМИ

2.1 Хмари точок та сітки

Повертаючись до початкової проблеми цієї дисертації, метою є завершення фігур. Оскільки хочемо вивчити як попередню форму, так і запропоновані підходи до завершення форми за допомогою 3D згорткових нейронних мереж, нам потрібні сіткові представлення спостережень, тобто хмари точок з KITTI і форм, тобто сітки з ShapeNet. Як зазначалося раніше, в основному покладаємося на сітки зайнятості для представлення спостережень, а для представлення фігур — як на сітки зайнятості, так і на знакові функції відстаней. Далі коротко і детально формалізуємо обидві модальності.

Хмари точок — це невпорядковані набори 3D точок. У нашій постановці задачі 1.3 хмари точок відповідають необробленій версії спостережень X:

Визначення 2.1 — хмара точок $P = \{p_1, \dots, p_N\} \subseteq \mathbb{R}^3$ — це множина точок з розмірами, що відповідають ширині (по горизонталі), висоті (по вертикалі) та глибині у такому порядку.

Зауважимо, що розмірності не узгоджуються з введеними розмірностями тензорів, тобто $H \times W \times D$. Хоча це може здатися незрозумілим, ми вирішили слідувати загальноприйнятій практиці в комп'ютерному зорі та глибокому навчанні, де перший вимір тензорів відноситься до висоти зображення або об'єму. Це означає, що осі висоти та ширини потрібно поміняти місцями при вокселізації хмар точок у сітки заповнення, які розглянемо пізніше.

Іншим джерелом інформації для нашої задачі є набори даних моделей автоматизованого проектування (САПР). У цій роботі ми припускаємо, що САПР-моделі надаються у вигляді трикутних сіток — за детальним описом ми звертаємося до. Щодо задачі 1.3, то вони відповідають необробленій версії набору фігур Y:

Визначення 2.2 — трикутна сітка $M = (V, F)$ визначається множиною вершин $V \subseteq \mathbb{R}^3$ та множиною трикутних граней $F \subseteq \{1, \dots, |V|\}^3$, де $f =$

(f_1, f_2, f_3) визначає трикутну грань, обмежену відповідними вершинами v_{f_1}, v_{f_2} та v_{f_2}. Грані неявно також визначають ребра E(F) між вершинами.

Ілюстрацію трикутної сітки з ShapeNet можна знайти на рисунку 2.1.

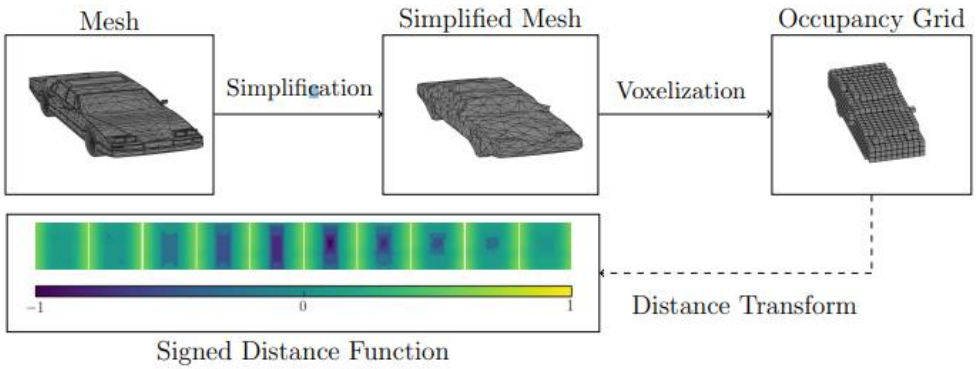

Рисунок 2.1 — Ілюстрація використаних модальностей, тобто сітки зайнятості та підписаних функцій відстані

Поняття суміжності та інцидентності природно поширюються на трикутні сіті. Зауважимо, що трикутна сіть визначає лише поверхню об'єкта. Без додаткових обмежень, як правило, важко міркувати про внутрішню та зовнішню частини поверхні, а також про те, чи є поверхня замкненою. Зазначимо що потім це питання природно приводить до концепції водонепроникних сіток. Водонепроникні сітки зазвичай визначаються як 2-багатовидові сітки без граничних ребер, детальніше див. додаток Б. Інша проблема, проілюстрована на рисунку 2.1, стосується дуже деталізованих сіток, тобто сіток, що складаються з великої кількості граней і вершин, часто в десятки тисяч. Тому спрощення сітей є важливою проблемою у комп'ютерній графіці. Пізніше представимо дуже практичний підхід до обчислення дуже грубих, спрощених сіток. У нашому випадку цей алгоритм вирішує обидві проблеми — отримання водонепроникних та спрощених сіток.

2.2 Сітки зайнятості та знакові функції відстані

Враховуючи хмари точок або водонепроникні сітки, сітки зайнятості є природним представленням для застосування методів глибокого навчання. Зокрема, тривимірні згорткові нейронні мережі здатні працювати

безпосередньо з наданою топологією і таким чином використовувати просторову інформацію. Робота безпосередньо з трикутними сітками або хмарами точок, навпаки, є менш простою щодо належного представлення (наприклад, для кодування граней), яке мало б бути інваріантним до порядку граней або точок і дозволяти використовувати локальну інформацію. Сітки зайнятості передбачають вокселізацію простору та визначають для кожного вокселя його зайнятість. Зокрема, воксель вважається зайнятим, якщо він лежить усередині фігури або перетинається з її поверхнею. Для хмар точок, воксель вважається зайнятим, якщо він містить хоча б одну точку.

Визначення 2.3 — сітка зайнятості є тензором $x \in \mathbb{R}^{H \times W \times D}$, де кожен елемент x_1 називається вокселем. Значення $x_1 = 1$ вказує на зайнятий воксель, тоді як $x_i = 0$ вказує на вільний воксель.

Сітки зайнятості можна трактувати як явне представлення форми. За умови, що H, W і D достатньо великі, тобто використовується висока роздільна здатність, можна зафіксувати навіть невеликі деталі форм. На рисунку 2.1 ми ілюструємо вокселізацію показаної спрощеної мережі.

У відмінність від сіток зайнятості, знакові функції відстані використовуються для неявного визначення форми. У загальному випадку знакова функція відстані може бути визначена наступним чином:

Визначення 2.4 — нехай $F: \mathbb{R}^3 \mapsto \mathbb{R}$ — це неперервна функція. Тоді поверхня форми неявно визначена нульовим рівнем набору $S := \{x \in \mathbb{R}^3 | F(x) = 0\}$ функції F. Як конвенція, F є від'ємним у межах форми і позитивним за межами форми.

Неявне представлення за допомогою функцій відстані спрощує проблему визначення внутрішньої та зовнішньої частини форми. Назва виникає з того факту, що F зазвичай вважається знаковою відстанню до найближчої точки на поверхні, наприклад, $|F(x)| = \min_{x_S \in S} \|x - x_S\|_2$.

Визначення 2.5 — знакова функція відстані — це тензор $x \in \mathbb{R}^{H \times W \times D}$, такий що x_i є від'ємною відстанню центру відповідного вокселя до поверхні форми. Усередині форми вона є від'ємною, а поза нею — позитивною. У

практиці, для спрощення, обчислюємо знакові функції відстані з сіток зайнятості. Це важливе розгалуження; використовуючи представлення сіток зайнятості, втрачаємо точність в залежності від використаної роздільної здатності. Щоб відновити вищу точність, можемо використовувати замість цього знакову функцію відстані, отриману з оригінальної поверхні.

Ремарка 2.1 — дано сітку зайнятості $x \in \{0, 1\}^{H \times W \times D}$, ми отримуємо відповідну функцію відстані $df(x) \in \mathbb{R}^{H \times W \times D}$ як $df_i(x) \in \min_{j, x_j = 1} \|x - x_S\|_2$. Аналогічно, функція відстані зі знаком $sdf(x)$ може бути визначена комбінуванням $df(x)$ і $df(1 - x)$з відповідними знаками.

На практиці перетворення від сіток зайнятості до (знакових) функцій відстані може бути обчислено за допомогою узагальнених трансформацій відстані. Для обчислення знака, необхідно знати внутрішність і зовнішність; для герметичних сіток внутрішність і зовнішність можуть бути визначені за допомогою сіток зайнятості та алгоритмів з'єднаних компонентів/заповнення заливкою.

Як визначено вище, вважаємо, що функція відстані $df_i(x)$ представляє відстань до наступного зайнятого вокселя $x_j = 1$, така що $df_i(1 - x)$ — це відстань до наступного вільного вокселя. Також виявили, що використання логарифмічної функції відстані — $lsdf(x)_i = sign(sdf(x)_i) \ln(1 + |sdf(x)_i|)$, що допомагає навчанню нейронних мереж – стратегія, яка також використовується для подібних зображень, наприклад, для прогнозування глибини. Інтуїтивно зрозуміло, що логарифм зменшує діапазон задачі прогнозування; однак, на відміну від простого масштабування, діапазон зменшується нелінійно, так що діапазон малих відстаней (тобто навколо нульового рівня) ефективно збільшується за рахунок більших відстаней.

Маючи сітки зайнятості або знакові функції відстані між множиною фігур Y та спостереженнями X, наш підхід до задачі 1.3 включає два кроки: по-перше, вивчаємо попередню форму, яка визначає простір дозволених фігур; по-друге, вивчаємо модель виведення для вбудовування спостережень у той самий простір латентних фігур. Попереднє знання форми визначає

спільний розподіл $p(y, z) = p(y|z)p(z)$ форм y та латентних кодів z. По суті, попередня форма являє собою вбудовування форм Y у низьковимірний латентний простір Z. Накладаючи попередню форму $p(z)$ на латентний простір, ми можемо генерувати форми, використовуючи $y \sim p(y|z)$ для $z \sim p(z)$. Крім того, завершення форми можна визначити над низьковимірним латентним простором, також вивчаючи вкладання $x \mapsto z$ спостережень x у латентний простір. У нашому формулюванні амортизованої максимальної правдоподібності це вкладання є детермінованим. Для запропонованого розширеного варіаційного автокодера це вкладання також є ймовірнісним, тобто виражається як $q(z|x)$.

Далі ми маємо намір вивчити попередню форму за допомогою варіаційних автокодерів на множині фігур $Y = \{y_1, \dots, y_M\}$.

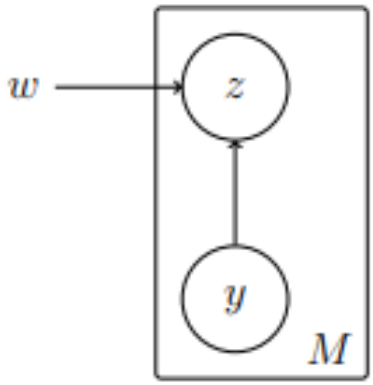

Рисунок 2.2 — Ілюстрація моделі розпізнавання для моделі розпізнавання $q(z|y)$, тобто кодера

Для сіток зайнятості або знакових функцій відстаней ми припускаємо для простоти сплющену версію $y \in \mathbb{R}^R \simeq \mathbb{R}^{H \times W \times D}$. Прихований простір тоді задається як $Z = \mathbb{R}^Q$ для малих $Q \ll R$. Варіаційний автокодер є реалізацією більш загальної моделі неперервної латентної змінної, яку можна легко узагальнити за допомогою двох графічних моделей на рисунках 2.2 і 2.3.

Хоча нас в основному цікавить генеративна модель p[y|z] при фіксованому попередньому p[z], навчання також вимагає вивчення моделі розпізнавання q[z|y] для того, щоб максимізувати загальну ймовірність $p(y)$.

$$p(y) = \int p(y,z)dz = \int p(y|z)p(z)dz \quad (2.1)$$

Для простих гауссівських моделей модель розпізнавання q[z|y], а також так звана гранична ймовірність зазвичай можна визначити аналітично, наприклад, див. обговорення імовірнісного аналізу головних компонентів у додатку B.

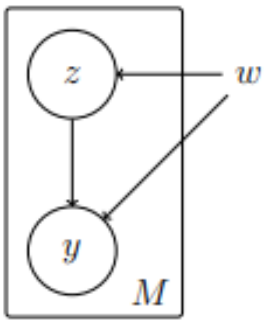

Рисунок 2.3 — Ілюстрація генеративної моделі $p(y|z)$, тобто декодер, який моделюється за допомогою нейронної мережі та параметрів w.

У випадку варіаційних автокодерів, як генеративна, так і розпізнавальна модель реалізуються за допомогою нейронних мереж. Тоді модель розпізнавання, а також гранична ймовірність можна лише наближено визначити — головним чином тому, що інтеграл у рівнянні 2.1 стає нерозв'язним.

2.3 Варіаційне виведення

Загалом, варіаційне виведення ставиться як задача знаходження модельного розподілу $q(z)$ для апроксимації істинних апостеріорних значень $p(z|y)$

$$q(z) = \underset{q}{\operatorname{argmin}}\, KL(q(z)|p(z|y)) \quad (2.2)$$

де дивергенція Кульбака-Лейблера KL — міра відстані, визначена на розподілах ймовірностей. Дивергенцію Кульбака-Лейблера можна переписати

так щоб отримати нижню межу нерозв'язної граничної ймовірності $p(y)$. Дивергенція Кульбака-Лейблера формально визначається як:

Визначення 2.6 — розбіжність Кульбака-Лейблера між двома розподілами ймовірностей $q(z)$ та $p(z|y)$ визначається як $KL\big(q(z)\big|p(z|y)\big) = \mathbb{E}_{q(z)}\left[\ln\frac{q(z)}{p(z|y)}\right]$, де $\mathbb{E}_{q(z)}$ позначає математичне сподівання відносно розподілу $q(z)$.

Уважний погляд на дивергенцію Кульбака-Лейблера показує, що проблема оптимізації у рівнянні 2.2 передбачає обчислення граничної ймовірності:

$$KL(q(z)|p(z|y) = \mathbb{E}_{q(z)}\left[\ln\frac{q(z)}{p(z|y)}\right] = \mathbb{E}_{q(z)}[\ln q(z)] - \mathbb{E}_{q(z)}[\ln p(z|y)] =$$
$$\mathbb{E}_{q(z)}[\ln q(z)] - \mathbb{E}_{q(z)}[\ln p(z,y)] + \ln p(y).$$

Перестановка лівої та правої частин приводить до нижньої межі доказу, яку також називають варіаційною нижньою межею $\ln p(y) = KL(q(z)|p(z|y) - \mathbb{E}_{q(z)}[\ln q(z)] + \mathbb{E}_{q(z)}[\ln p(z,y)] \geq$
$-\mathbb{E}_{q(z)}[\ln q(z)] + \mathbb{E}_{q(z)}[\ln p(z,y)] = -\mathbb{E}_{q(z)}[\ln q(z)] + \mathbb{E}_{q(z)}[\ln p(z)] +$
$\mathbb{E}_{q(z)}[\ln p(y|z)] = -KL(q(z)|p(z)) + \mathbb{E}_{q(z)}[\ln p(y|z)]$.

Початкова задача максимізації нерозв'язної граничної ймовірності $p(y)$ у рівнянні 2.1 потім апроксимується максимізацією доказової нижньої межі яку ми формально визначаємо як:

Визначення 2.7 — варіаційна нижня границя або доказова нижня границя, отримана із задачі 2.2 має вигляд:

$$-KL\big(q(z)\big|p(z)\big) + \mathbb{E}_{q(z)}[\ln p(y|z)] = \mathbb{E}_{q(z)}\left[\ln\frac{p(y,z)}{q(z)}\right] \tag{2.3}$$

У цьому загальному формулюванні нижньої межі достовірності розподіл моделі $q(z)$ може бути довільним. Однак у контексті моделей з латентними змінними має сенс зробити розподіл моделі залежним від у явно,

тобто $q(z|y)$, оскільки ми хочемо мати можливість відновити будь-яке y за відповідним латентним кодом z.

Тоді нижня межа достовірності в рівнянні (6.3) набуває вигляду автокодера, де $q(z|y)$ представляє кодер, а $p(y|z)$ — декодер.

Таким чином, варіаційний автокодер можна навчити, максимізуючи праву частину рівняння 2.3 після вибору відповідних параметризацій для розподілів $p(z)$ і $q(z|y)$.

2.4 Гаусівський варіаційний автокодер

У рамках варіаційних автокодерів модель розподілу $q(z|y)$ реалізовано як нейронну мережу; аналогічно, $p(y|z)$ змодельовано як нейронну мережу. У випадку гауссівських варіаційних автокодерів як $q(z|x)$, так і $p(z)$ вважаються гауссівськими, а саме: $q(z|y) = N\left(z; \mu(y; w), \mathrm{diag}\left(\sigma^2(y; w)\right)\right)$, $p(z) = N(z; 0, I_Q)$, де залежність від ваг нейронної мережі w є явною, тобто обидва середніх значення $\mu(y; w) \in \mathbb{R}^Q$, так і коваріаційна матриця $\mathrm{diag}\left(\sigma^2(y; w)\right) \in \mathbb{R}^{Q \times Q}$ прогнозуються за допомогою нейронної мережі з параметрами w, що підлягають оптимізації. Далі для стислості ми нехтуватимемо вагами w. Враховуючи нижню межу доказу з рівняння 2.3, тобто (2.3)$= -\mathrm{KL}\left(q(z|y)\middle|p(z)\right) + \mathbb{E}_{q(z)}[\ln p(y|z)]$, розбіжність Кульбака-Лейблера між $q(z|y)$ та $p(z)$ можна обчислити аналітично. Однак диференціювання нижньої межі з урахуванням прихованих ваг у $q(z|y)$ є проблематичним. Класичне наближення методом Монте-Карло математичного сподівання $\mathbb{E}_{q(z)}[\ln p(y|z)]$ вимагало б диференціювання через процес вибірки $z \sim q(z|y)$. Тому Кінгма та Веллінг запропонували так званий прийом репараметризації. Зокрема, випадкова величина $z \sim q(z|y)$ репараметризується за допомогою диференційованого перетворення $g(z, \epsilon)$ на основі допоміжної змінної, взятої з Гауссового блоку: $z_i = g_i(y, \epsilon_i) = \mu_i(y) + \epsilon_i \sigma_y^2(y)$, з $\epsilon_i \sim N(\epsilon; 0, 1)$. Загалом, для вибірки $y_m \in \mathbb{R}^R$,ціль, яку потрібно мінімізувати, має вигляд: $\mathcal{L}_{\mathrm{VAE}}(w) = \mathrm{KL}\left(q(z|y_m)\middle|p(z)\right) - \frac{1}{L}\sum_{l=1}^{L} \ln p(y_m|z_{l,m})$, де $z_{l,m} = g(\epsilon_{l,m}, y)$ і $\epsilon_{l,m} \sim N(\epsilon; 0, I_Q)$.

Тут L — це кількість вибірок, яку потрібно використати для оцінки помилки реконструкції методом Монте-Карло. На практиці $\mathcal{L}_{\mathrm{VAE}}$ з урахуванням втрат застосовується для міні-партій і L = 1 зазвичай є достатнім. Враховуючи ваги нейронної мережі w, процес генерації можна узагальнити наступним чином: побудувати $\mathrm{w} \sim \mathrm{p}(\mathrm{z}) = \mathrm{N}(\mathrm{z}; 0, \mathrm{I}_{\mathrm{Q}})$, і побудувати $\mathrm{y} \sim \mathrm{p}(\mathrm{y}|\mathrm{z})$. Аналогічно, розпізнавання виконується шляхом побудови $\mathrm{z} \sim \mathrm{q}(\mathrm{z}|\mathrm{y})$, тобто $\epsilon \sim \mathrm{N}(\epsilon; 0, \mathrm{I}_{\mathrm{Q}})$ і $\mathrm{z} = \mathrm{g}(\mathrm{y}, \epsilon)$. Для оцінювання, тобто для вимірювання ефективності розпізнавання, z зазвичай встановлюється на рівні $\mathrm{z} = \mathbb{E}_{\mathrm{q}(\mathrm{z}|\mathrm{y})}[\mathrm{z}]$; це може бути досягнуто шляхом безпосереднього розгляду $\mu(\mathrm{y})$.

Як зазначалося вище, розбіжність Кульбака-Лейблера двох гауссових розподілів можна легко обчислити безпосередньо. Оскільки ми розглядаємо діагональні коваріаційні матриці, розбіжність Кульбака-Лейблера є відокремлюваною на $1 \leq \mathrm{i} \leq \mathrm{Q}$. Тоді маємо: $KL\left(N\left(z_i; \mu_{1,i}, \sigma_{1,i}^2\right) \middle| N\left(z_i; \mu_{2,i}, \sigma_{2,i}^2\right)\right) = \frac{1}{2} \ln \frac{\sigma_{2,i}^2}{\sigma_{1,i}^2} + \frac{\sigma_{1,i}^2}{\sigma_{2,i}^2} + \frac{(\mu_{1,i} - \mu_{2,i})^2}{2\sigma_{2,i}^2} - \frac{1}{2}$.

