



Advancing Language-driven Scene Synthesis with AI Text Prompt Generation

Hongting Liu¹

MSc in Robotics and Computation

Supervisor: Dr. Baoru Huang

September 2024

¹**Disclaimer:** This report is submitted as part requirement for the MSc in Robotics and Computation at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

Human natural language input instructions are sometimes not well understood by the model. Existing LSDM (Language-driven Scene Synthesis using Multi-conditional Diffusion Model) frameworks integrate multiple conditions (such as text prompts, human actions, and existing scene objects) into the synthesis process to generate scenes. While this model has significantly improved the performance of previous single-condition models, the final scene generation results remain unsatisfactory.

This report introduces a method to enhance language-driven scene synthesis using AI-generated text prompts. This study is based on the original LSDM framework and uses the Groq AI API to generate text prompts at ten different temperature settings to create diverse language input. Additionally, this study adopts a comprehensive training method that combines multiple text prompts, significantly improving the model's generalisation ability and the accuracy of scene generation. These findings demonstrate the potential of AI-generated text prompts to enhance scene synthesis and prove the feasibility of integrating AI into model training.

Contents

1	Introduction	2
1.1	Background	2
1.2	Basic Principles of the Original Model (LSDM)	4
1.2.1	Multi-Conditional Framework	4
1.2.2	Diffusion-Based Framework	4
1.2.3	Guiding Points: Directing the Denoising Process	5
1.2.4	Improved Semantic and Spatial Consistency	5
1.2.5	Flexibility and Scalability	6
1.3	Motivation	6
1.3.1	Enhancing the Scene Synthesis Process through Text Prompts	6
1.3.2	Contributing to Large-Scale AI-Language Models	7
1.3.3	Exploring Parameter Influence on Model Performance	7
1.3.4	Strategic Contributions to AI Research	8
2	Methodology	9
2.1	Overview	9
2.1.1	LSDM Architecture	9
2.1.2	Theoretical Support and Application	11
2.1.3	Integration of Conditions	11
2.2	Kanban Board	11
2.3	PRO-teXt Dataset	12
2.3.1	Creation and Composition	12
2.3.2	Utility and Application	12
2.3.3	Impact on Research and Development	13
2.3.4	Availability	13
2.4	Groq AI Implementation	13
2.4.1	Overview of Groq AI	13
2.4.2	Text Prompt Replacement Using Groq AI	14
2.4.3	Implementation Details	14
2.5	Rationale for Focussing on SDM Baseline	15
3	Experiments	16
3.1	Experimental Setup	16
3.1.1	Training Machine Hardware	16
3.1.2	Training Operating System	17
3.1.3	Choice of AI API	18

3.1.4	API Rate Limit Comparisons	19
3.1.5	Decision and Implementation	20
3.2	Data Preparation	21
3.2.1	Original Dataset Acquisition	21
3.2.2	AI-Driven Text Prompt Modification	22
3.3	Training Process Overview	24
3.3.1	Setup and Configuration	24
3.3.2	Execution	25
3.3.3	Monitoring and Adjustments	26
3.4	Batch Processing Strategy	26
3.5	Problems Identified During the Conduct of the Experiment	28
3.5.1	Environment Configuration Issues	28
3.5.2	Training Process Issues: Text Prompt Format Inconsistencies	29
4	Results	31
4.1	Preliminary Comparison	31
4.2	Comprehensive Training	34
4.3	Visualization Tests	35
4.3.1	Comparison of Single Object Scene Generation	35
4.3.2	Comparison of Multi-object Scene Generation	36
5	Discussion	38
5.1	Analysis of Model Performance vs. Expected Results	38
5.2	Impact of Text Prompt Replacement on Natural Language Understanding	38
5.3	Benefits and Challenges in Model Training with Diverse Data	39
6	Future Improvement	40
7	Conclusion	42
A	The Code of Groq AI API Implementation	48
B	Training Result	51
B.1	Training Result of Original Text Prompt	51
B.2	Training Result of Batch 1	52
B.3	Training Result of Batch 3	53
B.4	Training Result of Batch 5	54
B.5	Training Result of Batch 8	55
B.6	Training Result of Batch 10	56
C	Comprehensive Training Result	57

List of Abbreviations

LSDM - Language-driven Scene Synthesis using Multi-conditional Diffusion Model

AI - Artificial Intelligence

SDM - Scene Dynamics Model

API - Application Programming Interface

GPU - Graphics Processing Unit

CPU - Central Processing Unit

RAM - Random-access Memory

SSD - Solid State Drive

WSL - Windows Subsystem for Linux

MLP - Multi-Layer Perceptrons

EMD - Earth Mover's Distance

CLIP - Contrastive Language-Image Pretraining

List of Figures

1	LSDM methodology overview from the original paper.	10
2	Screenshots of Kanban Board in use.	12
3	The architecture of override text prompt methodology overview. . . .	15
4	ChatGPT API limits are detailed in their official documentation. . .	19
5	Groq API limits are detailed in their official documentation.	20
6	How to upgrade ChatGPT API limits to the next tier.	21
7	Parts of training logs.	26
8	Python script used for checking the file lines.	30
9	Comparison of single object scene generation between the original and comprehensive batch.	36
10	Comparison of multi-object scene generation between the original and comprehensive batch.	36

List of Tables

1	Examples of different batches of grammar between different temperature parameters (T).	23
2	Preliminary comparison between different temperature parameters (T).	32
3	Comprehensive training result comparative.	34

Chapter 1

Introduction

This chapter provides an overview of the current landscape in scene synthesis, focusing on the evolution from traditional methods to advanced AI-driven approaches. It introduces the concept of diffusion models and their application in creating dynamic, interactive environments. The chapter also outlines the challenges in integrating multiple conditions, such as text prompts, human motion, and object interactions, into the scene synthesis process. Furthermore, it presents the motivation behind this research, which aims to enhance the capabilities of existing models by using optimised text prompts generated by AI language models. The chapter concludes by highlighting the potential contributions of this study to both the technical field of AI and its practical applications in areas such as virtual reality, robotics, and interactive simulations.

1.1 Background

Scene synthesis has used several methods to create dynamic and interactive settings for past endeavours, mainly in robotics, gaming, simulations, and virtual reality applications [1]. Historical approaches, such as rule-based or procedural generation, have been extensively used to construct static or programmed scenes. Nevertheless, these methods often lack flexibility and scalability, especially when confronted with complicated settings or requiring immediate adaptation [2]. Given the increasing complexity and interactivity of settings, there is a growing need for advanced and independent generating methods.

Especially in generative modelling, the emergence of machine learning has dramatically broadened the possibilities of this discipline. Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are two advanced deep-learning methods that have helped make scene generators that are more accurate and flexible. These models can extract knowledge from extensive datasets to create novel settings that are both varied and lifelike [3]. Despite the achievements of these mod-

els, shortcomings persist in their capacity to process multi-modal inputs (such as text, motion, and object characteristics) and produce scenes that are both logically consistent and visually precise.

A new development in AI called diffusion models has a lot of potential because they can get high-quality, diverse samples from sampling distributions that are hard to understand [4]. Using a gradual transformation of random noise into structured and meaningful data, these models are especially well-suited for jobs that need precise control over the creation process. In contrast to GANs or VAEs, diffusion models can accommodate a broader spectrum of variables during the generation process, therefore providing more control over the ultimately produced output [5]. Consequently, diffusion models have enjoyed growing acceptance in several applications, including image synthesis, text-to-image creation, and, more recently, scene synthesis.

Although diffusion models have shown considerable progress, some obstacles remain in their implementation for real-world scene synthesis. First, incorporating many factors, such as textual suggestions, interactions with objects, and human movement, continues to be complicated. Existing models often have difficulties managing complex interdependencies among various factors, leading to less precise or credible scene creation. Furthermore, diffusion models have shown much promise in producing high-quality results. Still, they need a lot of computing power and often need a lot of resources for both training and inference [6]. These elements challenge real-time applications, especially in fields such as virtual reality and robotics, where rapid and adaptable scene processing is essential.

Furthermore, the interaction between human language input and visually produced output by machines remains a fundamental obstacle. Conventional methods for scene synthesis based on language sometimes struggle to accurately represent the subtleties of actual language, especially when handling abstract or context-dependent commands. For example, a basic directive like ‘position a chair adjacent to the window’ might be understood in several ways based on the arrangement of space, categories of things, and even the inferred connections among objects. An ongoing research objective is to improve the comprehension of such instructions in the model while guaranteeing that the created scene corresponds to user expectations.

Considering these difficulties, it is evident that further scholarly investigation is necessary to enhance the capacity of diffusion models to generate complicated scenarios. This includes improving their management of multi-conditional inputs, optimising their processing efficiency, and enhancing their natural language comprehension skills. This study looks into how optimised text prompts made by AI language models like ChatGPT might help improve scene synthesis by making the outputs more in line with what users want. The purpose is to enhance current diffu-

sion models by adding a stronger, language-based method for creating scenes with multiple conditions. This will help AI and real-time environment creation in a big way.

1.2 Basic Principles of the Original Model (LSDM)

The model presented in the paper ‘Language-Driven Scene Synthesis Using Multi-Conditional Diffusion Model’ [7] introduces an innovative approach to scene synthesis. This model integrates multiple conditions (such as text prompts, human motion, and existing scene objects) into the synthesis process. Unlike previous models that relied on a single condition to direct the synthesis process, this approach employs a multi-conditional framework that enhances the capacity to generate comprehensive and contextualised scenes.

1.2.1 Multi-Conditional Framework

Conventional scene synthesis models often concentrate on a single input condition, such as text prompts or item locations, restricting their capacity to produce scenes with complicated contextual connections. Specifically, models that depend just on text inputs may have difficulties in accurately placing items in a manner that is both geographically and semantically significant, especially when there is a requirement to consider human movement or interactions between objects. The present model has implemented a multi-conditional framework to overcome these constraints by concurrently incorporating many inputs, hence enhancing the flexibility and responsiveness of the synthesis process.

The LSDM framework enables the model to systematically analyse and integrate different situations, including human postures, item locations, and textual descriptions, in a cohesive way. For instance, when provided with a text instruction such as ‘position the table adjacent to the chair and have an individual seated on the chair,’ the model takes into account not only the physical connections between the items (the table and the chair) but also the human posture and interaction (sitting on the chair). This comprehensive approach ensures that the physical and contextual elements of the created scenes and cues are consistent, resulting in more realistic scenes that better match user expectations.

1.2.2 Diffusion-Based Framework

The LSDM proposed approach uses a diffusion-based framework, a probabilistic model that facilitates a shift from noise to structured output using a denoising process. The effectiveness of diffusion models in producing high-quality and realistic data stems from their ability to reverse a noising process. This process begins

with random noise and is refined repeatedly to provide structured returns. Within scene synthesis, the model can produce scenes that go from abstract and noisy representations to completely actualised and complicated worlds [4].

By using the power of diffusion models to change the multi-conditional inputs in a way that makes sense and is consistent, this model stands out because it can handle conditions that depend on each other in complicated ways. The stepwise refinement process strengthens the model when dealing with uncertainty or missing inputs by letting it change and fix the scene-generating techniques on the fly.

1.2.3 Guiding Points: Directing the Denoising Process

‘Guiding points’ are used to direct the denoising process. These are explicitly expected throughout the diffusion process and act as anchors that guide the generation process towards realistic and semantically meaningful configurations. This set of guiding points serves as intermediary goals, offering spatial and contextual indicators that assist the model in aligning the produced scene with the input circumstances.

For example, in a scenario portraying a person engaging with an item, the guiding points may align with significant interaction locations, such as the point of contact between the hand and the object or the position of the feet about the ground. By predicting these guiding points early on in the diffusion process, the model makes sure that the picture it makes follows both the physical limits of the objects and the interactions that were described in the input [6]. This approach enables the model to sustain a significant level of semantic coherence throughout the generation process, guaranteeing that the end scene seems authentic and has logical connections between items and actors inside the scene.

1.2.4 Improved Semantic and Spatial Consistency

This model enhances the semantic and spatial coherence of the rendered scenes by including guiding points and the diffusion-based framework. Earlier models often had difficulties preserving coherence in complicated multi-object or human-object interactions, such as the ATISS model [8], leading to outputs that exhibited fragmentation or lacked objectivity.

This technique is particularly advantageous in applications like virtual reality, where users anticipate engaging and responsive experiences. This explicit grounding of scene production enables the model to produce scenes that dynamically adjust to user inputs in real-time, preserving the seamless interaction between the user and the environment.

