chchchch

A Hybrid Approach to Strike a Balance of Sampling Time and Diversity in Floorplan Generation

By

Azmeraw Bekele Yenew

Submitted to the School of Information Technology and Engineering

In partial fulfillment of the requirements for the degree of

Master of Science in Artificial Intelligence

Supervised by: Dr. Beakal Gizachew.

Addis Ababa Institute of Technology

Addis Ababa University

Addis Ababa, Ethiopia

May , 2024

APPROVAL

This is to certify that this thesis titled **"A Hybrid Approach to Strike a Balance of Sampling Time and Diversity in Floorplan Generation"** is prepared by Azmeraw Bekele Yenew and submitted in partial fulfillment of the thesis-option requirements for the Degree of Master of Science in Artificial Intelligence at School of Information Technology & Engineering, Addis Ababa Institute of Technology.

Name	**Signature**	**Date**
_____	_____	_____
(Advisor)		
_____	_____	_____
(External Examiner)		
_____	_____	_____
(Internal Examiner)		

ABSTRACT

Generative models have revolutionized various industries by enabling the generation of diverse outputs, and floorplan generation is one such application. Different methods, including GANs, diffusion models, and others, have been employed for floorplan generation. However, each method faces specific challenges, such as mode collapse in GANs and sampling time in diffusion models. Efforts to mitigate these issues have led to the exploration of techniques such as regularization methods, architectural modifications, knowledge distillation, and adaptive noise schedules. However, existing methods often struggle to effectively balance both sampling time and diversity simultaneously. In response, this thesis proposes a novel hybrid approach that amalgamates GANs and diffusion models to address the dual challenges of diversity and sampling time in floorplan generation. To the best of our knowledge, this work is the first to introduce a solution that not only balances sampling time and diversity but also enhances the realism of the generated floorplans. The proposed method is trained on the RPLAN dataset and combines the advantages of GANs and diffusion models while incorporating different techniques such as regularization methods and architectural modifications to optimize our objectives. To evaluate the effect of the denoising step, we experimented with different time steps and found better diversity results at T=20. The diversity of generated floorplans was evaluated using FID across the number of rooms, and the results of our model demonstrate an average 15.5% improvement over the state-of-the-art houseDiffusion model. Additionally, it reduces the time required for generation by 41% compared to the housediffusion model. Despite these advancements, it is acknowledged that the proposed research may encounter limitations in generating non-Manhattan floorplans and when dealing with complex layouts.

Key words: Diffusion model, Diversity,Floorplan, GAN, sampling time, mode collapse.

Acknowledgements

First and foremost, I would like to thank God. He has given me strength and encouragement throughout all the challenging moments of completing this thesis. I am truly grateful for His unconditional and endless love, mercy, and grace.

I would like also to express my heartfelt gratitude to Dr. Beakal Gizachew, my MSc advisor. His expertise, guidance, and unwavering support have been instrumental in shaping my academic journey. I am truly grateful for his invaluable insights, constructive feedback, and constant motivation. Working under his supervision has been an enriching experience that has contributed significantly to my personal and professional growth. I extend my sincerest appreciation to Dr. Beakala Gizachew for his mentorship and dedication throughout my MSc program.

I would also like to extend my heartfelt gratitude to the architecture teachers at Wolaita Sodo University, the Ph.D. civil engineering students at Addis Ababa University, and the architectural engineering students at Dilla University, particularly Selamawit, for their invaluable support in sharing architectural knowledge. Their willingness to collaborate, exchange ideas, and provide insights has been instrumental in broadening my understanding of the field. I would also like to express my sincere appreciation to the thesis examiners, namely Dr. Adane Letta, Dr. Fantahun Bogale, and Dr. Natnael Argaw, for dedicating their time and providing valuable feedback during the progress and mock presentations. I am also deeply appreciative of Mr. Amanuel Negash for his exceptional advising and insightful feedback, which have significantly contributed to the refinement of my work. Additionally, I would like to acknowledge the unwavering support and idea-sharing from my colleagues, Mr. Tadele melese, Mr. Robel Habtamu , Mr. Challa Bekabil and Mr. sintayehu zekarias. Their contributions have been invaluable in shaping my perspective and fostering a collaborative environment.

Dedication

To my family.

. . .

To my academic advisor.

. . .

To my best friends.

. . .

Contents

Abstract ii

Acknowledgements iii

Dedication iv

Abbreviaions vii

List of Figures x

List of Tables xi

1 Introduction 1

1.1 Motivation . . . 3
1.2 Statement of the Problem . . . 4
1.3 Research Question . . . 6
1.4 Objective . . . 6
1.4.1 General objective . . . 6
1.4.2 Specific objective . . . 6
1.5 Significance . . . 7
1.6 Scope . . . 8
1.7 Contribution . . . 8
1.8 Thesis structure . . . 10

2 Literature review 11

2.1 Background . . . 11
2.2 Computational Methods . . . 12
2.2.1 Deep Generative Learning Method. . . . 14
2.3 Graph Neural Network . . . 27
2.3.1 Graph Convolutional Networks (GCNs) . . . 28
2.3.2 Graph Attention Networks (GATs) . . . 29
2.3.3 Convolutional Message Passing Networks (CMPNs) . . . 30
2.4 Regularization techniques . . . 31
2.4.1 Loss function . . . 32
2.4.2 Minibatch Discriminator . . . 34
2.4.3 Spectral normalization . . . 36
2.5 Architectural Modifications . . . 37

2.6 Computational techniques to reduce Sampling time . . . 41
2.6.1 Knowledge Distilation . . . 41
2.6.2 Adaptive Noise Schedule . . . 43
2.6.3 SDE solver . . . 44
2.7 Metrics . . . 46
2.7.1 FID score . . . 46
2.7.2 Realism . . . 47
2.8 Related work . . . 48

3 Methodology **52**
3.1 Research Metodology . . . 52
3.2 Design and development . . . 54
3.2.1 Data Acquisition . . . 54
3.2.2 Data preprocessing . . . 56
3.2.3 Modeling . . . 59

4 Experiments **75**
4.1 Implementation Details . . . 75
4.2 Training stability . . . 77
4.3 Resuls . . . 79
4.3.1 Diversity . . . 79
4.3.2 Sampling Time Dynamics Across Models . . . 82
4.3.3 Realsim of Generated floorpans . . . 85
4.4 Statistical Significance . . . 89
4.4.1 Diversity . . . 89
4.4.2 Sampling time . . . 90
4.4.3 Realism . . . 90
4.5 Ablation Studies . . . 91
4.5.1 Number of denoising steps . . . 91
4.5.2 Effect of Used Techniques . . . 93
4.6 Discussion . . . 97
4.6.1 Key finidngs . . . 99
4.6.2 Limitations . . . 100

5 Conclusion and recommendation **102**
5.1 Conclusion . . . 102
5.2 Recommendation . . . 103

References **105**

Abbreviaions

Symbol	Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Network
Avg	Average
CAD	Computer-Aided Design
CGAN	Conditional Generative Adversarial Networks
CNN	Convolutional Neural Network
CMPN	Convolutional Message Passing Network
CV	Computer Vision
DNN	Deep Neural Network
DL	Deep Learning
DDIMs	Denoising Diffusion Implicit Models
DDPM	Denoising Diffusion Probabilistic Model
DSRP	Design Science Research Process
EM	Euler-Maruyama
Exp	Experiment
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GCN	Graph Convolution Network
GMMs	Gaussian Mixture Models
GPUs	Graphics Processing Units
GNN	Graph Neural Network
GT	Ground Truth
HMMs	Hidden Markov Models
InfoGAN	Information Maximizing Generative Adversarial Network
LSTM	Long Short-Term Memory
LSGAN	Least Squares Generative Adversarial Network
MCMC	Markov Chain Monte Carlo
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multilayer Perceptron
MSGAN	Multiscale Generative Adversarial Network
NLP	Natural Language Processing

ProGAN	Progressive Generative Adversarial Network
Rtv	Raster-to-Vector RNN
Recurrent Neural Network	
RL	Reinforcement Learning
SAG-MSG-GAN	Self-Attention Guided Multi-Scale Gradient GAN
SDGs	Sustainable Development Goals
SDE	Stochastic Differential Equation
SVM	Support Vector Machine
VAE	Variational Autoencoder

LIST OF FIGURES

1.1 Evolution of generative AI. Source: Yihan cao et al. [1] 2

2.1 Computational methods in floorplan generation. 13
2.2 Generative Adversarial Network. 15
2.3 Forward and backward Diffusion Process. 19
2.4 Auto regressive Process. 23

3.1 Research Design 52
3.2 From Data Acquisition to Deployment 55
3.3 Raw Floorplan. 57
3.4 Extracted Masks. 58
3.5 Graph representation. 59
3.6 Floorplan Generation Model. 60
3.7 Noise Scheduler. Source: Lin et al. [2] 62
3.8 Message passing operation across connected and nonconnected nodes 65
3.9 A schematic overview for generators operation. 66
3.10 A schematic overview for discriminators operation. 68
3.11 Forward and backward of or model training. 70
3.12 Pipeline for training algorithms 71

4.1 Adversarial training 78
4.2 Time Taken vs Number of Samples Generated 83
4.3 Generated floorplans for three models with the same bubble diagram as input constraint. 86
4.4 Color coding for each room. 86
4.5 Input the complex graph along with its corresponding generated floorplan. 100

LIST OF TABLES

2.1 Deep generative Model based floorplan Generation. 26
2.2 Comparison of Graph Neural Networks based on various properties. 31
2.3 Summary of variat GAN architectures and their proporty. 39
2.4 Summary of computational techniques to reduce sampling time 45
2.5 Comparative Analysis of Floorplan Generation Methods 51

4.1 Model hyperparameters. 76
4.2 Diversity in the FID scores. (↓) indicate the lower the better metrics. 80
4.3 Tabular comparison for the time taken vs number of samples generated. 83
4.4 Realism scores among House-GAN++, HouseDiffusion, and ours. 87
4.5 Effect of diffusion steps in diversity. Diversity is measured by the FID scores. (↓) indicate the-lower the-better metrics. 91
4.6 Effect of Different Techniques on Diversity at T=20. 93

Chapter 1

Introduction

In the realm of architectural design, interior planning, and spatial optimization, the creation of efficient floorplans stands as a foundational challenge. The arrangement of rooms, corridors, and open spaces within a building significantly influences its functionality, aesthetics, and overall usability. Over the years, architects and designers have employed manual techniques to create floorplans, relying on their expertise and creativity to meet the specific needs of clients and occupants [3]. However, the digital era has opened up new possibilities with advanced computer-aided design (CAD) (Carpo 2017) and the fusion of computational algorithms. This has sparked a revolution in floorplan generation, with various techniques ranging from Procedural Methods [4, 5] to machine learning techniques [6, 7, 8, 9, 10].

Building upon these advancements, machine learning has emerged as a transformative force across various domains, pushing the boundaries of computational capabilities and revolutionizing the field of data analysis. In particular, deep learning, inspired by the intricate workings of the human brain, has played a pivotal role in reshaping how computers learn from vast amounts of data [11]. The introduction of multi-layer perceptrons (MLPs) demonstrated the potential of deep architectures in learning hierarchical representations, enabling the extraction of complex patterns and features from data [12]. However, progress in training deep neural networks faced obstacles due to computational limitations and the scarcity of extensive labeled datasets, hindering the realization of their full potential. Nonetheless, breakthroughs in efficient training algorithms and the availability of large-scale datasets eventually paved the way for unleashing the true power of deep learning and its applications across various domains.

The breakthrough in deep learning occurred with the introduction of efficient training algorithms and the availability of large-scale datasets. This progress coincided with the emergence of Convolutional Neural Networks (CNNs) [13], which were originally conceptualized by LeCun et al. in the late 1980s but truly made their mark in the 2000s. Specifically designed to process grid-like data, such as images, CNNs utilize three key layers: the convolutional layer, the pooling layer, and the fully connected layer. Their introduction ushered in remarkable

achievements in image classification tasks [14, 15], leading to advancements in image detection, recognition [16, 17, 18], and even image generation [19, 20]. The proliferation of data from the internet, social media, and connected devices played a pivotal role in fueling the success of deep learning. Coupled with advancements in parallel computing and Graphics Processing Units (GPUs), the availability of large-scale datasets opened doors to exploring more complex and deeper architectures.

As deep learning continued to push the boundaries of data synthesis, generative models underwent a remarkable evolution, leading to a revolution across multiple domains by enabling the synthesis of realistic and diverse data samples. While statistical models like Gaussian Mixture Models (GMMs) [21] and Hidden Markov Models (HMMs) [22] initially laid the foundation for understanding data distributions and generating samples based on learned statistical properties, the introduction of deep learning techniques ushered in a new era in generative modeling. Notably, Generative Adversarial Networks (GANs) [23], diffusion models [24], autoregressive models [25], and Variational Autoencoders (VAEs) [26] emerged as a powerful deep learning model, pushing the boundaries of data synthesis across various domains.

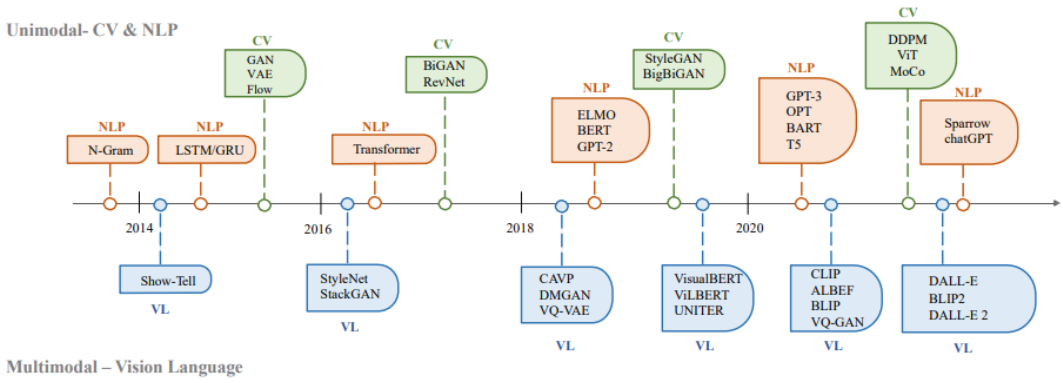

Figure 1.1: Evolution of generative AI. Source: Yihan cao et al. [1]

These advancements in generative modeling have found wide-ranging applications in computer vision [27, 28, 29], natural language processing [30], and data augmentation [31]. Moreover, the impact of these breakthroughs extends to architectural industries, particularly in the field of floorplan generation with tools like HouseGAN++ [6], HouseDiffusion [7], HouseGAN [8], END2END [9], and GTGAN [10], architects and designers now have powerful tools at their disposal to automate and enhance the process of floorplan creation. These generative models leverage deep learning techniques to generate diverse and innovative floorplan designs, enabling

architects to explore a wide range of possibilities and accelerate the design iteration process. By harnessing the capabilities of these models, architectural professionals can streamline their workflow, improve efficiency, and ultimately create floorplans that align with their clients' needs and preferences.