А при $\mu_{2,\mathrm{i}} = 0$ та $\sigma_{2,\mathrm{i}} = 1$ (і для простоти $\sigma_{\mathrm{i}}^2 := \sigma_{\mathrm{i}}^2(\mathrm{y}) = \sigma_{1,\mathrm{i}}^2$ та $\mu_{\mathrm{i}} := \mu_{\mathrm{i}}(\mathrm{y}) = \mu_{1,\mathrm{i}}$) випливає:

$$\mathrm{KL}\big(\mathrm{q}(\mathrm{z}_{\mathrm{i}}|\mathrm{y}) \big| \mathrm{p}(\mathrm{z}_{\mathrm{i}})\big) = -\frac{1}{2} \ln \sigma_{\mathrm{i}} + \frac{1}{2}\sigma_{\mathrm{i}}^2 + \frac{1}{2}\mu_{\mathrm{i}}^2 - \frac{1}{2} \qquad (2.4)$$

Інша частина завдання — це похибка реконструкції, тобто від'ємна лог-правдоподібність $-\ln \mathrm{p}(\mathrm{y}|\mathrm{z}_{\mathrm{l}})$. Це залежить від способу моделювання $\mathrm{p}(\mathrm{y}|\mathrm{z})$; у нашому випадку, $\mathrm{p}(\mathrm{y}|\mathrm{z})$ розкладається поелементно на вокселі — $p(y|z) = \prod_{i=1}^{R} p(y_i|z) \Rightarrow -\ln p(y|z) = -\sum_{i=1}^{R} \ln p(y_i|z)$.

Для форм у у вигляді сіток зайнятості ми використовуємо розподіл Бернуллі для моделювання окремих вокселів, тобто $\mathrm{p}(\mathrm{y}_{\mathrm{i}}|\mathrm{z}) = \mathrm{Ber}(\mathrm{y}_{\mathrm{i}};\ \theta_{\mathrm{i}}(\mathrm{z}))$, де ймовірності зайнятості $\theta_{\mathrm{i}}(\mathrm{z})$ прогнозуються за допомогою декодера. Від'ємна логарифмічна вірогідність тоді зводиться до двійкової перехресної ентропійної похибки. При роботі зі знаковими функціями відстані ми використовуємо гаусівський розподіл з фіксованою дисперсією для моделювання окремих вокселів, тобто $\mathrm{p}(\mathrm{y}_{\mathrm{i}}|\mathrm{z}) = \mathrm{N}(\mathrm{y}_{\mathrm{i}}; \mu_{\mathrm{i}}(\mathrm{z}), \sigma^2)$, де значення

$\mu_i(z)$ передбачаються декодером. У цьому випадку від'ємна лог-вірогідність призводить до середньоквадратичної похибки.

Для ілюстрації та реалізації ми визначаємо два додаткові шари:

Визначення 2.8 — гауссівський шар розбіжності Кульбака-Лейблера $\mathrm{KLD_N}$ приймає на вхід два тензори $\mu \in \mathbb{R}^{B\times Q}$ та $\sigma^2 \in \mathbb{R}^{B\times Q}$ і обчислює розбіжність Кульбака-Лейблера — $\sum_{b=1}^{B} \mathrm{KL}\left(\mathrm{N}\left(z; \mu_b, \mathrm{diag}(\sigma_b^2)\right) \middle| \mathrm{N}(z; 0, I_Q)\right)$, перед проходженням прогнозованого значення на $\mu_b, \sigma_b^2 \in \mathbb{R}^Q$ наступний шар (без змін) — $\mathrm{KLD_N}(\mu, \sigma^2) = (\mu, \sigma^2)$.

По суті, цей шар просто робить обчислення розбіжності Кульбака-Лейблера явним — в окремому шарі. Це означає, що на прямий прохід це не впливає. Однак важливо враховувати похідні від передбачених $\mu(y)$ і $\sigma^2(y)$, оскільки їх потрібно додавати до градієнтів декодера (що походять від втрат при реконструкції) під час зворотного поширення помилки, щоб врахувати розбіжність Кульбака-Лейблера під час навчання: $\frac{\partial \mathrm{KL}\left(q(z_i|y)\middle|p(z_i)\right)}{\partial \mu_i} = \mu_i$ та $\frac{\partial \mathrm{KL}\left(q(z_i|y)\middle|p(z_i)\right)}{\partial \sigma_i} = -\frac{1}{2\sigma_i} + \sigma_i$.

Після обчислення розбіжності Кульбака-Лейблера застосовується трюк репараметризації застосовується трюк репараметризації:

Визначення 2.9 — гауссівський шар репараметризації $\mathrm{repa_N}$ приймає на вхід два тензори $\mu \in \mathbb{R}^{B\times Q}$ та $\sigma^2 \in \mathbb{R}^{B\times Q}$ і обчислює один вихідний тензор:

$$\mathrm{repa_N}(\mu, \sigma^2) = \mu + \epsilon\sigma, \tag{2.5}$$

де $\epsilon \in \mathbb{R}^{B\times Q}$, $\epsilon_{b,i} \sim \mathrm{N}(\epsilon; 0, 1)$, і μ перемножуються поелементно.

Знову ж таки, зазначимо, що основною метою шару репараметризації є вибірка з $q(z|y)$ у диференційований спосіб. Тоді, наступний приклад ілюструє, як згортковий автокодер можна "покращити" до варіаційного автокодера можна "вдосконалити" до варіаційного автокодера за допомогою нещодавно введених шарів:

Приклад 2.1 — розглянемо на рисунку 2.4 нашу реалізацію варіаційного автокодера з чотирма ступенями згортки в кодері та декодері. У випадку кодера два повністю з'єднані шари незалежно обчислюють середнє $\mu(y) \in \mathbb{R}^Q$ і дисперсію $\sigma^2(y) \in \mathbb{R}^Q$ для одного і того ж входу. Потім обчислюється дивергенція Кульбака-Лейблера $KL(q(z|y)|p(z))$ і середнє значення та дисперсія передаються до шару репараметризації. Тут вибирається допоміжна змінна $\epsilon \sim N(\epsilon; 0, I_Q)$ для того, щоб зробити вибірку прихованого коду $z \sim q(z|y) = N(z; \mu(y), diag(\sigma^2(y)))$ за допомогою $z = g(y, \epsilon)$, q, а потім передається в декодер. Під час тестування шар репараметризації можна видалити, а передбачене середнє значення $\mu(y)$ безпосередньо передати в декодер. На ілюстрації ми також показуємо втрати чіткості при реконструкції $\mathcal{L}(\tilde{y}, y) = -\ln p(y|\tilde{z})$.

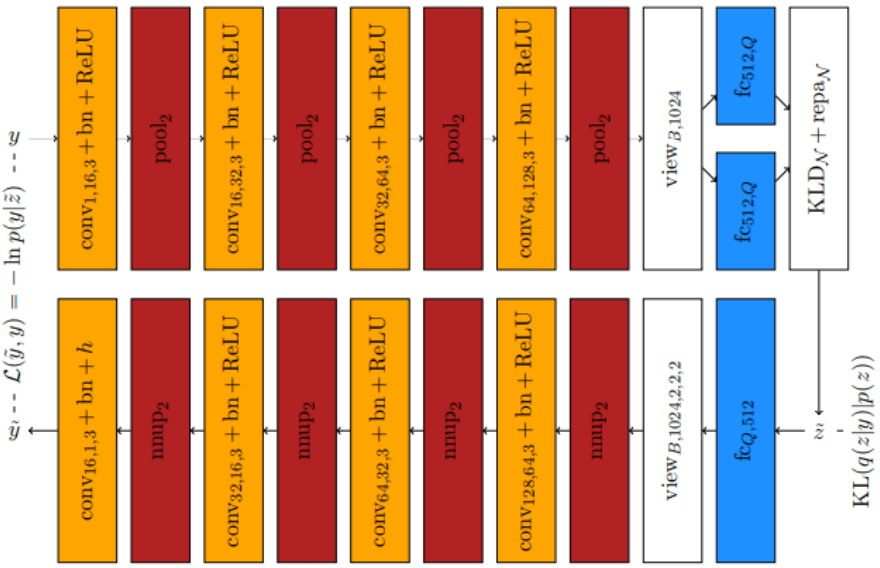

Рисунок 2.4 — Ілюстрація гауссівського варіаційного автокодера з чотирма згортковими каскадами як у кодері, так і в декодері.

На практиці, дозволити кодеру передбачати $\sigma^2(y)$ безпосередньо проблематично, оскільки дисперсія $\sigma^2(y)$ не може бути від'ємною. Тому ми слідуємо загальнодоступним реалізаціям і дозволяємо кодеру передбачати лог-варіації, тобто $l_i(y) := \ln \sigma^2(y)$.

Це гарантує, що дисперсія $\sigma^2(y) = \exp(l_i(y))$ завжди буде додатною. Розбіжність Кульбака-Лейблера з рівняння 2.4) і відповідна похідна від $l_i(y)$, а також прийом репараметризації з рівняння 2.5) легко адаптуються.

Навчання гауссівського варіаційного автокодера на практиці означає балансування — можливо, суперечливих — цілей, що відповідають втратам при реконструкції та розбіжності Кульбака-Лейблера. В той час як втрати при реконструкції можна інтуїтивно інтерпретувати, наприклад, як двійкову перехресну ентропійну похибку або масштабовану середньоквадратичну похибку у випадку Бернуллі та Гаусса відповідно, розбіжність Кульбака-Лейблера менш інтуїтивно зрозуміла. Тому може бути корисним відстежувати статистику латентного простору на утримуваному валідаційному наборі. Конкретно це означає моніторинг перших двох моментів прогнозованого середнього, тобто

$$\bar{\mu} = \frac{1}{QM'} \sum_{i=1}^{Q} \sum_{m=1}^{M'} \mu_i(y_m) \tag{2.6}$$

та

$$\mathrm{Var}[\mu] = \frac{1}{QM'} \sum_{i=1}^{Q} \sum_{m=1}^{M'} (\mu_i(y_m) - \bar{\mu})^2, \tag{2.7}$$

де M' — розмір валідаційної множини.

Додатково варто відстежувати перший момент прогнозованих логарифмічних варіацій:

$$\bar{l} = \frac{1}{QM'} \sum_{i=1}^{Q} \sum_{m=1}^{M'} \ln \sigma_i^2(y_m) = \frac{1}{QM'} \sum_{i=1}^{Q} \sum_{m=1}^{M'} l_i(y_m). \tag{2.8}$$

Зауважмо, що ці статистики обчислюються для всіх Q вимірів латентного простору — це доречно, оскільки попередній p(z) є одиничним гаусівським, так що всі виміри можуть розглядатися однаково. Під час навчання ці статистики повинні збігатися до одиничного гаусівського, тобто

$\bar{\mu} \approx 0$ і $\mathrm{Var}[\mu] \approx 1$, і мережа повинна бути здатна робити певні прогнози, тобто малі $\bar{I}$.

2.5 Варіаційний автокодер Бернуллі

Гауссівський варіаційний автокодер буде використано для вивчення латентного простору $\mathbf{z}$, тобто попередньої форми. Однак у випадку розширеного варіаційного автокодера для завершення форми ми маємо намір моделювати спільну ймовірність $p(x, y, z)$, вводячи випадкові величини для спостережень x_i , а також. Тут і y, і z вважаються латентними змінними, тому ми також повинні мати можливість моделювати дискретні латентні змінні — зокрема, бінарні латентні змінні, вважаючи у представленою у вигляді сітки заповнюваності. На жаль, це не так просто зробити в рамках представленої моделі.

Однак нещодавно дослідники змогли змоделювати як $p(z)$ та $q(z|y)$ за допомогою розподілів Бернуллі, тобто $p(z) = \prod_{i=1}^{Q} \mathrm{Ber}(z_i; 0.5)$ і $p(z|y) = \prod_{i=1}^{Q} \mathrm{Ber}(z_i; \theta_i(y))$, де $\theta_i(y)$ – передбачення за допомогою кодера, враховуючи вхідні дані y. Знову ж таки, нижня межа доказів має бути оптимізована, це означає для вибірки y_m: $\mathcal{L}_{\mathrm{VAE}}(w) = \mathrm{KL}\big(q(z|y_m)\big|p(z)\big) - \frac{1}{L}\sum_{l=1}^{L} \ln p\big(y_m\big|z_{l,m}\big)$, де $z_{l,m} \sim q(z|y)$. Дивергенцію Кульбака-Лейблера можна обчислити аналітично, однак диференціювання через процес дискретизації $z_{l,m} \sim q(z|y)$є проблематичним. На жаль, трюк репараметризації, який використовувався у випадку Гаусса, більше не застосовується. На даний момент пропонують альтернативний прийом репараметризації для загальних дискретних розподілів, які ми визначаємо наступним чином:

Визначення 2.10 — нехай $z \in \{1, \dots, K\}$ — випадкова величина; тоді z розподілена згідно з дискретним розподілом, тобто $z \sim \mathrm{Dis}(z; \pi)$, з параметрами $\pi = (\pi_1, \dots, \pi_K)$, якщо $p(z = k) = \pi_K$ та $\sum_{k=1}^{K} \pi_K = 1$. Представимо z за допомогою так званого однократного кодування, тобто $z \in \{0, 1\}^2$ таке, що $z_k = 1$ і $z_{k'} = 0$, $k' \neq k$, якщо відбувається подія k.

Репараметризація додатково базується на розподілі Гумбеля:

Визначення 2.11 — нехай $\epsilon \in \mathbb{R}$ — випадкова величина. Тоді вона розподілена згідно з розподілом Гумбеля, тобто $\epsilon \sim \mathrm{Gu}(\epsilon; \mu, \beta)$, з параметрами μ та β, якщо $p(\epsilon) = \exp\left(-\exp\left(\frac{\epsilon-\mu}{\beta}\right)\right)$. Стандартний розподіл Гумбеля є окремим випадком $\mu = 0$ і $\beta = 1$.

Ключовим висновком є те, що дуже легко зробити вибірку з розподілу Гумбеля: нехай $u_k \sim U(0, 1)$, тоді $\epsilon_k = \mu - \beta\log(-\log(u_k))$ розподілено згідно з $Gu(\mu, \beta)$. Стандартний розподіл Гумбеля також допомагає робити вибірки з дискретного розподілу. Нехай $\epsilon_1, \ldots, \epsilon_k \sim \mathrm{Gu}(\epsilon; 0, 1)$, тоді покладемо $z_k = 1$ для

$$k = \underset{k}{\operatorname{argmax}} \ln \pi_k + \epsilon_k \tag{2.9}$$

і z будуть розподілені відповідно до дискретного розподілу, тобто $z \sim \mathrm{Dis}(z; \pi)$; це називається трюком Гумбеля. Заміна argmax на його гладкий аналог, тобто softmax, дає остаточний трюк репараметризації:

$$z_k = \frac{\exp(\ln \pi_k + \epsilon_k)}{\sum_{k'=1}^{d} \exp(\ln \pi_{k'} + \epsilon_{k'})} \tag{2.10}$$

Інші науковці додатково додають параметр температури, тобто $(\ln \pi_k + \epsilon_k) / \lambda$, і показують, що для $\lambda \to 0$ рівняння 2.10 прямує до реального дискретного розподілу. Однак для простоти ми нехтуємо температурою λ у нашому випадку.

Розподіл Бернуллі є окремим випадком дискретного розподілу. Нехай $z \sim \mathrm{Ber}(z; \theta)$, $z = 1$ означає, що $\ln\theta + \epsilon_1 > \ln(1 - \theta) + \epsilon_2$, де ми використали рівняння 2.9, де $\pi_1 = 0$ ймовірність $z = 1$, а $\pi_2 = (1 - \theta)$ — ймовірність $z = 0$, і виразили арґмакс через нерівність.

Різниця двох випадкових величин Гумбеля $\epsilon_1 - \epsilon_2$ розподілена згідно з логістичним розподілом, де ми можемо зробити вибірку за допомогою $\epsilon_1 - \epsilon_2 = \ln(u) - \ln(1 - u)$ з $u \sim U(0, 1)$. Отже,

$$z = \begin{cases} 1 & \ln u - \ln(1 - u) + \ln \theta - \ln(1 - \theta) > 0 \\ 0 & \text{else} \end{cases},$$

яку можна зробити диференційованою за допомогою м'якої порогової операції, наприклад, сигмоїдної функції. Зі зміною позначень, нехай $\epsilon \sim U(0, 1)$ це призводить до

$$z_i = g(y, \epsilon) = \sigma(\ln \epsilon - \ln(1 - \epsilon) + \ln \theta_i(y) - \ln(1 - \theta_i(y))), \quad (6.11)$$

що є останнім використаним прийомом репараметризації.

Решта структури залишається незмінною; розбіжність Кульбака-Лейблера знову обчислюється аналітично, а похибка реконструкції.

2.6 Максимальна правдоподібність

Поки що ми можемо використовувати набір фігур $Y \subseteq \mathbb{R}^{H \times W \times D} \simeq \mathbb{R}^R$ з задачі 1.3 для того, щоб вивчити попередню модель $p(y, z)$ у можливо низьковимірному латентному просторі $Z = \mathbb{R}^R$. З цією метою ми ввели варіаційні автокодери, де генеративна модель $p(y|z)$ і (наближена) модель розпізнавання $q(z|y)$ реалізуються за допомогою тривимірних згорткових нейронних мереж. У нашому першому підході до завершення форми ми маємо намір навчитися моделі виведення

$$x \mapsto \tilde{z} \approx \operatorname*{argmax}_z p(y = x|z)p(z), \quad (2.11)$$

за допомогою набору спостережень $x \in \mathbb{R}^{H \times W \times D}$. Ми називаємо цей підхід амортизованою максимальною правдоподібністю, оскільки ми не розглядаємо спостереження x незалежно, а розуміємо мету максимальної правдоподібності як втрату, що дозволяє вивчити детерміноване вкладання $x \mapsto \tilde{z}$ у неконтрольованому середовищі. Враховуючи попередній варіаційний автокодер, це призводить до навчання нового кодера, а попередньо навчений декодер залишається фіксованим.

У запропонованому розширеному варіаційному автокодері ми натомість розуміємо спостереження x як випадкову величину. Для відповідного спільного розподілу p(x, y, z) ми можемо отримати нижню межу достовірності, припускаючи, що y і z статистично незалежні від x. Подібно до попередньої моделі, ми вивчаємо наближену модель розпізнавання q(z|x), яку можна розуміти як імовірнісне вбудовування $x \mapsto z$. На відміну від амортизованої максимальної правдоподібності, фактична мета, що пов'язує спостереження x з формами y, прихована в розбіжності Кульбака-Лейблера. Знову ж таки, модель можна навчати без нагляду. На додаток до навчання нового кодера, розширений варіаційний автокодер також реалізує модель спостережень p(x|y) за допомогою 3D-згорткової нейронної мережі.

Порівняно з амортизованою максимальною правдоподібністю, це потенційно дозволяє явно інтегрувати знання про модель спостереження, однак призводить до збільшення часу навчання — також тому, що вбудовування $x \mapsto z$ моделюється імовірнісно. Далі ми спочатку обговоримо загальний підхід максимальної правдоподібності до завершення фігури. Спочатку ми експериментували з різними втратами, щоб вивчити вкладання $x \mapsto y$ в неімовірнісній структурі..

Від'ємна логарифмічна ймовірність, що відповідає рівнянню 2.11, має вигляд $\underset{z}{\operatorname{argmin}} - \ln p(y|z) - \ln p(z)$. Оскільки p(y|z) є диференційованою моделлю, ми можемо застосувати градієнтний спуск після розкладання p(y|z) на вокселі: $-\ln p(y|z) = -\sum_{i=1}^{R} \ln p(y_i|z)$, де ми знову згладили представлення $y \in \mathbb{R}^{H \times W \times D} \simeq \mathbb{R}^{R}$. Розглядаючи фактичні спостереження xi , ми спочатку виявляємо, що ми не обов'язково маємо спостереження x_i для кожного вокселя i.

Формально ми записуємо $x \in \{0, 1, \perp\}^{R}$, де ⊥ позначає невідомі значення; на практиці ми просто ігноруємо відповідні індекси. Оскільки ймовірність $p(y_i = x_i|z)$ не може бути визначена для $x_i = \perp$, ми виключаємо їх з оптимізаційної задачі $\underset{z}{\operatorname{argmin}} - \sum_{i=1,\, x_i \neq \perp}^{R} p(y_i|z) - \ln p(z)$. Інтуїтивно припускаємо, що попередні p(z, y) є достатньо сильними, щоб, обмеживши p(y|z) лише для

кількох y_i певними значеннями (через спостереження $x_i = \perp$, отримати прогнози гарної форми. Це практична альтернатива розгляду всіх y_i з $x_i = \perp$, як прихованих змінних для оптимізації на додаток до z

2.7 Амортизована максимальна правдоподібність

Амортизуючи, тобто вивчаючи, максимальну ймовірність, ми маємо намір уникнути проблеми оптимізації, необхідної для виводу в попередньому розділі. З цією метою ми вводимо новий, детермінований кодер $z(x;w)$, призначений для представлення відображення $x \mapsto z(x; w) \approx \operatorname{argmin}_z p(y = x|z)p(z)$, тобто маємо намір навчитися безпосередньо передбачати рішення з максимальною правдоподібністю. На практиці кодер $z(x;w)$ також реалізується за допомогою 3D згорткових нейронних мереж, що наслідують архітектуру моделі розпізнавання $q(z|y)$.

Використовуючи генеративну модель, тобто $p(y|z)$, кодер $z(x;w)$ можна навчати, мінімізуючи втрати, отримані з формулювання максимальної правдоподібності. Однак, на відміну від попереднього розділу, ми також розглядаємо випадок знакових функцій відстані як представлення форми, де $p(y|z)$ моделюється за допомогою гаусівського розподілу — це стає проблематичним при оцінюванні $p(y_i = x_i|z)$, оскільки хі за своєю природою є бінарним (тобто зайнятий або не зайнятий). Загалом, це призводить до двох втрат, однієї для заповнюваності, отриманої за допомогою припущення про розподіл Бернуллі та для знакових функцій відстані.