1.2.5 Flexibility and Scalability

Furthermore, this approach has the added benefit of being versatile in managing diverse scene production activities. This multi-conditional architecture may be readily modified and add extra elements, such as illumination, camera angles, or even more conceptual factors like user preferences or emotional tone. This characteristic renders the model very adaptable for many applications, including robotics, virtual reality, gaming, and film production, where a wide range of distinct and ever-changing settings are necessary.

Undoubtedly, the LSDM is a notable breakthrough in the field of scene synthesis technology. By integrating a multi-conditional framework and using diffusion models, this approach offers a resilient method for producing complicated and contextually precise scenarios. The use of guiding points guarantees that the produced scenes retain both semantic and spatial coherence, making this approach especially suitable for real-time applications that need interaction and responsiveness.

1.3 Motivation

The motivation behind refining and extending the capabilities of scene synthesis models through the utilisation of enhanced text prompts is driven by several key factors. This project primarily focused on improving the accuracy, flexibility, and applicability of the synthesis process. Given the growing integration of scene synthesis into practical applications such as virtual reality, robotics, and automated content production, producing realistic and contextually suitable scenes from user-generated inputs has become a crucial problem. This research aims to tackle this problem using sophisticated AI language models and novel diffusion-based synthesis methods.

1.3.1 Enhancing the Scene Synthesis Process through Text Prompts

The main goal of this research is to incorporate text prompts that AI language models dynamically generate at different ‘temperatures’ to improve the logical consistency and authenticity of created scenarios. Temperature is a hyperparameter that controls the diversity of AI-generated text. Reduced temperatures provide more predictable results, while increased temperatures create a more comprehensive range of distinctive and innovative writing. This study aims to investigate the equilibrium between originality and fidelity in scene production by manipulating the temperature. This will enable the model to produce scenes that better correspond to user expectations.

The present adaptive system evaluates the adaptability and resilience of the initial model, therefore enabling the creation of scenes that can effectively react to

a diverse range of user inputs and practical scenarios. Unlike conventional scene synthesis models that depend on fixed or predetermined inputs, the capacity of the system to adapt text prompts in real-time enhances its responsiveness to user-driven customisation and interaction. This functionality is essential for real-time applications like virtual environments, interactive gaming, and automated content generation [9].

Moreover, the introduction of variability in the text prompts poses a challenge to the ability of the model to process increasingly complicated and subtle requirements effectively. Using a range of prompts, the ability of this model to generalise across different situations is tested. This makes the model more reliable in real-world situations where user inputs differ. In businesses that heavily depend on dynamic settings, such as simulations for autonomous systems or robotics, where the model must properly comprehend and handle both structured and unstructured user inputs, this flexibility is especially crucial [10].

1.3.2 Contributing to Large-Scale AI-Language Models

The second primary objective is to contribute to the broader AI language model development domain by enhancing the training procedures for large-scale models such as ChatGPT. Although AI has significantly emphasised language understanding and production, very little consideration has been given to adapting these models for multi-modal tasks, such as scene synthesis. This study looks into how enhanced text prompts might help both the visual synthesis process and teaching language models how to understand and respond to situations with many different factors. [11].

By examining the impact of improved prompts on scene production, this study may provide valuable insights for the development of future AI systems that need the simultaneous processing of complicated, multi-modal input such as text, photos, and 3D sceneries. This is consistent with the increasing need for sophisticated AI systems to understand and produce refined and contextually aware material [12]. Applications such as smart homes, autonomous vehicles, and virtual assistants need AI systems to comprehend language and the physical and geographical environment in which instructions are provided. Therefore, the approaches used in this study might have a crucial function in closing the divide between language processing and perception of real-world scenes.

1.3.3 Exploring Parameter Influence on Model Performance

Another important reason for doing this study is to comprehend the impact of different input factors, such as temperature settings and the format of text prompts, on the overall effectiveness of the scene synthesis model. This study provides a deep

understanding of the basic workings of high-performance language models when used for tasks other than just text production by looking at how these factors affect the quality and accuracy of the scenes that are produced.

For example, this experiment examines the impact of different text prompts on the directing points in the diffusion process, which subsequently alter the semantic and spatial precision of the produced scenes. Understanding how these interactions work is essential for making progress in automated scenario creation, where models need to be able to adapt and learn from a wide range of inputs in real-time. This will make the generative process more like human creativity and understanding [13].

1.3.4 Strategic Contributions to AI Research

Furthermore, this initiative aims to demonstrate a broader and more strategic goal beyond the specific technological improvements. This research is essential because it will help make AI systems that can solve hard generative problems easier by making textual data interact with other types of data, like 3D point clouds in scene synthesis. In addition, these systems will not only be more responsive to user input but will also produce scenarios and environments with greater adaptability, interactivity and alignment with human cognitive processes.

Finally, the combined contributions in this project to scene synthesis technology and AI language model training highlight its importance. A systematic approach is presented in the study that could make it easier for large-scale AI models, like ChatGPT, to understand and respond in multimodal settings that are aware of the situation. In addition, the results could provide valuable contributions to practical applications in areas such as virtual reality, robotics and interactive simulations, providing tangible benefits to businesses that depend on creating dynamic environments.

Fundamentally, this research aims to overcome the limitations of existing scene synthesis models and make a valuable contribution to the broader field of AI language model development. Improving the interaction between text prompts and scene generation can enhance technical capabilities and practical applications. Ultimately, this could push the boundaries of what artificial intelligence systems can achieve in understanding and creating complex, real-world environments.

Chapter 2

Methodology

This chapter outlines the methodological approach employed in this research to enhance language-driven scene synthesis. It begins with an overview of the LSDM architecture, detailing its essential components and innovative features. The chapter then describes the PRO-teXt dataset used for training and evaluation and introduces the Groq AI platform for text prompt generation and replacement. A significant focus is placed on implementing Groq AI to enhance text diversity and model robustness. The methodology also covers the project management approach using a Kanban board, ensuring efficient progress tracking and communication. Finally, the chapter explains the rationale behind focussing on the SDM baseline for optimisation, considering the constraints of computational resources and time. This comprehensive methodology aims to increase the scene precision of the synthesis, adaptability, and effectiveness when using natural language inputs.

2.1 Overview

As mentioned in section 1.2, this project is an optimisation and improvement based on LSDM. The LSDM model has been developed to synthesise scenes by integrating several conditions, including human movement, existing objects, and textual commands. The model is structured in such a way as to facilitate the efficient processing of these disparate inputs, which are then employed to generate 3D scenes contextually relevant.

2.1.1 LSDM Architecture

The most significant innovation of the LSDM model is its Guiding Points Network, which integrates all conditions to generate guiding points (hat-S). These guiding points play a crucial role in the denoising process of the diffusion model, directing it towards the generation of the final scene synthesis [14]. The following section

provides a detailed breakdown of the components and processes involved in the LSDM model, as depicted in the provided methodology overview (Figure 1) [7]:

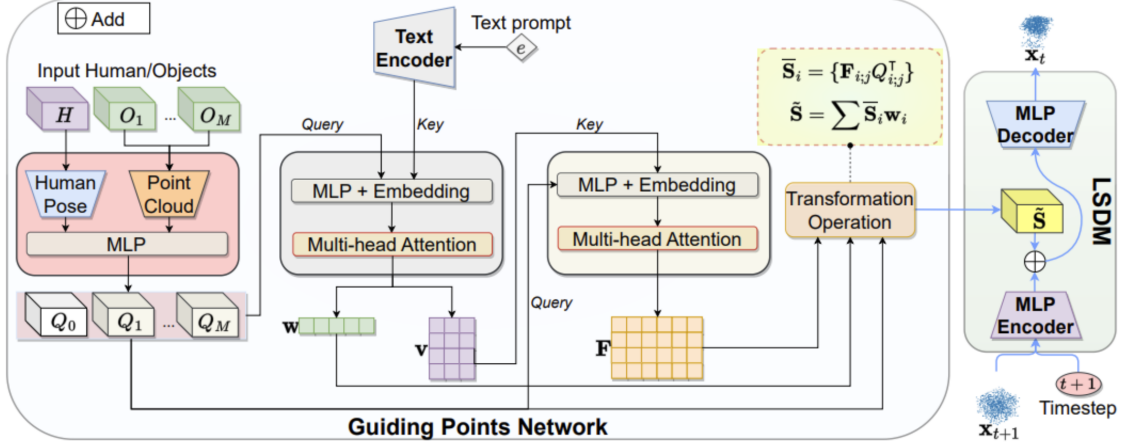


Figure 1: LSDM methodology overview from the original paper.

Here are the explanations of the essential components:

- **Input Layer:** The model can accept an input of human poses and a set of objects represented in a three-dimensional space. Each object, including the human pose, is converted into a point cloud representation, facilitating the integration of spatial and shape information into the model.
- **Feature Extraction:** The processing of point clouds through MLP enables the extraction of detailed features at the point level. A text encoder processes a text prompt, which encodes the text prompt from users, to extract pertinent linguistic features.
- **Guiding Points Network:** The extracted features from the point clouds and the text encoder are subjected to a multi-head attention mechanism. This mechanism is critically important in assessing and weighing the influence of each input feature on the scene synthesis. The model performs transformation operations based on the weighted features, integrating all the information to produce guiding points.
- **Denoising and Scene Generation:** The aforementioned guiding points are employed in the denoising stages of the diffusion process. They provide spatial and contextual cues that direct the noise reduction process, gradually refining the noise to synthesise the final scene. The iterative LSDM process through the MLP decoder and encoder ensures that the scene details improve with each timestep, getting closer to the desired output.

2.1.2 Theoretical Support and Application

The inclusion of guiding points, which facilitate the improvement of the denoising process, supports the theoretical foundation of the LSDM. This not only renders the model innovative but also empirically robust. The ability of this model to combine and understand different input types through a diffusion process makes it possible to create realistic, complex scenes that follow textual instructions.

2.1.3 Integration of Conditions

A noteworthy attribute of the LSDM model is its capacity to address multiple conditions concurrently. It is a substantial advancement over conventional scene synthesis techniques that may only consider isolated factors such as object placement or human motion. Integrating text prompts into the scene synthesis process enables the LSDM model to facilitate new avenues for user-driven customisation and interactive scene design [7].

2.2 Kanban Board

To ensure the efficient management of the research project and facilitate consistent communication with the supervisor, a digital Kanban board (Supported by Teamhood) was employed [15]. This tool enabled the systematic tracking of research activities, deadlines, and progress updates, thus facilitating a structured and transparent approach to handling research tasks. Furthermore, it allowed for flexible adjustments to the project timeline based on ongoing findings and feedback from weekly meetings with the advisor (Figure 2).

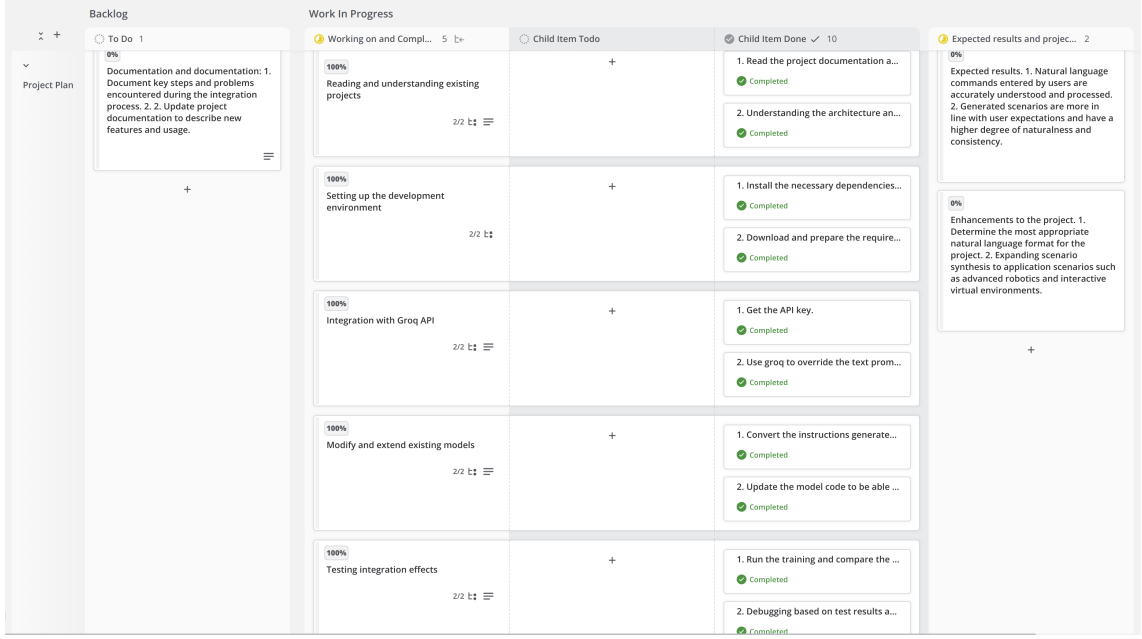


Figure 2: Screenshots of Kanban Board in use.