1.1 Motivation

A. Enhancing Floor Plan Diversity

The motivation behind conducting research on increasing the diversity of generated floor plans stems from the recognition that architectural spaces have a significant impact on people's lives. Traditional floor plans often lack diversity, limiting the range of options available to architects, designers, and individuals seeking unique and tailored spaces. By exploring and expanding the possibilities of floor plan design, we can unlock new opportunities for creativity, personalization, and functional optimization. This research aims to empower architects and designers to create environments that truly reflect the diverse needs, preferences, and identities of individuals and communities. By embracing diversity in floor plans, we can foster inclusive and inclusive spaces that enhance well-being, promote sustainability, and contribute to the overall quality of life.

B. Considering Sampling Time in Floor Plan Generation

Unlike existing methods that primarily prioritize the realism and visual fidelity of generated floor plans, our motivation lies in recognizing the significance of sampling time as a critical factor. While realism is important, it is equally crucial to consider the efficiency and speed of the generation process. By focusing on sampling time, we aim to develop innovative techniques that can generate diverse and visually appealing floor plans in a time-efficient manner. This research seeks to strike a balance between realism and computational efficiency, enabling architects and designers to explore a broader range of design options effectively and efficiently. By addressing the time aspect of floor plan generation, we can enhance productivity, streamline the design process, and ultimately facilitate quicker decision-making and realization of architectural visions.

C. Empowering Personalized Living Spaces

The demand for personalized and customized living spaces is on the rise, necessitating the development of generative models that can produce diverse floorplans. Individuals and families have distinct preferences and requirements when it comes to their ideal living spaces, including the arrangement of rooms, flow between spaces, and utilization of square footage. By developing an image floorplan generative model that prioritizes diversity, it becomes possible to cater to these individual needs and produce floorplans that align with specific lifestyle choices. Real-time generation coupled with diverse options would empower potential homeowners to visualize and customize their dream homes, leading to increased customer satisfaction and engagement in the real estate market.

1.2 Statement of the Problem

Generative models are a crucial element in the advancement of artificial intelligence, as they embody the aspiration to empower machines with the ability to create and imagine [1]. From early probabilistic models [21, 22] to the advent of deep learning architectures [23, 24, 26], the trajectory of generative models has been marked by an exponential growth in complexity and capability. While traditional methods relied on explicit probabilistic formulations, the emergence of deep generative models, notably exemplified by GANs [23], diffusion models [24], and variational autoencoders (VAEs) [26], revolutionized the field by enabling the learning of intricate data distributions directly from raw data [32].

However, the failure to capture and represent the full range of data distribution poses a fundamental challenge for most of these models [33]. When a generative model suffers from this challenge, it produces samples that are limited in variation, repetitive, or fail to encompass the entire spectrum of the patterns and modes found in the training data [34]. One prominent example of this challenge is observed in the generative adversarial networks (GANs) [23], which are known to suffer from the issue of mode collapse [35], leading to a lack of sampling diversity even when trained on multi-modal data. Mode collapse occurs when the generator network becomes stuck in the local minimum, causing it to produce samples that are similar to each other rather than sampling from the full range of possibilities in the distribution [34, 36].

To address this problem, researchers have explored various techniques. Karras et al. [37]

proposed the use of progressive growing GANs to mitigate mode collapse. Hoang et al. [38] investigated the effectiveness of employing multiple generators. Saad et al. [39] introduced techniques such as self-attention [40] and different regularization techniques [41, 42] to enhance sample diversity. Additionally, HouseGAN++ [6] employed conditioning and regularization techniques in progressive training to generate diverse floorplans. While the mentioned techniques contribute to enhancing diversity, they require further refinement in order to capture the full range of the distribution, especially when compared to the Diffusion model [43].

Unlike GANs [23], which have a low level of capturing data distribution, diffusion models [24] offer high generative diversity by effectively capturing the high-dimensional data distribution. Through the iterative diffusion process, which involves introducing random noise and transforming the input, diffusion models explore various regions of the distribution. This allows them to capture the intricate patterns and complex structures present in the training data, resulting in a wide range of generated outputs and shows demonstrated superiority over GANs in image synthesis [43]. However, the iterative nature of the diffusion process, with multiple diffusion steps, can lead to computational overhead and increased sampling time [33, 44]. This slowdown occurs because each diffusion step necessitates complex computations, involving the propagation of noise across multiple time steps, resulting in longer processing times compared to simpler generative models like GANs [23]. To alleviate this issue, multiple methods have been proposed, including knowledge distillation [45], learning an adaptive noise schedule [46], introducing non-Markovian diffusion processes [47], and using better SDE solvers for continuous-time models [48]. These methods either suffer from significant degradation in sample quality or still require many denoising steps [33].

From the above two scenarios, we understand that the challenge of striking a balance between the sampling time and the diversity of generated output in generative models encapsulates a fundamental dilemma. On one hand, achieving rapid generation is imperative for real-time applications such as image and video synthesis [27, 28], where prompt response times are essential. On the other hand, ensuring a wide range of diverse outputs is crucial for maintaining the fidelity and richness of the generated content, enabling the model to capture the complexity of real-world data distributions effectively. To effectively navigate this trade-off, it is crucial to adopt innovative approaches that optimize sampling efficiency while preserving the diversity and quality of the generated outputs. This will pave the way for practical and versatile

generative models applicable to various domains, including floorplan generation. Our proposed method aims to achieve these objectives by harnessing the advantages of diffusion models [49] and leveraging the strengths of GANs [23].

In our approach, the model is trained using a multimodal fashion [50], allowing us to optimize the denoising steps. By incorporating a GAN model at its core, we capitalize on its ability to generate diverse samples in adversarial training. This not only enhances the efficiency of the sampling process but also maintains the richness and variety of the generated outputs. Additionally, we employ a graph neural network [51] at a core to further enhance the realism and overall performance of the generated floorplans. This integration enables the model to capture intricate spatial relationships and dependencies within the floorplan data, resulting in more accurate and visually appealing outputs.

1.3 Research Question

RQ1: How do various regularization techniques affect the diversity of floorplan layouts generated?

RQ2: How can we achieve optimized sampling time and enhanced diversity through the combination of generative models?

1.4 Objective

1.4.1 General objective

To optimize sample generation, this study aims to develop a hybrid approach for floorplan generation that effectively balances sampling time and diversity.

1.4.2 Specific objective

- To investigate and analyze existing generative models in order to assess their strengths and weaknesses.
- To experiment with various regularization techniques that can improve the diversity of the generated samples.
- To optimize sampling time and achieve faster and more diverse sample generation by leveraging the strengths of different generative models.

- Conduct experimental evaluation and comparison to assess the performance of proposed techniques against existing methods.

1.5 Significance

Balancing the sampling time and diversity of floorplan generation holds significant importance for several reasons:

- Efficiency: Generating floorplans can be a time-consuming process, especially when dealing with complex layouts or large datasets. Balancing sampling time ensures that the generation process is efficient and practical for real-world applications. By reducing the time required for generation, it becomes feasible to generate floorplans on-demand, enabling faster iterations and improving productivity in architectural and real estate industries.

- Realism and Variety: Balancing diversity in floorplan generation ensures that the generated layouts exhibit a wide range of possibilities, capturing the natural diversity found in real-world floorplans. This is important because overly similar or repetitive floorplans can limit creativity and practical usability. By achieving a balance between sampling time and diversity, the generated floorplans are more likely to capture the richness and variety of real floorplan designs, providing architects, designers, and clients with a broader set of options to explore.

- User Preferences and Customization: Different users may have diverse preferences and requirements when it comes to floorplan designs. Some may prioritize spaciousness, while others may emphasize efficient space utilization or specific room arrangements. Balancing diversity ensures that the generated floorplans cater to a wider range of user preferences, allowing for customization and personalization. By considering various design possibilities within a reasonable time frame, the floorplan generation process becomes more adaptable and user-centric.

- Exploration of Design Space: Balancing sampling time and diversity allows for more comprehensive exploration of the design space. Architects and designers can experiment with different layouts, room configurations, and architectural elements to discover novel and innovative floorplan designs. This exploration enables the discovery of unconventional yet functional solutions, leading to unique and inspiring architectural creations.

- Sustainability and energy efficiency: The utilization of AI-driven computational methods, exemplified by Athens, showcases transformative potential in advancing Sustainable Development Goals (SDGs) such as SDG 11 and SDG 13 (UN, 2015; UN, 2019). By optimizing energy management and climate modeling, AI fosters greener urban environments. In architecture, AI-driven floorplan generation prioritizes sustainability metrics, enhancing resource efficiency and environmental quality.

Overall, balancing the sampling time and diversity of floorplan generation is essential for efficient, realistic, and customizable generation processes. It empowers architects, designers, and clients with a broader range of options, facilitates design exploration, and improves the overall quality and usability of generated floorplans.

1.6 Scope

The research scope encompasses the development and exploration of different training strategies and architectural innovations to address the trade-off between sampling diversity and sampling time in generative models for 2D residential floorplan generation. While the primary focus is on optimizing factors like diversity, sampling time, and realism, addressing the complexities introduced during training falls outside the scope of this study, despite their inadvertent impact. The research does not consider subjective issues such as personal preferences, cultural influences, or design choices in the generation of floorplans. The focus of the study is primarily on algorithmic and computational aspects of floorplan generation, optimizing factors like diversity, sampling time, and realism. While acknowledging the importance of subjective considerations, this research aims to explore and propose methods that primarily address technical aspects of floorplan generation. The research will involve evaluating the proposed approaches through comprehensive experiments, comparing their performance against existing models, and assessing metrics such as diversity measures and sampling time benchmarks.

1.7 Contribution

The aim of this research is to address the inherent trade-offs between sampling time and diversity in floorplan generation. Recognizing the importance of efficiently generating diverse and realistic floorplans, our study endeavors to contribute in three key areas.

- **Hybrid on graph:** To the best of our knowledge, our method pioneers the integration of a hybrid diffusion model with Generative Adversarial Networks (GANs) within graph neural networks. Current floorplan generative models [6, 7] only focus on a single model, which cannot address dual challenges. Our proposed method integrates the strengths of both the diffusion model and GANs, leveraging their complementary capabilities to significantly improve the efficiency of the sampling process. By seamlessly integrating these models within graph neural networks, we achieve remarkable advancements in generating diverse and high-quality samples while optimizing sampling time. This contribution not only enhances the efficiency of sampling procedures but also sets a new benchmark for the integration of diverse generative models within graph-based frameworks.

- **Architectural Modification:** By introducing an architectural modification that integrates an attention mechanism with convolutional message passing, this study addresses the oversight of weight-based message aggregation in existing message-passing networks [52], which leads to equal attention being given to all nodes. This study leverages the synergy between attention mechanisms and convolutional message passing to address a critical challenge in floorplan generation, enabling a broader spectrum of design possibilities and enriching the variability of generated floorplans while opening up new avenues for architectural exploration and creativity.

- **Modified Qualitative metrics:** We propose a qualitative metric that utilizes questionnaires to tangibly measure floorplan quality, offering valuable insights into the efficacy of our approach. This innovative metric provides a structured framework for assessing the qualitative aspects of generated floorplans, capturing nuanced aspects of design quality that may not be fully captured by traditional quantitative measures [8] alone. By incorporating user feedback through questionnaires, we ensure that our evaluation process aligns closely with human perception and preferences, ultimately enhancing the interpretability and applicability of our findings in real-world architectural contexts.

Through these endeavors, our research seeks to transform floorplan generation by effectively navigating the trade-off between sampling diversity and efficiency.

1.8 Thesis structure

The subsequent sections of this document delve into the development of the proposed technique. In Chapter 2, an extensive review of relevant literature and related works is provided, encompassing various aspects such as generative models in section 2.2, graph neural networks in section 2.3, regularization techniques in section 2.4, and techniques for reducing sampling time in section 2.6, metrics in section 2.7. Chapter 3 intricately details the methodologies employed, elucidating on the architectures and techniques utilized. In Chapter 4, experimental procedures and findings are presented. Finally, Chapter 5 culminates the study by succinctly summarizing its key contributions and offering valuable recommendations for future research endeavors.

Chapter 2

Literature review

This literature section delves into various methodologies that have been explored in the field of floorplan generation. It provides an in-depth analysis of different approaches and techniques employed by researchers to address the challenges and complexities associated with this subject matter. By reviewing a wide range of scholarly works and studies, this section aims to present a comprehensive overview of the existing methodologies, including their strengths, limitations, and contributions to the field.

2.1 Background

The evolution of floorplan generation from manual methods to computational approaches marks a significant advancement in architectural design practices. Traditionally, floorplans were created manually by architects and designers using drafting tools such as pencils, rulers, and stencils [3]. This manual process required meticulous attention to detail and precise measurements, making it time-consuming and labor-intensive. However, with the advent of computational methods, the process has undergone a remarkable transformation.

Computational methods for floorplan generation leverage the power of technology and computer algorithms to automate and streamline the design process. These methods enable architects and designers to create floorplans more efficiently, accurately, and flexibly. The earliest computational methods involved the use of computer-aided design (CAD) software [53], which allowed architects to create floorplans digitally. CAD software offered tools for drawing lines, shapes, and objects, as well as the ability to edit and manipulate elements with ease. This marked a significant improvement over manual drafting methods, as changes and modifications could be made more quickly and effortlessly.

As technology advanced, more sophisticated computational techniques were developed, including Procedural based [4, 5, 54, 55, 56] . This procedural method involved defining a set of design rules and constraints that governed the arrangement and relationships of architectural

elements within the floorplan. While procedural methods offer a structured approach to floorplan generation, they often struggle to capture the nuanced design elements and contextual relevance present in modern architecture

In recent years, the emergence of generative models and machine learning techniques has revolutionized floorplan generation even further. Generative models have the ability to learn patterns and generate novel floorplan designs based on existing examples. These models use complex neural network architectures to learn patterns and relationships from existing floorplan data and then generate new, coherent, and contextually relevant floor plans. This technology has gained significant attention in recent years due to its ability to produce highly realistic and diverse designs. Various generative models like Generative Adversarial Networks (GANs) [6, 8], Diffusion generative model [7], and Autoregressive Models [9] are employed for floorplan generation.

This advancement addresses various user needs and industry trends by revolutionizing the architectural and real estate sectors. By harnessing different algorithms, floorplan generation tools have become remarkably efficient, enabling the creation of accurate floorplans in a fraction of the time and effort required by manual drafting [57]. These tools offer a high degree of customization and flexibility, accommodating various design preferences and specific requirements. Moreover, it optimizes space utilization, ensuring that floorplans are practical and well-designed by considering factors such as traffic flow and natural lighting [58]. Additionally, this method provides valuable design assistance and inspiration by analyzing vast amounts of data, offering creative suggestions, and adhering to industry best practices [9]. Collaboration is enhanced through computational methods, facilitating the involvement of multiple stakeholders and enabling iterative design processes [59]. Lastly, the accessibility and democratization of AI-powered tools have revolutionized the field, empowering individuals without formal training to engage in architectural design.