2.8 Розширений варіаційний автокодер

Розглянуті вище підходи сформульовані в рамках теорії максимальної правдоподібності, де вважаємо x_i фактичними спостереженнями випадкових величин y_i . На противагу цьому, ми також можемо безпосередньо розширити графічну модель на рисунку 2.5, щоб також розглядати спостереження x_i як випадкову величину. На рисунку 2.6 ми ілюструємо можливе розширення, яке ми вирішили дослідити ближче.

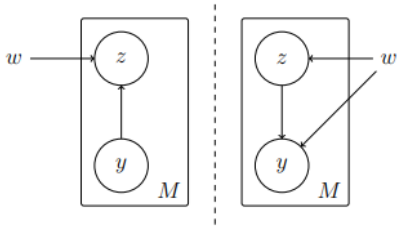

Рисунок 2.5 — Ілюстрація графічної моделі оригінального варіаційного автокодера

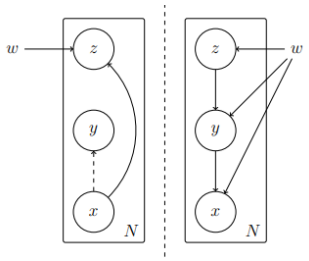

Рисунок 2.6 — Ілюстрація графічної моделі розширеного варіаційного автокодера розширеного варіаційного автокодера явно з урахуванням спостереження x

Основна ідея полягає в тому, щоб явно змоделювати процес спостереження p(x|y), щоб вивести інформацію про можливе значення y. Існує багато можливостей для моделювання p(x|y) — наприклад, вивчаючи його безпосередньо з вибірок $\{(x_n, y_n^*)\}$ або шляхом моделювання датчика. Тут залишаємо ці варіанти для подальшої роботи і моделюємо p(x|y) за допомогою нейронної мережі без прямого нагляду, вбудовуючи p(x|y) в рамки варіаційного виведення.

Після введення варіаційного висновку в підрозділі 2.3, докази нижню межу для моделі на рисунку 2.6 можна записати як: $-\mathrm{KL}\big(q(y,z|x)\big|p(y,z)\big) + \mathbb{E}_{q(y,z|x)}[\ln p(x|y)]$. У цьому формулюванні і y, і z розглядаються як латентні змінні відносно щойно введеної випадкової величини x. Точна структура q(y, z|x), а також p(y, z) поки що не визначена. Для останньої знову використовуємо попередньо навчену форму p(y, z) = p(y|z)p(z); для першої повинні зробити спрощувальні припущення. Зокрема, ми припускаємо, що y та z є статистично незалежними від x: $q(y,z|x) = q(y|x)q(z|x)$. Тут q(z|x) — відображення, яке

хочемо вивчити; p(y|x) розкладається на $q(y|x) = \prod_{i=1}^{R} q(y_i|x_i)$ і може бути легко змодельована для тих вокселів x_i, для яких ми маємо спостереження, тобто $x_i \neq \perp$. Зауважмо, що q(y|x) не є вивченою, тобто не моделюється за допомогою нейронної мережі — вона не є частиною моделі розпізнавання; однак q(y|x) дозволяє нам інтегрувати знання про спостереження в рамках розбіжності Кульбака-Лейблера: $\mathrm{KL}\big(q(y,z|x)\big|p(y,z)\big) = \mathbb{E}_{q(y,z|x)}\left[\ln\frac{q(y,z|x)}{p(y,z)}\right] = \mathbb{E}_{q(y,z|x)}\left[\ln\frac{q(y|x)q(z|x)}{p(y|z)p(y)}\right] = \mathbb{E}_{q(y,z|x)}\left[\ln\frac{q(y|x)}{p(y|z)}\right] + \mathbb{E}_{q(y,z|x)}\left[\ln\frac{q(z|x)}{p(z)}\right] = \mathrm{KL}\big(q(y|x)\big|p(y,z)\big) + \mathrm{KL}\big(q(z|x)\big|p(z)\big)$.

Нарешті, похибка реконструкції спрощується до $\mathbb{E}_{q(y,z|x)}[\ln p(x|y)] = \mathbb{E}_{q(y|x)}[\ln p(x|y)]$.

Тоді загальну мету можна записати так:

$$\mathcal{L}_{\mathrm{EVAE}}(w) = \mathrm{KL}\big(q(z|x)\big|p(z)\big) + \mathrm{KL}\big(q(y|x)\big|p(y,z)\big) - \mathbb{E}_{q(y|x)}[\ln p(x|y)], \quad (2.12)$$

де дивергенція Кульбака-Лейблера KL(q(y|x)|p(y|z)) неявно пов'язує спостереження x з можливою формою y.

Як і раніше, обидві дивергенції Кульбака-Лейблера можна обчислити аналітично. Лише похибка реконструкції $\mathbb{E}_{q(y|x)}[\ln p(x|y)]$ має бути апроксимована за допомогою Монте-Карло; для конкретної вибірки x_n це означає, що $\mathbb{E}_{q(y|x)[}\ln p(x|y)] = -\sum_{l=1}^{L}\ln p(x_n|y_{l,m})$. Тут g_y представляє прийом репараметризації Бернуллі а g_z — гауссівський еквівалент. Це також означає, що y_i моделюється за допомогою розподілу Бернуллі, так само як і x_i , тоді як z моделюється за допомогою гауссового розподілу. Щоб зробити це явним, запишемо: $p(z) = \mathrm{N}\left(z; \mu(x), \mathrm{diag}\big(\sigma^2(x)\big)\right)$, $p(y_i|z) = \mathrm{Ber}(y_i;\ \theta_i(z))$, $p(x_i|y) = \mathrm{Ber}(x_i;\ \rho_i(y))$, де параметри μ, σ^2, а також θ_i , ρ_i моделюються за допомогою нейронних мереж; зокрема, $\theta_i(z)$ береться з попередньо навченої форми, щоб обмежити виведення форми розумними формами.

Загалом, це визначає генеративну модель розширеного варіаційного автокодера, як показано на рисунку 2.7 праворуч.

У рівнянні 2.12 для обговорення залишаються лише розбіжності Кульбака-Лейблера KL(q(y|x)|p(y|z)) та модель розпізнавання q(z|x). Наближений апостеріор q(z|x) моделюється аналогічно до q(z|y). Для розбіжності Кульбака-Лейблера дотримуємося формулювання та використання методу максимальної правдоподібності: $q(y_i|x_i = 1) = \mathrm{Ber}(y_i; 1)$, $q(y_i|x_i = 0) = \mathrm{Ber}(y_i; 0)$, ігноруючи неспостережувані вокселі $x_i = \perp$, оскільки припускаємо достатньо сильну попередню форму, яка здатна заповнити неспостережувані вокселі. Дивергенція Кульбака-Лейблера KL(q(y|x)|p(y|z)) неявно намагається підігнати передбачену форму y до спостережень x.

Більшість з представлених підходів — за винятком, наприклад, пакетної нормалізації, введеної схеми ініціалізації ваг або деконволюції/дискретизації — використовуються вже кілька десятиліть, принаймні, у схожих формах. Однак багато цікавих ідей було запропоновано лише нещодавно, і, як наслідок, глибоке навчання знайшло широке застосування в комп'ютерному зорі.

3 ВИБІР ДАНИХ ДЛЯ ДОСЛІДЖЕННЯ

3.1 3D приклад та ShapeNet

Для нашого формулювання завершення фігури, тобто задачі 1.3, потрібні два джерела даних: набір фігур Y, щоб дізнатися попередню фігуру, і спостереження X, щоб дізнатися максимальну правдоподібність або розширений варіаційний автокодер. Однак для дослідницьких цілей ми створили три синтетичні набори даних, які також включають базові фігури Y*, що відповідають X для оцінки: набір прямокутників у 2D, набір кубоїдів у 3D і набір автомобілів з ShapeNet у 3D. Крім того, що ці набори даних містять як спостереження X, так і базову істину Y*, вони також відображають прогрес, досягнутий в ході виконання дисертації. Для стислості представимо лише експерименти у 3D; Приклад з синтетичного набору 2D з роздільною здатністю в 1D можна знайти у додатку Г. Крім того, ми представляємо експерименти на хмарах точок, наданих KITTI. У цьому випадку ми використовуємо моделі автомобілів з ShapeNet як набір форм Y і надані 3D-обмежувальні рамки для вилучення відповідних спостережень X. Однак, на KITTI не можемо кількісно оцінити запропоновані підходи через відсутність базових форм Y*. Далі представляємо наш процес генерації синтетичних наборів 3D-даних, тобто набору даних 3D-кубоїдів і ShapeNet, та вилучення спостережень з KITTI.

Наші 3D-дані складаються з довільно масштабованих і повернених кубоїдів або автомобілів; кубоїди генеруються на льоту, тоді як ми використовуємо моделі автомобілів з ShapeNet. Моделі надаються у вигляді трикутних сіток, тому нам потрібно розглянути наступні кроки обробки: вокселізація та заповнення фігур; рендеринг, зворотне проектування та вокселізація спостережень; і, нарешті, постобробка для обчислення знакових функцій відстаней. Це забезпечить нас як спостереженнями, так і істинними фігурами; щоб вивчити попередню фігуру і змоделювати реалістичний випадок, ми потім вокселізуємо і заповнюємо окремий набір фігур. Загальний процес також проілюстровано на рисунку 3.1.

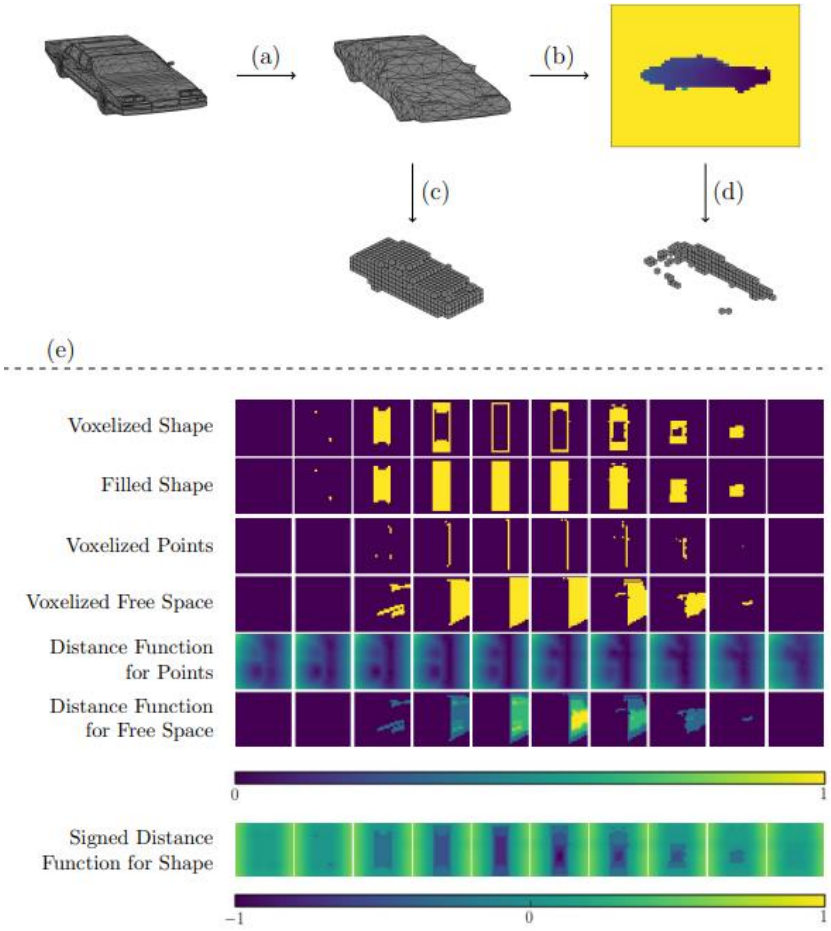

Рисунок 3.1 — Ілюстрація процесу генерації даних на прикладі набору даних ShapeNet набору

3.1.1 Попередня обробка сіток та вокселізація

Вважаємо за краще працювати з водонепроникними сітками; для 3D-кубоїдів можемо контролювати це, оскільки автоматично генеруємо сітки. У випадку з ShapeNet це проблематично. Крім того, ShapeNet може містити дуже складні моделі, де нам потрібно мати справу з 100000 граней. Часто ці моделі містять рівень деталізації, який нас не цікавить, оскільки він буде втрачений під час вокселізації, особливо в низькій роздільній здатності. Тому вирішили обчислити так звану напівопуклу оболонку.

Розглянемо алгоритм, що спочатку вибирає множину точок P із заданої сітки M = (V, F). Опис алгоритму представлений у додатку Д. Потім обчислюється опукла оболонка — стандартна задача обчислювальної геометрії. Щоб зменшити кількість початкових вершин, використовується підхід для пересічення початкової сітки. Опукла оболонка, $M^{(0)}$, потім

ітеративно уточнюється шляхом мінімізації втрат $\mathcal{L}(V^{(t)}) = \sum_{v \in V^{(t)}} \min_{v \in P} \|v - p\|_2^2 + \sum_{(i,j) \in E(V^{(t)})} (\|v_i - p_j\|_2^2 - \mu)$ з використанням градієнтного спуску. Тут μ — середня довжина ребра початкової сітки $M^{(0)}$. Проблема такого формулювання полягає в тому, що на будь-якій ітерації t сітка $M^{(t)}$ може більше не містити множину точок P. У цьому випадку вершини змінюються як $V^{(t)} := (1 + \alpha)V^{(t)}$ так, що $P \subseteq Vol(V^{(t)})$, тобто множина точок P міститься в об'ємі, що охоплюється $V^{(t)}$, і сітка $M^{(t)}$ знову перемальовується. Результати спрощення показано на рисунку 3.2.

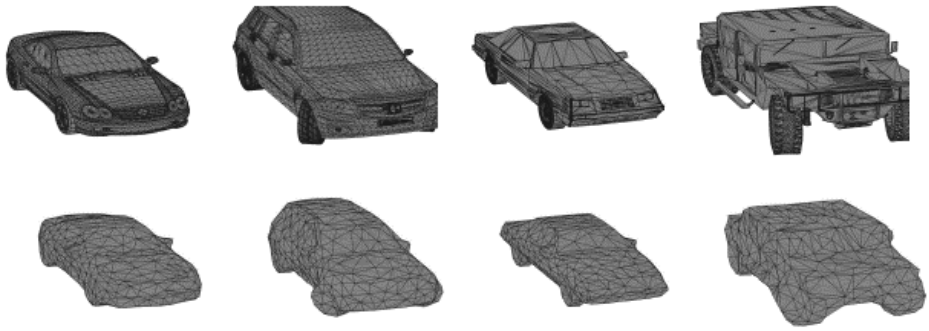

Рисунок 3.2 — Ілюстрація застосованого підходу до спрощення сітки на вибраних вручну прикладах з ShapeNet.

Після отримання простих, водонепроникних сіток виконується вокселізація за допомогою тестів на перетин трикутників і вокселів. На практиці масштабуємо, трансформуємо та розбиваємо всі моделі до $[0, 1]^3$, які потім розбиваємо на вокселі, вирівняні по осях $H \times W \times D$. Для моделей автомобілів слід бути обережним, щоб уникнути перекосу моделей під час масштабування, переміщення та заповнення. Всі осі масштабуються однаково, і можемо розглянути можливість невеликого випадкового обертання, перекладу та масштабування для збільшення даних.

3.1.2 Заповнення,візуалізація сіток та вокселізація спостережень

Для того, щоб отримати правильні сітки заповнення, тобто також ідентифікувати внутрішні вокселі, використовуємо алгоритм заливки/з'єднаних компонент. Оскільки працюємо з водонепроникними

сітками, внутрішні та зовнішні частини фігур чітко розділені. Для низьких роздільних здатностей, наприклад, 32^3, цей підхід працює дуже добре.

Для спрощення ми використовуємо рендерер MatLab на основі OpenGL для отримання зображень глибини з попередньо оброблених сіток. За допомогою параметрів камери, що використовуються OpenGL, пікселі глибинного зображення можуть бути спроектовані у 3D простір. Задавши порогове значення глибини, знаємо, які пікселі відповідають точкам на поверхні сіті.

Керуючи роздільною здатністю глибинного зображення, а також фокусною відстанню, опосередковано контролюємо розрідженість отриманої хмари точок. Використовуючи трасування променів, можемо додатково отримати вільний простір.

Визначення 3.1 — (спрощена) 2D проекційна камера — це кортеж (K, R, t), де $K = \begin{bmatrix} f_u & 0 & u \\ 0 & f_v & v \\ 0 & 0 & 1 \end{bmatrix}$ —це внутрішня матриця камери, $R \in \mathbb{R}^{3*3}$ — матриця обертання, а $t \in \mathbb{R}^3$ — вектор трансляції. Тут f_u та f_v — це фокусні відстані по горизонталі та вертикалі відповідно, а $(u, v)^T$ визначає головну точку 2D зображення — неявно визначаючи також його роздільну здатність, $2u \times 2v$, припускаючи, що $(u, v)^T$ представляє центральний піксель.

Загальна проекційна матриця камери (K, R, t) задається $P = K[R\ t] \in \mathbb{R}^{3*4}$ і визначає проекцію точки $p = (p_1, p_2, p_3, 1)^T$ в однорідних координатах на відповідний піксель

$$x = \begin{bmatrix} \widetilde{x_1} \\ \widetilde{x_3} \\ \widetilde{x_2} \\ \widetilde{x_3} \\ 1 \end{bmatrix} \text{ з } \widetilde{x_1} = Pp.$$

Обернена проекція визначається як її псевдообернена $P^+ = (P^T P)^{-1} P^T \in \mathbb{R}^{4\times3}$. Промінь, що відповідає пікселю x в однорідних координатах, тобто $x = (x_1, x_2, 1)^T$, може бути отриманий як:

$$p = \begin{bmatrix} d * \frac{r_1}{r_3} \\ d * \frac{r_1}{r_3} \\ d \end{bmatrix}.$$

На практиці також можемо працювати в системі координат камери так, що $R = I$ і $t = 0$. В результаті матриця зворотної проекції спрощується до $P^+ = K^{-1}$. Потім знімаємо фігури при різних поворотах навколо (вертикальної) осі висоти — відповідно до різних точок зору камери. Після зворотного проектування можна використовувати тести перетину променів і вокселів, а зайняті вокселі можна визначити за допомогою тестів перетину точок і вокселів. Обертаємо і переводимо спостереження до $[0, 1]^3$ і використовуємо той самий поділ на вокселі, вирівняні по осях $H \times W \times D$, як і раніше. Щоб отримати вокселі вільного простору, розглядаємо тільки промені до точок на відповідній фігурі, бо не враховуємо промені, що відповідають точкам на фоні. Це розумно, оскільки промені від точок фону не можуть бути надійно використані на реальних даних, наприклад, на KITTI.

3.1.3 Шум

Щоб імітувати реальні умови, ми хочемо ввести штучний шум. Ми вручну перевірили багато зразків з набору даних KITTI і вирішили визначити два параметри шуму, λ_{hit} і θ_{ignore}.

Перший визначає експоненціальний розподіл:

Визначення 3.2 — нехай $\varepsilon \in \mathbb{R}$, $\varepsilon \geq 0$, є випадковою величиною. Тоді вона розподілена за експоненціальним законом, тобто $\varepsilon \sim \text{Exp}(\varepsilon; \lambda)$, з параметром λ, якщо функція густини ймовірності задана у вигляді

$$p(\varepsilon) = \lambda \exp(-\lambda\varepsilon).$$

Використовуючи вибірку з оберненим перетворенням, отримуємо вибірки з цього розподілу, використовуючи $u = U(0,)$ та

$$\varepsilon = \frac{-\ln u}{\lambda}.$$

Для кожного пікселя зображення глибини ми вибираємо значення похибки $\varepsilon \sim Exp(\varepsilon; \lambda_{hit})$ з експоненціального розподілу і додаємо це значення до фактичного значення глибини. Це розумно, оскільки завжди буде невід'ємним, а $p(\varepsilon)$ зменшується експоненціально зі зростанням . Ймовірність θ_{ignore} визначає, наскільки ймовірно, що спостереження буде проігноровано. У цьому випадку значення глибини відповідного пікселя встановлюється на максимальну глибину.

3.2 KITTI

Кожна наземна 3D-обмежувальна рамка надається у вигляді її центру, тобто переведення з центру датчика, розмірів по ширині, висоті та глибині, а також кута повороту вздовж (вертикальної) осі висоти. Для вокселізації обмежувальні рамки обертаються для вирівнювання по осі, а потім масштабуються до одиничного куба, тобто $[0, 1]^3$. Знову ж таки, ми переконуємося, що всі осі масштабуються однаково, щоб спостереження не були перекошені.

Для вокселізації вільного простору використовуємо той самий підхід, що й раніше, тобто трасування променів. Знову ж таки, розглядаємо частковий вільний простір, оскільки датчик Velodyne має труднощі з відбиваючими та прозорими поверхнями. Зокрема, виявили, що багато променів проходять через анотовані автомобілі та точки потрапляння на задньому плані. Враховуючи частковий вільний простір, цей ефект зменшується шляхом трасування променів, які відповідають точкам у межах 3D-обмеження. Потім обчислюємо перетини променів з усіма вокселями, використовуючи той самий поділ на вокселі, що й раніше, щоб уникнути помилок.