2.3 PRO-teXt Dataset

The PRO-teXt dataset comes from a new way of solving 3D human pose ambiguity by adding 3D scene constraints, which was first described by Mohamed Hassan et al. in their work presented at ICCV 2019 [16]. This dataset is essential for training models to understand and predict human motion in naturally occurring environments. It is relevant for applications that require precise interaction between humans and their environment.

2.3.1 Creation and Composition

The dataset has been carefully assembled to include 12 different 3D scenes accompanied by RGB sequences of 20 people interacting within these scenes. It uses the SMPL-X model, which significantly refines the capture of human pose by incorporating realistic interaction with scene geometry. This model tuning allows for a much more accurate representation of human movement, considering environmental constraints such as object contact and inter-penetration avoidance.

2.3.2 Utility and Application

The dataset helps the creation of more advanced pose estimation algorithms by limiting the human model so it can not go outside the scene and by letting it interact with its surroundings realistically. These algorithms may provide results that are

both visually realistic from a camera standpoint and conform to the physical characteristics of the scene, therefore guaranteeing the predictability of the estimations in three-dimensional space.

2.3.3 Impact on Research and Development

A significant development in human posture assessment techniques is the incorporation of scene limitations. In addition to enhancing algorithmic accuracy, the PRO-teXt dataset is a vital resource for research focused on comprehending human-scene interaction. It facilitates several applications ranging from augmented reality to motion analysis in sports and ergonomics, emphasising the reciprocal impact between the set posture and its contextual environment.

2.3.4 Availability

The dataset, along with the code and ancillary materials, will be made available to the research community to ensure that it can contribute widely to advancing both academic and applied sciences in human motion analysis. The dataset can be accessed and used for educational and research purposes under certain conditions, thus encouraging further research and development in the field.

2.4 Groq AI Implementation

This section detailed the implementation of the Groq AI platform in enhancing the text prompt inputs for a scene synthesis model. The method flow of the Implementation of the Groq API is also introduced to clearly understand how to leverage the capabilities of Groq AI to optimise the quality of input data and the overall performance of the model.

2.4.1 Overview of Groq AI

The Groq AI platform is renowned for its capacity to process voluminous datasets and complicated computational operations with remarkable efficiency [17]. In particular, Groq provides API that facilitates the training and inference of models at unprecedented speeds and with unparalleled accuracy in the context of deep learning and machine learning applications. In this project, the input of users in the form of natural language is primarily employed to generate scenes. Consequently, the visual interpretation of a sentence assumes great significance [18]. The primary benefit of the platform is in its meticulously designed hardware and software co-design, allowing it to save power use while ensuring consistently high throughput rates. This is

a crucial characteristic for language models that process large volumes of text data [19].

2.4.2 Text Prompt Replacement Using Groq AI

In this project, I utilise the Groq AI API to enhance and supplant the text prompt within the dataset. The following considerations inform these steps:

- **Enhanced text diversity:** Given that the text prompts in the original dataset may only have limited semantic complexity, integrating varied Groq AI-produced text into the dataset can increase text variety. This technique enables the model to acquire more comprehensive language terms, enhancing comprehension and ensuring more precise production of scene descriptions that correspond to the ideas of the users.

- **Enhanced model robustness:** Including text produced at different temperature conditions enables the model to undergo training in the wider variety of text inputs, improving its capacity to adapt and withstand different language use.

- **Experimental control:** Efficient control of the experimental procedure is implemented to guarantee the dependability of the obtained findings. The Groq AI API enables meticulous manipulation of text creation traits, such as inventiveness, which is manageable by changing temperature settings. It also allows the repetition of and control of studies to guarantee dependable and scientifically rigorous outcomes.

2.4.3 Implementation Details

The particular methodology for implementation is as follows:

1. **API call:** An API call is made to set the parameters of the API according to the requirements of each scene. These parameters include the length, complexity, and temperature settings of the text. After that, a request is sent through the Groq AI API to get the generated text.
2. **Data preprocessing:** The text is received through a preprocessing step, including syntax checking and semantic integration, to ensure its suitability for subsequent model training.
3. **Model training input:** The processed text is employed as the input for the model, supplanting the original text prompts for training the scene synthesis model.
4. **Performance monitoring:** Throughout the training process, the response of the model to the new text is monitored, its impact on scene generation is evaluated, and any necessary adjustments are made.

In light of this concept, an amendment was made to the text prompt input architecture of the original project. The particular process is illustrated in the figure below (Figure 3):

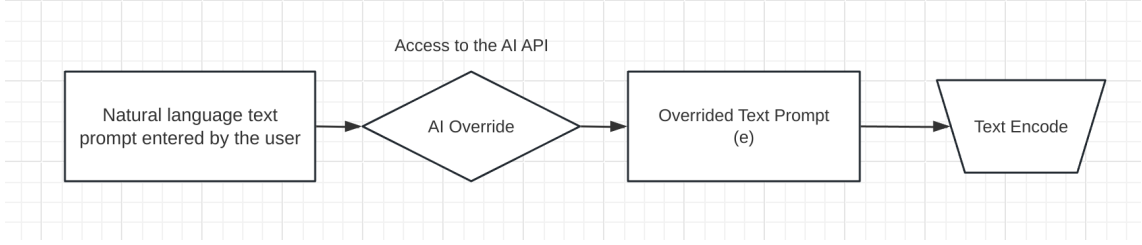


Figure 3: The architecture of override text prompt methodology overview.

This approach allows this model to enhance the quality of the data input and the overall performance of the model, thereby enabling it to generate scenarios that are more natural and in line with user expectations.

2.5 Rationale for Focussing on SDM Baseline

The original GitHub project included many baselines for scene synthesis and model training. Considering the substantial computing resources and time needed to train each baseline, choosing one for targeted optimisation was imperative. I determined that the ensuing work should concentrate on the SDM (Scene Dynamics Model) baseline.

Within the given hardware and time limitations for this project, the SDM baseline was determined to be the most suitable option for optimisation. This approach achieved an ideal equilibrium between the computing requirements and the possibility for substantial performance improvements [20]. Optimisation of the SDM baseline allowed the discovery of universal enhancements that may be extended to other baselines, laying the groundwork for broader applicability in the field.

Chapter 3

Experiments

This chapter details the experimental setup and procedures employed in this research to enhance language-driven scene synthesis. It begins with a comprehensive overview of the hardware specifications and operating system choice, emphasising using a high-performance personal laptop running Ubuntu 24.04 LTS for its advantages in scientific computing. The chapter then delves into the selection process for the AI API, comparing Groq AI and ChatGPT and justifying the choice of Groq AI based on its superior rate limits and cost-effectiveness. Much of the chapter is dedicated to data preparation, describing the acquisition of the PRO-teXt dataset and the AI-driven text prompt modification process using Groq AI. The training process includes setup, execution, and monitoring strategies, focussing on batch processing and optimisation techniques. Finally, the chapter addresses various challenges encountered during the experiment, particularly in environment configuration and text prompt format inconsistencies, providing solutions and recommendations for future implementations. This comprehensive experimental approach aims to rigorously test and validate the proposed enhancements to the scene synthesis model.

3.1 Experimental Setup

This section mainly introduces the initial preparation work for the experiment, including the choice of experimental hardware, system environment and AI API.

3.1.1 Training Machine Hardware

All of the training process is run on my personal laptop. The following are the hardware component details of my laptop:

- **GPU:** NVIDIA GeForce RTX 4090, equipped with 24 GB GDDR6X VRAM [21]. This GPU is well-known for its high processing power and efficiency, making it ideal for deep learning tasks involving large datasets and complex neural network architectures [22].

- **CPU:** The Intel Core i7-14700K is a high-end desktop processor featuring 12 cores (8 performance and 4 efficient cores) and 20 threads [23]. The CPU was selected based on its robust performance in handling multiple tasks and its capacity to efficiently manage the computational demands of data preprocessing and model training.
- **RAM:** The 64 GB DDR5 memory provides sufficient capacity to accommodate the substantial memory requirements of large-scale machine-learning models during the training phase.
- **Storage:** A 4 TB NVMe SSD is provided to ensure rapid data read and write speeds, essential for reducing training time when dealing with large data volumes.

The selected hardware components were chosen to optimise the training speed and model performance, thus providing a robust foundation for conducting extensive experiments and achieving reliable results.

3.1.2 Training Operating System

In scientific computing, the selection of an operating system can considerably impact the efficiency, performance and ease of development of a project [24]. Concerning the operating system selection, I decided to install the native Ubuntu 24.04 LTS, a Linux distribution system, on my laptop. This section presents the rationale for choosing Ubuntu over Windows for projects.

The Ubuntu 24.04 LTS operating system offers a ‘minimal’ installation option, which installs only those applications deemed essential for basic functionality, such as the Firefox web browser and system tools. This approach aims to provide a streamlined setup, reducing the number of unnecessary components [25]. The system maintains its capacity to support various hardware and software by implementing ongoing compatibility enhancements. Performance enhancements are attributable, in part, to the incorporation of the most recent Linux kernel, which facilitates more effective management of hardware resources and enhances system responsiveness [26]. This is especially useful for subsequent environment configuration and project development.

Ubuntu 24.04 LTS is supported for five years, with an extended support option available through Ubuntu Pro, which offers an additional seven years of support, resulting in a total support period of 12 years. This makes it a stable and long-term option for enterprises and professional users [27]. This means that after the development of this project is completed, the original system environment can be used for subsequent work for a long time.

Compared with Windows, Linux (Ubuntu) has a lot of advantages for scientific computing. First and foremost, Ubuntu is an open-source operating system. The open-source framework of Ubuntu permits unparalleled customisation and modification without licencing constraints, rendering it an optimal choice for bespoke scientific research requirements [28]. In addition, Linux also has several essential Python libraries, including NumPy, SciPy, and OpenCV, which are natively supported and optimised on Linux platforms. This support encompasses fundamental functionalities and the management of complicated dependencies by utilising productive package managers, such as Advanced Package Tool (APT) [29].

Furthermore, Linux systems are renowned for their superior performance in managing extensive computations and multiple processes, a critical requirement for scientific applications. This capability is essential for real-time processing tasks and high-load computations, prevalent in data-intensive fields[30].

Finally, a sizeable and dynamic community reinforces a robust support system for those utilising the Linux operating system. This, in turn, facilitates a more straightforward process of troubleshooting and the implementation of enhancements within the development ecosystem [31].

However, there are many challenges with Windows in scientific computing. Windows may present compatibility challenges, particularly when libraries necessitate extensive system access or specific architectural requirements, such as those about Mesh-to-SDF, OpenMesh, and Chumpy used in this project. These issues originate from significant discrepancies between the fundamental architectural and file system management paradigms of the Windows and Linux operating systems [32]. Development tools like IPDB, TensorBoard, and others integrate more seamlessly with Linux, offering a more robust and efficient development workflow than Windows environments.

The technical evaluation and comparative analysis results indicate that Ubuntu provides a more conducive environment for scientific computing than Windows. The combination of open-source flexibility, robust library support, superior performance, and an active community renders Ubuntu the optimal choice for this project. The alignment of these strengths will likely enhance the efficiency and effectiveness of this project, ensuring a streamlined development process and reliable execution of computational tasks.

3.1.3 Choice of AI API

For the experimental setup of the project, selecting the appropriate AI API was critical to ensure efficient text prompt generation and replacement. The project required the generation of multiple text prompts across multiple batches, each containing 183 text prompt documents. Given the data generation scale, the rate limits

of API were an important consideration. In the next section, I will describe the reasons for choosing Groq AI API usage in more detail.