2.2 Computational Methods

The evolution of computational methods has profoundly impacted various areas across science, engineering, and technology, extending to fields like floorplan generation. As illustrated in Figure 2.1, computational methods are integral to this process, encompassing both procedural methods [54, 60] and deep generative learning techniques [23, 24, 26]. Procedural methods

involve the automatic generation of complex structures, scenes, or content using algorithms, rules, and parameters [61]. Moreover, within the realm of procedural methods, specific algorithms play crucial roles in the floorplan generation process. Subdivision algorithms [4], for instance, iteratively refine an initial geometric shape by dividing it into smaller, more detailed components, enhancing the structure with each subdivision step. Complementarily, Inside Out algorithms [5] initiate floorplan generation from interior spaces, gradually expanding outward to ensure efficient space utilization and seamless connectivity. Tip Placement algorithms [55] strategically position key architectural elements such as entrances, corners, and landmarks to guide the layout and flow of the floorplan, enriching both functionality and aesthetics. Furthermore, Growth-based algorithms [62] simulate organic growth processes, gradually expanding and evolving the floorplan layout based on predefined rules and constraints. This dynamic approach often yields natural-looking and adaptive designs that effectively respond to spatial requirements and environmental factors.

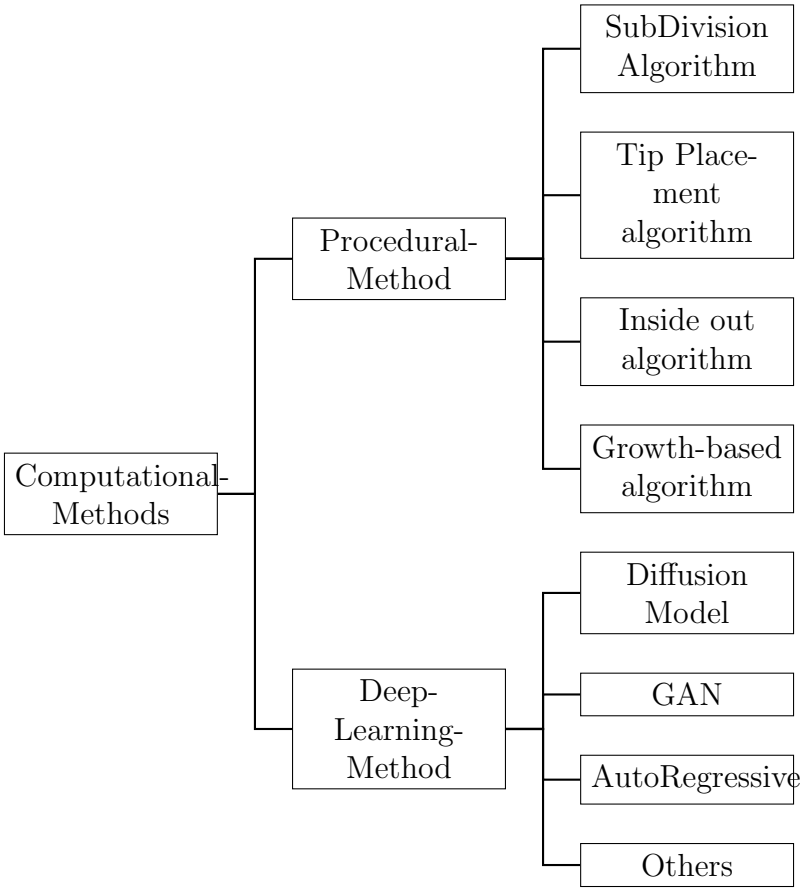

Figure 2.1: Computational methods in floorplan generation.

While procedural methods have long been effective at generating structured and rule-based

content, they often face limitations when it comes to capturing the intricate details and complexity found in real-world data. Procedural techniques rely on predefined algorithms and rules, which can result in repetitive and predictable outputs. This restricts their ability to accurately represent the rich and diverse nature of real-world environments. Moreover, procedural methods can be time-consuming and require significant manual effort to fine-tune parameters and achieve desired results [61]. They may struggle to achieve high levels of realism and variability, as they lack the ability to learn and generalize from large datasets. Procedural methods also face challenges in capturing the nuances of natural textures, lighting, and spatial relationships, resulting in outputs that may appear artificial or lacking in visual appeal. The evolution towards deep generative models has been driven by the need to overcome these limitations.

2.2.1 Deep Generative Learning Method.

Deep generative models have a rich history in artificial intelligence, particularly in the context of the development of machine learning techniques. While the origins of deep learning can be traced back to the 1950s, early models such as Hidden Markov Models [63], Naive Bayes [64], and Gaussian Mixture Models [65] played foundational roles in statistical modeling and data generation. These models, although not deep in architecture, laid the groundwork for later advancements in deep generative modeling. A deep generative model belongs to a category of machine learning models that harness deep neural networks to understand and represent complex data distributions. Unlike conventional discriminative models [66, 67], which primarily focus on predicting labels or making classifications, deep generative models aim to grasp the underlying structure of the data and generate new samples that resemble the original data. By leveraging multiple layers of neural networks, these models can capture intricate patterns and dependencies within the data.

Deep generative models, such as Generative Adversarial Networks (GANs) [23], autoregressive models [25], and Diffusion models [24], have demonstrated impressive capabilities in tasks like image generation [27, 28] and text synthesis [68]. These advancements have greatly influenced generative modeling, with applications ranging from creative art generation to scientific data synthesis. In the context of floorplan design, GANs, autoregressive models, and diffusion models have played a significant role, revolutionizing the floorplan generation process and introducing innovative techniques for transformative solutions.

A. Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [23] represent a breakthrough in generative modeling, capable of creating realistic data samples, including image synthesis [69, 70], data augmentation [71], super-resolution[72], and style transfer [73, 74]. At their core, GANs consist of two neural networks: a generator and a discriminator, as shown in Figure 2.2. The generator network learns to generate data, such as images, from random noise samples. Simultaneously, the discriminator network learns to distinguish between real data and data generated by the generator. The training process involves a competitive interplay between these two networks: as the generator strives to produce increasingly realistic samples to fool the discriminator, the discriminator adapts to become more adept at discerning real from fake data. Through this adversarial training dynamic, both networks improve iteratively until an equilibrium is reached, ideally resulting in the generation of highly realistic synthetic data.

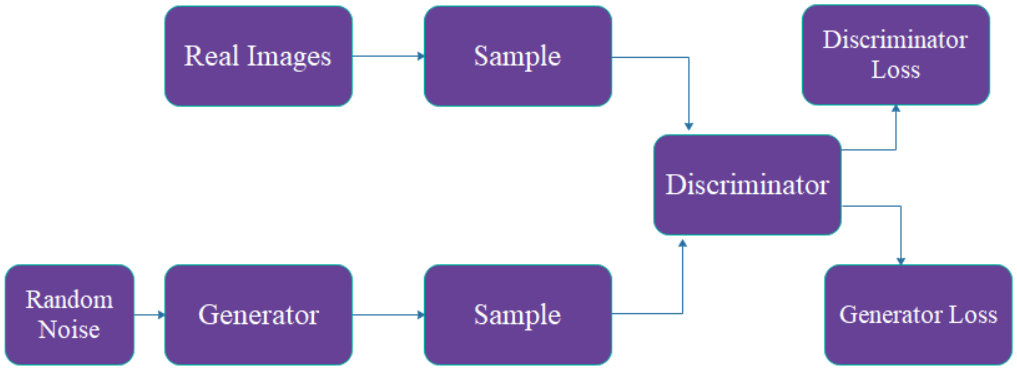

Figure 2.2: Generative Adversarial Network.

One of the earliest and most influential GAN variants is the Vanilla GAN [23] proposed by Ian Goodfellow and his colleagues in 2014. As shown in the training algorithm (Algorithm 1), it involves a simple architecture where the generator creates samples from random noise, and the discriminator distinguishes between real and fake data. Despite its simplicity, Vanilla GANs can produce high-quality samples. To address challenges such as mode collapse and training instability, researchers have developed numerous GAN variants. For instance, Deep Convolutional GANs (DCGANs) [75] introduced convolutional layers to both the generator and discriminator networks, enabling the generation of high-resolution images with more realistic

details. Conditional GANs (cGANs) [76] extend the basic GAN framework by conditioning the generator on additional information, such as class labels, enabling controlled generation of specific classes of data. Despite their remarkable success, GANs pose challenges such as mode collapse [35], training instability [77], and evaluation difficulties. Researchers continue to explore novel architectures, training techniques, and applications to address these challenges and further enhance the capabilities of GANs.

Algorithm 1 Training a Generative Adversarial Network (GAN)

Require: Training dataset D, learning rate η, number of training iterations T, discriminator network $D(\cdot)$, generator network $G(\cdot)$
Ensure: Trained discriminator and generator parameters

Initialize discriminator and generator parameters θ_D and θ_G
for $t = 1$ to T **do**
    Sample a mini-batch from D
    Sample random noise $\mathbf{z}$ from a prior distribution
    Generate fake data samples $\mathbf{x}_{\text{fake}} \leftarrow G(\mathbf{z}; \theta_G)$
    Compute discriminator loss $L_D \leftarrow -\mathbb{E}_{\mathbf{x}\sim D}[\log D(\mathbf{x}; \theta_D)] - \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z};\theta_G)}[\log(1 - D(\mathbf{x}; \theta_D))]$
    Update discriminator parameters $\theta_D \leftarrow \theta_D - \eta \cdot \nabla_{\theta_D} L_D$
    Sample new random noise $\mathbf{z}$ from a prior distribution
    Generate fake data samples $\mathbf{x}_{\text{fake}} \leftarrow G(\mathbf{z}; \theta_G)$
    Compute generator loss $L_G \leftarrow -\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z};\theta_G)}[\log D(\mathbf{x}; \theta_D)]$
    Update generator parameters $\theta_G \leftarrow \theta_G - \eta \cdot \nabla_{\theta_G} L_G$
end for

The training algorithm for a Generative Adversarial Network (GAN) as outlined in Algorithm 1 involves a series of iterative steps to train both the discriminator and the generator networks. Initially, the discriminator and generator parameters are randomly initialized. Subsequently, for each training iteration, a mini-batch of real data samples is sampled from the training dataset, and random noise vectors are sampled from a prior distribution. The generator then produces fake data samples from the noise vectors. The discriminator computes its loss by distinguishing between real and fake data samples, while the generator computes its loss by attempting to fool the discriminator. These losses are then used to update the parameters of the discriminator and generator networks using gradient descent with a specified learning rate. This process is repeated for a predetermined number of training iterations, resulting in the trained discriminator and generator networks capable of generating realistic data samples.

The paper by Zheng et al. [**?**] contributes to this growing body of research by introducing a novel approach for generating floorplans. The proposed method aims to transform input

images featuring design boundaries into detailed interior designs within those boundaries. This approach involves training on two distinct datasets and incorporates preprocessing steps such as boundary production and masking. The model's training is conducted using Pix2pixHD [78] on labeled datasets comprising resized images and their corresponding masked versions, illustrating a comprehensive strategy for floorplan generation that integrates advanced machine learning techniques. While the model tackles compatibility challenges, it faces drawbacks such as extended training times and difficulty in accurately placing key spaces like living rooms, kitchens, and bedrooms. Moreover, issues related to achieving visual and functional realism and customization remain to be addressed.

Lim et al. [79] introduces a method for generating architectural floor plans leveraging grid data and AI techniques. The operation of this approach involves analyzing grid-based representations of spaces and applying GAN to generate floor plans automatically. Its contribution lies in providing a systematic and efficient way to create architectural layouts, particularly useful in scenarios where manual design is time-consuming or impractical. By harnessing AI, the system can produce floor plans that optimize space utilization and adhere to architectural constraints. However, limitations may arise from the complexity of accurately representing spatial relationships within grid data and the potential for generated designs to lack the nuanced creativity of human-designed floor plans. Additionally, the effectiveness of the system may depend on the quality and comprehensiveness of the input grid data, which could impact the diversity and suitability of the generated floor plans.

In ArchiGAN by Chaillou et al. [80] a comprehensive generative framework is presented, capable of generating entire apartment building designs by leveraging multiple generative models. Its performance is marked by the ability to produce diverse and realistic designs across different architectural elements. The primary contribution lies in its holistic approach to design, incorporating various generative models to address different aspects of the process. Yet, challenges may arise from coordinating multiple models within the stack and capturing the full spectrum of architectural styles and preferences.

The paper presented by Shidong Wang et al. [81] for generating floorplans, aims to generate diverse floorplans for residential buildings that meet the conditions of human-environment interaction outlined in the activity map. Unlike other methods, they use a human activity map which is extracted from the input boundary and used to guide the floorplan generations

from the input boundaries. This human activity maps is performed either automatically with a GAN model trained from synthetic human-activity maps or semi-automatic approach by using bi-RRT [82] based on user-specified furniture locations. To produce the vectorized floorplans the paper proposes two-stage approaches, the first stage is named as ActFloorr-GAN and aims to synthesize a rasterized human activity map from the input boundary. Then this pixel-wise representation is converted into a vectorized way in the second stage. While they didn't provide the exact measurement, they tried to address functional realism by using a human activity map. While they generate diverse floorplans, limitations arise from the reliance on accurate activity recognition, which may introduce errors, and from the potential struggle with complex activity patterns not adequately represented in the training data.

A new approach introduced by Nauata et al. [8] is HouseGAN by involving a generative adversarial network with graph constraints, where both the generator and discriminator utilize a relational architecture. The central concept is to incorporate the constraint within the relational network's graph structure. To achieve this, the model employs conv-MPN (Convolutional Message Passing Network) [52] for graph updates and upsampling in the generator, as well as downsampling in the discriminator. However, it's essential to recognize the approach's limitations, including restricted rectangular shape generation, the absence of room size incorporation, and the omission of door placements in the graph's edges due to spatial adjacency, all of which suggest potential avenues for future refinements and expansions of the proposed method.

An improved iteration of HouseGAN [8], which we refer to as HouseGAN++ [6], has been introduced. HouseGAN++ merges a relational GAN constrained by graphs with a conditional GAN to address the issue of baselines. This integration allows for iterative improvement, as a previously generated layout serves as the next input constraint. Notably, this research unveils the effectiveness of a simple non-iterative training approach known as component-wise GT-conditioning in training such a generator. Moreover, the iterative generator presents a new avenue for refining chosen metrics through meta-optimization strategies by regulating the timing of input constraint passage during the iterative layout enhancement process.

Research conducted by Tang et al. [10] introduced Graph Transformer GANs (GTTGAN), a notable advancement in the field. GTTGAN operates by seamlessly integrating graph transformer networks into the GAN framework, thereby enabling the generation of house designs that adhere to graph-based constraints. The model operates by first encoding the graph-

based constraints representing architectural features and layout requirements. It then employs transformer-based architectures to generate house designs that adhere to these constraints, ensuring structural coherence and architectural integrity. The primary contribution of GTTGAN lies in its innovative approach to leveraging graph representations for guiding the generation of house designs, offering a solution that balances design creativity with architectural constraints. By incorporating graph constraints into the GAN architecture, GTTGAN streamlines the design process and provides architects and developers with a tool for efficiently exploring diverse design possibilities. However, limitations may arise from the complexity of encoding and interpreting graph-based constraints, as well as from potential challenges in training the model effectively with limited or noisy data. Additionally, while GTTGAN offers valuable automation in the design process, it may not fully capture the intricacies and nuances of human-designed house plans, potentially limiting its applicability in certain architectural contexts.