Використання всіх вокселізованих спостережень для вивчення завершення форми не є реалістичним — особливо тому, що багато

спостережень містять лише дуже мало спостережуваних точок і помилковий вільний простір.

Тому відфільтрували вокселізовані спостереження, щоб отримати простіший набір даних. Вимагаємо, щоб спостерігалося щонайменше $n_{1,min}$ точок і $n_{0,min}$ вокселів відповідали вільному простору:$\sum_{i=1}^{R} \mathbb{1}[x_i = 1] \underset{!}{\rightarrow} n_{1,min}$ та $\sum_{i=1}^{R} \mathbb{1}[x_i = 0] \underset{!}{\rightarrow} n_{0,min}$. І також вимагаємо, щоб відстань від 3D-обмеження до датчика Velodyne була меншою за t_{max}. На практиці, ці три обмеження гарантують, що отриманий набір даних є керованим для вирішення в ході цієї дипломної роботи. Більш складні завдання можна вирішити, зменшивши $n_{1,min}$ і $n_{0,min}$ та збільшивши t_{max} відповідно.

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОБОТИ МОДЕЛЕЙ ЗАПОВНЕННЯ ФІГУР

У цьому розділі представлено експериментальні результати щодо підходів до заповнення фігур, розглянутих у розділі 2, на всіх наборах даних, представлених у розділі 3. Не всі з розглянутих формулювань працюють однаково добре. Спочатку провели експерименти на нашому синтетичному наборі даних 2D прямокутників, див. додаток Е, і виявили, що метод максимальної правдоподібності (ML) працює не дуже добре. Для наочності наводимо лише експерименти на наших синтетичних 3D-даних, тобто кубоїдах та автомобілях з ShapeNet, а також на реальних даних, наприклад, KITTI. Загалом, розглядаємо варіаційні автокодери (VAE) для навчання попередніх форм і амортизованої максимальної правдоподібності (AML), а також розширені варіаційні автокодери (EVAE) для завершення форми. Спочатку обговоримо експериментальну установку, а потім розглянемо експерименти з 3D-кубоїдами та автомобілями, а також з KITTI.

4.1 Налаштування для експериментів: архітектура, навчання та оцінка

Реалізували всі розглянуті підходи у фреймворку глибокого навчання Torch. Реалізації можна розглядати як прототипи; Щоправда, не оптимізували їх щодо часу виконання або споживання пам'яті. Попередню обробку та генерацію даних, а також оцінювання виконували мовами Python та C++. Для всіх експериментів використовуємо роздільну здатність $H \times W \times D = 32^3$ і припускаємо рівномірний розподіл $[0, 1]$ на 32^3 вокселі для вокселізації. Як уже зазначалося, отримуємо знакові функції відстаней з відповідних сіток заповнення за допомогою перетворення відстаней. Детально про штучно доданий шум, а також доповнення даних обговорюється у відповідних розділах. Далі ми коротко опишемо використані архітектури та метрики оцінювання.

Архітектури, що використовуються для наших експериментів, є простими. Хоча ми експериментували з глибшими та складнішими

архітектурами, зокрема зі з'єднаннями з пропуском, залишковими одиницями та архітектурами, що базуються на початкових даних, ці зміни не мали значного впливу. Кодер і декодер складаються з чотирьох етапів згорткових шарів, включаючи пакетну нормалізацію, нелінійність ReLU та максимальне об'єднання/дискретизацію за найближчим сусідом. Збільшуємо ширину мережі (тобто кількість каналів) при зменшенні просторового розміру карт ознак і використовуємо $3 \times 3 \times 3$ ядра згортки з нульовим заповненням і вікнами $2 \times 2 \times 2$, що не перекриваються, для максимального об'єднання і дискретизації за методом найближчого сусіда. Для $32 \times 32 \times 32$, таким чином, зменшуємо просторовий розмір до $2 \times 2 \times 2$ перед обчисленням прихованого коду. Як для AML, так і для EVAE нові кодери, тобто z(x;w) та q(z|x), навчаються з нуля, але слідують архітектурі попередньої форми. У цьому випадку відповідна генеративна модель p(y|z) завжди залишається фіксованою. Для AML додатково видаляємо повністю зв'язаний шар, який передбачає дисперсію прихованого коду, щоб отримати детермінований кодер. Для EVAE додатковий декодер p(x|y) складається з семи етапів згортки, включаючи пакетну нормалізацію та нелінійності ReLU. При прогнозуванні заповнюваності використовуємо сигмоїдну нелінійність; для прогнозування знакових функцій відстані ми використовуємо тотожність, тобто без нелінійності.

Для оцінки ми використовуємо абсолютну похибку між прогнозом та істиною для обох представлень, тобто для функцій заповнюваності та підписаних відстаней. Для прогнозу $y \in \mathbb{R}^{H*W*D}$ та істини $y^* \in \mathbb{R}^{H*W*D}$ ми усереднюємо за всіма просторовими вимірами:

$$\mathrm{Abs}(y, y^*) = \frac{1}{HWD} \sum_{i_1=1}^{H} \sum_{i_2=1}^{W} \sum_{i_3=1}^{D} |y_1 - y_i^*|.$$

Завжди повідомляємо середнє значення на валідаційному наборі (або на партіях під час навчання). Додатково використовуємо абсолютну похибку

після встановлення порогових значень для прогнозованих форм, тобто після отримання належних сіток зайнятості. Для сіток зайнятості встановлюємо поріг прогнозованої ймовірності зайнятості на рівні 0,5, а для функцій знакових відстаней — на рівні 0 (тут від'ємні значення відповідають зайнятим вокселям). Ці порогові значення також використовуються для наших 3D-візуалізацій. Абсолютну похибку після порогового значення ми позначаємо як $\text{Abs}_{\text{thresh}}$. Загалом, абсолютну похибку легко інтерпретувати, наприклад, вона забезпечує чітку нижню межу (яка дорівнює $\text{Abs} \geq 0$) і є порівнянною між наборами даних і методами.

Розглядаємо від'ємну логістичну ймовірність як міру. Однак дійшли висновку, що від'ємна логістична вірогідність не підходить для оцінювання. По-перше, від'ємну логістичну ймовірність важче інтерпретувати, оскільки вона сильно залежить від моделі (наприклад, VAE або EVAE). Зокрема, окрім втрат на реконструкцію, всі моделі також включають (можливо, зважені) попередні. Крім того, для EVAE втрати при реконструкції не є метою, яку насправді намагаємося оптимізувати для завершення форми. По-друге, для всіх наборів даних від'ємна лог-правдоподібність залежить від кількості спостережуваних вокселів. Таким чином, ефективність запропонованих підходів до завершення фігури не можна порівнювати з ефективністю реконструкції попередньої фігури, яка була б природною базовою лінією. По-третє, від'ємна логарифмічна вірогідність для спостережень Бернуллі за своєю природою зміщена в бік "непевних" прогнозів; це означає, що від'ємна логарифмічна вірогідність віддає перевагу непевним прогнозам, а не дуже певним прогнозам з невеликою кількістю помилок. Це є розумним під час навчання, коли явно хочемо змоделювати невизначеність, але перешкоджає справедливому оцінюванню. Загалом,вирішили не повідомляти про від'ємні логічні ймовірності.

4.2 3D приклад

Почнемо з експериментів на нашому наборі даних синтетичних 3D-кубоїдів. Тут можемо представити як кількісні, так і якісні результати на

контрольованому, простому наборі даних. Для цього створили набори даних трьох рівнів складності, дотримуючись процедури, описаної в розділі 3: легкий, складний і середній, з деталями в таблиці 4.1. Зі зростанням складності зменшується кількість спостережень і збільшується рівень шуму, що лежить в основі даних. Складний випадок має відображати реальні умови, які можна знайти на KITTI. У таблиці 4.1 додатково наводимо деякі основні статистичні дані, наприклад, відсоток зайнятих вокселів, щоб отримати уявлення про те, як працюватимуть тривіальні прогнози, або відсоток спостережуваних і вільних вокселів, щоб показати, який рівень спостереження є доступним.

Таблиця 4.1 — Огляд згенерованих наборів даних 3D-кубоїдів

	3D простий	3D середній	3D складний
Розмір тренінгу (попередній/послідовний)	10000/10000		
Розмір валідації	1000		
Роздільна здатність H × W × D	32 × 32 × 32		
Роздільна здатність 2u × 2v	48 × 64	24 × 32	
Шум λ_{hit}	0	50	50
Шум θ_{ignore}	0	0	0.1
Спостережувані вокселі	1.43%	0.53%	0.48%
Вокселі вільного простору	10.73%	7.07%	8.32%
Зайняті вокселі	16.24%	16.17%	16.21%

Почнемо з обговорення форми VAE перед тим, як перейти до проблеми заповнення форми. Оскільки виявили, що ML погано працює у 2D випадку, ми виключаємо експерименти на 3D даних.

4.2.1 Попередній вибір форми

Для попереднього навчання розмір Q латентного простору має вирішальне значення. Виявлено, що на практиці відповідний розмір можна

визначити, спостерігаючи за ходом навчання VAE для різних розмірів Q. Для подальшого обговорення визначили Q = 15 як відповідний, і відповідні криві навчання показані на рисунку 4.1.

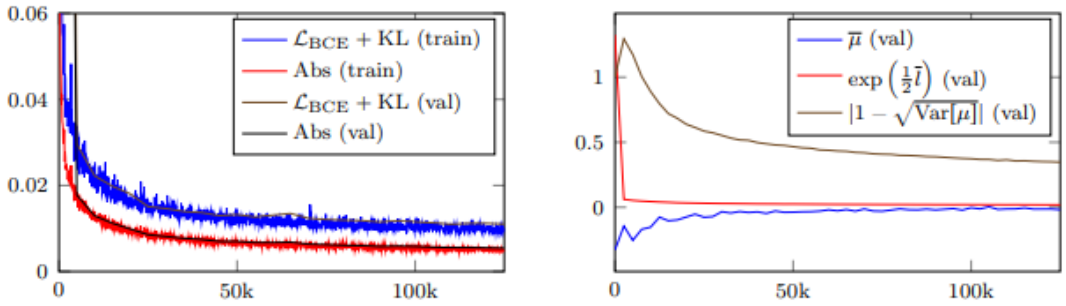

Рисунок 4.1 — Навчальні криві для VAE з Q = 15, навченого на наборі даних 3D кубоїдів

Важливим критерієм для оцінки Q є латентний простір, тобто чи збігається навчання з латентним простором. Зазвичай на це вказують низькі прогнозовані лог-варіації та статистика прогнозованих середніх, яка повільно нагадує одиничний гаусівський розподіл (тобто нульове середнє та одинична дисперсія). Крім того, отримана помилка реконструкції може бути використана як індикатор. Наприклад, для Q = 5 абсолютна похибка не опускається нижче Abs = 0.04. Враховуючи, що зайнято лише 16,2% вокселів (див. табл. 4.1), ця похибка все ще є дуже великою. Для Q = 15 і вище, навпаки, похибка зменшується до Abs = 0.0054 або нижче. Нарешті, також розглянемо випадкові вибірки з використанням генеративної моделі. Як видно з рисунків 4.2 і 4.3, випадкові вибірки мають відповідний вигляд і здебільшого нагадують кубоїди. Зауважмо, що через обертання кубоїди можуть виглядати не прямокутними, якщо показувати горизонтальні зрізи відповідних об'ємів. Загалом, не використали всі можливості щодо гіперпараметрів та часу навчання, але задоволені отриманими результатами при використанні Q = 15.

На 2D прикладах зрозуміли, що прогнозування як заповнюваності, так і знакових функцій відстані є більш ефективним порівняно з прогнозуванням лише знакових функцій відстані. Для 3D зробили аналогічне спостереження і показали якісні результати на рисунку 4.2. Знову ж таки, використовуємо

Q = 15, і досягаємо абсолютної похибки Abs = 0.0064 для заповнюваності та Abs = 0.071 для знакових функцій відстаней. Після встановлення порогу для прогнозованих репрезентацій для отримання сіток заповнюваності (на рівні 0,5 для ймовірностей заповнюваності та на рівні 0 для знакових функцій відстаней), абсолютна похибка падає до Abs_{thresh} = 0.0045 та Abs_{thresh} = 0.0053, відповідно.

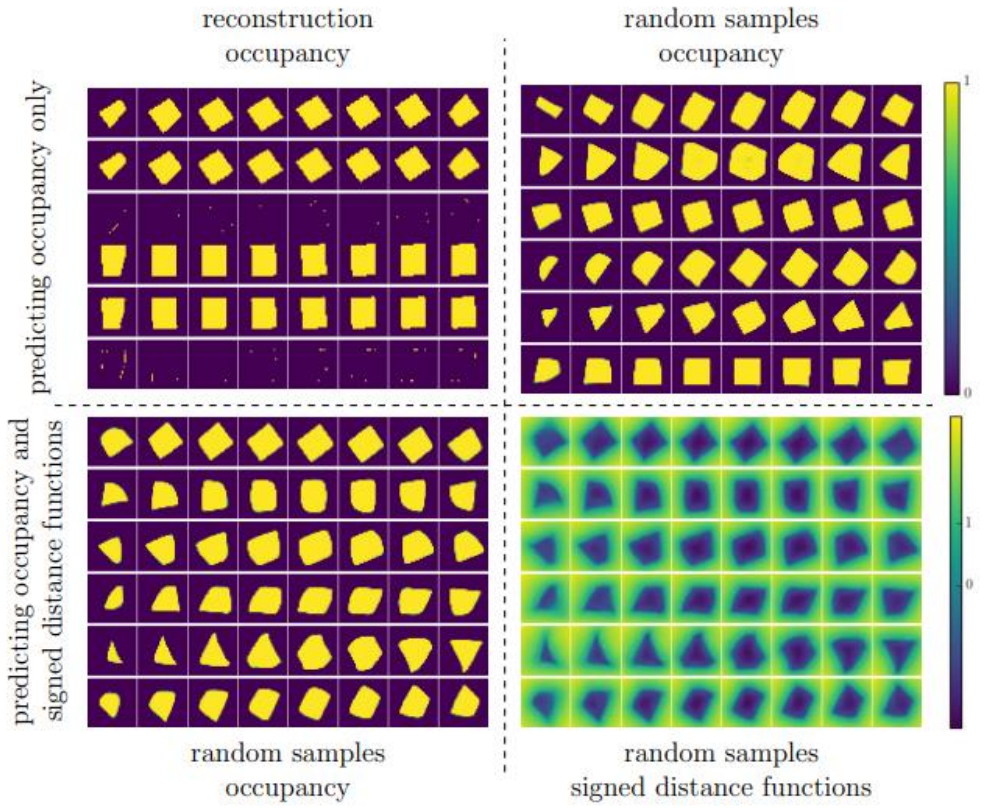

Рисунок 4.2 — Якісні результати для навченого попередника форми VAE з Q = 15 на наборі даних 3D кубоїдів

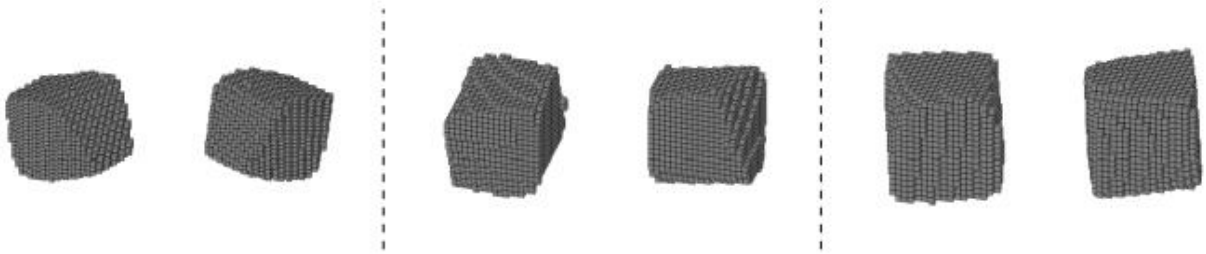

Рисунок 4.3 — Три випадкові приклади форми VAE з Q = 15 на 3D кубоїдах набір даних, навчений лише на заповнюваності

Це означає, що обидва представлення можуть бути використані для отримання сіток зайнятості з низькою похибкою. Випадкові вибірки також виглядають придатними і чітко зображують кубоїди в більшості випадків; однак, вважаємо, що ці вибірки є дещо менш чіткими порівняно з прогнозуванням лише заповнюваності.

4.2.2 Амортизована максимальна правдоподібність

Для AML в основному наводимо експерименти на помірних(moderate) і складних(hard) завданнях, оскільки виявили, що AML дуже добре працює на помірних завданнях. Перш за все, зазначимо, що вага k на від'ємній лог-вірогідності попередника, тобто $\ln p(z)$, має вирішальне значення для успішного навчання. На рисунку 4.4 показано криві навчання для помірного випадку при використанні k = 15.

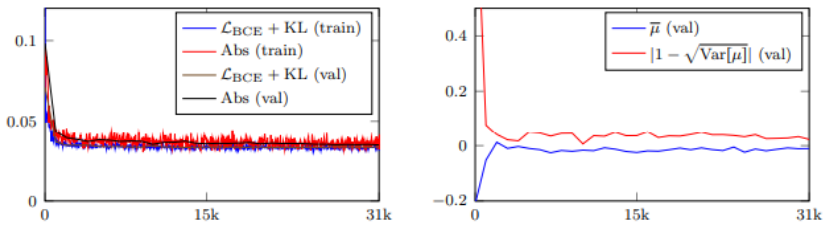

Рисунок 4.4 — Криві навчання для AML з використанням лише заповнюваності та попереднього VAE з Q = 15 на наборі даних 3D кубоїдів, зокрема, для помірного випадку

Загалом мережа досягає абсолютної похибки Abs = 0.036. Якщо вага k буде недостатньо великою, то мережа швидко відхилиться від одиничного гауссівського попередника. Це можна спостерігати при моніторингу латентного простору, тобто спостережувані статистики $\sqrt{\text{Var}[\mu]}$ і $\bar{\mu}$ значно відхиляються від одиничного гауссового попереднього. Якщо k вибрано надто великим, то попереднє "руйнується", тобто $\sqrt{\text{Var}[\mu]}$ наближається до нуля. Якщо ж зважено правильно, то пріоритет дбає про дотримання одиничного значення Гаусса на перших кількох ітераціях. Виявили, що узгодження з попередніми даними стає важливим, оскільки велика частина навчання

відбувається на ранніх ітераціях. Це також видно на рисунку 4.4; здається, що AML потребує лише кілька епох, щоб "навчитися" робити висновок. Експериментально виявили, що k = 15 добре працює для помірного випадку, однак для складного випадку необхідно k = 30. На жаль, не знайшли жодного емпіричного правила для встановлення k, тому довелося вдатися до методу проб і помилок. Загалом, вважаємо, що навчання AML є складним у 3D; експерименти з гіперпараметрами стають все більш важливими.

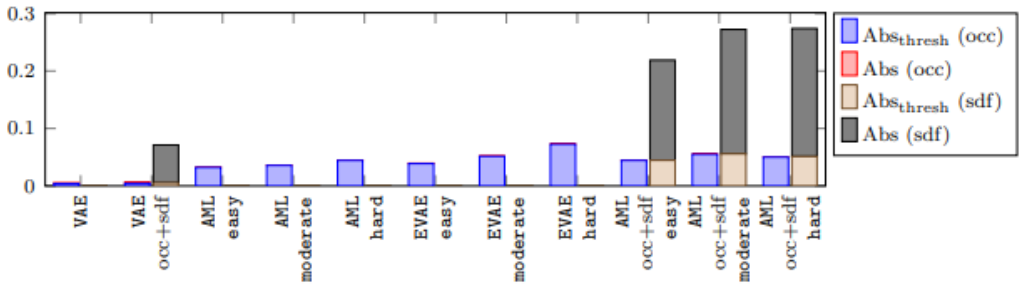

Рисунок 4.5 — Абсолютна похибка Abs та її порогове значення $\text{Abs}_{\text{thresh}}$

На рисунку 4.5 показано порівняння попередніх результатів з отриманими абсолютними похибками для легких, середніх та важких завдань. Якісні результати для випадку середньої складності можна знайти на рисунку 4.6. AML працює досить добре на помірній складності. Однак виявили, що на ефективність сильно впливають спостереження за шумним вільним простором у важкому випадку. Особливо проблеми викликають ігноровані промені. Після ретельнішого дослідження вдалися до простого підходу, щоб уникнути цих труднощів — зважуємо спостереження $x_i = 0$, що відповідають вільному простору, на

$$\rho_i = \frac{\sum_{m=1}^{M} y_{m,i}}{m},\ Y = \{y_m\}_{m=1}^{M} \subseteq \{0,1\}^R. \tag{4.1}$$

Вагу ρ_i можна інтерпретувати як статистику вільного простору, тобто ймовірність того, що воксель і не буде зайнятий протягом попередньої навчальної вибірки. Ця концепція також проілюстрована на рисунку 4.6. Додатково експериментували з використанням $\rho_i^{\lambda}, \lambda \in (0,1)$ і визначили, що

$\lambda = 0.5$ працює добре. В результаті ми змогли отримати майже однакову продуктивність на середніх і важких рівнях складності. В цілому, це обговорення також показує вплив індивідуального зважування вокселів для того, щоб впоратися із зашумленими спостереженнями.