3.1.4 API Rate Limit Comparisons

Both the Groq AI API and the ChatGPT API were considered. However, their rate limits were significantly different, which influenced the final choice:

- **ChatGPT API Limits:** The ChatGPT API offers a free tier that allows up to 3 requests per minute and 200 requests per day [33]. Users must spend a minimum amount to unlock higher tiers to increase their usage limits. For example, reaching Tier 1 requires a payment of at least \$5, which allows up to 500 requests per day and 30 requests per minute. This limit was insufficient for the needs of this project, given the high number of hits required daily.

Free	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Free tier rate limits This is a high level summary and there are per-model exceptions to these limits (e.g. some legacy models or models with larger context windows have different rate limits). To view the exact rate limits per model for your account, visit the limits section of your account settings.					
MODEL	RPM	RPD	TPM	BATCH QUEUE LIMIT	
gpt-3.5-turbo	3	200	40,000	200,000	
text-embedding-3-large	3,000	200	1,000,000	3,000,000	
text-embedding-3-small	3,000	200	1,000,000	3,000,000	
text-embedding-ada-002	3,000	200	1,000,000	3,000,000	
whisper-1	3	200	-	-	
tts-1	3	200	-	-	
dall-e-2	5 img/min	-	-	-	
dall-e-3	1 img/min	-	-	-	

Free	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Tier 1 rate limits This is a high level summary and there are per-model exceptions to these limits (e.g. some legacy models or models with larger context windows have different rate limits). To view the exact rate limits per model for your account, visit the limits section of your account settings.					
MODEL	RPM	RPD	TPM	BATCH QUEUE LIMIT	
gpt-4o	500	-	30,000	90,000	
gpt-4o-mini	500	10,000	200,000	2,000,000	
gpt-4-turbo	500	-	30,000	90,000	
gpt-4	500	10,000	10,000	100,000	
gpt-3.5-turbo	3,500	10,000	200,000	2,000,000	
text-embedding-3-large	3,000	-	1,000,000	3,000,000	
text-embedding-3-small	3,000	-	1,000,000	3,000,000	
text-embedding-ada-002	3,000	-	1,000,000	3,000,000	
whisper-1	50	-	-	-	
tts-1	50	-	-	-	
tts-1-hd	3	-	-	-	
dall-e-2	5 img/min	-	-	-	
dall-e-3	5 img/min	-	-	-	

(a) ChatGPT API limits in Free Tier.

(b) ChatGPT API limits in Tier 1.

Figure 4: ChatGPT API limits are detailed in their official documentation.

- **Groq AI API Limits:** In contrast, the Groq AI API allows up to 30 requests per minute and up to 14,400 requests per day, far exceeding the limitations of the ChatGPT free tier [34].

Limits

These are the rate limits for your organization

Chat Completion

ID	REQUESTS PER MINUTE	REQUESTS PER DAY	TOKENS PER MINUTE	TOKENS PER DAY
gemma-7b-it	30	14,400	15,000	(No limit)
gemma2-9b-it	30	14,400	15,000	(No limit)
llama-3.1-70b-versatile	100	14,400	131,072	1,000,000
llama-3.1-8b-instant	30	14,400	131,072	1,000,000
llama-guard-3-8b	30	14,400	15,000	(No limit)
llama3-70b-8192	30	14,400	6,000	(No limit)
llama3-8b-8192	30	14,400	30,000	(No limit)
llama3-groq-70b-8192-tool-use-preview	30	14,400	15,000	(No limit)
llama3-groq-8b-8192-tool-use-preview	30	14,400	15,000	(No limit)
mixtral-8x7b-32768	30	14,400	5,000	(No limit)

Speech To Text

ID	REQUESTS PER MINUTE	REQUESTS PER DAY	AUDIO SECONDS PER HOUR	AUDIO SECONDS PER DAY
distil-whisper-large-v3-en	20	2,000	7,200	28,800
whisper-large-v3	20	2,000	7,200	28,800

Figure 5: Groq API limits are detailed in their official documentation.

3.1.5 Decision and Implementation

The practical constraints of request limits and the associated costs primarily influenced the decision to select an appropriate AI API. During the experimental phase, the project required extensive generation and replacement of text prompts, which required numerous API calls. The free level of ChatGPT only allows 3 requests per minute and 200 requests per day, which was insufficient for the needs of the project. As Figure 6 shows, users must spend more than \$5 on the free tier to upgrade to the next tier to unlock more requests [33]. Since this project calls the API mainly for text replacement, the amount of tokens actually spent is minimal, far from the cost of unlocking the next tier. This is a crucial reason for abandoning the ChatGPT API.

Usage tiers

You can view the rate and usage limits for your organization under the [limits](#) section of your account settings. As your usage of the OpenAI API and your spend on our API goes up, we automatically graduate you to the next usage tier. This usually results in an increase in rate limits across most models.

TIER	QUALIFICATION	USAGE LIMITS
Free	User must be in an allowed geography	\$100 / month
Tier 1	\$5 paid	\$100 / month
Tier 2	\$50 paid and 7+ days since first successful payment	\$500 / month
Tier 3	\$100 paid and 7+ days since first successful payment	\$1,000 / month
Tier 4	\$250 paid and 14+ days since first successful payment	\$5,000 / month
Tier 5	\$1,000 paid and 30+ days since first successful payment	\$50,000 / month

Figure 6: How to upgrade ChatGPT API limits to the next tier.

Groq AI offers a more suitable alternative with its ability to handle 30 requests per minute and up to 14,400 requests per day without a mandatory financial limitation. This capacity effectively supported the high frequency of requests required to process the datasets efficiently.

Therefore, Groq AI was selected for its superior service support, which matched the high-frequency, low-token usage pattern, ensuring efficient data processing of the project without unnecessary expenditure. This choice allowed for the seamless integration of generated text prompts, maintaining project continuity and cost-effectiveness.

3.2 Data Preparation

This section provides a detailed introduction to the preparation process of the original dataset.

3.2.1 Original Dataset Acquisition

The first step was to acquire the original PRO-teXt dataset, which contains various scenarios encapsulating human-object interactions in different 3D environments. This provided the baseline against which the modifications were applied using AI-generated text prompts. Here is the extension of the PRO-teXt dataset provided by LSDM contributors [7]: <https://forms.gle/AutfNYQEF6K9DRYs7>

3.2.2 AI-Driven Text Prompt Modification

To enhance the descriptive quality of the text prompts in the dataset and improve the resultant scene synthesis, I employed a sophisticated AI-driven approach utilising the Groq AI text generation API. API calls are implemented based on the Groq official documentation [35]. This section provides a detailed account of the automated script developed to facilitate the modification of text prompts across multiple batches, each exhibiting varying degrees of linguistic diversity. Full code details are provided in Appendix A.

This Python script supplied is an essential tool in the data preparation pipeline. The functionality of this system is to extract original text autonomously prompts from a designated directory, activate the Groq AI API to create altered versions of these prompts at various ‘temperatures,’ and then store these changes in separate batches for future use in training the models.

In the context of artificial intelligence (AI) language models, temperature is a parameter that determines the degree of randomness in the generated text. In this script, temperatures are set at values between 0.5 and 0.95, with increments of 0.05, generating ten distinct batches of text. The objective of each batch is to introduce a varying degree of novelty and creativity in the text prompts, thereby enabling an investigation of the impact of such variations on the fidelity of scene synthesis.

The table below (Table 1) displays the grammatical structure of natural language text in each training batch after the AI API has overwritten the original text prompt file at different temperatures:

List of different batches of grammar.	
Batch name	Grammar
Original batch	Place a single bed under me and next to the sofa.
Batch_1 (T = 0.5)	Put a single bed beneath me and beside the sofa.
Batch_2 (T = 0.55)	Could you position a single bed under me and adjacent to the sofa?
Batch_3 (T = 0.6)	I need a single bed placed right under where I'm standing, and it should also be next to the sofa.
Batch_4 (T = 0.65)	Arrange for a single bed to be situated directly underneath me and by the side of the sofa.
Batch_5 (T = 0.7)	Please ensure a single bed is set up under my current position and right alongside the sofa.
Batch_6 (T = 0.75)	I would appreciate it if a single bed could be installed beneath where I am now and near the sofa.
Batch_7 (T = 0.8)	Can you organize to have a single bed placed beneath where I'm located, as well as near the sofa?
Batch_8 (T = 0.85)	It would be helpful if you could install a single bed directly underneath me and also position it close to the sofa.
Batch_9 (T = 0.9)	I'm requesting that a single bed be positioned below my current spot and also to be placed adjacent to the sofa.
Batch_10 (T = 0.95)	Please accommodate a single bed right beneath my standing area and ensure it's placed next to the sofa as well.

Table 1: Examples of different batches of grammar between different temperature parameters (T).

Then, the process is initiated by verifying the existence of the designated directory structure, wherein the modified prompts are to be stored. For each temperature setting, a new subdirectory is created, thus ensuring that each batch of modified prompts is organised and readily accessible.

In the next step, process each text prompt in the source directory by reading the original material, transmitting it to the Groq AI API with the desired temperature, and retrieving the customised text. The API request is designed to include a precise directive to the model to concentrate on rebuilding the text prompt precisely without augmenting it with unnecessary material. These changes guarantee that the revised prompts stay faithful to the initial purpose but with the possibility of enhanced clarity or descriptiveness.

In addition, an error-handling mechanism is incorporated to effectively handle possible problems that may arise during API interactions, such as unsuccessful requests or unforeseen answers. Furthermore, to adhere to API rate restrictions, the script incorporates a pause mechanism that stops for 60 seconds after every 30 queries. The implementation of this measure serves to avoid surpassing the allowed threshold of requests per minute, guaranteeing that the utilisation adheres to the operating parameters of the API.

After receiving a modified prompt, it is merged with any leftover text that was not updated during the prompt modification process (such as a target model number). Subsequently, the merged text is reinstated in the relevant batch directory with the identical file name as the original, preserving a uniform file structure that enables easy incorporation into the training procedure.

A logging function has also been added to offer immediate feedback on the files, record their progress during the operation that has been processed, and indicate intervals when it stops to adhere to rate restrictions. Implementing this logging is essential for monitoring the performance of the script and resolving any problems that may arise during its execution.

3.3 Training Process Overview

The training process is an integral part of the experimental setup and aims to optimise the performance of the scene synthesis model by iteratively updating its parameters. This section outlines the details of the training process, including initialisation, execution, and monitoring of training across multiple datasets. All project development and experimental methods are based on the GitHub project LSDM, created by @andvg3. The original Git repository project code is accessible via the following link [7]: <https://github.com/andvg3/LSDM.git>.

3.3.1 Setup and Configuration

The experiments use PyTorch, a widely used library for deep learning applications [36]. The configuration comprises bespoke datasets, model definitions, and training procedures, which are encapsulated in Python scripts. This ensures a modular and replicable framework.

I employ bespoke dataset classes `ProxDataset_txt` [16], which have been devised to facilitate the management of disparate elements of the PRO-teXt dataset. The class mentioned above are responsible for loading data, applying transformations, and preparing data batches for training purposes. The data is subjected to a pre-processing process to align it with the input requirements of the model. This process includes applying techniques such as normalisation and augmentation, as defined in the dataset classes.

The model architecture is defined separately, facilitating flexibility in experimentation with different configurations. The number of layers (`n_layer`), the number of heads in the transformer (`n_head`), and other architectural features are defined at the outset of the model training process. These parameters are of great consequence in adapting the model to the intricacies and complexities inherent to the dataset.

The critical training parameters include the learning rate (LR), the number of

epochs (epochs), the batch size, and the optimisation settings. The AdamW optimiser is employed due to its efficacy in addressing sparse gradients and adaptive learning rate management capacity. Training is conducted over a specified number of epochs, with the implementation of checkpoints and logging to facilitate monitoring progress and performance.

3.3.2 Execution

The training data is systematically analysed in batches, allowing the model to adjust its weights and enhance its ability to generalise gradually. The effectiveness of this approach in managing large datasets and complicated models is remarkable since it minimises memory use and accelerates the training process.

Batching is a crucial component of this training session. Each batch undergoes 1,000 epochs during individual batch training, which aligns with the training protocol used on the original dataset. This large number of epochs is chosen to guarantee thorough training of the model using each unique collection of text prompts, hence enabling deep learning and appropriate fine-tuning of model performance.

In addition, To avoid overfitting during training, the number of epochs is restricted to 500 while training across all batches. The strategic reduction seen in this study demonstrates a well-balanced strategy to effectively use the many text alterations included in the whole dataset. This method improves the generalisation of the model without overfitting to any one batch of data. The rationale for this choice is based on the results of the first experiments, which showed that the improvement in model accuracy beyond this stage in the multi-batch training setting became less significant.