B. Diffusion Models

The paper Deep Unsupervised Learning using Nonequilibrium Thermodynamics [83] published in 2015 introduced the concept of diffusion models as a method for deep unsupervised learning. Drawing inspiration from non-equilibrium statistical physics, the authors proposed a novel approach to modeling data distributions. The core idea was to iteratively degrade the structural information in a data distribution through a forward diffusion process. They then developed a reverse diffusion process to restore the original structure, resulting in a flexible and tractable generative model. This approach provided a powerful framework for unsupervised learning, enabling the generation of high-quality samples and the exploration of complex data distributions.

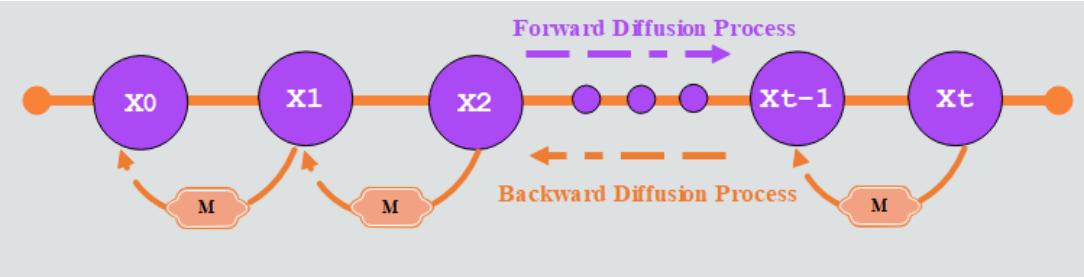

Figure 2.3: Forward and backward Diffusion Process.

As shown in Figure 2.3, the diffusion model comprises two processes Forward and Backward, where the forward process involves generating noise through a fixed noise vector, containing random values or samples from basic distributions like Gaussian noise, with the vector's size

aligned with the size of the intended generated data, such as images or text sequences. The noise vector is passed through a sequence of diffusion steps.

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_t = (1-\beta_t)x_{t-1}, \Sigma_t = \beta_t I) \tag{2.1}$$

The equation (equation 2.1) above embodies the forward diffusion process within a probabilistic framework. This equation characterizes how a random variable x evolves over discrete time steps, with x_t representing its state at time t. The conditional distribution $q(x_t|x_{t-1})$ captures the likelihood of x_t given the previous value x_{t-1}, and is modeled as a Gaussian distribution $\mathcal{N}$. The mean μ_t of this distribution is determined by $(1-\beta_t)x_{t-1}$, reflecting how x_t depends on its prior state x_{t-1} with the influence controlled by the parameter β_t. Additionally, the covariance matrix Σ_t is specified as β_t times the identity matrix I, regulating the variability of x_t and indicating the level of uncertainty. Noise scheduling [84] is an important aspect of diffusion models that involves adding the right amount of noise to arrive at an isotropic Gaussian distribution with various types of schedules like linear [49], cosine [84], or combined approaches that determine how the noise increases over time.

Another integral component of the diffusion model is the Reverse Process, which involves a sequence of iterative steps that typically extend over hundreds to thousands of iterations. During each iteration of the reverse process, the noise vector undergoes sequential updates aimed at refining its representation. This refinement is accomplished through conditioning based on the actual training data. By progressively applying these iterative updates, the model learns to align the noise vector with the underlying patterns and structures present in the training data.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.2}$$

In the backward diffusion process (Equation 2.2), the model iteratively updates the noise vector by conditioning it on the actual training data. In the conditional distribution where x_t represents the current value in the diffusion process, and x_{t-1} represents the previous value. This conditional distribution is assumed to follow a multivariate normal distribution, with mean $\mu_\theta(x_t, t)$ and covariance $\Sigma_\theta(x_t, t)$. These mean and covariance parameters are learned during the training of the diffusion model. To refine the noise vector in the backward diffusion process, the model samples from the conditional distribution. This sampling step allows the noise

vector to progressively align with the target data distribution, capturing the complex patterns and dependencies present in the training data. By iteratively applying the backward diffusion process, the model updates the noise vector over a series of steps, often spanning from hundreds to thousands. This sequential refinement facilitates the generation of high-quality samples that closely resemble the training data. The backward diffusion process, along with the forward diffusion process, enables the diffusion generative model to generate diverse and realistic samples by leveraging the learned conditional distribution and the associated mean and covariance parameters.

Training diffusion models involves an iterative process where data undergoes denoising and reconstruction to learn the underlying probability distribution. Algorithm 2 outlines the steps for training a diffusion model based on a given dataset. First, the dataset is shuffled and divided into smaller parts known as mini-batches. Within each mini-batch, the algorithm executes denoising steps to enhance the quality of the generated samples. Next, random noise sampled from a standard normal distribution is used to create new samples by leveraging the model architecture. These generated samples are evaluated for quality using a denoising loss function. Subsequently, the model parameters are updated using gradient descent, with the learning rate dictating the step size. By repeating this process for a specified number of iterations, the diffusion model progressively learns to produce high-quality samples that closely resemble the characteristics of the training dataset.

Algorithm 2 Training a Diffusion Model

Require: Training dataset D, number of denoising steps S, learning rate η, model architecture f(model_architecture), number of training iterations T

Initialize model parameters θ
for $t = 1$ to T **do**
    Shuffle and split D into mini-batches
    for each mini-batch B in D **do**
        for $s = 1$ to S **do**
            Sample noise $\epsilon_s \sim \mathcal{N}(0, I)$
            Generate samples $x_t^s \leftarrow f(\text{model\_architecture})(\theta, x_t^{s-1}, \epsilon_s)$
        end for
        Compute denoising loss $L_t \leftarrow \text{DenoisingLoss}(x_t^S, B)$
        Update model parameters $\theta \leftarrow \theta - \eta \cdot \nabla L_t$
    end for
end for

Diffusion models have found diverse applications across various domains, owing to their

ability to generate high-quality samples and capture complex data distributions. In the field of computer vision, diffusion models have been employed for tasks such as image synthesis [49], image inpainting [85], and style transfer. Additionally, diffusion models have been applied in natural language processing for tasks like text generation [86] and language modeling, where they have demonstrated the ability to generate coherent and contextually relevant text. Furthermore, the incorporation of diffusion models spans various layout generation applications, including document layout generation [87, 88], where diffusion models organized the arrangement of document elements to shape comprehensive layouts. Demonstrating this versatility, the houseDiffusion generative model [7], as indicated in Table 2.1, endeavors to generate vectorized floorplans seamlessly using diffusion processes. They introduces a groundbreaking approach that leverages a diffusion model and a core Transformer architecture[89] for the generation of intricate vector-based floorplans. These floorplans, comprised of interconnected polygons outlining rooms and doors, are created through a process guided by attention masks based on graph-constraints. This process involves a combined discrete and continuous noise reduction approach, resulting in accurate geometric relationships among architectural elements.

C. Autoregressive Models

Autoregressive models are a class of generative models that capture dependencies within sequential or structured data. As shown in figure 2.4, the model is designed to generate new samples by estimating the conditional probability of each element in the sequence given the previous elements. By iteratively generating data elements, autoregressive models excel at capturing intricate patterns, making them especially effective in scenarios where the order and context of elements matter significantly. Autoregressive models have shown significant success in various domains, including natural language processing [90, 91] and computer vision [92, 93]. Notably, autoregressive models have witnessed substantial advancements, with innovative architectures like Transformer-based models [89] achieving exceptional performance in language understanding and generation tasks.

$$p(x) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i|x_{<i}) \tag{2.3}$$

The equation 2.3, above encapsulates the foundational concept of an autoregressive model, widely employed across various generative tasks. In this context, x denotes a sequence of

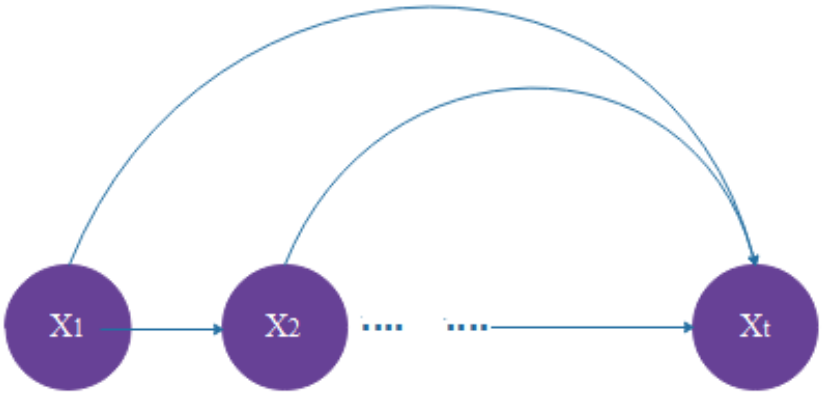

Figure 2.4: Auto regressive Process.

elements, often corresponding to pixels within images, while x_i signifies the ith element within the sequence. The equation's essence lies in expressing the probability distribution of the entire sequence x as a product of conditional probabilities. Each element's likelihood $p(x_i|x_{<i})$ is intricately modeled with respect to all prior elements, thus capturing complex dependencies existing within the sequence. This formulation allows autoregressive models to generate new samples by iteratively predicting each element in the sequence, resulting in the generation of outputs that exhibit complex patterns and closely resemble the characteristics of the training data.

The training algorithm for autoregressive models, as depicted in Algorithm 3, follows an iterative optimization approach to update the model parameters by minimizing prediction errors over sequential data. The algorithm processes a training dataset, splits it into mini-batches, and predicts each element in a sequence while updating hidden states iteratively. The loss incurred at each prediction step is accumulated to compute the batch loss, and model parameters are adjusted using backpropagation through time. Through a series of training iterations, this process enables autoregressive models to capture sequential dependencies and generate coherent sequences, making it fundamental for tasks like natural language processing [94, 95] and image processings [96].

Algorithm 3 Training an Autoregressive Model

Require: Training dataset D, model architecture f(model_architecture), learning rate η, number of training iterations T
Ensure: Trained model parameters θ

Initialize model parameters θ
for $t = 1$ to T **do**
    Shuffle and split D into mini-batches
    for each mini-batch B in D **do**
        Initialize hidden state h_0
        for $i = 1$ to sequence length L **do**
            Input x_i from B and previous hidden state h_{i-1}
            Compute model prediction
            $y_i \leftarrow f(\text{model\_architecture})(\theta, x_i, h_{i-1})$
            Compute loss $L_i \leftarrow \text{Loss}(y_i, x_{i+1})$
            Update hidden state h_i
        end for
        Compute batch loss $L_B \leftarrow \sum_{i=1}^{L} L_i$
        Update model parameters $\theta \leftarrow \theta - \eta \cdot \nabla L_B$
    end for
end for

The paper by Liu et al. [9] introduces a novel autoregressive approach to synthesizing floorplans using 1-D vector sequences, enhancing user interaction and customization. The framework consists of a two-stage process involving a draft stage and a multi-round refining stage. The initial floorplan sequence is generated using a graph convolutional network (GCN) [97] and an autoregressive transformer network [89]. A panoptic refinement network (PRN) refines the design in the second stage, aided by a geometric loss to ensure proper room connectivity. As shown in table 2.1, in contrast to prior methods this vectorized approach produces more realistic and functional designs, achieving higher usability and visual appeal by using panoptic refinement network (PRN). The framework's effectiveness is demonstrated through experiments on real-world floorplan data, showcasing its superiority over previous state-of-the-art methods.

D. Others

In contrast to the previously mentioned generative models such as GANs, diffusion models, and autoregressive models, this approach delves into the realm of floorplan generation using distinct techniques like Convolutional Neural Networks (CNNs) [13], Graph convolutional Networks (GCNs) [97] or other techniques.

Prior to the utilization of conventional generative models [23, 24, 25] in the domain of floor-

plan generation, researchers employed a variety of deep learning techniques to explore innovative approaches. In 2018, Liu et al. [98] proposed a Unified Framework for Floorplan Reconstruction from 3D Scans. The research focuses on automating indoor floorplan reconstruction by utilizing a smartphone's RGBD streams captured while walking through a house. The proposed solution, FloorNet, introduces a unique deep neural architecture that effectively addresses the challenge of processing vast 3D space data. FloorNet employs three neural network branches: PointNet for 3D point processing, CNN with 2D point density images for enhanced local spatial reasoning, and CNN with RGB images to utilize full image information. These branches exchange intermediate features to harness the strengths of all architectures. A benchmark was established using RGBD video streams from 155 residential spaces, demonstrating FloorNet's efficacy in improving reconstruction quality through both qualitative and quantitative evaluations.

Addressing the intricate challenge of generating 3D house models based on linguistic descriptions, Chen [99] introduces a House Plan Generative Model (HPGM) that uniquely divides the process into two sub-tasks: constructing layouts and synthesizing textures. To effectively tackle these tasks, two specialized modules, the Graph Conditioned Layout Prediction Network (GC-LPN) and the Language Conditioned Texture GAN (LCT-GAN), are proposed. These modules focus on generating floor plans and corresponding interior textures guided by provided descriptions. The generation of building layouts that fulfill the specified requirements is facilitated by the Graph Conditioned Layout Prediction Network (GC-LPN), which integrates adjacent information into the extracted features using a Graph Convolutional Network (GCN) [97], thereby enhancing the performance of layout generation.

The predominant focus of existing research revolves around addressing issues of compatibility and visual realism in the context of floorplan generation. However, both [98] and [99] encounter challenges when attempting to generate diverse floorplans within a single sampling, thereby compromising diversity. Furthermore, the mechanisms by which these models efficiently conduct sampling remain largely unaddressed. Another significant gap in the literature pertains to the issue of functional realism, a dimension that most papers fail to adequately consider or explore in their floorplan generation methodologies.

Another study by Hu et al. [100] introduces an automated approach to floorplan generation that fuses deep neural networks with user-guided design. Their framework, Graph2Plan, employs a layout graph and user-defined constraints to produce floorplans that adhere to layout

and boundary requirements. By allowing users to input room counts and constraints, the system retrieves floorplans from a database and uses Graph2Plan to convert layout graphs into refined room representations. The neural network, trained on a sizable annotated dataset, employs graph neural networks and conventional image convolution to process layout graphs, building boundaries, and raster floorplan images. The method's versatility and quality are demonstrated through its ability to accommodate diverse user inputs.

Table 2.1: Deep generative Model based floorplan Generation.

Methods	Name	Diversity	Compatibility	Visual Realism	Functional Realism	Sampling Time
GAN	Nauata et al. [6]	✓	✓	✓	-	-
	Nauata et al [8]	✓	✓	✓	-	-
	Tang et al. [10]	✓	✓	✓	-	-
	Chen et al. [101]	-	✓	-	-	
	Schiller et al. [102]	-	✓	-	-	
	Zheng et al. [?]	✓	✓	-	-	-
	Lim et al. [79]	✓	✓	-	-	-
	Chaillou et al. [80]	-	✓	-	-	
	Wang et al. [81]	-	✓	-	✓	-
	Liu et al. [103]	-	✓	✓	-	-
Diffusion Model	Shabani et al. [7]	✓	✓	✓	-	-
Autoregressive	Liu et al. [9]	✓	✓	✓	✓	-
Others	Liu et al. [98]	-	✓	✓	-	-
	Hu et al. [100]	✓	✓	✓	-	-
	Chen et al. [99]	-	✓	✓	-	-
	Wu et al. [104]	-	✓	✓	-	-

The summary table in Table 2.1 provides an overview of floorplan generation using various generative models. It outlines the different desirable properties achieved through this approach. Also, the "✓" symbol signifies that the papers have addressed the specified constraint, while the "-" symbol indicates that the constraint was not taken into consideration by those papers.