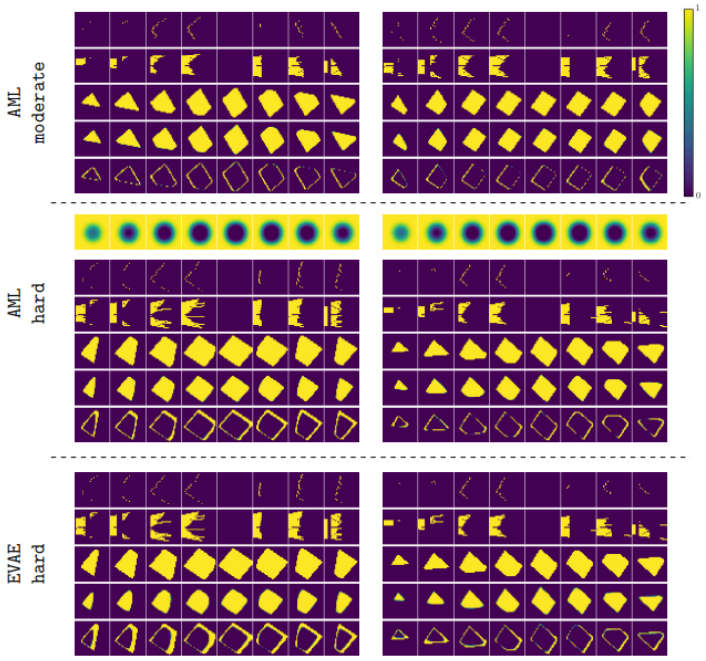

Рисунок 4.6 — Якісні результати для AML на наборі даних 3D кубоїдів з попереднім VAE та Q = 15

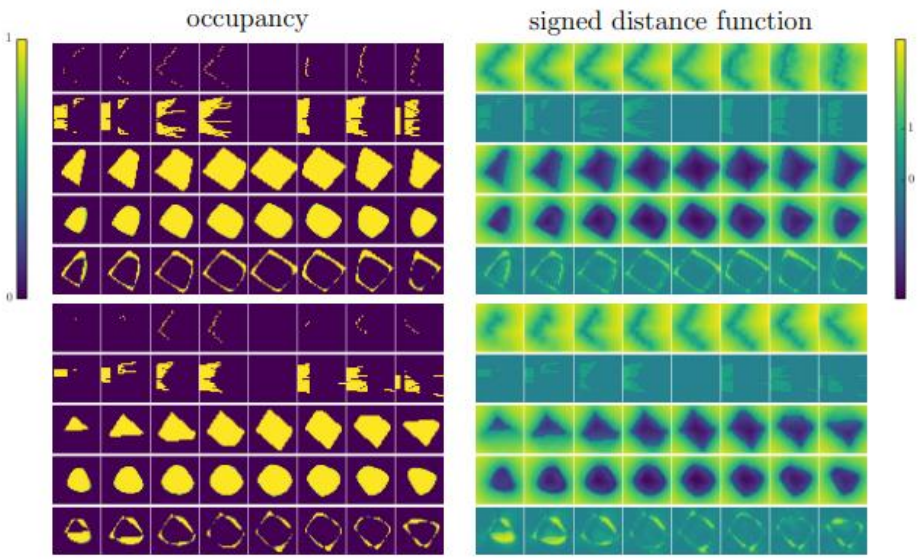

Рисунок 4.7 — Якісні результати для AML з використанням попереднього VAE з Q = 15, навченого на обох функціях заповнюваності та знакових функцій відстані

На рисунку 4.5 також показано результати, отримані при прогнозуванні як заповнюваності, так і знакових функцій відстані. Відповідні якісні результати для жорсткого випадку можна знайти на рисунку 4.7. Виявили, що і в помірному, і в жорсткому випадку передбачені фігури трохи більші за цільові фігури.

Зокрема, модель, здається, вдається до "стандартних", краплеподібних форм, які є близькими до середньої форми. Вже знаємо, що вивчати та прогнозувати знакові функції відстані важче, ніж просто заповнюваність. Прогнози, схожі на краплі, можна також пояснити занадто великою вагою k, що неявно обмежує модель формами, близькими до середньої. На жаль, широке дослідження та налаштування гіперпараметрів не було можливим в рамках обмеженого часу цієї роботи. Приклад зі згенерованих наборів даних 3D-кубоїдів показано в додатку E.

4.2.3 Розширений варіаційний автокодер

Для EVAE отримуємо подібні результати, як показано вище, див. рисунок 4.5. Наприклад, у легкому випадку досягається абсолютна похибка Abs = 0,034 — це лише трохи гірше, ніж AML. Для помірного та складного випадків досягаються похибки ~ 0,042 та ~ 0,057. Це гірше, ніж AML, але ми також відзначаємо, що не витратили стільки часу на налаштування гіперпараметрів і, можливо, недооцінили необхідний час навчання, що є більш актуальним для EVAE, оскільки він включає значно більше параметрів. Також відзначаємо, що спостережувана продуктивність контрастує з 2D випадком, EVAE дещо випереджає AML, що також вказує на важливість часу навчання — у 2D випадку, здається, виділили достатньо часу для навчання. Для складного випадку приклади можна знайти на рисунку 4.6. Ми також бачимо, що AML та EVAE дають дуже схожі результати, за винятком того, що EVAE дещо недооцінює справжній розмір кубоїдів. Втім, це не дивно, оскільки обидва підходи оптимізують схожу задачу, тільки у випадку EVAE це відбувається за допомогою розбіжності Кульбака-Лейблера. Маємо намір провести подальші експерименти щодо EVAE у майбутній

роботі, але, враховуючи надані докази, віддаємо перевагу AML через менший час навчання та дещо кращу продуктивність.

4.3 ShapeNet

ShapeNet — це наш перший набір даних, що містить реалістичні об'єкти, зокрема автомобілі. Пізніше ShapeNet також буде використовуватися для тренування форми перед завершенням на KITTI. Спочатку ми вручну відкинули 262 з 3514 спрощених сітей, які не могли бути автоматично масштабовані та повернуті до $(0, 1)^3$ для подальшої обробки. Розділили моделі, що залишилися, на дві навчальні множини — для попереднього та виводу — та валідаційну множину, що відповідає часткам 0,45 : 0.45 : 0.1. До кожної моделі застосовуємо сім випадкових перетворень, включаючи незначне масштабування, обертання та переклад сіток; крім того, перевертаємо кожен варіант. В результаті отримали набори даних, наведені в таблиці 4.2.

4.3.1 Попередній вибір форми

Для попереднього VAE знову використовуємо Q = 15. З навчання можна зробити висновок, що набір даних ShapeNet дещо легше навчити. Зокрема, дивлячись на рисунок 4.8, ми помічаємо, що і реконструкція, і латентний простір вивчаються швидше.

Таблиця 4.2 — Огляд згенерованих наборів даних ShapeNet

	3D простий	3D середній	3D складний
Розмір тренінгу (попередній/послідовний)	20496/20496		
Розмір валідації	4550		
Роздільна здатність H × W × D	32 × 32 × 32		
Роздільна здатність 2u × 2v	48 × 64	24 × 32	
Шум λ_{hit}	0	50	50
Шум θ_{ignore}	0	0	0.075
Спостережувані вокселі	0.62%	0.31%	0.304%
Вокселі вільного простору	3.91%	3.75%	4.37%
Зайняті вокселі	5.54%	5.51%	5.52%

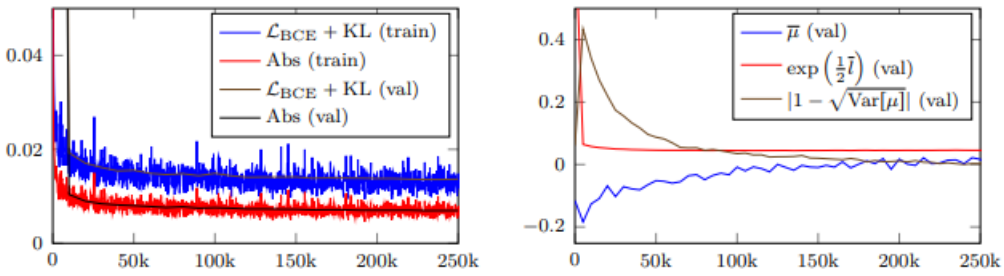

Рисунок 4.8 — Криві навчання для VAE з Q = 15, який навчався тільки на заповнюваності на наборі даних ShapeNet

Однак важко сказати, чи є ShapeNet за своєю суттю простішим за набір даних 3D кубоїдів. З одного боку, ми розглядаємо лише незначні обертання навколо всіх осей; з іншого боку, автомобілі демонструють більшу внутрішньокласову варіацію порівняно з кубоїдами. Загалом, модель досягає абсолютної похибки Abs = 0.0073 та $\text{Abs}_{\text{thresh}}$ = 0.0048 після порогового значення. Однак, слід зазначити, що лише ~ 5,54% вокселів зайняті на першому місці (див. таблицю 4.2). Ми також помічаємо, що модель є менш визначеною з огляду на випадкові вибірки на рисунках 4.9 та 4.10, тобто прогнози виглядають менш чіткими. Особливо щодо коліс та даху, модель має труднощі.

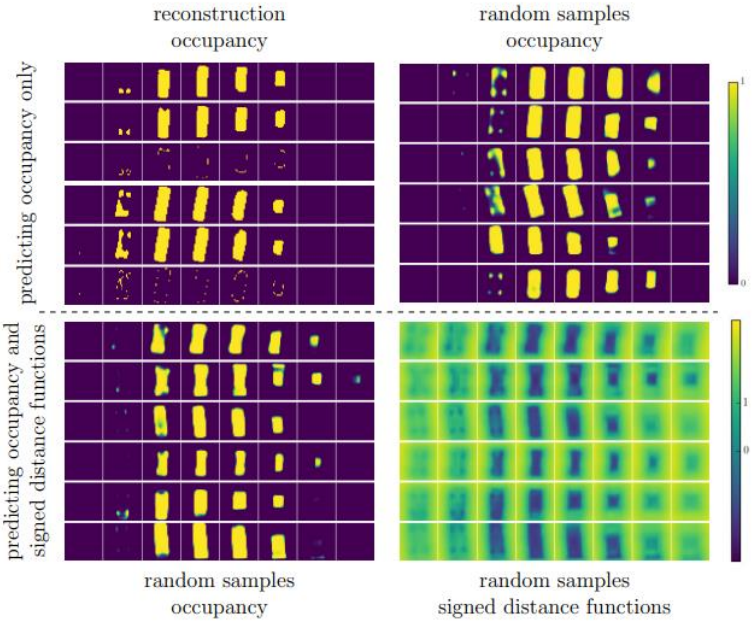

Рисунок 4.9 — Якісні результати з урахуванням реконструкції та випадкових вибірок для попереднього VAE, Q = 15

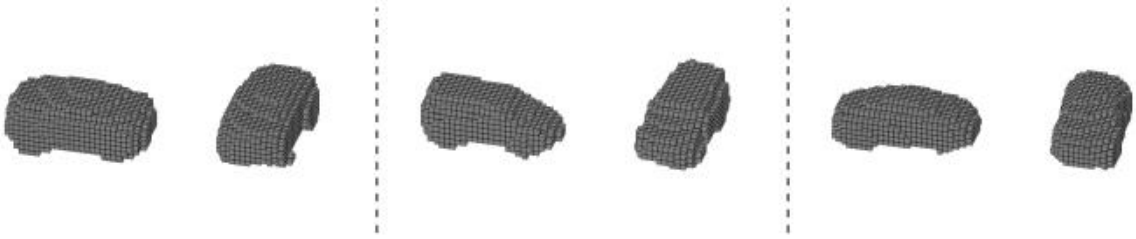

Рисунок 4.10 — 3D-візуалізації випадкових вибірок, отриманих через VAE, попередньо навченого з Q = 15 і заповненням тільки на наборі даних ShapeNet

Загалом, ми задоволені отриманими результатами — для експериментів на KITTI ми також матимемо більше навчальних даних.

У ShapeNet також навчили попередню модель працювати як з функціями заповнення, так і зі знаковими функціями відстаней. Впершетакож продемонстрували, чому віддаємо перевагу роботі зі знаковими функціями відстані: можна отримати трикутні сітки з точністю до суб-вокселя, наприклад, за допомогою алгоритму маршируючих кубів . Знову ж таки, використовуємо $Q = 15$ для досягнення порогової абсолютної похибки $Abs_{thresh} = 0.0048$ (заповнюваність) і $Abs_{thresh} = 0.0053$ (функція відстані зі знаком), відповідно. Сітки, що відповідають випадковим вибіркам, показано на рисунку 4.11.

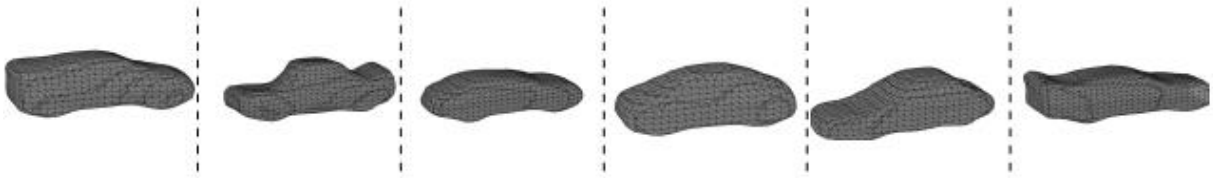

Рисунок 4.11 — 3D-візуалізації сітчастих випадкових вибірок з VAE, навчених як на функціях заповнення, так і на знакових функціях відстаней

Крім того, на рисунку 4.9 показуємо обидві модальності, щоб проілюструвати пов'язану з ними невизначеність. Загалом, вважаємо, що випадкові вибірки є доречними. Хоча деякі випадкові вибірки виглядають досить дивно, не важко уявити, що на них зображені автомобілі. Підкреслімо, що сітки виглядають досить гладкими, хоча підписані функції відстані, на яких

навчалася модель, спочатку були отримані з сіток заповнюваності. Припускаємо, що це результат імовірнісного формулювання, тобто навчання VAE. Зокрема, кодер прогнозує гаусівський розподіл, де всі вибірки повинні призвести до правильної реконструкції. Це змушує модель плавно інтерполювати між фігурами.

4.3.2 Амортизована максимальна правдоподібність

Для AML дотримуємося процедури набору даних 3D-кубоїдів; на рисунку 9.12 показуємо кількісні результати для легкого, помірного та важкого випадків. Знову ж таки, важливо пам'ятати, що зайнято лише приблизно 5,54% вокселів. Таким чином, результати для середнього та важкого випадків виглядають досить погано: Abs = 0,023 та Abs = 0,027 відповідно.

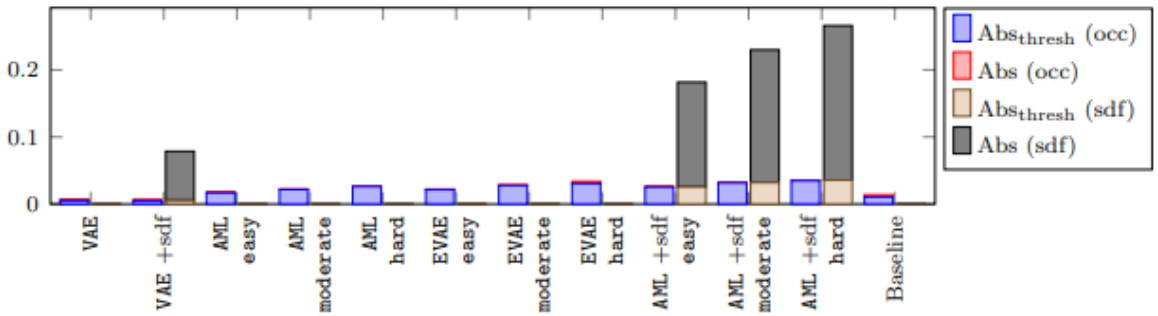

Рисунок 4.12 — Абсолютна похибка Abs та її порогове значення Abs_{thresh}

Це також можна спостерігати при розгляді якісних результатів на рисунках 4.13 і 4.14, які показують результати для важкого випадку.

Виявили, що в багатьох випадках прогнози є дуже невизначеними. При розгляді 3D-візуалізації це ще більше підкреслюється через порогове значення. Для помірного випадку, навпаки, результати в багатьох випадках виглядають більш правдоподібними. Не впевнені, чи можна пояснити ці спостереження попередньою моделлю (яка, наприклад, могла бути недостатньо навченою), чи варіаціями в наборі даних, які дозволяють використовувати ці моделі. Також може бути корисним посилити вимоги

щодо від'ємної лог-вірогідності для попередньої p(z) — це припускає, що латентний простір є "надійнішим" близько до 0, ніж до хвоста гаусівського закону. Це може запобігти вивченню малоймовірних форм і потенційно зменшити вплив вагової ініціалізації та стохастичного навчання в перші кілька епох. Загалом, ми бачимо потенціал для покращення шляхом налаштування гіперпараметрів та більш тривалого навчання.

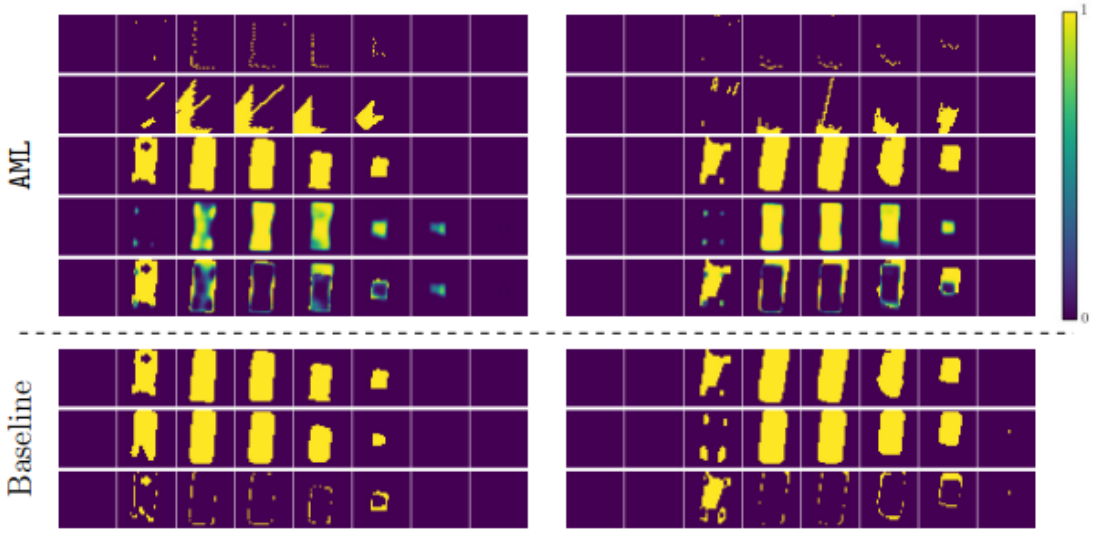

Рисунок 4.13 — Якісні результати AML для важкої складності набору даних ShapeNet

При прогнозуванні як заповнюваності, так і знакових функцій відстані, не можемо досягти продуктивності, порівнянної з випадком, коли прогнозується лише заповнюваність.

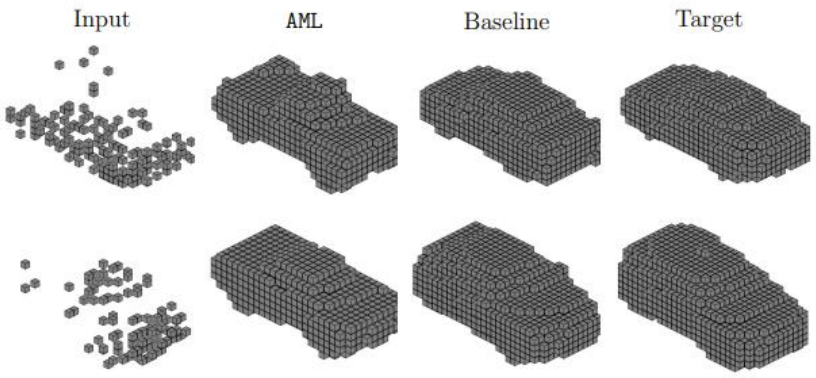

Рисунок 4.14 — 3D-візуалізації для порівняння AML та контрольованої базової лінії за складністю набору даних ShapeNet

Особливо в помірних і важких випадках ефективність значно падає від Abs = 0,026 до Abs = 0,032 або вище (для заповнюваності). Не можемо сказати, чи це є недоліком знакових функцій відстані як модальності, чи це пов'язано з попередньою формою, чи з процедурою навчання. На відміну від випадку з заповнюваністю, не витратили так багато часу на налаштування параметрів. Точне порівняння можна знайти на рисунку 4.12, а якісні результати показуємо на рисунках 4.15 та 4.16.