During the training phase, the primary goal is to reduce the reconstruction loss, quantifying the difference between the produced outputs and the observed data. Moreover, the performance of the model across many dimensions is comprehensively evaluated using other metrics, including the Chamfer distance and category accuracy. Thorough descriptions of complete metrics will be provided in the results section.

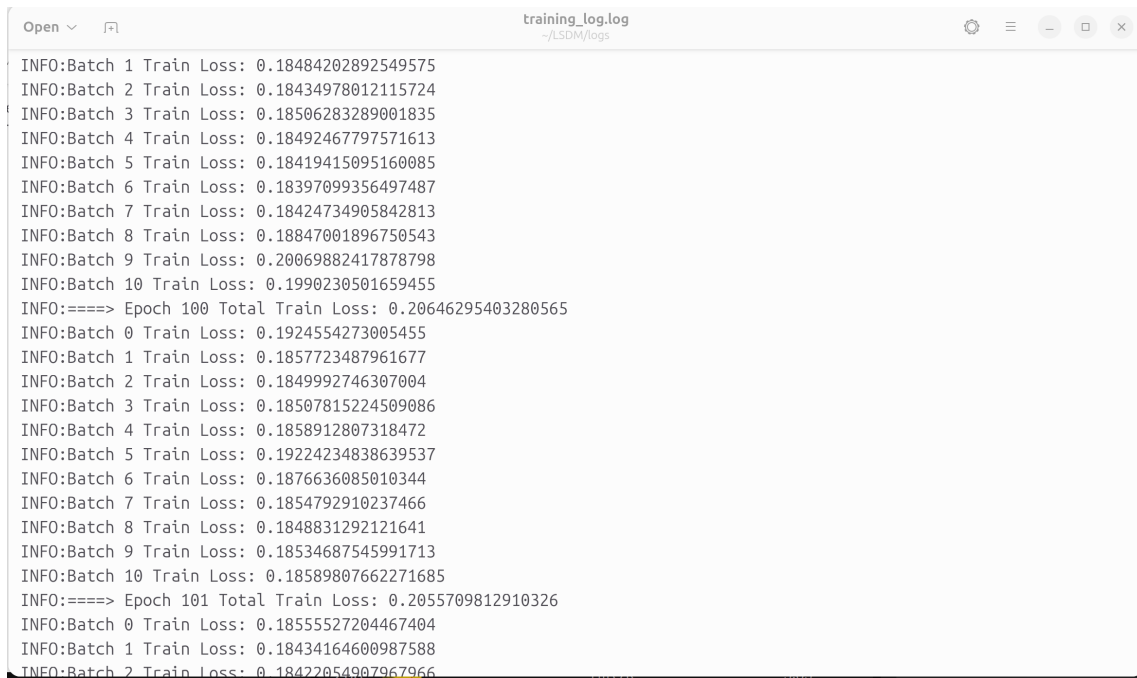
Furthermore, the model parameters are updated using gradient descent, which is helped by the optimiser. The backpropagation technique is used to compute the gradients of the loss function. Subsequently, the optimiser adjusts the model weights based on this computation. A learning rate scheduler is used to enhance the effectiveness of the training process. This component adjusts the learning rate based on predetermined rules or in response to the training loss, enabling the avoidance of local minima and optimising convergence rates.

3.3.3 Monitoring and Adjustments

Some tools and methods are essential to monitor the progress of the project training.

- The training process incorporates a system of regular checkpoint saving after ten times epochs. This enables the training to be resumed from a designated point, thus preventing the loss of progress and facilitating the assessment of model states at various stages of training.

- Training progress is monitored by maintaining comprehensive logs, which document loss metrics and operational parameters (Figure 7). Such records are of paramount importance for debugging, performance assessment, and the fine-tuning of training strategies.



```
Open  [F] training_log.log
~/LSDM/logs

INFO:Batch 1 Train Loss: 0.18484202892549575
INFO:Batch 2 Train Loss: 0.18434978012115724
INFO:Batch 3 Train Loss: 0.18506283289001835
INFO:Batch 4 Train Loss: 0.18492467797571613
INFO:Batch 5 Train Loss: 0.18419415095160085
INFO:Batch 6 Train Loss: 0.18397099356497487
INFO:Batch 7 Train Loss: 0.18424734905842813
INFO:Batch 8 Train Loss: 0.18847001896750543
INFO:Batch 9 Train Loss: 0.20069882417878798
INFO:Batch 10 Train Loss: 0.1990230501659455
INFO:====> Epoch 100 Total Train Loss: 0.20646295403280565
INFO:Batch 0 Train Loss: 0.1924554273005455
INFO:Batch 1 Train Loss: 0.1857723487961677
INFO:Batch 2 Train Loss: 0.1849992746307004
INFO:Batch 3 Train Loss: 0.18507815224509086
INFO:Batch 4 Train Loss: 0.1858912807318472
INFO:Batch 5 Train Loss: 0.19224234838639537
INFO:Batch 6 Train Loss: 0.1876636085010344
INFO:Batch 7 Train Loss: 0.1854792910237466
INFO:Batch 8 Train Loss: 0.1848831292121641
INFO:Batch 9 Train Loss: 0.18534687545991713
INFO:Batch 10 Train Loss: 0.18589807662271685
INFO:====> Epoch 101 Total Train Loss: 0.2055709812910326
INFO:Batch 0 Train Loss: 0.18555527204467404
INFO:Batch 1 Train Loss: 0.18434164600987588
INFO:Batch 2 Train Loss: 0.18422054907967966
```

Figure 7: Parts of training logs.

- Concurrent with the training process, a validation procedure is conducted intermittently to assess the model against a distinct data set. This enables the monitoring of the performance of the model in terms of generalisation and the avoidance of overfitting. It needs to run separately from the training process.

3.4 Batch Processing Strategy

Batch processing represents a fundamental element of my training scheme, facilitating the efficient and practical training of complicated neural network models on expansive datasets. This section outlines the methodology underlying the batch processing approach, including the partitioning of datasets, the implementation of

batch processing during training, and the optimisation techniques employed to enhance computational efficiency and model performance.

1. Generation of Text Prompt Batches:

The Groq AI platform was employed to substitute the original text prompts from the dataset with AI-generated alternatives. This process was repeated ten times, with the ‘temperature’ parameter adjusted on each occasion to introduce variability and richness in the generated text prompts. Adding these changes made it easier to explore a bigger semantic space. This led to the creation of a more flexible and accurate model for analysing and putting together scene data based on textual inputs.

2. Dataset Segmentation into Batches:

The ten distinct AI-modified versions of the dataset were divided into ten batches. The division of the data allowed for more efficient management of the computational resources and the implementation of targeted training regimes on selected batches. Each batch was stored separately, thus enabling the training process to be diversified and managed at a granular level.

3. Selective Batch Training:

Due to limited time resources and hardware equipment, five out of the ten batches were selected for the preliminary training phase. The objective of this process is to assess the efficacy of the AI-modified text prompts in improving the performance of the model. By training the model on these chosen batches, it will be possible to gain insights into the performance variations brought about by various textual modifications, which will allow for the adjustment of the strategies used.

4. Comprehensive Training Across All Batches:

Following the preliminary selective training phase, a new model was constructed utilising all ten batches, including the original unmodified text prompts. This comprehensive training approach was designed to leverage the diversity of text modifications to build a model with enhanced generalisation capabilities. The model could learn to handle various textual inputs by training across all variations, improving its robustness and accuracy in scene synthesis.

5. Optimisation and Learning Rate Management:

The AdamW optimiser was employed throughout the batch processing, which is renowned for its effectiveness in handling sparse gradients and for adaptive learning rate adjustments [37]. This approach guaranteed that each training batch was optimised to achieve the best possible outcomes, with model weights updated to maximise learning efficiency from each data set.

6. Checkpointing and Evaluation:

To protect against data loss and facilitate comprehensive assessments, I have incorporated systematic checkpointing into the training procedures. The model states were saved at predetermined intervals, thus enabling the resumption of training in the event of an interruption and the evaluation of models trained at different points in time. This approach was instrumental in identifying the optimal model configuration based on the empirical performance data collected during the validation process.

3.5 Problems Identified During the Conduct of the Experiment

Virtualisation and system issues are common challenges developers encounter in machine learning [38]. While setting up and executing my project based on the original project GitHub repository, I faced several challenges, particularly regarding software dependencies and environmental configuration. The following section outlines the key issues encountered and the solutions applied to address them. Original project GitHub repository link [7]: <https://github.com/andvg3/LSDM.git>

3.5.1 Environment Configuration Issues

Most of these problems were encountered when choosing the operating system platform in the early stage of the experiment. The corresponding solutions to these problems determine why the native Linux system is used instead of using WSL and VirtualBox on the Windows platform.

1. Installation Challenges with PyTorch3D on WSL:

- **Issue:** From the original document, ‘pytorch3d=0.7.5’ is recommended. The installation of PyTorch3D on the WSL encounters difficulties due to incompatibility issues.
- **Solution:** The installation process was completed successfully using an older version of PyTorch3D (‘pytorch3d=0.7.3’), which was compatible with the system requirements on WSL.

2. Torchvision Detection Failures:

- **Issue:** Following the installation of Torchvision, it was impossible to detect any activity, which suggests that there may be compatibility issues with the PyTorch version that was installed.

- **Solution:** The specific versions of PyTorch (**'pytorch-2.4.0'**) and Torchvision (**'torchvision-0.19.0'**) were employed, ensuring compatibility between the two.

3. CUDA Library Errors:

- **Issue:** An **'ImportError'** was encountered, which appeared to be related to **'libc10_cuda.so'**. This suggested that the PyTorch3D library was attempting to load CUDA-related shared libraries not found on the system.
- **Solution:** The reinstallation of CUDA and PyTorch was an effective solution to the problem, although it occasionally resurfaced, suggesting underlying configuration inconsistencies.

4. CUDA Version Mismatch:

- **Issue:** The README instructions recommended CUDA 11.6, whereas the installed PyTorch version was compatible with CUDA 11.2. This discrepancy led to an **'ImportError'** for **'libtorch_cuda_cu.so'**.
- **Solution:** Such errors can be avoided by ensuring that the PyTorch and CUDA versions are compatible with the requirements of the project in question.

5. VirtualBox Execution Errors:

- **Issue:** The execution of scripts within a VirtualBox environment (Ubuntu 24.04) yielded an **'Illegal instruction (core dumped)'** error, which was not observed when utilising WSL.
- **Solution:** This issue typically arises due to incompatibilities between the CPU instruction set and other components. This problem may be mitigated by running the project on a native Linux system or ensuring the virtualised environment supports all necessary CPU instructions.

Recommendation: To avoid the issues mentioned above, it is recommended that future implementations utilise a native Linux system. The native system ensures superior compatibility and performance for the complex computations AI models require. This approach should result in the elimination of inconsistencies and limitations that are inherent to virtualised environments such as WSL and VirtualBox.

3.5.2 Training Process Issues: Text Prompt Format Inconsistencies

As previously stated in Section 3.4, the Groq AI API was employed to generate ten distinct versions of a text prompt file, each with the same underlying meaning.

During the training phase of the project, a significant issue was identified concerning the format of the text prompts that the AI had rewritten. Despite the

original text prompt files being structured with the natural language command on the first line, followed by object and target object codes on the second and third lines, respectively, inconsistencies were observed in the outputs generated by the AI.

Even though the API call scripts made it clear to rewrite only the first line with the natural language command, there were times when the outputs generated by AI included extra lines that were not needed. Such discrepancies can potentially disrupt the training process, introducing format inconsistencies incompatible with the expected input structure for model training.

To address this issue and guarantee uniformity in the text file format before commencing training, a Python script was devised to automate the verification process for each text file within the specified batch directories. The script verifies that each text file contains precisely three lines and notifies the user of discrepancies.

Here is the Python script used for checking the file lines (Figure 8):



```
check.py > ...
1  import os
2
3  def check_file_lines(directory):
4      for filename in os.listdir(directory):
5          if filename.endswith(".txt"):
6              filepath = os.path.join(directory, filename)
7              with open(filepath, 'r') as file:
8                  lines = file.readlines()
9                  if len(lines) != 3:
10                     print(f"File {filename} has {len(lines)} lines.")
11
12 directory_path = 'data/protext/proxd_train/context_versions/batch_10' # replace into the real data path
13 check_file_lines(directory_path)
```

Figure 8: Python script used for checking the file lines.