- **Method:** represents a specific approach or Model used to create a new floorplan.
- **Name:** highlights the paper name or used method with reference.
- **Diversity:** refers to the count of unique floorplans generated during a single sampling instance. It encompasses a range of distinct designs that adhere to specified constraints and layout arrangements.

- **Compatibility :** The ability of generated floorplans to seamlessly fit and adhere to the given design boundaries and spatial requirements.
- **Visual Realism:** The extent of similarity between generated designs and actual architectural layouts.
- **Functional Realism:** The extent to which generated floorplans accurately represent functional and logical relationships between rooms and spaces, ensuring practicality and usability.
- **Sampling Efficiency:** The effectiveness of the floorplan generation process in exploring diverse design possibilities while using minimal computational resources and iterations.

2.3 Graph Neural Network

In recent years, Convolutional Neural Networks (CNNs) [13] have made remarkable strides in deep learning, particularly in tasks related to computer vision [105, 106, 107]. With their impressive effectiveness, CNNs excel at extracting features from grid-like data, such as images, leading to significant advancements in the field. However, applying CNNs to structured data, such as tabular data [108] or structured grids [6, 7], poses distinct and significant challenges. This is especially evident in the context of floorplan data [8], which encompasses elements like walls, doors, and rooms, each entailing intricate spatial relationships that may not seamlessly align with CNNs' grid-based approach. As a result, tasks such as room classification or furniture layout optimization are affected. The variable dimensions of floorplans pose further obstacles, given CNNs' requirement for fixed-size inputs, potentially leading to information loss or distortion. Additionally, the intricate dependencies among floorplan elements, such as room adjacency or hierarchical structures, pose challenges for standard CNN architectures [13], which are primarily designed for local pattern recognition.

The rise of Graph Neural Networks (GNNs) [51] tackles this challenge by explicitly modeling graph structures and using message-passing techniques to capture complex dependencies. GNNs are specifically designed to operate on graph-structured data, enabling effective learning and inference tasks in graph domains. A graph [10] itself is a mathematical representation composed of nodes and edges, providing a flexible framework to represent complex relationships and interactions between elements. GNNs [51] leverage this structure to extract meaningful

representations and perform computations that take into account the inherent connectivity and dependencies within the graph. Graph Neural Networks (GNNs) have exhibited exceptional performance across a broad spectrum of tasks that inherently involve graph-based data. These tasks span social network analysis, Traffic State Predictions [109, 110], recommendation systems, object detection [111, 112], text classification [113, 114], and molecular chemistry [115], all of which inherently rely on graph-based data structures.

Thanks to their capacity to capture geometric and structured information, graph neural networks, particularly models like Convolutional Message Passing Networks (CMPN) [116], have found practical application in floorplan generation. The process of floorplan generation [6, 8] entails orchestrating the layout and arrangement of rooms, walls, and other architectural components within a building. By representing the floorplan as a graph, with rooms as nodes and connections as edges, GNNs can effectively leverage their spatial reasoning capabilities to produce optimal floorplans. Within the realm of GNNs, a diverse array of models exists, including Graph Convolutional Networks (GCNs) [97], Graph Attention Networks (GATs) [117], and Convolutional Message Passing Networks (CMPNs) [116]. Each variant offers distinct characteristics tailored to specific graph-based tasks, thereby enriching the versatility and applicability of GNNs across various domains.

2.3.1 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs) [97] are a class of neural networks designed to operate on graph-structured data. They extend the concepts of convolutional neural networks (CNNs) [13] to non-Euclidean domains [118], enabling the processing of data represented as graphs. GCNs have gained significant attention due to their effectiveness in various tasks, including social network analysis [119, 120], molecular graphs [121], recommendation systems [122]. and link prediction [123].

At its core, a GCN operates by performing message passing between neighboring nodes in a graph. This message passing mechanism allows each node to aggregate information from its neighbors, incorporating local graph structure into node representations. In a typical GCN architecture, the operation proceeds through multiple layers, with each layer refining node representations based on aggregated information from neighboring nodes.

GCN utilizes fully connected encodings and is capable of capturing both local and limited

global information from graph-structured data. It demonstrates high scalability and computational efficiency, making it suitable for analyzing large-scale graphs. However, when it comes to handling three-dimensional (3D) data, GCN's capabilities are somewhat limited. Despite this constraint, GCN remains a powerful tool for graph analysis tasks, offering a balance between capturing local and global information while maintaining efficiency and scalability.

2.3.2 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) [117] are a class of neural networks designed to operate on graph-structured data. They leverage attention mechanisms [40] to dynamically weight the contributions of neighboring nodes during message passing, allowing for flexible and adaptive information aggregation. Its flexibility and effectiveness make GAT a versatile tool applicable in diverse domains, including recommendation systems [124], bioinformatics [125, 126], and natural language processing [127].

At its core, a GAT [117] operates by assigning attention coefficients to neighboring nodes, indicating their importance in contributing to the representation of the central node. These attention coefficients are learned during training and can vary for each node and each layer, allowing GATs to capture complex relationships within the graph. In a typical GAT architecture, the operation proceeds through multiple layers, with each layer refining node representations based on attention-weighted aggregations of neighboring nodes.

GAT utilizes fully connected encodings and is capable of capturing both local and global information from graph-structured data. It employs attention mechanisms to assign varying importance to different neighbors during information aggregation, allowing it to focus on relevant nodes and edges. This attention mechanism enhances the network's ability to handle complex relationships within the graph. Although GAT demonstrates moderate scalability and computational efficiency, it offers superior performance in capturing global information compared to the Graph Convolutional Network (GCN). Furthermore, GAT, like GCN, has limited capabilities in handling three-dimensional (3D) data. Overall, GAT is a powerful GNN model that leverages attention mechanisms to effectively capture both local and global information, making it particularly well-suited for tasks that require considering the importance of different graph elements.

2.3.3 Convolutional Message Passing Networks (CMPNs)

Convolutional Message Passing Networks (CMPNs) [116] are a class of neural networks designed to operate on graph-structured data. They combine the principles of convolutional operations and message passing mechanisms to effectively process information on graphs. At its core, a CMPN operates by performing message passing between nodes in a graph, guided by a convolutional mechanism akin to convolutional neural networks (CNNs) on grid-like data. This allows CMPNs to extract hierarchical features from the graph structure, capturing both local and global graph properties.

As shown in equation 2.4, the feature vectors associated with nodes are spread across a volume. To avoid collisions and ensure that all information is retained in the message, a simple pooling operation is applied, aggregating the features from all neighboring nodes. To update a feature vector, the pooled features are then passed through a CNN. The CNN takes the concatenated input of the current feature vector and the pooled features from neighboring nodes.

$$f_v \leftarrow \mathrm{CNN}\left(f_v; \mathrm{Pool}_{w \in N(v)} f_w\right) \tag{2.4}$$

The resulting updated feature vector is denoted as fv and is obtained by applying the CNN operation to the concatenation of fv and the pooled features from the neighboring nodes. By using this pooling and CNN combination, the Conv-MPN approach achieves feature updates.

During each iteration, CMPN updates the node features by aggregating messages from neighboring nodes [128]. This aggregation process allows nodes to incorporate information from their local neighborhoods. The messages are computed based on the features of the sender node and the relationship between the sender and receiver nodes. This relationship is typically defined by the edge weights or adjacency matrix of the graph. After aggregating the messages, CMPN applies a learnable transformation to update the node features. This transformation can be a simple linear operation or a more complex neural network layer. By iteratively repeating the message passing and feature updating steps, CMPN allows the nodes to exchange information and refine their features based on the graph structure and the information carried by the messages. This iterative process enables CMPN to capture and propagate information throughout the graph, capturing complex dependencies and patterns within the data.

Unlike the fully connected encodings used by GAT and GCN, CMP employs convolutional

operations for encoding the graph. This allows CMP to capture local information in a structured and hierarchical manner. Additionally, CMP has the capability to capture both local and global information, similar to GAT and GCN. Its computational efficiency and scalability are high, making it suitable for handling large-scale graphs. Notably, CMP stands out in its ability to handle three-dimensional (3D) data effectively, surpassing the limited capabilities of both GAT and GCN in this regard. By combining convolution, merging, and pooling operations, CMP offers a comprehensive approach to graph analysis, enabling robust encoding, aggregation, and pooling of information from the graph structure.

Table 2.2 presents a comparative analysis of three prominent graph-based models: Graph Convolutional Network (GCN) [97], Graph Attention Network (GAT) [117], and Convolutional Message Passing (CMPN) [116], highlighting their respective strengths across various desirable properties.

Table 2.2: Comparison of Graph Neural Networks based on various properties.

Property	**GAT**	**GCN**	**CMPN**
Encodings	Fully connected	Fully connected	Convolve
Capture Local Information	Yes	Yes	Yes
Capture Global Information	Yes	Limited	Yes
Scalability	Moderate	High	High
Computational Efficiency	Moderate	High	High
Handling 3D data	Limited	Limited	High

2.4 Regularization techniques

Regularization techniques are strategies employed during the training of neural networks to prevent overfitting and improve generalization performance [129]. These techniques typically entail the addition of extra terms to the loss function or adjustments to the network architecture to impose constraints on the model's parameters. They aid in combating overfitting [130], enhancing generalization [129], and refining the quality of generated images in models like GANs [39, 131, 132]. For generative models, regularization contributes to improving training stability [132], preventing mode collapse [39], and augmenting the quality and diversity of generated samples by imposing constraints, fostering diversity, and facilitating smoother optimization dynamics during training. Several regularization techniques commonly utilized

for generative models include loss functions, spectral normalization, batch normalization, and minibatch discrimination.

2.4.1 Loss function

The goal of training a machine learning model is to minimize the loss function [133], leading to improved performance and accuracy in a given task. A loss function is a mathematical measure that quantifies the difference between the predicted output of a model and the true output, guiding the learning process by providing feedback on the error [134]. The choice of a loss function depends on the problem's nature and the model type, where classification tasks require specific loss functions like cross-entropy [133], hinge [135], or softmax loss [136], while regression tasks benefit from mean squared error (MSE)[137] or mean absolute error (MAE) [138] as suitable loss functions.

In the realm of generative modeling, where the goal is to produce realistic data samples resembling those from a given distribution, various loss functions have emerged to guide the training process and enhance model performance. One of the pioneering approaches, the minimax loss [23], laid the groundwork for subsequent advancements in the field. Initially introduced within the framework of Generative Adversarial Networks (GANs) [23], the minimax loss aims to optimize a game between two neural networks: the generator G and the discriminator D. Mathematically, the minimax loss can be expressed as:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))] \tag{2.5}$$

Here, x represents real data samples drawn from the true data distribution p_{data}, and z represents random noise drawn from a prior distribution p_z. The objective is straightforward yet profound—the generator seeks to minimize this loss function, while the discriminator aims to maximize it. Through this adversarial training process, the generator learns to produce samples that are increasingly indistinguishable from real data, while the discriminator becomes adept at distinguishing between real and generated samples.

However, despite its conceptual elegance, the original minimax loss formulation suffered from several limitations. Training instability and mode collapse, where the generator produces limited or repetitive samples, were common challenges encountered during optimization. To address these issues, Wasserstein Generative Adversarial Networks (WGANs) [139] introduced

a novel loss function based on the Wasserstein distance,[140] also known as the Earth Mover's distance. Unlike traditional GANs [23], which measure divergence between probability distributions using metrics like the Jensen-Shannon divergence [141] or Kullback-Leibler divergence , WGANs utilize the Wasserstein distance [140] for more stable and meaningful training. The Wasserstein distance measures the "distance" between two probability distributions by calculating the minimum amount of "work" required to transform one distribution into another. In the context of WGAN, this means minimizing the discrepancy between the distribution of real data and the distribution of generated data.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim P_z}[D(G(z))] \tag{2.6}$$

The equation 2.6 defining the Wasserstein Generative Adversarial Network (WGAN) loss encapsulates the fundamental principle behind WGAN's approach to training generative models. At its core, the equation seeks to minimize the discrepancy between the expected discriminator outputs for real and generated samples. The first term, $\mathbb{E}_{x\sim p_{\text{data}}}[D(x)]$, represents the expected output of the discriminator when presented with real data samples drawn from the true data distribution p_{data}. The second term, $\mathbb{E}_{z\sim p_z}[D(G(z))]$, represents the expected output of the discriminator when presented with generated samples produced by the generator from random noise z drawn from a prior distribution p_z. By optimizing the difference between these two expectations, the WGAN loss encourages the generator to produce samples that closely match the distribution of real data.

While WGANs [139] represented a significant step forward in generative modeling, the Wasserstein GAN with Gradient Penalty (WGAN-GP) [132] further refined the training process by addressing a key limitation of WGANs—the need for weight clipping in the discriminator. WGAN-GP introduced a gradient penalty [142] term to the loss function, penalizing the discriminator for having gradients with a norm different from unity. By penalizing the norm of the discriminator's gradient with respect to interpolated points between real and generated samples, the Gradient Penalty encourages smoother training dynamics and mitigates issues like mode collapse. This technique is often used in combination with the WGAN loss, resulting in the Wasserstein Gradient Penalty (WGAN-GP) loss as shown in equation 2.7.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{\text{data}}}[D(x)] - \mathbb{E}z \sim p_z(z)[D(G(z))] + \lambda \mathbb{E}\hat{x} \tag{2.7}$$

Another notable approach in generative modeling is the hinge loss [135], which has found application in both discriminative and generative models. Unlike traditional loss functions such as cross-entropy [133], hinge loss focuses on maximizing the margin between classes, making it particularly well-suited for tasks involving binary classification or margin-based objectives. In the context of generative modeling, hinge loss has been utilized in models like Adversarial Autoencoders (AAEs), where it serves as a discriminator loss function. By maximizing the margin between real and generated samples, hinge loss encourages the discriminator to distinguish more effectively between the two, thereby enhancing the quality of generated samples.

The hinge loss function can be defined as:

$$\text{Hinge Loss} = \mathbb{E}_{x\sim p_{\text{data}}}[\max(0, 1 - D(x))] + \mathbb{E}_{z\sim p_z}[\max(0, 1 + D(G(z)))] \tag{2.8}$$

where $D(x)$ represents the discriminator's output for real samples, and $D(G(z))$ represents the discriminator's output for generated samples.