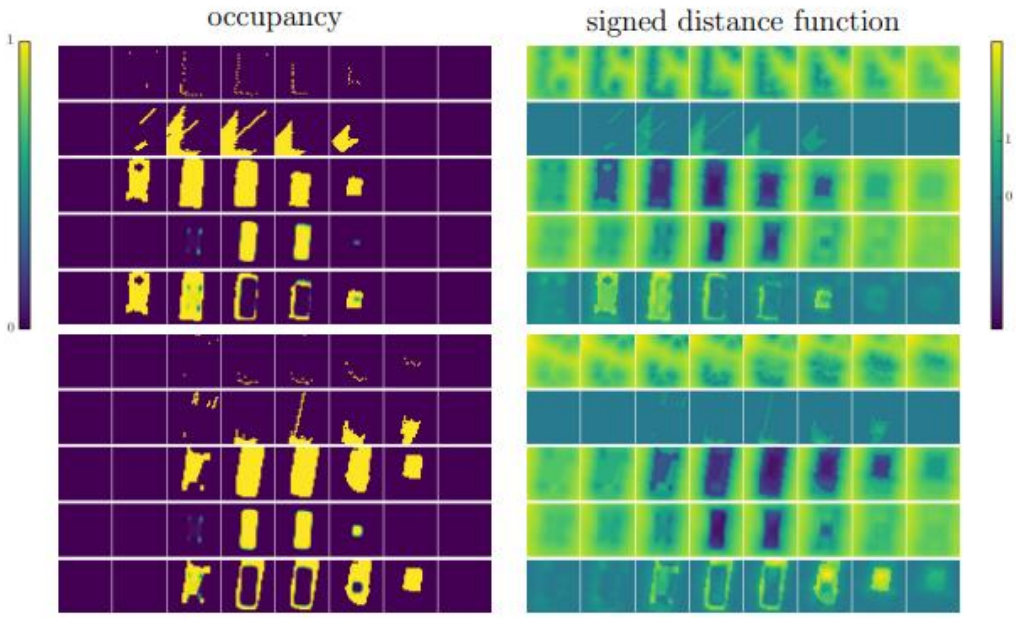

Рисунок 4.15 — Якісні результати для AML, використовуючи обидва розміщення та підписаних функцій відстаней

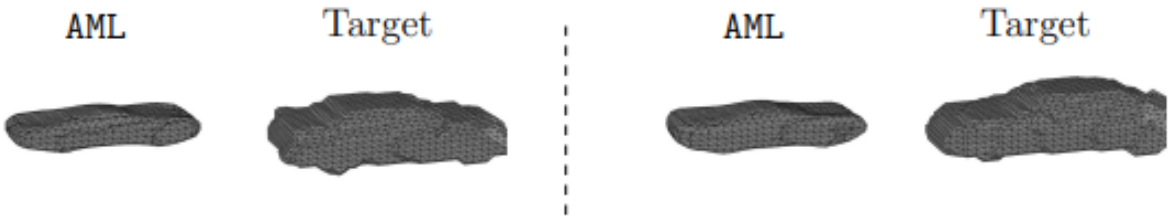

Рисунок 4.16 — Якісні результати для AML після використання маркізних кубів,щоб отримати сітки з прогнозів і цілей

Розглядаючи якісні результати, ммможемо, однак, оцінити ефект згладжування на попередніх; сітчасті прогнози виглядають дуже привабливо — на відміну від сітчастих цілей. Тут стає очевидним, що наші підписані

функції відстані отримані з сіток заповнюваності. Також бачимо падіння продуктивності; деякі прогнози не відповідають цілям так добре, як раніше; особливо тому, що AML постійно недооцінює розмір автомобілів. Приклад зі згенерованих наборів даних 3D-кубоїдів наведений в додатку Ж. Проблеми з прогнозуванням знакових функцій відстані, схоже, є однаковими для всіх наборів даних і моделей; загалом, вважаємо, що альтернативне представлення може бути легшим для вивчення, наприклад, нормалізоване, дискретизоване або просто заміна логарифму.

4.3.3 Розширений варіаційний автокодер

Хоча EVAE показав дещо гірші результати, ніж AML на наборі даних 3D кубоїдів, ми все одно включили кількісні результати на рисунку 4.12. На ShapeNet різниця в продуктивності між AML та EVAE стає більш помітною. Однак також хочемо зазначити, що, знову ж таки, не доклали стільки зусиль до налаштування параметрів і навчання порівняно з AML. Зокрема, підозрюємо, що вага на дивергенції Кульбака-Лейблера може мати суттєве значення. На жаль, не змогли дослідити цю проблему глибше. Тим не менш, показали, що фреймворк також застосовний до реальних об'єктів — навіть якщо тільки простий випадок призводить до відповідної продуктивності.

4.3.4 Контрольована базова лінія

Для ShapeNet також підготували контрольовану базову лінію. Для справедливого порівняння ми використали архітектуру попередня форми з Q = 15, щоб вивчити відображення $x_n \mapsto y_n^*$ безпосередньо з синтетичних даних. Не помітили суттєвої різниці між навчанням на тій самій архітектурі та вилученням розбіжності Кульбака-Лейблера і відповідного шару репараметризації; для справедливості ми дотримувалися першого підходу. Розглянули лише важкий випадок. Пізніше також оцінимо, наскільки добре вивчена модель узагальнюється на KITTI. Рисунок 4.12 показує, що супервізія здатна наблизитися до результатів реконструкції попередньої форми з абсолютною похибкою приблизно Abs = 0.014, тим самим перевершуючи всі

інші представлені підходи для завершення форми. Якісні результати показано на рисунках 4.13 та 4.14 у порівнянні з AML. Загалом, контрольована базова лінія потенційно може також перевершити AML на KITTI, враховуючи, що використана модель спостереження досить близько нагадує датчик Velodyne від KITTI. Загалом, не дивно, що контрольована базова лінія перевершує AML, враховуючи, що AML використовує лише частину, зокрема 4,06% у помірному випадку, інформації під час навчання.

4.4 KITTI

Використовуючи ShapeNet, можемо вивчити попередню фігуру, що дозволяє нам виконати завершення фігури у KITTI, однак, не маємо доступу до істинних форм. Після обговорення в розділі 4.2 ми відфільтрували надані базові 3D-обмеження, використовуючи $n_{1,min} = 150$, $n_{0,min} = 1500$ і $t_{max} = 30$. Це означає, що вимагаємо, щоб принаймні 150 вокселів були зайняті, а 1500 вокселів відповідали вільному простору. Крім того, розглядаємо тільки обмежувальні рамки в межах 30 м від датчика.

Загалом, отримали 1928 окремих спостережень за автомобілями, які ми розділили на навчальну вибірку з 1714 та валідаційну з 214 зразків. В середньому, ~ 0,634% вокселів є спостережуваними точками; ~ 5,98% вокселів є вільним простором. У цьому відношенні набір даних може бути дещо простішим, ніж жорсткий набір даних ShapeNet, де лише 0,304% вокселів вважаються зайнятими. Отриманий набір даних є особливо складним через відповідні шумові патерни.

4.4.1 Амортизована максимальна правдоподібність

Використовуючи AML, можемо отримати розумні завершення форми із зашумлених спостережень. З попереднього навчання, модель є перенавчена, що використовується для ShapeNet, на всіх 40992 зразках з обох навчальних наборів, див. таблицю 4.2. На рисунку 4.17 спочатку показуємо спостереження, тобто спостережувані точки і вільний простір, разом з прогнозами форми для декількох прикладів з валідаційного набору. Також

показуємо вплив використання вагових коефіцієнтів з рівняння 4.1, а також навчання за попередніми даними на комбіновані навчальні множини ShapeNet. Як і очікувалося, використання зважених втрат має значний вплив на якість прогнозованих фігур.

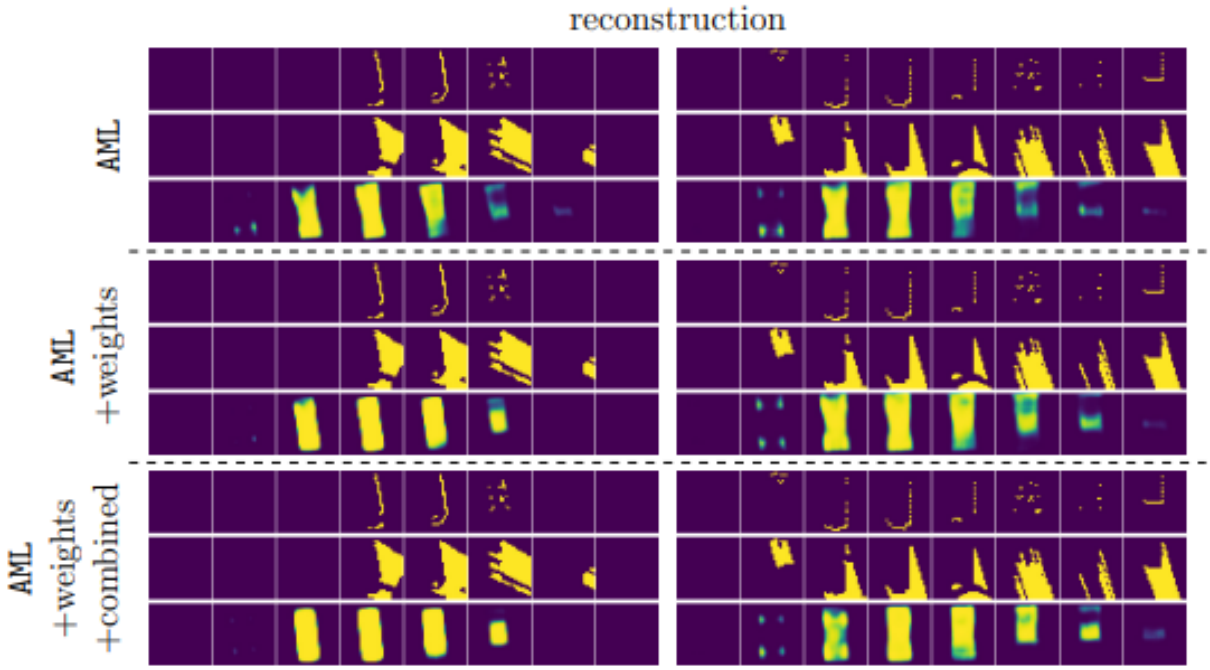

Рисунок 4.17 — Якісні результати для AML на витягнутому наборі даних KITTI

Крім того, також помітили перевагу використання більшої кількості навчальних даних для попередника. Для подальшого обговорення завжди використовуємо вагові коефіцієнти, а також сильніший попередник. На рисунку 4.18 ми показуємо 3D-візуалізації, що відповідають вокселізованим спостереженням та передбаченим формам.

Бачимо, що передбачені форми збігаються зі спостережуваними точками і чітко зображують автомобілі. Однак прогнозам все ще бракує значного рівня деталізації навколо коліс та даху.

Це можна розглядати як компроміс між завадостійкістю, інтегрованою за допомогою сильної попередньої обробки, та орієнтацією на деталі. Дивно, але передбачені форми виглядають краще, ніж на жорсткому наборі даних на основі ShapeNet, тобто на рисунку 4.14.

Це підтверджує нашу інтуїцію, що спостереження, отримані з KITTI, є дещо простішими. Загалом, AML може правильно прогнозувати орієнтацію та приблизну форму спостережуваних автомобілів, але також залишає простір для вдосконалення щодо деталей.

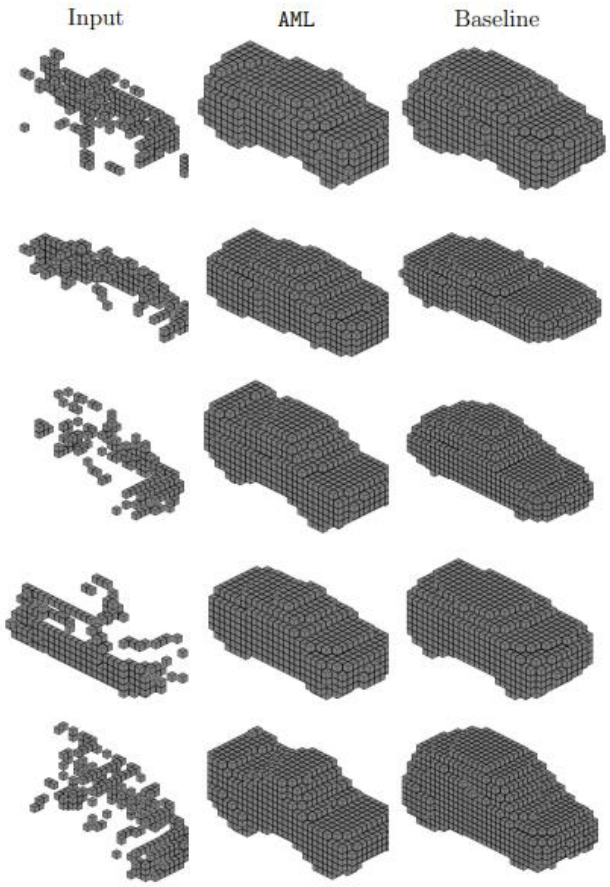

Рисунок 4.18 — Порівняння AML та контрольованої базової лінії на KITTI

Бачимо, що передбачені форми збігаються зі спостережуваними точками і чітко зображують автомобілі. Однак прогнозам все ще бракує значного рівня деталізації навколо коліс та даху. Це можна розглядати як компроміс між завадостійкістю, інтегрованою за допомогою сильної попередньої обробки, та орієнтацією на деталі. Дивно, але передбачені форми виглядають краще, ніж на жорсткому наборі даних на основі ShapeNet, тобто на рисунку 4.14. Це підтверджує нашу інтуїцію, що спостереження, отримані з KITTI, є дещо простішими. Загалом, AML може правильно прогнозувати орієнтацію та

приблизну форму спостережуваних автомобілів, але також залишає простір для вдосконалення щодо деталей.

Також використовували ShapeNet, попередньо навчений як на функціях заповнюваності, так і на знакових функціях відстані. Однак виявили, що результати були дещо гіршими порівняно з ShapeNet. Як і раніше, підозрюємо, що для покращення результатів знадобиться довший час навчання та точне налаштування гіперпараметрів. Тим не менш, на рисунку 4.19 показано кілька прикладів прогнозів як у вигляді сіток заповнюваності, так і у вигляді сіток.

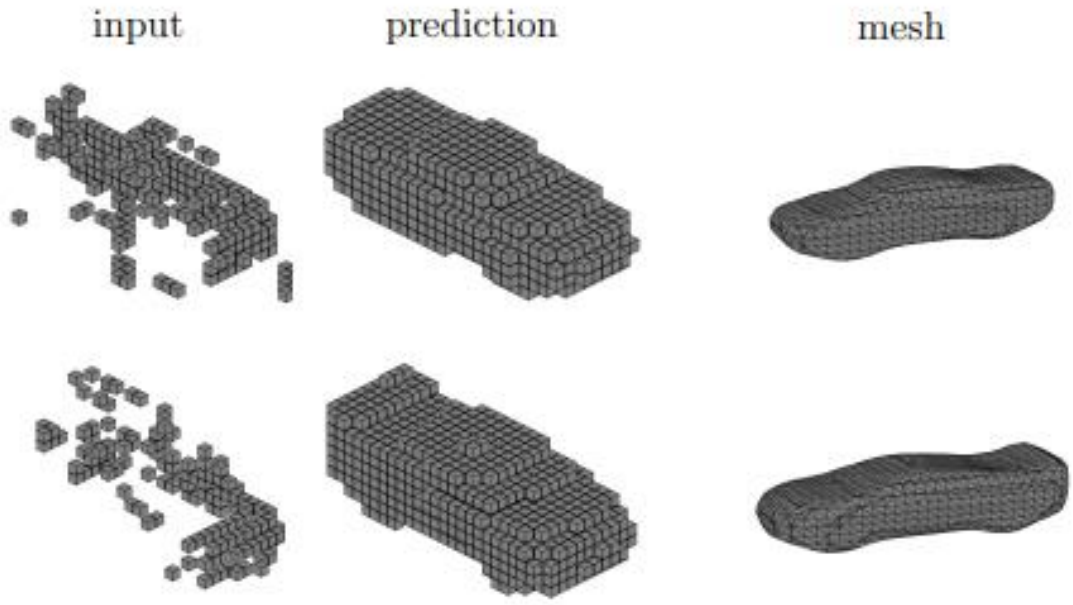

Рисунок 4.19 — 3D-візуалізація прогнозованих форм з використанням AML, що прогнозує як заповнюваність та підписані функції відстані

Як і у складному випадку з ShapeNet, пріоритет, схоже, віддається невеликим і тонким автомобілям. Ця тенденція підкреслюється сітками, отриманими за допомогою маршових кубів, і може бути пояснена невизначеністю, пов'язаною з реконструкцією таких деталей, як колеса і дах. Знову ж таки, можна зробити висновок, що знакові функції відстаней, хоча і є зручними для побудови сіток, є більш складними для вивчення в умовах слабкого контролю. Додатковий приклад зі згенерованих наборів даних KITTI наведено в додатку К.

4.4.2 Розширений варіаційний автокодер

Дослідили, наскільки контрольовану базову лінію можна узагальнити для KITTI. На рисунку 4.18 показано якісні результати застосування навченої моделі до спостережень KITTI. Як видно, модель прогнозує розумні автомобілі. На відміну від AML, контрольована модель не має проблем з прогнозуванням деталей вздовж даху або коліс. Для деяких зразків, однак, прогнози виглядають дуже схожими. Загалом, контрольована базова лінія, здається, прогнозує дещо більш когерентні форми; але точна оцінка і порівняння неможливі без анотацій наземної істини. Загалом, вважаємо, що між AML і контрольованою базовою лінією немає значного розриву з точки зору візуальної якості відповідних прогнозів.

Узагальнюючи, тренування форми попереднього розподілу (VAE) та моделей інференції форми на 3D-даних вимагає належного налаштування гіперпараметрів через тривалість тренування, особливо при низьких роздільних здатностях. Предствлено, що VAE здатні вивчати форми, використовуючи зайнятість та функції відстані зі знаком.

Щодо завершення форми, AML та EVAE виявилися ефективними, з AML трошки переважає EVAE. На ShapeNet результати не є однозначними, особливо на складних рівнях. Обмежений час завадив провести більше експериментів, але представлені результати свідчать про можливість вивчення завершення форми при слабкому нагляді та сильних формах пріоритетів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

За мету проведення технологічного аудиту є взято оцінювання комерційного потенціалу розробки, в нашому випадку – навченої нейромережі для відновлення 3D-об’єктів із набору попередньо відсканованих точок.

Дуже схожим аналогом нашого рішення може бути програмне забезпечення Geomagic Wrap від компанії Artec 3D, який дана компанія продає за ціною $9500 або 348336,22 грн (за курсом станом на 05.12.2023 року: $1 = 36.67₴) за 1 цифрову копію.

Для проведення комерційного та технологічного аудиту рекомендовано залучати не менше 3-х незалежних експертів та із застосуванням п’ятибальної системи оцінювання згідно 12-ти критеріями, що приведені в таблиці 5.1.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
№	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1.

1	2	3	4	5	6
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1.

1	2	3	4	5	6
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Всі результати оцінювання по кожному параметру занесено до таблиці 5.2 і оцінити рівень його комерційного потенціалу, порівнявши дані разом з відповідними рівнями комерційного потенціалу розробки із таблиці 5.3.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами:		
1	2	4	4
2	4	4	2
3	3	4	4
4	4	2	3
5	3	3	4
6	3	4	4
7	4	3	2
8	4	2	3
9	2	4	4
10	3	4	2
11	4	3	2
12	4	4	3

Продовження таблиці 5.2.

Сума балів	40	41	43
Середньоарифметична сума балів	(40 + 41 + 43) / 3 = 41,33		

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Судячи з таблиці, можемо зазначити, що рівень комерційного потенціалу розробки є високим. Таку позначку можна досягти за причини того, що нейромережа завдяки самонавчанню може достатньо точно і детально відтворити модель, і ще й так щоб кінцевий користувач міг зрозуміти всі нюанси і деталі про реконструйований цифровим шляхом об’єкт. Зокрема можна виділити, що цю нейромережу можна легко інтегрувати у сервіси різних сфер життя.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Усі розрахунки щодо витрат на виконання науково-дослідної роботи поділимо по певним пунктам:

— витрати на заробітню плату;

— витрати на матеріали та обладнання;

— загальновиробничі витрати;

— амортизаційні відрахування;

Основна заробітна плата працівника $З_о$ визначається за даною формулою (5.1)

$$З_о = \frac{М}{Т_р} * t \ \ [грн], \tag{5.1}$$

де М — місячний посадовий оклад робітника, грн;

$Т_р$ — число робочих днів в місяці, 22 дні;

t — число днів роботи робітника.

Для створення нашого проекту можна залучити таких спеціалістів потрібного нам стеку технологій: Deep Learning Researcher, Neural Network Architect, Data Processing Engineer, Deep Learning Project Manager.

Усі виконані розрахунки занесемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Deep Learning Researcher	40000	2000	22	3636363,63
Neural Network Architect	35000	1800	22	2863636,36
Data Processing Engineer	25000	1300	22	1477272,72
Deep Learning Project Manager	45000	2200	22	4500000
Всього				12477272,71

Розмір додаткової заробітної плати робітників зазвичай розраховується як 15% від витрат на заробітну плату робітників, тому:

$$З_д = З_о * \frac{15\%}{100}, \tag{5.2}$$

$$З_д = 12477272{,}71 * \frac{15\%}{100} = \ 371590{,}9 \text{ (грн.)}$$

Нарахування на заробітну плату розраховують як 22% суми основної і додаткової заробітних плат.

$$Н_з = (З_о + З_д) * \frac{22\%}{100} \quad (5.3)$$

$$Н_з = (12477272{,}71 + 371590{,}9) * \frac{22\%}{100} = 2891420{,}39 \ \text{(грн.)}$$

Витрат на матеріали чи комплектуючі не передбачається, так як для нашого продукту не потрібно витрачати такі ресурси і тому вони будуть дорівнювати нулю.