This utility script performs verification of each text file within the specified directory. Subsequently, it outputs the filename and the number of lines if the file does not adhere to the expected format of three lines per file. This is an indispensable tool for maintaining data integrity and ensuring that the training data fed into the model is correctly formatted. This avoids potential errors during model training arising from improperly formatted input data.

Chapter 4

Results

This chapter presents a comprehensive analysis of experiment results conducted to improve language-driven 3D scene synthesis. The study explores the impact of varying temperature parameters in text prompt generation on the quality and accuracy of synthesized scenes. Results include comparison results between various batches with different temperature settings and the original dataset, comparison results between the unified dataset and the original dataset and visual results for these two comparisons.

4.1 Preliminary Comparison

The preliminary results from the experiments using the modified text prompts across different batches compared to the original dataset are summarised in the table below. The train numbers of epochs are 1000, the same as the original train. Full training results of these six batches can be found in Appendix B.

Preliminary results list					
Batch name	Final Chamfer distance	Final EMD	Final F1 score	Category accuracy	Top 3 accuracy
Original batch	0.8037	0.7152	0.5505	0.8500	100.0000
Batch_1 (T = 0.5)	0.3082	0.5793	0.4271	0.9000	100.0000
Batch_3 (T = 0.6)	1.2468	0.7979	0.4922	0.3500	40.0000
Batch_5 (T = 0.7)	1.5485	0.8660	0.3801	0.8000	85.0000
Batch_8 (T = 0.85)	0.6860	0.8743	0.3413	0.8000	85.0000
Batch_10 (T = 0.95)	1.1936	0.8902	0.2771	0.9000	100.0000

Table 2: Preliminary comparison between different temperature parameters (T).

Where:

- **Chamfer Distance:** ‘Chamfer Distance’ describes a metric measuring the difference between two sets of points. It is frequently employed to ascertain the degree of resemblance between generated three-dimensional models or other point cloud data and authentic data. The calculation is performed by taking each point from one set of points and identifying it in the different set of points closest to it. The resulting distances are then averaged to produce a single value representing the similarity between the two sets of points [39]. The chamfer distance metric is widely used in 3D model reconstruction, machine vision, and point cloud processing. In these applications, a smaller value of the chamfer distance indicates that the model-generated output is more closely aligned with the real-world data.
- **EMD:** The Earth Mover’s Distance, also called the Wasserstein Distance, measures the discrepancy between two probability distributions. In the context of a point cloud, it quantifies the effort required to transform one set of points into another [40]. The EMD is employed in many applications to evaluate the degree of similarity between the generated samples and the target set of samples. A lower EMD value indicates more proximity between the two data sets.
- **F1 Score:** The F1 score is the harmonic mean of precision and recall and is frequently employed in classification tasks. The harmonic mean of precision (the proportion of samples predicted to be positive that are positive) and recall (the proportion of samples predicted to be positive that are positive) is a

measure of classification accuracy [41]. The F1 score is instrumental in cases of category imbalance, as a higher F1 score indicates that the model has achieved a superior balance between precision and recall.

- **Category Accuracy:** Classification accuracy is the ratio of correctly classified samples to the total number of samples. It is the most direct indicator of classification performance [42]. This is employed to assess the overall efficacy of a classification model. A higher accuracy rate indicates enhanced model performance.
- **Top 3 Accuracy:** Top 3 accuracy is a classification metric that indicates how frequently a model predicts the top three most likely categories to the correct category [43]. In classification problems, mainly when there are many categories and some are similar, Top 3 accuracy provides additional flexibility in evaluating the model. A perfect Top 3 accuracy of 100% signifies that the model consistently includes the correct category in its top three most probable predictions.

Analysis:

1. **Chamfer Distance and EMD:** A decrease in the values of chamfer distance and EMD generally improved the geometric accuracy of the intended models. Batch_1, at a temperature of 0.5, showed the greatest improvement in both measures, suggesting that a moderate variation in the text prompts can enable more accurate 3D scene reconstructions. Conversely, the highest temperatures (Batch_5 and Batch_10) yielded the poorest results, indicating that excessive variability can impair the quality.
2. **F1 Score:** The F1 score, which quantifies the accuracy of the test, exhibited the highest values in the original batch and a notable decline in subsequent batches subjected to elevated temperatures. This decline in accuracy means that the enhanced creativity observed at elevated temperatures is not optimally aligned with the precise requirements of the scene synthesis task.
3. **Category and Top 3 Accuracy:** While the accuracy of the categories remains high across the majority of batches, there is a notable discrepancy in the top 3 accuracy results. Batch_3, at a temperature of 0.6, exhibited markedly inferior top 3 accuracy, suggesting that the scenes generated at this temperature exhibited a notable divergence from the target categories.

Conclusion: The disparate temperatures employed during the generation of text prompts exert a discernible influence on the efficacy of the scene synthesis model. Lower temperatures balance creativity and fidelity to the original dataset, enhancing overall performance metrics. The results demonstrate the influence of changing the

temperature parameter (T) employed during text prompt generation on the quality and precision of the synthesised scenes. That shows that precise calibration of the input and the variability of the text is needed to get the best quality output from language-driven scene synthesis models.

4.2 Comprehensive Training

The comprehensive training process entailed the integration of all ten AI-modified text prompt batches, in conjunction with the original text prompts, to create a unified dataset for model training. The objective of this approach was to evaluate the impact of blending various prompt modifications on model performance in a comprehensive manner. The following table summarises the results obtained from training on the original batch and the aggregated dataset. The full result can be found in Appendix C.

Preliminary results list					
Batch name	Final Chamfer distance	Final EMD	Final F1 score	Category accuracy	Top 3 accuracy
Original batch	0.8037	0.7152	0.5505	0.8500	100.0000
Batch_all (included the original batch)	0.5223	0.4545	0.7375	1.0000	100.0000

Table 3: Comprehensive training result comparative.

Analysis:

1. **Chamfer Distance and EMD:** The comprehensive dataset showed a notable enhancement in both chamfer distance and EMD when the model was trained. The chamfer distance decreased from 0.8037 to 0.5223, and the EMD from 0.7152 to 0.4545, indicating a substantial enhancement in geometric accuracy and the ability of the model to approximate the target geometries closely.
2. **F1 Score:** The F1 score increased from 0.5505 to 0.7375, indicating a more optimal balance between precision and recall. This improvement suggests that the model, trained on a diverse set of prompts, is more effective at recognising and generating the correct elements of the scene, thereby enhancing the overall accuracy of scene synthesis.
3. **Category and Top 3 Accuracy:** The category accuracy reached a perfect

score of 1.0000, representing a notable improvement from 0.8500. This outcome illustrates that the diverse training inputs have markedly enhanced the capacity of the model to classify scenes with precision. The top three accuracy scores remained at 100.0000, confirming consistent performance even as the complexity of the training data increased.

Conclusion: The findings from the comprehensive training program demonstrate a distinct advantage in integrating AI-modified text prompts with the original dataset. This training method made the model more accurate across several different metrics. It also showed that giving the model access to a wider range of textual inputs can help it become more stable and accurate. These findings highlight the significance of incorporating diverse training data in developing sophisticated AI systems, which could prove advantageous for applications in automated content generation and augmented reality systems. Further research should examine the extent of input diversity and its relationship with the capacity of the model to generalise across diverse scene synthesis tasks.

4.3 Visualization Tests

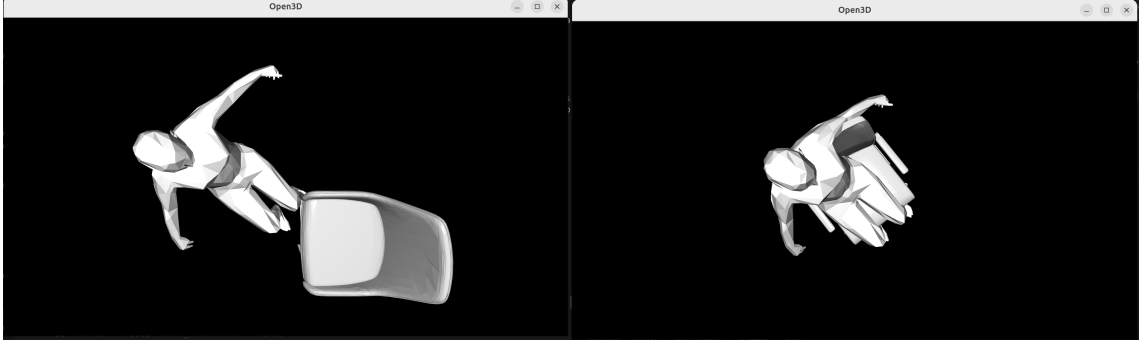
In this section, the visualisation test results between the original batch and the comprehensive batch are shown below.

Following the example from the original GitHub repository document on visual content generation, I used the same `<sequence_name>` and `<interaction_name>` to generate the visual content. Where `sequence_name` is the name of the human motion, and `interaction_name` is the name of the human pose [7].

To control the variables, all visualisations were generated using the original batch trained model and comprehensive trained model, all other conditions being identical.

4.3.1 Comparison of Single Object Scene Generation

The valid test text prompt below is ‘Place a chair behind me’.



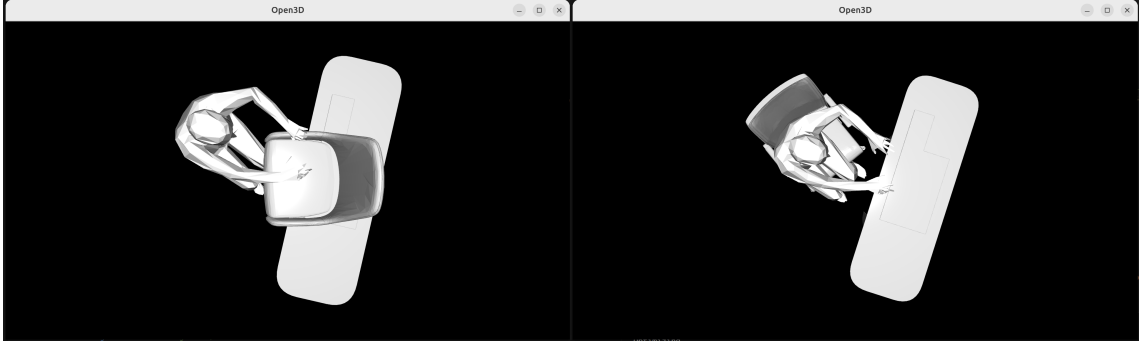
(a) Visualisation result of the original model. (b) Visualisation result of optimised model.

Figure 9: Comparison of single object scene generation between the original and comprehensive batch.

As shown in Figure 9, the ‘human’ successfully sits on the generated chair in the enhanced model.

4.3.2 Comparison of Multi-object Scene Generation

The valid test text prompt below is ‘Place a chair under me and a desk in front of me’.



(a) Original model multi-object scene result. (b) Optimised model multi-object scene result.

Figure 10: Comparison of multi-object scene generation between the original and comprehensive batch.

The result shows that the original model placed the table correctly during the multi-object scene generation. However, the same as the single object scene generation, the chair has left the wrong position.

Conclusion: The comparative analysis of the results obtained from this visualisation demonstrates that the model that has undergone comprehensive training effectively enhances the accuracy of scene generation. In single-object scene generation, the optimised model correctly positioned the chair behind the person, achieving the desired interaction. In multi-object scene generation, the optimised model correctly placed the table and chair in the expected position, whereas the original model

misplaced the chair. These tests demonstrate that the optimised model is more accurate in generating single or multi-object scenes based on the natural language text prompt input by the user, thereby improving the quality and reliability of the generated scenes.

Chapter 5

Discussion

A discussion was conducted based on the comprehensive experimental process and results.

5.1 Analysis of Model Performance vs. Expected Results

Experimental results show that the variation in text prompts generated at different temperature settings can affect the ability of the model to learn effectively. For example, a lower temperature setting (0.5) produces more consistent and predictable text prompts, resulting in better chamfer distance and EMD performance. On the other hand, higher temperatures introduce more randomness into the text prompts, negatively impacting performance (as results in Batch_10).