Overall, the choice of loss functions in GANs significantly influences training dynamics and model performance. While the original MinMax loss [23] fosters adversarial competition, it often leads to instability and mode collapse. WGAN loss [139], employing Wasserstein distance, enhances stability, further WGANGP [132] reinforced by the Gradient Penalty technique. Selecting the appropriate loss function hinges on specific application objectives, with options like LSGAN for sharper samples and divergence-based losses for distribution alignment. Careful consideration of gradient behavior and convergence properties is essential to balance stability and sample quality, crucial for achieving stable training and generating high-quality and diverse samples.

2.4.2 Minibatch Discriminator

The concept of mini-batch discrimination [42] was conceived to address certain limitations in conventional GAN [23] architectures, particularly in scenarios where the discriminator struggles to effectively differentiate between real and generated images. In standard GAN [23] setups, the discriminator operates on individual images, making its judgments based solely on each image's

features. However, this approach can sometimes lead to issues such as mode collapse, where the generator produces limited varieties of outputs, or unstable training dynamics.

Mini-batch discrimination [42] operates by introducing a mechanism for the discriminator to consider information from multiple images within a mini-batch, rather than evaluating each image in isolation. By aggregating information across the mini-batch, the discriminator gains a broader context for making its judgments, allowing it to discern more nuanced differences between real and generated images. This can lead to more stable training dynamics and encourage the generator to produce a wider variety of outputs.

Minibatch discrimination has been applied in various contexts within the realm of generative modeling, yielding notable improvements in sample quality and training stability. In their paper, Salimans et al. [143] introduced minibatch discrimination as a technique to improve the performance of GANs. By incorporating minibatch discrimination into the discriminator architecture, they demonstrated enhanced sample diversity and reduced mode collapse. The motivation behind their approach was to address the limitations of traditional GAN training, particularly regarding sample quality and diversity. Through experiments on various datasets, they showcased the effectiveness of minibatch discrimination in producing high-quality, diverse samples.

Zhang et al. [144] proposed Self-Attention Generative Adversarial Networks (SAGANs), which utilize self-attention mechanisms to capture long-range dependencies in images. In their work, they incorporated minibatch discrimination as part of the self-attention mechanism, enabling the discriminator to capture global statistics across the minibatch. By doing so, they achieved significant improvements in sample quality and realism. Their motivation stemmed from the need to address the limitations of traditional GAN architectures in capturing long-range dependencies and producing coherent samples.

Karras et al. [37] introduced Progressive Growing of GANs (PGGANs), a method for training GANs in a progressive manner by incrementally increasing the resolution of generated images. Minibatch discrimination played a crucial role in their approach by facilitating the training of high-resolution images. They utilized minibatch discrimination to improve sample diversity and stability throughout the training process. Their motivation was to address the challenges of training GANs on high-resolution images, which often suffer from mode collapse and training instability.

2.4.3 Spectral normalization

Spectral normalization [41] is a technique that plays a crucial role in stabilizing the training process and maintaining the diversity of image generation. By normalizing the spectral norm of weight matrices in the generator network, it addresses issues like vanishing or exploding gradients, which can lead to mode collapse and limited image diversity. Spectral normalization ensures stable training by constraining the Lipschitz constant [145] of each layer, preventing signal magnitude amplification. This stability allows the generator to explore a wider range of the latent space, capturing diverse patterns and generating a richer variety of images, thereby enhancing the diversity of the generated outputs.

In their paper, Miyato et al. [77] introduced spectral normalization as a technique to stabilize the training of GANs. They demonstrated that spectral normalization applied to the discriminator significantly improves training stability and mitigates mode collapse. Their motivation stemmed from the need to address the challenges of training GANs on complex datasets, where traditional optimization techniques often struggle to converge. Through experiments on various benchmark datasets, they showcased the effectiveness of spectral normalization in producing high-quality, diverse samples.

Zhang et al. [146] proposed a novel architecture called BigGAN, which achieves state-of-the-art performance in terms of sample quality and diversity. In their work, they incorporated spectral normalization into both the generator and discriminator networks of BigGAN. By doing so, they were able to stabilize the training process and produce high-resolution images with remarkable fidelity and diversity. Their motivation was to push the boundaries of generative modeling and demonstrate the effectiveness of spectral normalization in handling large-scale GANs trained on high-resolution datasets.

Karras et al. [147] introduced StyleGAN, a groundbreaking architecture for high-fidelity image generation with fine-grained control over image attributes. In their work, they employed spectral normalization as a regularization technique to improve the stability of StyleGAN training. By incorporating spectral normalization into the discriminator network, they achieved smoother optimization dynamics and better convergence properties. Their motivation was to enable the training of large-scale GANs capable of producing high-resolution images with unparalleled realism and diversity.

In conclusion, the integration of spectral normalization in GAN architectures, as demon-

strated by Miyato et al. [77], Zhang et al. [146], and Karras et al. [147], has significantly enhanced training stability and sample quality. These seminal papers showcase the effectiveness of spectral normalization in mitigating challenges such as mode collapse and vanishing gradients, paving the way for the development of high-fidelity and diverse generative models. Moving forward, spectral normalization remains a key technique in advancing the capabilities of GANs for various applications in image generation and beyond.

2.5 Architectural Modifications

By expanding the model's capacity to capture diverse patterns and features, architectural modifications enable it to generate outputs that exhibit greater variability and richness [148]. These modifications often entail intricate adjustments to the model's architecture, such as incorporating additional layers [147], introducing novel connectivity patterns [148], or leveraging attention mechanisms [39, 144] to capture intricate dependencies within the data.

Progressive Growing of GANs (ProGAN) [37] is a significant advancement in generative adversarial networks (GANs) that tackles the challenge of generating high-resolution and diverse images. ProGAN introduces a progressive training scheme where the generator and discriminator are gradually grown, starting from low-resolution images and progressively increasing the resolution during training. This technique allows for more stable training by initially learning coarse-level features and then refining them progressively. ProGAN also employs minibatch standard deviation [42], equalized learning rate, and pixel-wise feature normalization [149] to enhance training stability and improve the quality of generated images. The key contribution of ProGAN is its ability to generate high-resolution images with fine details, surpassing the limitations of previous GAN architectures. However, ProGAN's training can be computationally expensive and requires substantial computational resources. Additionally, achieving optimal stability and quality in training ProGAN can still be challenging and may require careful hyperparameter tuning and architectural adjustments.

StyleGAN [147] is a groundbreaking generative model that has made significant contributions to the field of image synthesis. It introduces several key techniques to enhance the diversity, control, and realism of generated images. StyleGAN employs a two-component architecture consisting of a mapping network and a synthesis network, allowing for fine-grained control over various image attributes. It utilizes progressive growing [150] to train the model

progressively from low to high resolutions, ensuring stable training. Additionally, it incorporates noise injection [151], style mixing regularization, adaptive instance normalization [152], and stochastic variation to encourage diversity and prevent mode collapse. The contributions of StyleGAN include improved image quality, enhanced control over image synthesis, and the ability to generate highly realistic and diverse images. However, it also has some gaps, such as limited scalability for high-resolution images and high computational and memory requirements, which can pose challenges for practical applications and training on large-scale datasets.

StyleGAN2 [153] builds upon the success of StyleGAN [147] by introducing improved architecture and training techniques. It incorporates path length regularization, which encourages smooth and continuous changes in the latent space, leading to improved convergence during training. StyleGAN2 also features a generator architecture with skip connections that preserves fine details and textures, resulting in higher-quality generated images. Furthermore, it benefits from a larger and more diverse training dataset called "FFHQ," consisting of high-resolution human face images, which enhances generalization and image quality. While StyleGAN2 achieves impressive results, it still has some limitations, such as high computational and memory requirements, making it resource-intensive and time-consuming to train. Additionally, controlling specific attributes or interpretability of the underlying latent space remains challenging. Researchers continue to explore solutions to address these gaps and further refine the capabilities of StyleGAN2 and its applications in the field of generative models.

BigGAN [146] is a state-of-the-art generative model known for its impressive scalability and high-quality image synthesis. It introduces several key techniques and innovations to address the challenges of generating high-resolution images. BigGAN utilizes a deep convolutional architecture [13] with a conditional GAN [76] setup, allowing for control over the generated images based on class labels. It incorporates techniques like self-attention mechanisms [40] to capture global dependencies and spectral normalization [155] to stabilize training. BigGAN also introduces truncation trick, which controls the trade-off between image quality and diversity. The contributions of BigGAN include the generation of highly realistic and diverse images at high resolutions, surpassing the capabilities of previous models. However, BigGAN has some gaps, such as controlling specific image attributes beyond class labels remains a challenge, limiting fine-grained control. Despite these gaps, BigGAN represents a significant advancement in generative models, showcasing the potential for generating high-quality images with improved

Table 2.3: Summary of variat GAN architectures and their proporty.

Variants	Architectural novelity	Training Stability	Mode Collapse	Gaps
ProGAN [37]	Progressive growing with Equalized Learning Rate, Minibath discrimination	Improved stability by gradual network growth & Equalized Learning Rate	Reduced mode collapse through gradual learning & mini bath Discrimination	Difficulty in training on large-scale datasets
StyleGAN [147]	Progressive growing with noise injection	Improved stability by Noise Injection	Reduced mode collapse with Progressively growth	Limited scalability for high-resolution images
StyleGAN2 [153]	Skip connection & Mapping Networks	Improved stability via Path length regularization	Reduced mode collapse through skip connection	High computational and memory requirements.
BigGAN [146]	Self-Attention Mechanisms with spectral normalization	Can suffer from instability at larger scales	Reduced mode collapse with class-conditional training	Limited flexibility in controlling specific image attributes
CycleGAN [154]	Dual Generator Discriminator with Cycle consistency loss	Generally stable training by identity mapping loss	Reduced with Cycle loss & duality	Requirement for paired data from both domains,
MS-GAN [148]	Multi-scale discriminator architecture	Improved stability with multi-scale analysis	Reduced mode collapse with multi-scale discrimination	Complex training process and high memory requirements
MSSA-GAN [39]	Multi-scale self-attention mechanism	Improved stability and feature capture	Reduced mode collapse with attention-based modeling	Problems in Interpretability and Control

scalability and control.

CycleGAN [154] is designed for image-to-image translation tasks, where the goal is to learn mappings between two domains without paired training data. It introduces a cycle consistency loss, which enforces that the translated images can be successfully reversed back to the original domain. This regularization promotes stable training and improves the preservation of important content during translation. However, in complex domains, CycleGAN can still exhibit mode collapse, where it fails to capture the full diversity of the target distribution. Despite this limitation, CycleGAN produces good sample quality for image-to-image translation tasks and offers faster sampling times compared to some other GAN variants.

Multi-Scale Gradients GAN (MSG-GAN) [148] introduces a hierarchical architecture that operates at multiple resolutions simultaneously, enabling the generation of diverse images with fine-grained details. MSG-GAN introduces a multi-scale gradient blending technique, where gradient from multiple resolutions are combined to enhance the generation process. This technique allows the model to capture both fine-grained details and global coherence in the generated images. Additionally, MSG-GAN incorporates spectral normalization [155] to stabilize the training process and reduce mode collapse. The major contribution of MSG-GAN is its ability to generate highly realistic and visually appealing images with improved texture quality and fine details. The multi-scale discrimination also helps reduce mode collapse by encouraging the generator to generate samples that are consistent across different resolutions. As a result, MS-GAN generates high-quality samples with multi-scale details, albeit with slower sampling times due to the increased computational complexity. However, a potential gap of MSG-GAN lies in the increased computational complexity due to the multi-scale blending technique, which may require significant computational resources and longer training times.

The Self-Attention Guided Multi-Scale Gradient GAN (SAG-MSG-GAN)[39] represents a groundbreaking advancement in the realm of generative modeling, offering a novel approach to enhancing both diversity and realism in generated images. This innovative architecture combines the strengths of multi-scale gradient networks [148] and self-attention mechanisms [40] to achieve unprecedented levels of quality and variety in synthesized samples. SAG-MSG-GAN operates across multiple resolutions simultaneously, allowing it to capture intricate details and features at different scales. Moreover, the integration of self-attention mechanisms enables the model to dynamically focus on relevant regions of the input space, thereby enhancing the diversity of

generated samples while maintaining realism. Consequently, SAG-MSG-GAN produces high-quality samples with attention-driven synthesis, albeit with slower sampling times due to the additional computational overhead.

2.6 Computational techniques to reduce Sampling time

The efficiency of sampling time has a profound impact on the usability and real-time capabilities of generative models, directly influencing interactive experiences and adaptability to changing conditions [156]. Faster sampling times are essential for dynamic responses to user input and large-scale data generation, particularly in applications like deep learning model training. The understanding and improvement of sampling time are crucial for successful deployment in real-world scenarios. Several factors, such as model complexity, dataset size, hardware infrastructure, and application requirements, contribute to sampling time and involve trade-offs [157]. Complex models may offer higher quality [49] but require more computation, while larger datasets necessitate more time for sampling. Hardware capabilities influence computation speed, with faster hardware enabling quicker sampling, although potentially at a higher cost.

Several techniques have been suggested to improve the sampling time of generative models, including Knowledge distillation [45], Stochastic differential equations [47], Adaptive noise schedule [46], and Non-markovian process [158]. Each of these approaches plays a vital role in optimizing system efficiency, with the shared objective of enhancing the overall efficiency of generative models. These methods, either directly or indirectly, contribute to improving sampling time and overall performance, aligning with the pursuit of efficiency in generative models.

2.6.1 Knowledge Distilation

Knowledge distillation is a technique used in generative modeling to compress a large teacher model into a smaller student model while transferring the valuable information embedded in the teacher's output distribution [45, 159]. By minimizing the cross-entropy between the student's output distribution and the teacher's output distribution, the student model can achieve better performance than if trained independently. This approach is particularly useful in supervised learning tasks where the teacher's output captures additional information not present in simple one-hot labels. For instance, in image classification tasks with multiple plausible categories, the teacher's output distribution reflects ambiguity, while a one-hot label provides only a single

category.

One key requirement for successful knowledge distillation is that the function being learned must be deterministic. In most supervised learning scenarios, this condition is straightforward to satisfy since models produce the same output given the same input. However, in generative modeling, this condition becomes non-trivial. Stochastic models that employ MCMC [160] techniques like Langevin dynamics, such as score-based and energy-based models, cannot be used directly as teacher models due to their inherent stochasticity. This limitation hinders their applicability in knowledge distillation.

Nevertheless, a recent advancement in generative modeling called denoising diffusion implicit models (DDIMs) [161] shows promise for knowledge distillation. DDIMs are a type of generative model that produces high-quality samples and offers faster sampling speed compared to other iterative generative models. Notably, the generative process of DDIMs is deterministic, making them suitable candidates for knowledge distillation. Their deterministic nature enables the transfer of knowledge from DDIMs to student models, facilitating the compression of the larger teacher model into a smaller and faster student model. Denoising diffusion implicit models (DDIMs) [161] are implicit probabilistic models that transform a latent variable using a deterministic function to model data. However, evaluating this function can be computationally expensive, requiring multiple forward passes through a neural network. To overcome this computational burden, the paper [45] proposes distilling the knowledge from the expensive "teacher" function into a faster "student" network. In this approach, the student network approximates the teacher's output distribution by utilizing only a single network evaluation and the latent variable.