Амортизаційні відрахування для всього обладнання, що використовувались для проведення розробки розраховується згідно формулі:

$$А = \frac{Ц}{Т_в} * \frac{t_{вих}}{12} \ \text{[грн.]}. \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Згідно нашого проекту, розрахуємо амортизаційні відрахування саме для комп'ютер. Його балансова вартість складає 44000 грн, термін використання — 2 роки, фактичний час використання — 9 місяців.

Тоді:

$$А_{обл} = \frac{44000}{2} * \frac{9}{12} = \ 16500 \text{ (грн.)}$$

Далі за таким же принципом розраховується амортизація і на решту обладнання і приміщення. Усі розрахунки щодо амортизації матеріальних речей будуть представлені у таблиці 5.5. Амортизація нематеріальних речей

не буде відбуватись, так як вартість ОС та інших нематеріальних ресурсів (PyCharm, Visual Studio) не перевищує 20000 грн, тому їх вартість буде включена повністю — $В_{нем.ак.}$ = 12000 грн.

Таблиця 5.5 — Амортизаційні відрахування матеріальних речей

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна периферія (Dell XPS 15 9500)	44000	2	9	16500
Офісне обладнення (меблі)	150000	4	9	56250
Приміщення	430000	20	9	161250
Всього				234000

Ціни на електроенергію для підприємств варіюються в залежності від типу клієнта, а також від розподільних компаній, які надають послуги розподілу електроенергії. Тарифи визначаються Державною комісією з регулювання енергетики та комунальних послуг і залежать від рівня напруги, де може бути 1 або 2 рівень. Крім того, вони різняться для підприємств та населення, створюючи різноманітні тарифи для різних категорій споживачів. Визначення вартості електроенергії для підприємства може бути виконане за такою формулою:

$$В_е = В * П * Ф * К_п, \quad (5.5)$$

де В — вартість 1 кВт/год електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання — 0,2 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

Кп — коефіцієнт використання потужності — 0,9.

$$В_е = 0{,}9 * 0{,}2 * 8 * 39 * 6{,}2 = 428{,}04 \text{ (грн.)}$$

До інших та загальновиробичих витрат відносяться витрати, які не було зазначено в інших категоріях і можуть бути враховані в собівартість розробки на підставі конкретних параметрів. Розрахунок витрат за цією статтею визначається у діапазоні від 50% до 100% від суми основної заробітної плати. Рохрахуємо це за наступною формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (5.6)$$

де $Н_{ів}$ — норма нарахувань по іншим витратам.

Тоді:

$$І_в = 12848863{,}61 * \frac{95\%}{100\%} = 12206420{,}43 \text{ (грн.)}$$

Загальновиробничі витрати включають у себе різноманітні витрати, що виникають в процесі управління організацією та виробництва продукції. Це охоплює витрати на управління, винаходи та раціоналізацію, підготовку та навчання персоналу, набір робочої сили, банківські послуги, освоєння виробництва, науково-технічну інформацію, рекламу тощо.

Загальновиробничі витрати розраховуються як відсоток від суми основної заробітної плати дослідників. Конкретно, ці витрати становлять 100% до 150% від суми основної заробітної плати працівників, і розраховуються за формулою:

$$Н_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ — норма нарахувань загальновиробничих витрат.

$$Н_{нзв} = 12848863{,}61 * \frac{145\ \%}{100\ \%} = 18617150{,}03 \text{ (грн)}$$

Загальні витрати на проведення розробки визначається сумою всіх попередніх витрат — заробітні плати, амортизаційні відрахування, витрати на електроенергію, загальновиробничі, тощо:

$$В_{заг} = 312477272{,}71 + 371590{,}9 + 2891420{,}39 + 234000 + 12000 + 428{,}04 + 12206420{,}43 + 18617150{,}03 = 46800282{,}59 \text{ (грн)}$$

Загальні витрати для завершення проектування та розробки проекту розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η — етап виконання проекту.

Для визначення показника η є певні його критерії в залежності від стадії виконання роботи:

— η = 0,1 — науково-дослідна робота;
— η = 0,2 — технічне проектування;
— η = 0,3 — розробка документації;
— η = 0,4 — розробка технологій;
— η = 0,5 — створення прототипу для досліджень;
— η = 0,7 — створення промислового прототипу;
— η = 0,9 — впровадження розробки в експлуатацію.

Оскільки наша розробка знаходиться на етапі створення дослідницького прототипу, то η = 0,5:

$$ЗВ = \frac{46800282{,}59}{0{,}5} = 93600565{,}18 \text{ грн}$$

5.3 Розрахунок економічної ефективності та обґрунтування економічної доцільності комерціалізації науково-технічної розробки

У конкурентному середовищі, для потенційного інвестора, успішним результатом при впровадженні науково-технічної розробки буде збільшення чистого прибутку. Це зростання чистого прибутку відкриває можливості для додаткових інвестицій, поліпшує фінансові показники та підвищує конкурентоспроможність. Обране значення η = 0,5 вказує на те, що розробка знаходиться на етапі створення дослідного зразка.

Для визначення потенційного зростання чистого прибутку важливо:

а) уточнити період часу, протягом якого очікується впровадження результатів розробки;

б) визначити термін, протягом якого потенційний інвестор очікує позитивних результатів (наприклад, протягом перших 3 років);

в) оцінити ринковий попит на розробку та ідентифікувати ключових зацікавлених сторін цього попиту;

г) визначити ринкову ціну для аналогічних технологічних рішень.

При оцінці економічної ефективності важливо враховувати часові фактори, оскільки інвестиції мають витрати у часі. Ключові показники для оцінки інноваційних проектів включають абсолютний економічний ефект, внутрішню норму доходності та термін окупності. Надалі викладений аналіз можна застосовувати до різних напрямків розробки, враховуючи конкретні умови та сценарії.

У даному випадку, прогнозований економічний виграш буде визначений з урахуванням наступних вихідних даних на основі формули (5.10):

$$\Delta П_i = (\pm \Delta Ц_0 * N + Ц_0 * \Delta N)_i * \lambda * \rho(1 - \frac{\vartheta}{100}), \qquad (5.10)$$

де $\pm \Delta Ц_0$ — це зміна вартості програмного продукту (збільшення чи зменшення) внаслідок впровадження результатів науково-технічної розробки протягом аналізованих періодів часу;

N — кількість споживачів, які використовували аналогічний продукт у рік до впровадження результатів нової науково-технічної розробки;

$Ц_0$ — основний оціночний показник, що визначає діяльність підприємства у році після впровадження результатів наукової розробки;

$Ц_b$ — вартість програмного продукту у рік до введення результатів розробки;

ΔN — збільшення кількості споживачів продукту протягом аналізованих періодів часу внаслідок покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість, де ставка податку на додану вартість дорівнює 20%, і λ = 0,8333;

ρ — коефіцієнт, який враховує рентабельність продукту;

ϑ — базова ставка податку на прибуток, у 2023 році згідно Податкового кодексу України він становить 18%.

Внаслідок інтеграції наукової розробки покращиться якість конкретного продукту, що призводить до підвищення його ринкової ціни на 200000 грн. Кількість користувачів зросте протягом першого року на 4000 осіб, другого — на 10000, а третього — на 25000. Реалізація продукції до впровадження результатів наукової розробки складала 500 користувачів, а її ціна — 35000 грн, тоді:

$$\Delta П_1 = (200000 \cdot 500 + 35000 \cdot 4500) \cdot 0.8333 \cdot 0.3 \cdot \left(1 - \frac{18}{100}\right) = 4692166{,}67 \text{ грн}$$

$$\Delta П_2 = (200000 \cdot 500 + 35000 \cdot 14500) \cdot 0.8333 \cdot 0.3 \cdot \left(1 - \frac{18}{100}\right) = 10365980{,}51 \text{ грн}$$

$$\Delta П_3 = (200000 \cdot 500 + 35000 \cdot 29500) \cdot 0.8333 \cdot 0.3 \cdot \left(1 - \frac{18}{100}\right) = 321590329{,}27 \text{ грн}$$

Можемо підмітити що зазначений комерційний ефект від впровадження результатів розробки за три роки становить приблизно 336648476,45 грн

Обчислюємо чистий прибуток, який може отримати потенційний інвестор внаслідок можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.11)$$

де $\Delta П_i$ — приріст щорічного чистого прибутку;

Т — тривалість часу, вимірювана в роках, під час якої стають очевидними результати реалізованого проекту;

τ — ставка дисконтування або щорічний прогнозований рівнем інфляції в країні, що знаходиться в діапазоні від 0,05 до 0,15;

t — період часу, роки;

$$ПП = \left(\frac{4692166,67}{(1+0,1)^1}\right) + \left(\frac{10365980,51}{(1+0,1)^2}\right) + \left(\frac{321590329,27}{(1+0,1)^3}\right) = 4265606,97 + 8561963,39 + 241674368,71 \approx 255575939,07 \text{ грн}$$

Наступним кроком є визначення величини початкових інвестицій (PV), необхідних для введення та виходу розробки на ринок. Це можна здійснити за допомогою відповідної формули:

$$PV = k_{інв} * ЗВ \quad (5.12)$$

де $k_{інв}$ — коефіцієнт вказує на те, скільки грошових активів потрібно для введення і виходу продукту на ринок, щоб здобути одиницю прибутку;

ЗВ — загальні витрати на розробку.

$$PV = 2 * 93600565,18 = 187201130,36 \text{ грн}$$

Аби вирахувати абсолютну ефективність інвестицій, скористаємось наступною формулою:

$$E_{абс} = (ПП - PV), \quad (5.13)$$

де ПП представляє приведену вартість всіх чистих прибутків, отриманих підприємством (організацією) від реалізації результатів наукової розробки у гривнях, а PV позначає теперішню вартість інвестицій, що дорівнює загальним витратам на проведення науково-технічної розробки та оформлення її результатів, також в гривнях.

$$E_{абс} = 255575939{,}07 - 93600565{,}18 = 161975373{,}89 \text{ грн}$$

Якщо $E_{абс} > 0$, це означає, що приведена вартість всіх чистих прибутків більша за теперішню вартість інвестицій, тоді інвестиції є ефективними.

Аби вирахувати щорічну ефективність інвестицій, скористаємось наступною формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

де $E_{абс}$ — абсолютна ефективність інвестицій, грн;

PV — теперішня вартість інвестицій, грн;

$Т_ж$ — життєвий цикл наукової розробки, роки.

Тоді:

$$E_в = \sqrt[3]{1 + \frac{161975373{,}89}{187201130{,}36}} - 1 = 0{,}255.$$

Отже, для визначення зацікавленості інвестора вкладати кошти в науковий проект розглядається питання порівняння розрахованого абсолютного ефекту від інвестицій ($E_{абс}$) з мінімальною (бар'єрною) ставкою дисконтування (мін). Мінімальна (бар'єрна) ставка дисконтування (мін) обчислюється за формулою: мін = d + f, де d — середньозважена ставка за

депозитними операціями в комерційних банках (дорівнює 0,09), а f — показник, що враховує ризикованість вкладень (зазвичай f = 0,05). Таким чином, $\tau min = 0,09 + 0,05 = 0,14$. Якщо розрахований абсолютний ефект перевищує мінімальну ставку дисконтування (мін), це свідчить про зацікавленість інвестора вкладати гроші в дану наукову розробку. У випадку, який розглядаємо, $E_в = 25,5\%$, що значно менша за мінімальну ставку дисконтування , яка становить 14%. Таким чином, інвестор може бути зацікавлений вкладати кошти в дану наукову розробку. Абсолютний ефект вказує на те, що очікуваний прибуток від інвестицій більший ніж вартість вкладених коштів, що може бути дуже привабливим для інвестора.

Також визначимо термін окупності інвестицій у реалізацію проект:

$$T_{ок} = \frac{1}{E_в} \tag{5.14}$$

$$T_{ок} = \frac{1}{0,255} = 3,92 \text{ (років)}$$

Оскільки термін окупності є більшим за 3 роки — це може вказувати на те, що вкладені кошти в проект повертаються повільно. В цьому випадку може бути легше визначити час повернення інвестицій, і ризик інвестування в проект може збільшитися.

5.4 Результати до економічної частини

В даному розділі було проведено комерційний та технологічний аудит розробки із залученням трьох незалежних експертів та оцінка комерційного потенціалу показала, що рівень є високим. Прогнозовані витрати на проектування та реалізацію системи складають 46800282,59 грн. Загальні витрати оцінені у 93600562,18 грн. Також було зпрогнозовано прибуток у сумі 255575939,07 гривень. Розрахунки терміну окупності та прибутку за три роки показують, що розробка є досить привабливою для інвесторів, із терміном окупності 3 роки і 11 місяців що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У вступі представлено завершення форми як проблему реконструкції окремого об'єкта на основі часткового спостереження, наприклад, з одного погляду. Доведено, що попередні знання про форму відіграють вирішальну роль у сприйнятті форми загалом і завершенні форми зокрема. Окреслено нещодавні успіхи у вивченні моделей фігур за допомогою глибоких генеративних моделей та використання попередніх даних для завершення фігур на основі даних Підходи до завершення фігур, засновані на навчанні, навпаки, безпосередньо вивчають завершення фігур у контрольованому середовищі, більшість з них використовують глибокі нейронні мережі. Хоча ці підходи дозволяють робити ефективні висновки, тобто завершення фігури є "просто проходом вперед" у навченій мережі, їх застосування обмежене вимогою анотованих навчальних даних. Підходи, керовані даними, можуть бути безпосередньо застосовані до реальних даних, оскільки завершення фігури — і, отже, висновок — є мінімізацією енергії. Гіпотеза полягає в тому, що використання генеративних моделей фігур дає змогу навчитись заповнювати фігури в неконтрольованому середовищі. В результаті, завершення фігури все одно буде ефективно виконуватися навченою мережею, уникаючи при цьому необхідності в анотованих наборах даних.

З метою експериментальної перевірки нашої гіпотези запропоновано та реалізували два імовірнісних фреймворки для вивчення завершення форми автомобілів на KITTI. Зокрема, базуючись на варіаційному автокодері в якості попереднього навчання, ми поставили задачу завершення форми як задачу максимальної правдоподібності над вивченим латентним простором. Дотримуючись ідеї амортизованого висновку, ми навчили кодер безпосередньо передбачати рішення з максимальною правдоподібністю, тобто вивчати вбудовування спостережень у латентний простір форми, інтерпретуючи задачу максимальної правдоподібності як неконтрольовану втрату. Як альтернативний підхід також розглядається безпосереднє інтегрування спостережень як випадкової величини в модель латентної

змінної, тобто в варіаційний автокодер. Врахування відповідного спільного розподілу дозволило нам отримати нижню межу достовірності та навчити розширений варіаційний автокодер, який неявно також навчається завершенню форми. Обидва підходи можна навчати на KITTI, використовуючи машини з ShapeNet для попереднього розпізнавання фігур. Крім того, доповнення форми, тобто висновок, ефективно виконується навченою мережею. Експериментально показано, що обидва підходи оптимізують схожу задачу. На синтезованому наборі даних на основі ShapeNet, обидва підходи здатні конкурувати з повністю контрольованою базовою лінією як кількісно, так і якісно. На прикладі KITTI продемонстровано, що запропоновані підходи дійсно застосовні до реальних даних і можуть навчатися завершенню фігур у неконтрольованому середовищі.

Отже, представлені експерименти свідчать на користь нашої гіпотези про те, що сильні попередні знання про форму дозволяють навчитися завершувати фігури без нагляду. Це може стати важливим першим кроком у міркуваннях про фігури, а також про сцени в умовах слабкого контролю шляхом явного використання сильних попередніх знань про фігури, які можна вивчити на синтетичних даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wang, H., Raj, B., & Xing, E. P. On the origin of deep learning. [Електронний ресурс]. — Режим доступу до ресурсу: https://arxiv.org/abs/1702.07800.

2. Підцерковний Є. О., Кожем'яко А. В. Навчена нейромережа завершення 3D-форм об'єктів з розріджених хмар 3D-точок. Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/19682

3. Girdhar, R., Fouhey, D. F., Rodriguez, M., & Gupta, A. Learning a predictable and generative vector representation for objects. European Conference on Computer Vision, ст. 484–499. 2016.

4. Wu, J., Zhang, C., Xue, T., Freeman, B., & Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3D generative adversarial modeling. У Advances in Neural Information Processing Systems, ст. 82–90. 2016.

5. 3D ShapeNets: A deep representation for volumetric shapes / Zhirong Wu et al. 2015 IEEE conference on computer vision and pattern recognition (CVPR), Boston, MA, USA, 7 — 12 червня 2015. 2015.

6. Computer vision — ECCV 2016 workshops / ed. by G. Hua, H. Jégou. Cham : Springer International Publishing, 2016.

6. Dai A., Qi C. R., NieBner M. Shape completion using 3d-encoder-predictor cnns and shape synthesis. 2017 IEEE conference on computer vision and pattern recognition (CVPR), Honolulu, HI, 21 — 26 червня 2017. 2017.

7. Dense Object Reconstruction with Semantic Priors / S. Y. Bao et al. 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23—28 червня 2013. 2013.

8. Dense Reconstruction Using 3D Object Shape Priors / A. Dame et al. 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23 — 28 червня 2013. 2013.

9. Engelmann F., Stückler J., Leibe B. Joint Object Pose Estimation and Shape Reconstruction in Urban Street Scenes Using 3D Shape Priors. Lecture Notes in Computer Science. Cham, 2016. P. 219 — 230.

10. Fan H., Su H., Guibas L. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 21 — 26 червня 2017. 2017.

11. Geiger A., Lenz P., Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, 16 — 21 червня 2012. 2012.

12. Guney F., Geiger A. Displets: Resolving stereo ambiguities using object knowledge. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7 — 12 червня 2015. 2015.

13. Improved adversarial systems for 3D object generation and reconstruction. [Електронний ресурс]. — Режим доступу до ресурсу: https://proceedings.mlr.press/v78/smith17a.html

14. Learning a predictable and generative vector representation for objects / R. Girdhar et al. Computer vision — ECCV 2016. Cham, 2016. P. 484—499.

15. Menze M., Heipke C., Geiger A. JOINT 3D ESTIMATION OF VEHICLES AND SCENE FLOW. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences. 2015. II-3/W5. P. 427—434.

16. OctNetFusion: learning depth fusion from data / G. Riegler et al. 2017 international conference on 3D vision (3DV), Qingdao, 10—12 жовтня 2017. 2017.

17. Point cloud completion using extrusions / O. Kroemer et al. 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, 29 November — 1 грудня 2012. 2012.

18. ShapeNet: An information-rich 3D model repository. [Електронний ресурс]. —— Режим доступу до ресурсу: http://arxiv.org/abs/1512.03012

19. Unsupervised Learning of 3D Structure from Images. List of Proceedings. [Електронний ресурс]. — Режим доступу до ресурсу: https://papers.nips.cc/paper_files/paper/2016/hash/1d94108e907bb8311d8802b48fd54b4a-Abstract.html

20. VConv-DAE: Deep Volumetric Shape Learning Without Object Labels. [Електронний ресурс]. — Режим доступу до ресурсу: http://arxiv.org/abs/1604.03755

21. Vision meets robotics: The KITTI dataset / A. Geiger et al. The International Journal of Robotics Research. 2013. Vol. 32, no. 11. P. 1231—1237.

22. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти денної та заочної форм навчання. / Укладачі О.Д. Азаров, О.В. Дудник, С.І. Швець − Вінниця : ВНТУ, 2023. − 57 с.

23. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. 42 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

“29” вересня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

“Штучна нейромережа глибокого навчання для моделювання 3D-форм об'єктів з розріджених хмар 3D-точок”

08-54.МКР.038.00.000 ПЗ

Науковий керівник: доцент
к.т.н. каф.ОТ

_____ Кожем'яко А. В.

Студент групи 2КІ-22м

_____ Підцерковний Є. О.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Основною актуальністю роботи введення амортизованого виводу як способу передбачення з високою ймовірністю, що дозволяє прогнозувати рішення на основі спостережень для розв'язання складних завдань, знижуючи обчислювальні витрати та забезпечуючи більш ефективні підходи.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — розробка імовірнісного підходу для нейромережі щодо завершення 3D-фігур з великою кількістю невідомих параметрів на основі попередніх знань про форму, отриманих із об'єктів у великих наборах даних.

2.2 Призначення розробки полягає в здатності нейромережі вирішувати проблему через один прямий прохід, що може бути важливим для практичних задач з відновленням об'єктів на основі обмежених спостережень.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу проблеми та методів у Deep Learning.

3.2 Обґрунтування вирішення і представлення фігур та форм.

3.3 Аналіз наборів даних, обраних для дослідження.

3.4 Проведення екпериментів із обраними наборами даних.

3.5 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — нейромережа має виконувати амортизований вивід, що дозволяє нейромережі передбачати рішення з максимальною ймовірністю безпосередньо на основі відповідних спостережень.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Обґрунтування проблеми, методів та технік у глибокому навчанні	19.09.2023	25.09.2023	Розділ 1
2	Аналіз представлення фігур та форми	1.10.2023	19.10.2023	Розділ 2
3	Вибір даних для дослідження	20.10.2023	23.10.2023	Розділ 3
4	Екпериментування	23.10.2023	30.10.2023	Розділ 4
4	Підготовка економічної частини	10.11.2023	23.11.2023	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	3.12.2023	11.12.2023	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.12.2023	12.12.2023	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Опис водонепроникних сіток

Щоб доповнити грубе уявлення про водонепроникні сітки як про закриті поверхні з чітко визначеними внутрішньою і зовнішньою сторонами, надамо відповідне математичне обґрунтування для уточнення.