In addition, the LSDM model used for scene synthesis relies on guide points in the diffusion process for denoising. Suppose these points are disturbed by inconsistent or overly creative text prompts. In that case, the model may not be able to generate accurate scene representations, which indirectly leads to performance degradation in some batches.

Finally, due to time and hardware resource constraints, the high demands on memory and processing power during training can lead to suboptimal model performance.

5.2 Impact of Text Prompt Replacement on Natural Language Understanding

The results show that simple text prompts that are more structured and predictable, such as the first set of prompts generated at lower temperatures, tend to improve the ability of the model to interpret natural language instructions effectively. In contrast, prompts generated at higher temperatures introduce variability and randomness that

are difficult for the model to interpret consistently, resulting in poor scene synthesis results.

Text prompts at higher temperatures provide more linguistic variety, which may improve the training dataset but confuse the model and reduce its effectiveness if the model is resilient enough to cope with this complexity. These findings suggest that while linguistic variety is crucial to developing adaptive models, finding a middle ground between originality and clarity is necessary to ensure accurate model learning.

The performance improvement of the fully trained model demonstrates how alternating prompts at various temperatures allow the model to generalise better. However, suppose too many prompts deviate from the central semantic theme. In that case, the model may begin to overfit specific language patterns, reducing its ability to handle a variety of inputs in real-world scenarios.

5.3 Benefits and Challenges in Model Training with Diverse Data

- **Benefits:** Augmenting linguistic variety using AI-generated prompts at varying temperatures enhances the resilience of the model in processing diverse input sources. Several model performance measures, such as chamfer distance, EMD, and category accuracy, get better when data from a synthetic training dataset with many batches is used. Diverse training data improve the ability of the model to accept ambiguous or less structured language input, increasing flexibility in comprehending and processing natural language instructions in practical applications like robotics, virtual reality ('VR'), or gaming.

- **Challenges:** Increased data diversity brings challenges of longer training time and higher memory usage. Processing diverse language inputs requires more computational resources (e.g., hardware equipment and time resources), especially When the model data increases and the samples become more complex.

Chapter 6

Future Improvement

This research revealed many potential areas for future improvement and development that have the potential to augment the performance and utility of the model significantly.

Presently, the model is constrained to reading and processing text prompt files that alone consist of three lines. An area for future enhancement would be expanding this capability, enabling the model to process longer and more complicated cues effectively. The proposed expansion would allow the incorporation of more instructions or complex situations, enhancing the use of the model.

Another potential area for future research is developing an application seamlessly incorporating real-time AI APIs, such as ChatGPT. Users would be able to provide natural language instructions that AI could dynamically rephrase and improve before input into the model for rendering scenes. An implementation of such a feature has the potential to significantly enhance user engagement and accessibility by providing instant feedback and increasing the intuitiveness of users without technical expertise in the system. Although the existing model works offline, efficiently generating scenes in real-time remains difficult. Since scene synthesis requires significant processing time, real-time performance may not be achievable even if the system were placed online. Possible enhancements may include rationalising the synthesis process or integrating more efficient hardware accelerators to minimise user delay. Furthermore, asynchronous processing might enable users to place requests in a queue and get notifications once the scene has been created.

The present model uses the CLIP text encoder to evaluate alternative text encoders. The text-embedding-3 model, newly presented by OpenAI, has the potential to enhance performance by directly transforming text into embedding vectors. More investigation might be conducted to see whether substituting CLIP with this more recent model improves the quality and precision of the produced scenes. Therefore, considering this proposal is still based on speculation, conducting experiments to verify its effectiveness is essential.

Finally, optimising dataset and point cloud variety could be a concern. Currently, the dataset used for model training is constrained in size and variety. This limitation restricts the capacity of the model to produce a diverse array of scenarios that extend beyond the particular items it was trained on. Future research should prioritise the augmentation of the dataset, namely by integrating a more comprehensive range of point cloud data, to enable the model to produce more adaptable and innovative scenarios. Implementing this would provide a more extensive range of interactions and situations more closely matched with user expectations.

Through the resolution of these aspects, the project has the potential to develop into a more resilient and user-centric system that accommodates a broader range of use cases, therefore rendering it appropriate for real-time applications in fields such as robotics, virtual reality, gaming, and interactive simulations.

Chapter 7

Conclusion

This research has explored the enhancement of language-driven scene synthesis through AI-generated text prompts. By using the Groq AI API to create diverse text inputs at different temperature settings, the results have demonstrated significant improvements in the performance and flexibility of the basic LSDM framework.

According to metrics like chamfer distance and EMD, these experiments showed that lower temperature settings in text prompt generation produced more consistent and predictable outputs, improving model performance. Conversely, while introducing more language diversity, higher temperature settings often led to decreased model accuracy due to the increased complexity and potential inconsistency of the generated text prompts.

A key finding of this study is the importance of striking a balance between linguistic diversity and semantic coherence in the training data. The comprehensive training approach, which included both text prompts generated at various AI API temperature settings and the original dataset in the training process, enhanced the ability of the model to generalise across a broader range of natural language inputs. This comprehensive training method improves the various indicators of the model, indicating a more robust and adaptable scene synthesis model.

In addition, this study opens up several avenues for future research. Some of these are making the model better at handling longer and more complicated text prompts, making it possible to create scenes in real-time, looking into other ways to encode text like text-embedding-3 model from OpenAI, and adding more 3D scenes and object interactions to the dataset.

In conclusion, this research demonstrates the potential of AI-generated text prompts to significantly enhance the performance and versatility of language-driven scene synthesis models. By combining AI and balancing language diversity and semantic coherence, more powerful and adaptable systems can be developed that can interpret a wide range of complex natural language instructions. These advancements pave the way for more intuitive and powerful applications in robotics,

virtual reality, and interactive simulations, bringing us closer to seamless human-AI interaction in complex 3D environments.

Bibliography

- [1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qi Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- [2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. *Communications of the ACM*, 45(3):43–47, 2005.
- [3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [4] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.
- [5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.
- [6] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.
- [7] An Dinh Vuong, Minh Nhat Vu, Toan Nguyen, Baoru Huang, Dzong Nguyen, Thieu Vo, and Anh Nguyen. Language-driven scene synthesis using multi-conditional diffusion model. *arXiv preprint arXiv:2310.15948*, 2023.
- [8] A. Paschalidou, M. Kar, M. Shugrina, K. Kreis, A. Geiger, and S. Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 12013–12026, 2021.
- [9] Xiujun Li and Jianfeng Gao. Prompt learning for vision-language models: A survey and beyond. *arXiv preprint arXiv:2112.08352*, 2021.
- [10] Chenguang Zhou, Qian Zhang, Jian-Guang Lou, Lijun Zhang, Xue Xu, Zixuan Liu, Mu Song, Li Hu, and Lidan Shou. Coco-lm: Correcting and contrasting

- text sequences for language model pretraining. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5463–5475, 2021.
- [11] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.
 - [12] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
 - [13] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.
 - [14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
 - [15] Teamhood. Kanban view - teamhood. <https://teamhood.com/knowledge-base/views/kanban/>, 2024. Accessed: 2024-09-01.
 - [16] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J. Black. Resolving 3D human pose ambiguities with 3D scene constraints. In *International Conference on Computer Vision*, pages 2282–2292, October 2019.
 - [17] Venkat Vishwanath, Murali Emani, Valerie Taylor, Ian Foster, Salman Habib, Michael E Papka, Anakha V Babu, Henry Chan, Mathew J Cherukara, Jose M Monsalve Diaz, et al. 2022 ai testbed expeditions report. Technical report, Argonne National Laboratory (ANL), Argonne, IL (United States), 2022.
 - [18] C Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. Learning the visual interpretation of sentences. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.
 - [19] Murali Emani, Zhen Xie, Siddhisanket Raskar, Varuni Sastry, William Arnold, Bruce Wilson, Rajeev Thakur, Venkatram Vishwanath, Zhengchun Liu, Michael E Papka, et al. A comprehensive evaluation of novel ai accelerators for deep learning workloads. In *2022 IEEE/ACM international workshop on performance modeling, benchmarking and simulation of high performance computer systems (PMBS)*, pages 13–25. IEEE, 2022.

- [20] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29, 2016.
- [21] NVIDIA Corporation. Nvidia geforce rtx 4090 graphics cards. <https://www.nvidia.com/en-gb/geforce/graphics-cards/40-series/rtx-4090/>, 2023. Accessed: 2024-09-01.
- [22] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.
- [23] Intel Corporation. Intel core i7-14700k processor (33m cache, up to 5.60 ghz) specifications. <https://www.intel.com/content/www/us/en/products/sku/236783/intel-core-i7-processor-14700k-33m-cache-up-to-5-60-ghz/specifications.html>, 2023. Accessed: 2024-09-01.
- [24] Kainat Afridi, Jamshid Ali Turi, Barirah Zaufishan, and Joanna Rosak-Szyrocka. Impact of digital communications on project efficiency through ease of use and top management support. *Heliyon*, 9(7), 2023.
- [25] Bobby Borisov. Ubuntu 24.04 lts (noble numbat) released, this is what’s new, 2024. Accessed: 2024-09-02.
- [26] Lucas Rees. Ubuntu 24.04 lts vs 22.04 lts: A comparison guide and what’s new, 2024. Accessed: 2024-09-02.
- [27] Ubuntu Community Hub. Ubuntu 24.04 lts (noble numbat) release notes, 2024. Accessed: 2024-09-02.
- [28] Ankush Das. 11 reasons why linux is better than windows, 2024. Accessed: 2024-09-02.
- [29] Disha Misal. Linux vs windows: Which is the best os for data scientists?, 2024. Accessed: 2024-09-02.
- [30] Josphat Mutai. Linux vs. windows: A comprehensive look at the pros and cons, 2023. Accessed: 2024-09-02.
- [31] Abhimanyu Krishnan. Linux vs windows: Which operating system should you choose?, 2024. Accessed: 2024-09-02.
- [32] Data Science Central. Linux vs windows: Pros and cons for data science, 2024. Accessed: 2024-09-02.
- [33] OpenAI. Api rate limits and quotas. <https://platform.openai.com/docs/guides/rate-limits/usage-tiers?context=tier-free>, 2024. Accessed: September 1, 2024.

- [34] Groq. Api rate limits and quotas. <https://console.groq.com/settings/limits>, 2024. Accessed: September 1, 2024.
- [35] Groq. Api rate limits and quotas. <https://console.groq.com/docs/api-reference#chat>, 2024. Accessed: September 1, 2024.
- [36] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: solution for edge computing applications*, pages 87–104, 2021.
- [37] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [38] Robert P Goldberg. Survey of virtual machine research. *IEEE computer magazine*, 7(6):34–45, 1974.
- [39] Haoqiang Fan, Hang Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2463–2471, 2017.
- [40] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.
- [41] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [42] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.
- [43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

Appendix A

The Code of Groq AI API Implementation

```
1 import os
2 import requests
3 import json
4 import time
5
6 APIKEY = 'YOUR_APIKEY'
7 API_URL = 'https://api.groq.com/openai/v1/chat/completions'
8
9 def generate_prompts_and_save(source_dir, target_dir_base):
10     if not os.path.exists(target_dir_base):
11         os.makedirs(target_dir_base)
12
13     files = [f for f in os.listdir(source_dir) if f.endswith('.txt')]
14     temperatures = [0.5 + 0.05 * i for i in range(10)] # Temperature list for
15     ↪ controlling text diversity
16     request_count = 0
17
18     for index, temp in enumerate(temperatures):
19         target_dir = os.path.join(target_dir_base, f'batch-{index + 1}')
20         os.makedirs(target_dir, exist_ok=True)
21
22         for filename in files:
23             source_file_path = os.path.join(source_dir, filename)
24             target_file_path = os.path.join(target_dir, filename)
25
26             with open(source_file_path, 'r', encoding='utf-8') as file:
```

```

26         lines = file.readlines()
27         original_text = lines[0].strip() # The first line is the text
           ↪ to be overwritten
28         remaining_text = ''.join(lines[1:]) # The remaining rows
           ↪ remain
29
30         generated_text = generate_text_with_api(original_text, temp)
31         full_text = generated_text + '\n' + remaining_text # Merge the
           ↪ generated text with the retained text
32
33         with open(target_file_path, 'w', encoding='utf-8') as file:
34             file.write(full_text)
35
36         print(f"Processed {filename} with temp {temp}")
37         request_count += 1
38
39         if request_count % 30 == 0:
40             print("Pausing for 60 seconds to comply with API rate limits.")
41             time.sleep(60)
42
43 def generate_text_with_api(text, temperature):
44     headers = {
45         "Authorization": f"Bearer {API_KEY}",
46         "Content-Type": "application/json"
47     }
48     data = {
49         "model": "llama3-8b-8192",
50         "messages": [
51             {"role": "system", "content": "You are a helpful assistant. What I
           ↪ am sending to you now is the text prompt of my project.
           ↪ These text instructions will be converted into embedding
           ↪ vectors and combined with my other point cloud data to
           ↪ generate the scene corresponding to the text prompt. Now I
           ↪ need you to help me overwrite this text prompt to improve the
           ↪ accuracy of the model. No extra words are needed, just
           ↪ ONLY provide the rewritten language."},
52             {"role": "user", "content": text}
53         ],
54         "max_tokens": 100,