The paper by Luhman et al. [45] introduces a novel technique for applying knowledge distillation in iterative generative models, consolidating a multi-step denoising process into a single step to enhance sampling efficiency. This streamlined method significantly boosts sampling speed without requiring adversarial training [23], as demonstrated by the Denoising Student model's ability to generate high-quality samples comparable to those produced by GANs [23] and VAE [26] on datasets like CIFAR-10 and CelebA. This approach allows for the compression of knowledge into a more efficient model, enabling the student network to mimic the behavior of the teacher network while significantly reducing computational costs. The usage of knowledge distillation in DDIMs provides a practical and efficient solution for applications

where computational resources are limited or real-time performance is essential. However, there is a concern regarding the initialization of the student network. Since the student network is trained to model the data, initializing it with a teacher network that models the noise may not be ideal. Secondly, both the teacher network and the student network are conditioned on time, specifically at timestep T, which corresponds to the highest noise level. These challenges raise important considerations regarding the initialization and conditioning of the networks, affecting their performance and ability to accurately model the underlying data distribution.

As a whole, performing knowledge distillation in image generative models can pose several challenges. One problem lies in preserving the fine-grained details and intricate structures of the generated images during the distillation process. The compressed "student" model may struggle to capture the full complexity of the "teacher" model's output distribution, leading to a loss of fidelity and the generation of less visually appealing images. Additionally, the choice of an appropriate loss function for distillation is critical. The commonly used cross-entropy loss may not fully capture the perceptual or structural similarities between the generated images. This can result in a mismatch between the teacher and student distributions, leading to sub-optimal knowledge transfer. Furthermore, the computational cost of training a large "teacher" model and distilling its knowledge into a smaller "student" model can be significant, requiring substantial resources and time. Balancing model size, efficiency, and performance becomes a crucial consideration in the knowledge distillation process for image generative models.

2.6.2 Adaptive Noise Schedule

In diffusion generative models, an adaptive noise schedule [46] refers to a dynamic strategy for controlling the level of noise added to the data at each step of the generative process. The diffusion model gradually refines an initial noise-corrupted image or signal into a realistic sample. The adaptive noise schedule adjusts the magnitude of noise added at each step based on the progression of the generation process. Initially, high levels of noise are added to the input, allowing for easy sampling from a simple distribution. As the model progresses, the noise level decreases, enabling finer details to be added to the generated sample. Adaptive noise schedules are crucial for balancing the trade-off between preserving details and maintaining sample diversity throughout the generation process, ultimately leading to the creation of high-quality and diverse samples.

The paper Noise Estimation for Generative Diffusion Models [162] proposes a novel approach to enhance sampling time in generative diffusion models. The method introduces a neural network capable of dynamically adjusting noise parameters step-by-step during the denoising process, thus optimizing synthesis results without requiring separate tuning for each step. Unlike conventional methods [49] that predefine the steps of the reverse process, this approach estimates noise levels in the data and schedules subsequent denoising steps accordingly. By adapting noise parameters dynamically, the model can efficiently adjust noise levels for any number of steps, resulting in faster and more effective sample generation. This not only simplifies training but also significantly improves synthesis results while keeping computational costs low, addressing the issue of slow inference and high computational requirements associated with generative diffusion models.

2.6.3 SDE solver

Stochastic Differential Equation (SDE) [47] solvers are mathematical algorithms that approximate the solutions of differential equations incorporating random noise. In diffusion probabilistic models [49], using advanced SDE solvers can reduce sampling time and improve the efficiency of the sampling process. These solvers, such as stochastic Runge-Kutta methods [163] or adaptive step-size algorithms, offer better numerical stability and precision compared to traditional solvers. By allowing larger time steps and reducing the number of iterative sampling steps, SDE solvers enable faster exploration of the probability distribution and generation of high-quality samples. Additionally, they enhance the modeling capabilities of diffusion models by accurately capturing long-range dependencies and complex dynamics. However, the choice of SDE solver involves a trade-off between computational efficiency and accuracy.

The paper Gotta Go Fast When Generating Data with Score-Based Models [48] tackles the persistent challenge of slow data generation in score-based generative models. These models rely on forward diffusion processes to transform data into noise and generate samples by reversing this process. However, the bottleneck often arises from the numerical solvers used to approximate the stochastic differential equations (SDEs) [47], necessitating numerous evaluations of the score network. To address this, the authors propose a novel SDE solver tailored specifically for score-based generative models. This solver, featuring adaptive step sizes, is meticulously designed to optimize performance and sample quality. Remarkably, it achieves high-quality

Table 2.4: Summary of computational techniques to reduce sampling time

Method	**Advantages**	**Disadvantages**
Knowledge Distillation	• Speeds up sampling by training a smaller, faster model to mimic the behavior of the original model. • Reduces the number of iterative sampling steps. • Can be used to compress large models.	• Requires an additional training step to distill knowledge from the original model. • Loss of some fine-grained details or complexity compared to the original model. • Performance depends on the quality of the distillation process.
Adaptive Noise Schedule	• Adjusts the noise schedule dynamically based on the properties of the target distribution. • Can improve sampling efficiency and convergence speed. • Reduces the number of steps needed to generate high-quality samples.	• Requires additional computational overhead to estimate the properties of the target distribution. • Complexity in determining the optimal adaptation strategy. • May introduce additional hyperparameters to tune.
SDE Solvers	• Enables better exploration and approximation of the target distribution. • Provide lower gradient variance, which can enhance the stability of learning processes.	• Complexity in choosing the appropriate solver for the specific modeling task. • Trade-off between accuracy and computational efficiency.

sample generation with only two score function evaluations, significantly reducing sample rejections and outperforming the commonly used Euler-Maruyama (EM) solver [164]. The proposed method demonstrates data generation speeds 2 to 10 times faster than the EM solver while maintaining or improving sample quality, particularly excelling in generating high-resolution images. Furthermore, the solver eliminates the need for manual step size tuning, simplifying the data generation process. In essence, by introducing this efficient SDE solver with adaptive step sizes tailored for score-based models, the authors make a significant contribution to accelerating data generation in score-based generative models, ensuring faster and higher-quality sample generation.

2.7 Metrics

The evaluation of deep learning models is crucial as it allows us to assess their performance, understand their limitations, and make informed decisions [165]. It is important because, without proper evaluation, we cannot determine the effectiveness of a model or its generalizability to new data and leads to incorrect predictions or decisions. When it comes to evaluating generative models, which are designed to generate new data samples, several techniques can be employed. One commonly used approach involves measuring the diversity of the generated samples using metrics such as the Inception Score or Frechet Inception Distance [166]. These metrics provide objective measures to evaluate the diversity of the generated samples. They allow us to quantify the quality of the generated data and compare different generative models based on their performance.

In addition to objective metrics, human evaluation can also be employed to assess the subjective quality of the generated samples. Human evaluators can provide valuable insights regarding the realism, coherence, and overall quality of the generated data. By incorporating human judgment, we can gain a more comprehensive understanding of the generative model's performance and its ability to produce samples that are indistinguishable from real data.

2.7.1 FID score

The Fréchet Inception Distance (FID) [166] score is a widely used metric for evaluating the diversity of generated images in generative models such as Generative Adversarial Networks (GANs). It measures the similarity between the distributions of real images and generated

images by comparing their feature representations.

To calculate the FID score, the feature vectors of a set of real images and a set of generated images are extracted using a pre-trained Inception model [167]. Let's denote the mean and covariance of the feature vectors for the real images as μ_{real} and Σ_{real}, respectively. Similarly, the mean and covariance of the feature vectors for the generated images are denoted as μ_{gen} and Σ_{gen}, respectively.

The FID score is derived from the Fréchet distance [168], which measures the distance between two multivariate Gaussian distributions. It is computed using the following formula:

$$\text{FID}(\mu_{\text{real}}, \Sigma_{\text{real}}, \mu_{\text{gen}}, \Sigma_{\text{gen}}) = \|\mu_{\text{real}} - \mu_{\text{gen}}\|^2 + \text{Tr}(\Sigma_{\text{real}} + \Sigma_{\text{gen}} - 2(\Sigma_{\text{real}} \cdot \Sigma_{\text{gen}})^{1/2}) \tag{2.9}$$

Here, $\|\cdot\|$ denotes the Euclidean distance, and $\text{Tr}(\cdot)$ represents the trace of a matrix. The FID score is essentially the sum of the squared Euclidean distance between the means of the two distributions and a term that accounts for the covariance matrices. A lower FID score indicates a closer match between the distributions of real and generated image features, suggesting higher diversity and quality of the generated images.

2.7.2 Realism

Realism metrics in the context of generating floorplans refer to the evaluation criteria used to assess how realistic and plausible the generated floorplans appear [8]. These metrics aim to measure the extent to which the generated floorplans resemble real-world floorplans in terms of their layout, spatial organization, and architectural conventions. Realism metrics provide quantitative or qualitative measures of the fidelity and authenticity of the generated floorplans, allowing researchers and practitioners to assess the quality and realism of the generated results. The degree of realism in the generated house layouts is assessed through an evaluation process involving users with extensive knowledge of architecture.

In contrast to previous research [6, 7, 9, 10], which relied on ratings collected from a review group consisting of 12 graduate students and 10 professional architects to derive realism scores, and where reviewers assigned one of four ratings to each layout pair (better [+1], worse [-1], moderate [0]), our approach introduces a more objective evaluation method. By using a questionnaire-based assessment, we aim to minimize potential subjectivity and biases inherent in

human ratings. This shift towards objectivity provides a more reliable and consistent evaluation process for assessing the realism of layout designs. Instead of relying solely on subjective judgments, our approach employs predefined criteria to systematically evaluate layout pairs, enabling a more standardized and rigorous assessment process. By gathering insights from this structured evaluation, we can make more informed and objective assessments of layout quality, enhancing the reliability and robustness of our findings.

2.8 Related work

At the heart of generative models' effectiveness lies the optimization of two critical elements: diversity and efficiency in sample generation [33]. In recent years, researchers have delved into inventive approaches, algorithms, and architectural adjustments to tackle these challenges. By focusing on enhancing diversity, which ensures the generation of varied and realistic samples, and improving efficiency, which streamlines the generation process, these efforts aim to push the boundaries of generative model capabilities. Through innovative techniques and advancements in generative models, researchers strive to unlock new potentials, enabling applications across various domains. In the subsequent section, we will discuss related works that are directly relevant to the techniques used in our study.

HouseGAN [8] introduces a novel approach to graph-constrained generative adversarial networks (GANs) [23], aiming to incorporate constraints into the graph structure of its relational networks. This research represents a pioneering endeavor in employing deep learning generative models for such purposes. By leveraging the graph structure, HouseGAN [8] effectively encodes the desired constraints, resulting in floorplan generation with improved adherence to architectural guidelines. Furthermore, HouseGAN exhibits moderate diversity in the generated floorplans through the incorporation of a progressive growing mechanism [37]. Although it does not directly address the sampling time issue inherent in GANs due to their single-shot noise injection nature, HouseGAN achieves faster generation speeds compared to alternative approaches in the field. While it is at the forefront of generating floorplans using deep generative models, it still encounters challenges in accurately producing floorplans with doors and non-Manhattan layouts.

Building upon the foundation laid by HouseGAN [8], HouseGAN++ [6] introduces a powerful framework for floorplan generation. It combines a graph-constrained relational GAN

and a conditional GAN [76], with the integration of a non-iterative training process known as component-wise GT-conditioning to train the generator. Additionally, HouseGAN++ leverages meta-optimization techniques [169]. These techniques have a twofold effect on the generation process. Firstly, they enhance the diversity of the generated floorplans by allowing the model to capture and preserve high-level layout features while refining the details. This results in a wide range of visually distinct and architecturally viable floorplans. Secondly, the combination of the non-iterative training process and the intelligent use of meta-optimization techniques contributes to a fast sampling time. This enables efficient and rapid generation of floorplans, making HouseGAN++ [6] a compelling choice for real-time design exploration and iterative floorplan refinement. While HouseGAN++ demonstrates improved performance compared to its baseline, it may still encounter difficulties in generating complex floorplans and occasionally generates separate floorplans.

Another research on floorplan generation, HouseDiffusion [7] introduced by Shabani et al., presents an innovative approach to vector floorplan generation using a diffusion model. This model employs a two-fold inference objective aimed at denoising the 2D coordinates of room and door corners: treating noise as a continuous quantity for precise inversion of the continuous forward process and treating the final 2D coordinate as a discrete quantity to establish geometric incident relationships such as parallelism, orthogonality, and corner-sharing. These objectives aim to broaden the distribution and enhance the diversity of generated floorplans, albeit at the cost of increased generation time. Despite its proficiency in generating diverse floorplans, HouseDiffusion faces challenges with slow sampling times. This can be attributed to several factors, including the complexity of the diffusion model [24] involving multiple iterations and computations, as well as the computational demands of the Transformer architecture [89]. Furthermore, the graph-conditioned nature of the floorplan generation task and the need to incorporate constraints further contribute to the slower sampling time. Additionally, the combination of continuous and discrete denoising steps adds complexity and time required for sampling, collectively resulting in the observed slower sampling time in HouseDiffusion [7].

In line with the methodology proposed in Housediffusion [7], End-to-end Graph-constrained Vectorized Floorplan Generation [9] also endeavors to synthesize floorplans as sequences of 1-D vectors, thereby enhancing user interaction and enabling tailored design customization. To generate high fidelity vectorized floorplans, they propose a novel two-stage framework, includ-

ing a draft stage and a multi-round refining stage. In the first stage, they encode the room connectivity graph input by users with a graph convolutional network (GCN) [97], then apply an autoregressive transformer network [170] to generate an initial floorplan sequence. To polish the initial design and generate more visually appealing floorplans, they further propose a novel panoptic refinement network (PRN) [9] composed of a GCN [97] and a transformer network [170]. The PRN takes the initial generated sequence as input and refines the floorplan design while encouraging the correct room connectivity with our proposed geometric loss.

Another research that generates floorplans like houseGAN++ [6] is GTGAN [10]. GTGAN is a novel approach that aims to generate floorplans and roofs using the LIFULL HOME's dataset . It incorporates graph convolution networks and Transformers to capture local and global interactions for graph-constrained house generation. The key contributions of GTGAN include the introduction of connected node attention (CNA) and non-connected node attention (NNA) modules, which enable the modeling of relationships between nodes in the graph. The approach consists of a graph Transformer [171] based generator, a node classification-based discriminator, and a graph-based cycle-consistency loss. GTGAN [10] achieves significant performance improvements compared to existing methods, although it may have slower sampling times due to the combination of graph convolution networks and Transformers, along with the incorporation of graph node attention modules. Despite this, GTGAN prioritizes generating realistic and visually appealing results.

Table 2.5 provides a comprehensive summary of various approaches employed for floorplan generation utilizing different generative models. Each method is evaluated based on several comparable properties such as model type, employed techniques, and architectural considerations. The diversity and sampling column in the table visually represents the performance of these methods, with a green triangle (▲) indicating superior performance, blue triangle (▲) moderate, and a red triangle (▲) indicating inferior performance in the respective areas.