Спочатку надамо термінологію:

— самоперетин — це перетин двох граней однієї і тієї ж сітки;

— небагатовимірне ребро має більше двох інцидентних граней;

— зірка вершини — це об'єднання всіх її інцидентних граней;

— небагатовимірна вершина — це вершина, при видаленні якої відповідна зірка не з'єднується з нею;

— сітка є 2-багатовимірною, якщо вона не містить ні самоперетинів, ні ребер, ні вершин, що не є багатовимірними, ні вершин, що не є багатовимірними.

Загалом, 2-багатовидні сітки є кращими за довільні сітки, оскільки багато алгоритмів і додатків не застосовуються до не-багатовимірних сіток. У нашому випадку, однак, визначення 2-багатовидових сіток мотивовано лише необхідністю формального визначення водонепроникних сіток (які іноді також називають закритими сітками). Інтуїтивно зрозуміло, що єдиним обмеженням, якого бракує 2-багатовидовим сіткам, є поняття "замкненості", тобто чіткої внутрішньої та зовнішньої частини. Це стає очевидним при розгляді визначення ребра, що не є многовидом, яке також допускає ребра з однією інцидентною гранню, так звані граничні ребра.

Визначення А.1 — 2-множинна сітка називається водонепроникною, якщо кожне ребро має рівно дві інцидентні грані, тобто не існує граничних ребер.

У програмному забезпеченні, наприклад, у MeshLab, можна легко ідентифікувати та позначати вершини, ребра, а також граничні ребра, які не є многовидом, що допомагає проектувати та працювати з трикутними сітками.

ДОДАТОК В

Неімовірнісний підхід

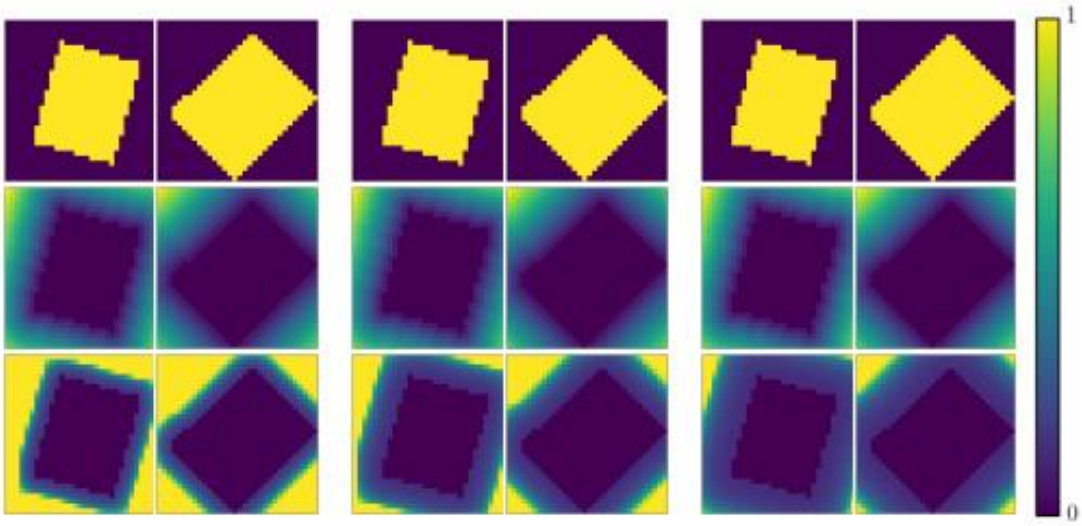

Рисунок В.1 — Ілюстрація апроксимації функції відстані для T = 3, 5, 7 ітерацій (зліва направо)

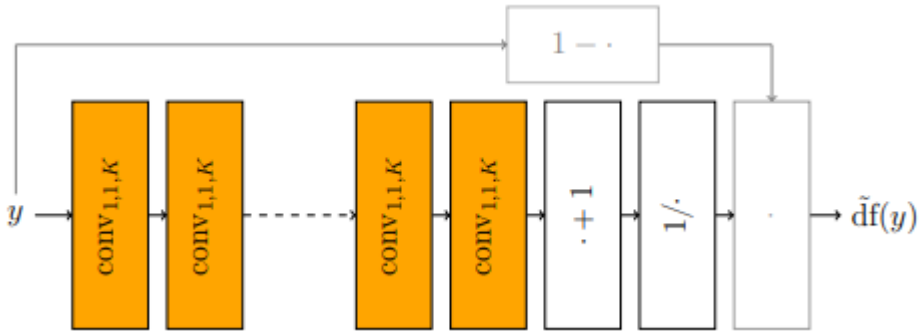

Рисунок В.2 — Ілюстрація шарів, необхідних для апроксимації функції відстані сітки заповнюваності з урахуванням ймовірностей заповнюваності y

ДОДАТОК Г

Приклад з синтетичного набору 2D

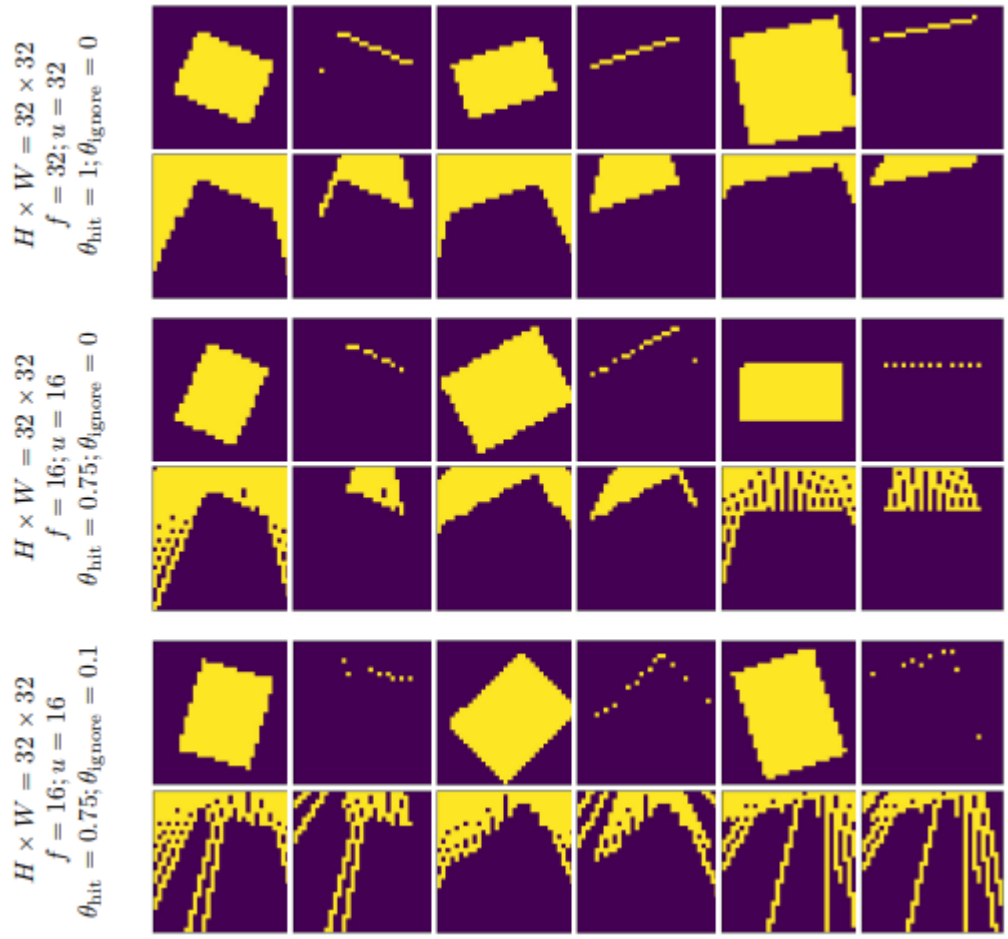

Рисунок Г.1 — Приклад з синтетичного набору 2D роздільною здатністю 1D

ДОДАТОК Д

Опис алгоритму напівопуклої оболонки для отримання водонепроникних спрощених сіток

Вхідні дані — трикутна сітка M.

Вихідні дані — спрощена сітка $M^{(T)}$.

Розглянемо структуру алгоритму для побудови опуклої оболонки та ізотропного відтворення поверхні за допомогою прямої передискретизації, яку опишемо покроково:

1) побудувати вибірку точок P з M;

2) обчислити опуклу оболонку $M^{(0)} = (V^{(0)}, F^{(0)})$ з M;

3) пересікти $M^{(0)}$ за допомогою прямої передискретизації для ізотропного відтворення поверхні;

4) для t = 0 до (T – 1) — якщо $M \subseteq \mathrm{Vol}(V^{(t)})$, то знайти найменше $\alpha > 0$ таке, як $P \subseteq \mathrm{Vol}((1+\alpha)V^{(t)})$, тоді визначимо, що $V^{(t)} \coloneqq (1+\alpha)V^{(t)}$, далі пересікти $M^{(t)}$ за допомогою прямої передискретизації для ізотропного відтворення поверхні та визначити вираз — $V^{(t+1)} = V^{(t)} - \gamma \nabla \mathcal{L}(V^{(t)})$;

5) повернути $M^{(t)}$.

ДОДАТОК Е

Приклад зі згенерованих наборів даних 3D-кубоїдів

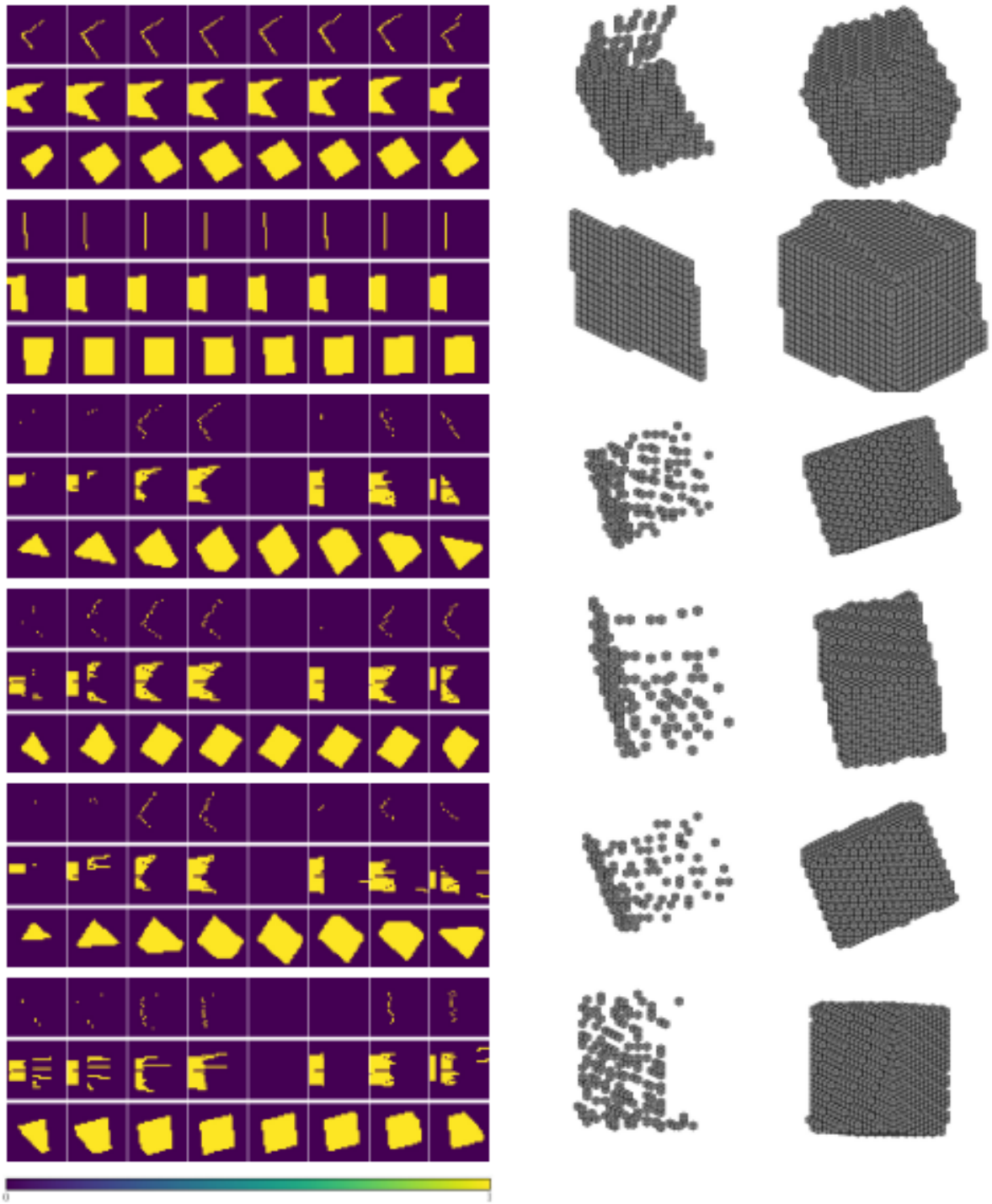

Рисунок Е.1 — Приклад зі згенерованих наборів даних 3D-кубоїдів

ДОДАТОК Ж

Приклад зі згенерованих наборів даних ShapeNet

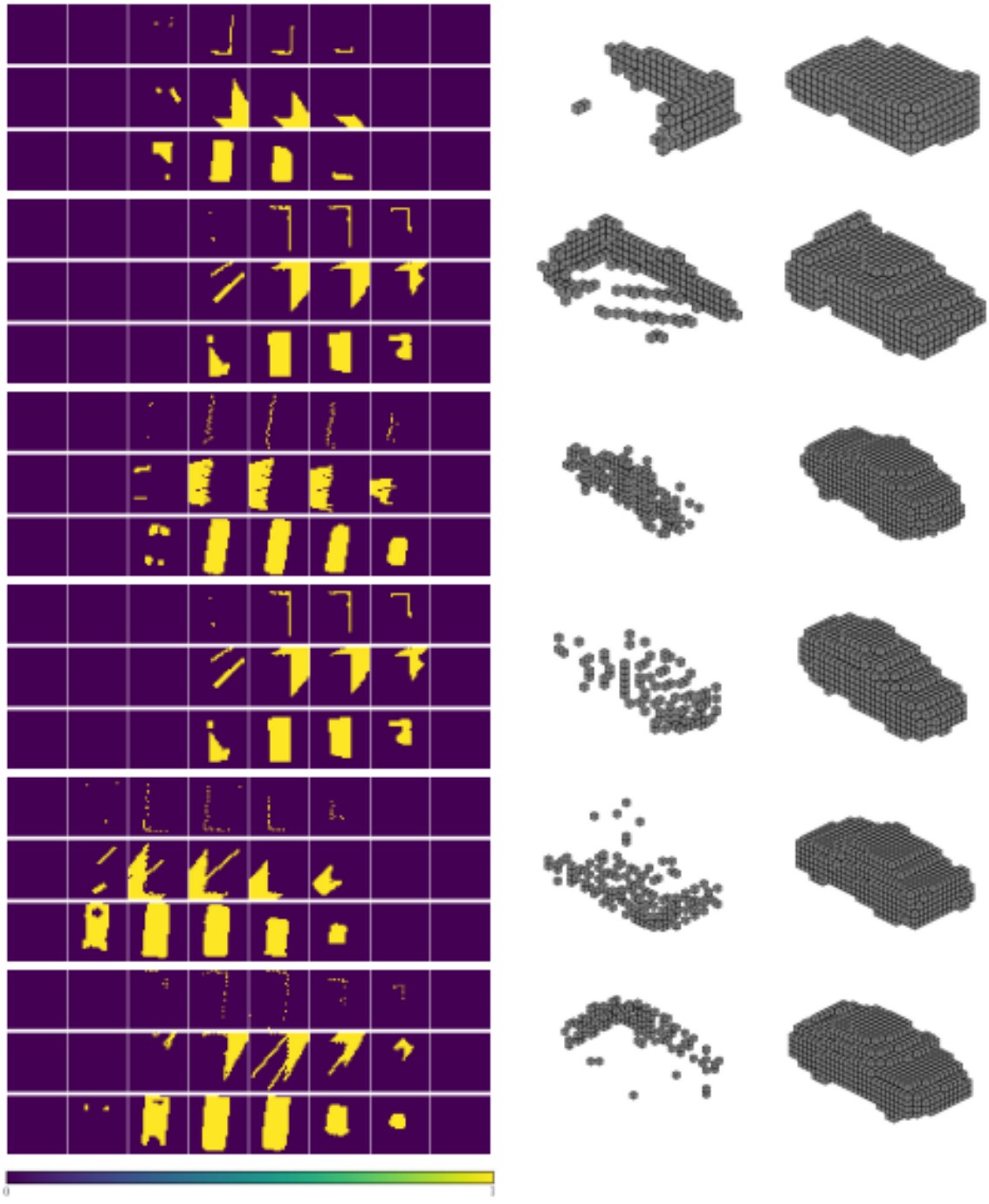

Рисунок Ж.1 — Приклад зі згенерованих наборів даних ShapeNet

ДОДАТОК И

Приклад зі згенерованих наборів даних KITTI

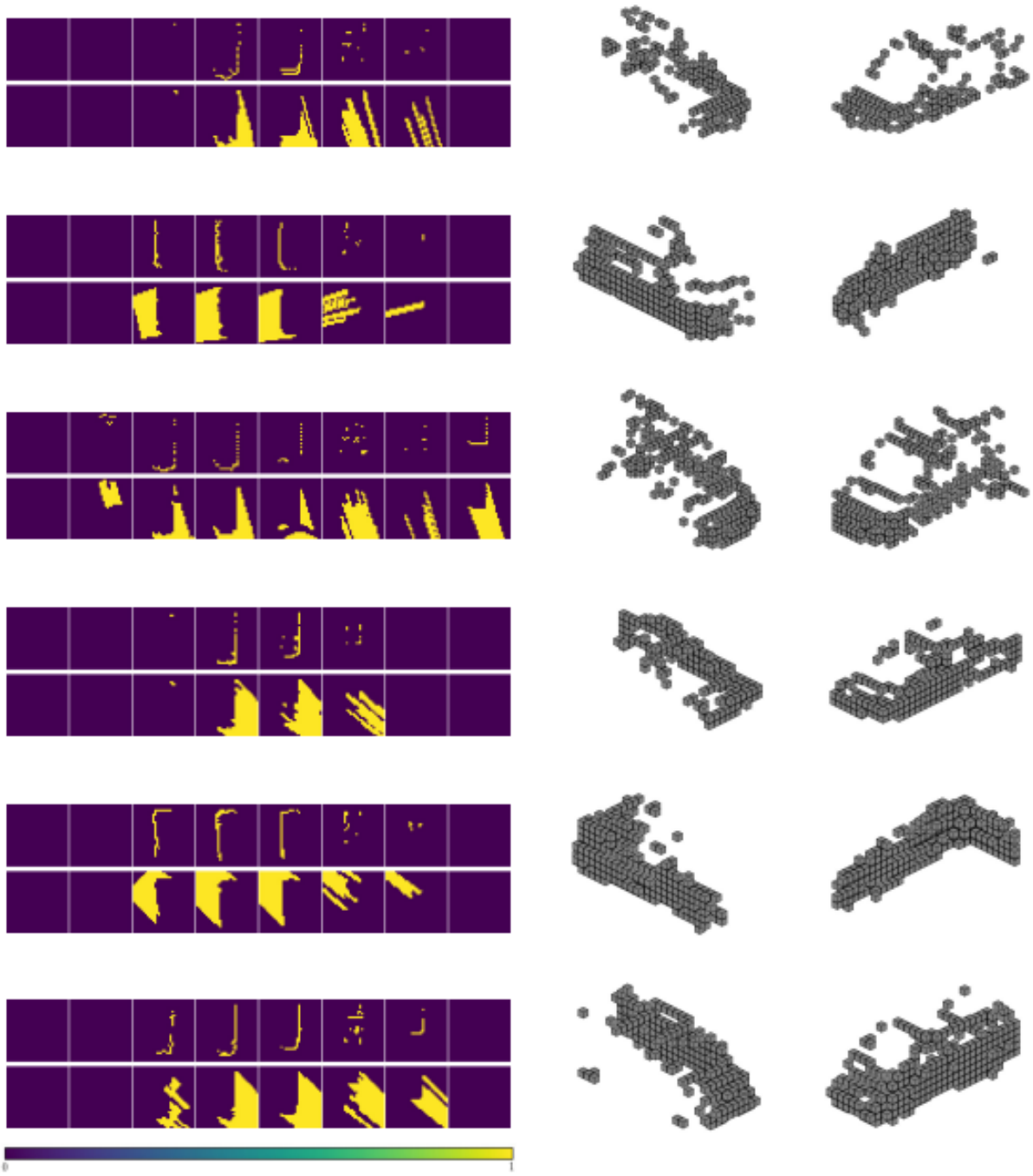

Рисунок И.1 — Приклад зі згенерованих наборів даних KITTI

ДОДАТОК К

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Штучна нейромережа глибокого навчання для моделювання 3D-форм об'єктів з розріджених хмар 3D-точок _____

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 98,3% Схожість 1,7%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Підцерковний Є. О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Кожем'яко А. В.
(підпис) (прізвище, ініціали)