```

```

55     "temperature": temperature,
56     "n": 1
57 }
58 response = requests.post(API_URL, headers=headers, data=json.dumps(data)
    ↪ )
59 if response.status_code == 200:
60     response_data = response.json()
61     generated_text = response_data['choices' ][0][' message'][' content']. strip
    ↪ ()
62     return generated_text
63 else:
64     print(f"Failed to generate text: {response.status_code} {response.text
    ↪  }")
65     return "Error in text generation"
66
67 source_directory = 'data/protext/proxd_train/context'
68 target_directory_base = 'data/protext/proxd_train/context_versions'
69
70 generate_prompts_and_save(source_directory, target_directory_base)

```


Appendix B

Training Result

B.1 Training Result of Original Text Prompt

```
1 Chamfer distance for seq N0Sofa_00034_02: 1.3394
2 Chamfer distance for seq N3Office_00034_02: 0.0038
3 Chamfer distance for seq N0Sofa_00034_02_3: 0.0866
4 Chamfer distance for seq N3OpenArea_03301_02: 0.3418
5 Chamfer distance for seq N3Office_00034_05: 1.7068
6 Chamfer distance for seq MPH112_00169_03: 0.0163
7 Chamfer distance for seq MPH112_00151_03: 2.2346
8 Chamfer distance for seq MPH112_00151_05: 0.0103
9 Chamfer distance for seq MPH112_00151_04: 0.0088
10 Chamfer distance for seq MPH11_00150_04: 0.4046
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.0731
12 Chamfer distance for seq MPH11_00150_05: 0.0165
13 Chamfer distance for seq N3Office_00034_04: 0.0072
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0029
15 Chamfer distance for seq N3Office_00034_02: 0.5318
16 Chamfer distance for seq MPH112_00169_05: 9.2357
17 Chamfer distance for seq N3OpenArea_03301_05: 0.0390
18 Chamfer distance for seq N3OpenArea_03301_04: 0.0046
19 Chamfer distance for seq N3Office_00034_03: 0.0065
20 Chamfer distance for seq MPH112_00169_04: 0.0035
21 Final Chamfer distance: 0.8037
22 Final EMD: 0.7152
23 Final F1 score: 0.5505
24 Category accuracy: 0.8500
25 Top 3 accuracy: 100.0000
```

B.2 Training Result of Batch 1

```
1 Chamfer distance for seq N0Sofa_00034_02: 0.2324
2 Chamfer distance for seq N3Office_00034_02: 0.1087
3 Chamfer distance for seq N0Sofa_00034_02_3: 0.0238
4 Chamfer distance for seq N3OpenArea_03301_02: 1.1688
5 Chamfer distance for seq N3Office_00034_05: 0.2274
6 Chamfer distance for seq MPH112_00169_03: 0.0373
7 Chamfer distance for seq MPH112_00151_03: 0.0266
8 Chamfer distance for seq MPH112_00151_05: 0.0379
9 Chamfer distance for seq MPH112_00151_04: 0.0368
10 Chamfer distance for seq MPH11_00150_04: 0.0906
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.1090
12 Chamfer distance for seq MPH11_00150_05: 0.5934
13 Chamfer distance for seq N3Office_00034_04: 0.0416
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0194
15 Chamfer distance for seq N3Office_00034_02: 0.0413
16 Chamfer distance for seq MPH112_00169_05: 2.6163
17 Chamfer distance for seq N3OpenArea_03301_05: 0.1837
18 Chamfer distance for seq N3OpenArea_03301_04: 0.0165
19 Chamfer distance for seq N3Office_00034_03: 0.0177
20 Chamfer distance for seq MPH112_00169_04: 0.5350
21 Final Chamfer distance: 0.3082
22 Final EMD: 0.5793
23 Final F1 score: 0.4271
24 Category accuracy: 0.9000
25 Top 3 accuracy: 100.0000
```

B.3 Training Result of Batch 3

```
1 Chamfer distance for seq N0Sofa_00034_02: 1.2790
2 Chamfer distance for seq N3Office_00034_02: 0.0052
3 Chamfer distance for seq N0Sofa_00034_02_3: 3.1066
4 Chamfer distance for seq N3OpenArea_03301_02: 0.0839
5 Chamfer distance for seq N3Office_00034_05: 9.6798
6 Chamfer distance for seq MPH112_00169_03: 0.0086
7 Chamfer distance for seq MPH112_00151_03: 2.9344
8 Chamfer distance for seq MPH112_00151_05: 0.0063
9 Chamfer distance for seq MPH112_00151_04: 0.0045
10 Chamfer distance for seq MPH11_00150_04: 0.0754
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.0383
12 Chamfer distance for seq MPH11_00150_05: 0.0109
13 Chamfer distance for seq N3Office_00034_04: 0.0052
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0019
15 Chamfer distance for seq N3Office_00034_02: 0.1545
16 Chamfer distance for seq MPH112_00169_05: 2.8212
17 Chamfer distance for seq N3OpenArea_03301_05: 2.1423
18 Chamfer distance for seq N3OpenArea_03301_04: 0.0355
19 Chamfer distance for seq N3Office_00034_03: 0.0082
20 Chamfer distance for seq MPH112_00169_04: 2.5343
21 Final Chamfer distance: 1.2468
22 Final EMD: 0.7979
23 Final F1 score: 0.4922
24 Category accuracy: 0.3500
25 Top 3 accuracy: 40.0000
```

B.4 Training Result of Batch 5

```
1 Chamfer distance for seq N0Sofa_00034_02: 0.3906
2 Chamfer distance for seq N3Office_00034_02: 0.0662
3 Chamfer distance for seq N0Sofa_00034_02_3: 0.0867
4 Chamfer distance for seq N3OpenArea_03301_02: 0.2264
5 Chamfer distance for seq N3Office_00034_05: 9.4324
6 Chamfer distance for seq MPH112_00169_03: 0.0163
7 Chamfer distance for seq MPH112_00151_03: 9.4799
8 Chamfer distance for seq MPH112_00151_05: 0.0116
9 Chamfer distance for seq MPH112_00151_04: 0.0232
10 Chamfer distance for seq MPH11_00150_04: 0.1787
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.0914
12 Chamfer distance for seq MPH11_00150_05: 0.6178
13 Chamfer distance for seq N3Office_00034_04: 1.5073
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0142
15 Chamfer distance for seq N3Office_00034_02: 0.0134
16 Chamfer distance for seq MPH112_00169_05: 8.6050
17 Chamfer distance for seq N3OpenArea_03301_05: 0.0959
18 Chamfer distance for seq N3OpenArea_03301_04: 0.0493
19 Chamfer distance for seq N3Office_00034_03: 0.0149
20 Chamfer distance for seq MPH112_00169_04: 0.0492
21 Final Chamfer distance: 1.5485
22 Final EMD: 0.8660
23 Final F1 score: 0.3801
24 Category accuracy: 0.8000
25 Top 3 accuracy: 85.0000
```

B.5 Training Result of Batch 8

```
1 Chamfer distance for seq N0Sofa_00034_02: 0.4644
2 Chamfer distance for seq N3Office_00034_02: 0.0549
3 Chamfer distance for seq N0Sofa_00034_02_3: 0.0165
4 Chamfer distance for seq N3OpenArea_03301_02: 0.4729
5 Chamfer distance for seq N3Office_00034_05: 4.7982
6 Chamfer distance for seq MPH112_00169_03: 0.0226
7 Chamfer distance for seq MPH112_00151_03: 2.9242
8 Chamfer distance for seq MPH112_00151_05: 0.0237
9 Chamfer distance for seq MPH112_00151_04: 0.0201
10 Chamfer distance for seq MPH11_00150_04: 0.0530
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.0689
12 Chamfer distance for seq MPH11_00150_05: 0.0968
13 Chamfer distance for seq N3Office_00034_04: 0.0584
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0196
15 Chamfer distance for seq N3Office_00034_02: 0.0102
16 Chamfer distance for seq MPH112_00169_05: 3.1115
17 Chamfer distance for seq N3OpenArea_03301_05: 0.4410
18 Chamfer distance for seq N3OpenArea_03301_04: 0.1365
19 Chamfer distance for seq N3Office_00034_03: 0.4759
20 Chamfer distance for seq MPH112_00169_04: 0.4513
21 Final Chamfer distance: 0.6860
22 Final EMD: 0.8743
23 Final F1 score: 0.3413
24 Category accuracy: 0.8000
25 Top 3 accuracy: 85.0000
```

B.6 Training Result of Batch 10

```
1 Chamfer distance for seq N0Sofa_00034_02: 0.6180
2 Chamfer distance for seq N3Office_00034_02: 0.0340
3 Chamfer distance for seq N0Sofa_00034_02_3: 0.3409
4 Chamfer distance for seq N3OpenArea_03301_02: 0.9957
5 Chamfer distance for seq N3Office_00034_05: 4.3111
6 Chamfer distance for seq MPH112_00169_03: 0.0663
7 Chamfer distance for seq MPH112_00151_03: 6.6966
8 Chamfer distance for seq MPH112_00151_05: 0.0340
9 Chamfer distance for seq MPH112_00151_04: 0.0434
10 Chamfer distance for seq MPH11_00150_04: 0.0528
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.0565
12 Chamfer distance for seq MPH11_00150_05: 0.0637
13 Chamfer distance for seq N3Office_00034_04: 1.8527
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0099
15 Chamfer distance for seq N3Office_00034_02: 0.3582
16 Chamfer distance for seq MPH112_00169_05: 7.3398
17 Chamfer distance for seq N3OpenArea_03301_05: 0.0982
18 Chamfer distance for seq N3OpenArea_03301_04: 0.0593
19 Chamfer distance for seq N3Office_00034_03: 0.0295
20 Chamfer distance for seq MPH112_00169_04: 0.8119
21 Final Chamfer distance: 1.1936
22 Final EMD: 0.8902
23 Final F1 score: 0.2771
24 Category accuracy: 0.9000
25 Top 3 accuracy: 100.0000
```

Appendix C

Comprehensive Training Result

```
1 Chamfer distance for seq N0Sofa_00034_02: 0.5659
2 Chamfer distance for seq N3Office_00034_02: 0.0021
3 Chamfer distance for seq N0Sofa_00034_02_3: 0.0013
4 Chamfer distance for seq N3OpenArea_03301_02: 0.7551
5 Chamfer distance for seq N3Office_00034_05: 5.0572
6 Chamfer distance for seq MPH112_00169_03: 0.0032
7 Chamfer distance for seq MPH112_00151_03: 0.0011
8 Chamfer distance for seq MPH112_00151_05: 0.0031
9 Chamfer distance for seq MPH112_00151_04: 0.0032
10 Chamfer distance for seq MPH11_00150_04: 0.0560
11 Chamfer distance for seq N0Sofa_00034_02_4: 0.0037
12 Chamfer distance for seq MPH11_00150_05: 0.0032
13 Chamfer distance for seq N3Office_00034_04: 0.0030
14 Chamfer distance for seq N3OpenArea_03301_03: 0.0007
15 Chamfer distance for seq N3Office_00034_02: 0.0011
16 Chamfer distance for seq MPH112_00169_05: 3.7081
17 Chamfer distance for seq N3OpenArea_03301_05: 0.0054
18 Chamfer distance for seq N3OpenArea_03301_04: 0.0012
19 Chamfer distance for seq N3Office_00034_03: 0.0402
20 Chamfer distance for seq MPH112_00169_04: 0.2317
21 Final Chamfer distance: 0.5223
22 Final EMD: 0.4545
23 Final F1 score: 0.7375
24 Category accuracy: 1.0000
25 Top 3 accuracy: 100.0000
```