From the above table 2.5, it becomes evident that the majority of floorplan generative models have primarily focused on aspects such as realism and diversity, while largely neglecting the time required to generate floorplans. The emphasis has been on producing visually appealing and diverse floorplans without giving due consideration to the speed of the generation process. This oversight is significant because, in real-world applications, the time required to generate floorplans can be a critical factor, especially when dealing with large-scale or time-sensitive

Table 2.5: Comparative Analysis of Floorplan Generation Methods

Name	**Model**	**Architecture**	**Techniques**	**Diversity**	**Sampling time**
House GAN[8]	GAN	CMPN	Up/Down Sampling PostProcessing	▲	▲
House-GAN++[6]	GAN	CMPN	GT-conditioning Up/Down Sampling Meta-Optimization	▲	▲
HouseDiffusion[7]	DM	Transformer	Continues Denoising Vectorization Discrete Denoising	▲	▲
EndToEnd [9]	AUM	Transformer	GCN Vectorization Panoptic Refinement	▲	▲
GTGAN [10]	GAN	GCN+ViTs	cycle-consistency loss Node classification	▲	▲

projects. By disregarding the time aspect, these models may fall short in meeting the practical requirements and constraints of users who need quick and efficient floorplan generation. Therefore, our research aims to bridge this gap by developing models that not only prioritize realism and diversity but also take into account the time required to generate floorplans, ensuring a balance between quality and efficiency.

Chapter 3

Methodology

The objective of this work is to propose a method that balances the sampling time and diversity while maintaining the realism of the generated floorplan. This is accomplished through the integration of a GAN (Generative Adversarial Network) [23] and diffusion model [24], coupled with various regularization and training methodologies. The forthcoming subsection, referenced as Subsection 3.1, provides an elaboration on the research methodology adopted in this pursuit.

3.1 Research Metodology

To conduct our research, we have chosen to adopt the Design Science Research Process (DSRP) [172] as our methodology. DSRP is a systematic and iterative approach widely employed in information systems and computer science for developing practical artifacts. By embracing the DSRP methodology, we aim to effectively address our objective of resolving an existing problem through a novel approach. This systematic and rigorous approach ensures the structured development and evaluation of an innovative solution for the challenge of floorplan generation, specifically focusing on achieving a balance between sampling time and diversity. Adhering to the DSRP, we identify key problem areas, define clear research objectives, design and develop a robust floorplan generation technique, demonstrate and evaluate its effectiveness, and communicate our research findings. We followed the following steps throughout the research process which is also depicted in Figure 3.1.

Figure 3.1: Research Design

- **Problem identification:**

  To identify the research problem, we took a multifaceted approach, conducting an in-depth

exploration of generative models and the existing landscape of floorplan generation techniques. Our initial step involved extensive research to gather insights and knowledge on the subject matter. Subsequently, we undertook a comprehensive literature review, carefully examining studies, papers, and publications relevant to floorplan generation. This thorough analysis enabled us to identify common problems and challenges consistently mentioned across different sources. Equipped with this information, we meticulously analyzed and scrutinized the identified problems, seeking to understand their underlying causes and implications. Through this critical evaluation, we were able to carefully select and define the research problem, which served as the core focus of our investigation as outlined in section 1.2. This rigorous process ensured that our research efforts were directed towards addressing a significant and relevant issue within the realm of floorplan generation, establishing a solid foundation for our subsequent research endeavors.

- **Objective Formulation:**

  After identifying the research problem, we proceeded to formulate a research question, as outlined in section 1.3. To effectively address these research questions, we formulated clear aims and objectives for our study. We initially established a general objective in Subsection 1.4, which provided a comprehensive direction and purpose for our project, serving as the guiding principle for our research. Subsequently, we defined specific objectives that allowed us to break down the general objective into smaller, more manageable components. These specific objectives acted as actionable steps, providing a focused and structured approach to our research endeavors. By establishing these specific objectives, we created a well-defined roadmap that facilitated our progress and ensured we stayed on track throughout the research process.

- **Design and development:** During this pivotal step, we introduce a groundbreaking approach that harnesses the combined capabilities of Generative Adversarial Networks (GANs) and diffusion models to address the delicate balance between sampling time and diversity. Our method capitalizes on the strengths of each model, leveraging GANs' ability for efficient sampling and diffusion models' capacity to generate diverse samples. To further enhance our design and development process, we delve into exploring various techniques, including regularization techniques and architectural modifications. These

explorations aim to refine our approach, ensuring its effectiveness in achieving our objectives.

- **Experimentation:** Through systematic experimentation, we rigorously assess the impact of different regularization techniques on model performance, aiming to mitigate overfitting and enhance generalization capabilities. Additionally, we explore the potential synergies of hybrid models, combining the strengths of multiple approaches to achieve superior performance in floorplan generation tasks, as shown in chapter 4 ablation studies. By conducting experiments on these parameters, we strive to uncover insights that advance the field and contribute to the development of more effective and robust generative models for floorplan generation.

- **Evaluation:**

  In this step, we evaluate the performance of our solution based on predefined metrics and criteria. As shown in chapter 4, We measure the diversity of generated images using established evaluation methods and analyze the sampling time efficiency. Additionally, we gather feedback from domain experts and users to gain insights into the practical implications of our solution.

3.2 Design and development

The design and development phase adheres to typical machine learning training procedures, illustrated in Figure 3.2, which outlines five main operations: Data Acquisition, Preprocessing, Training, Post-Processing, and Deployment. Subsection 3.2.1 elucidates the data acquisition process, while subsection 3.2.2 delves into preprocessing techniques. Subsection 3.2.3 outlines the modeling of the diffusion-based GAN. Additionally, subsection 3.2.3 discusses a method for visualizing generated floorplans through post-processing. However, the deployment step is not addressed in this work, as it falls outside the scope of our research.

3.2.1 Data Acquisition

Acquiring pertinent data is pivotal for effectively investigating and validating our research hypotheses. To ensure the reliability and authenticity of our findings, we have meticulously selected the RPLAN dataset [173], comprising an extensive compilation of 60,000 real residential

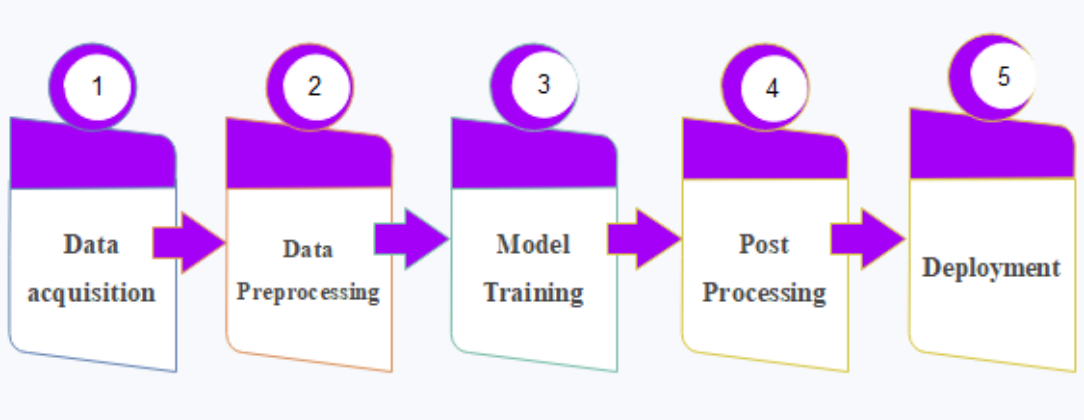

Figure 3.2: From Data Acquisition to Deployment

building layouts. Each floor plan in the dataset is meticulously annotated at the pixel level, providing semantic information. These floor plans typically encompass 5 to 8 rooms with sizes ranging from 10 to 20 square meters, all including a living room. Furthermore, the overall size of the floor plans within the dataset varies between 60 to 120 square meters. We opted for the RPLAN dataset due to its relevance and representativeness, closely aligning with the objectives of our study. Crafted by professional architects, each floor plan in this dataset is represented as pixel images, notable for its richness in architectural intricacies and diversity in design styles, thus rendering it an ideal resource for our research objectives. Leveraging this meticulously curated dataset, we endeavor to extract meaningful insights, validate our proposed methodologies, and advance the state-of-the-art in our domain of study.

In order to train the model, the floorplan samples are divided into four groups based on the number of rooms: 5, 6, 7, or 8 rooms. During the training phase, the samples in each group are excluded to prevent methods from merely memorizing them [8]. This exclusion encourages the development of innovative methods that can generate new floorplan layouts and adapt to diverse room configurations within each group. Subsequently, during the evaluation process, the excluded groups serve as a benchmark to assess the methods' generalization abilities. By evaluating the methods on these unseen room configurations, their performance can be accurately evaluated. Furthermore, during the testing phase, the methods are tested on completely unseen floorplan samples to validate their effectiveness in generating diverse and novel layouts. This approach ensures that the methods are robust, capable of handling various room configurations, and capable of producing reliable and creative floorplan solutions.

3.2.2 Data preprocessing

Recognizing the impact of directly using acquired floorplans on performance, we undertake a crucial step: preprocessing. As we prepare for model training, we embark on a journey of refining our raw RGB floorplan images (depicted in Figure 3.3). Inspired by the innovative techniques found in HouseGAN++ [6], we meticulously craft a series of preprocessing steps. Essentially, our preprocessing journey entails rescaling [174], vectorization [175], and graph formation [8]. Firstly, we meticulously resize each image, ensuring uniformity and consistency at a resolution of 256x256. This step lays the foundation for subsequent transformations, providing our models with a standardized canvas to work on [174].

Next, we delve into the intricate world of floorplan vectorization used to capture the essential information. Unlike conventional approaches that heavily rely on techniques such as edge detection [176] and filtering [177], our research takes a different path by adopting the "Raster-to-Vector" (RtV) method proposed by Eli et al. [175] in their work on revisiting floorplan transformation. Conventional methods often depend on a series of low-level image processing heuristics, which may struggle to accurately capture the intricate details and semantic information present in the input images [178]. In contrast, the RtV method [175] combines convolutional neural networks [13] with Integer Programming [179] to overcome these limitations and generate precise vectorized representations of our floorplans. By leveraging a Convolutional Neural Network (CNN) [13] for wall junction extraction and pixel-wise room classification, the RtV method [175] effectively captures both geometric and semantic information at a granular level. Subsequently, utilizing Integer Programming [179], the RtV method generates primitives representing spatial structures such as walls and doors, ensuring adherence to inherent constraints within the floorplans, such as room enclosure and structural consistency.

The vectorized output of the floorplan encapsulates key details about room types, spatial boundaries, and room connectivity. Categorizing rooms aids in their classification, while precise coordinates delineate room boundaries. Each room specifies its starting and ending coordinates, facilitating visualization and spatial analysis. Additionally, connections between rooms offer insights into room adjacency and floorplan organization. Together, these components provide a comprehensive understanding of the floorplan's layout and spatial relationships.

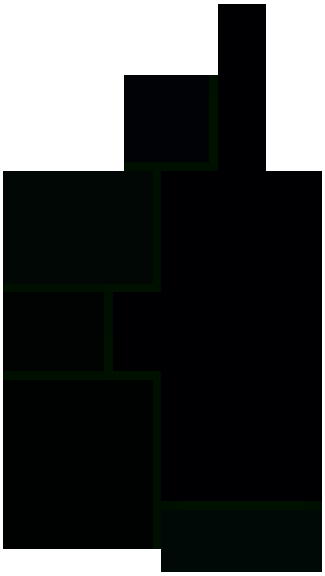

Figure 3.3: Raw Floorplan.

In the visualization illustrated in Figure 3.4, each extracted mask represents a unique room type and is delineated using a grayscale representation. These masks symbolize various room classifications according to the predefined room class, where each room type is assigned a unique numerical identifier. For instance, the living room is represented by the value 1, the kitchen by 2, the bedroom by 3, the bathroom by 4, the balcony by 5, the entrance by 6, the dining room by 7, and the study room by 8. Additional classifications include storage areas represented by 10, the front door by 15, unknown regions by 16, and interior doors by 17.

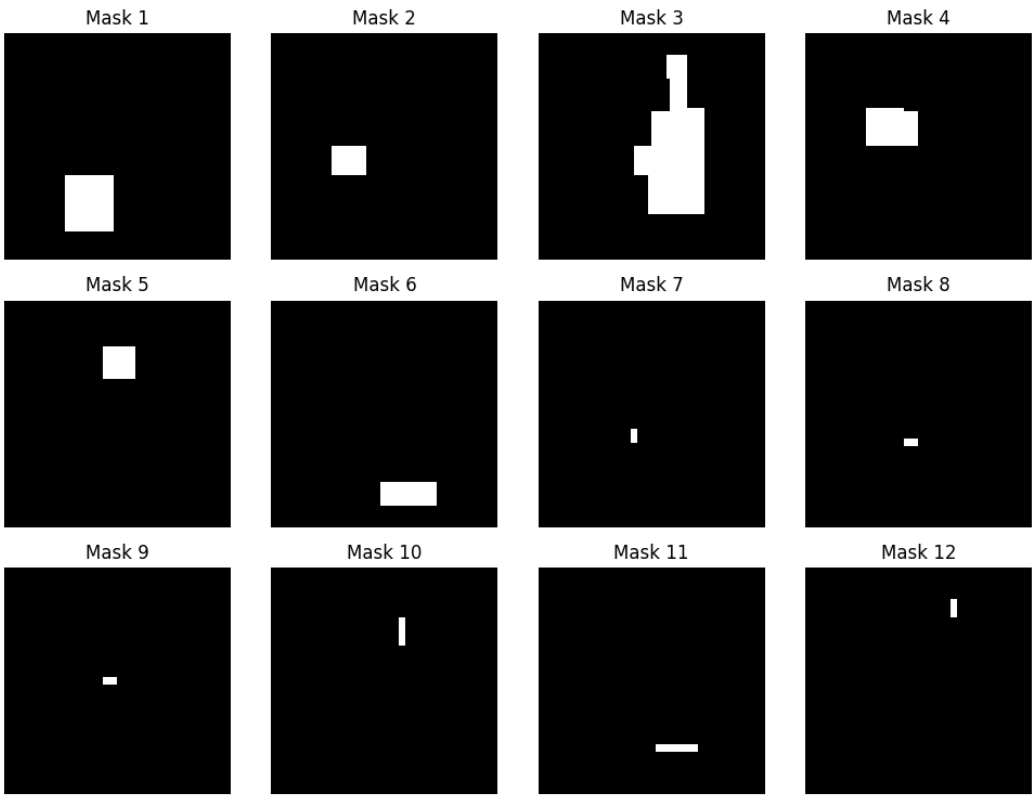

Figure 3.4: Extracted Masks.

After converting the raster floorplan into a vectorized representation, the subsequent step involves creating a graph that depicts the connectivity of architectural components like rooms, walls, and doors [8]. This graph-based representation offers several benefits. Firstly, graphs provide a structured and organized approach to capture the spatial relationships and connections between different elements of the floorplan [51]. This information is crucial for generating floorplans that are coherent and realistic, ensuring that the generated layouts adhere to architectural principles. Secondly, representing the floorplan as a graph enables the utilization of graph-based algorithms and techniques [8]. These algorithms can leverage the connectivity information encoded in the graph to generate new floorplan layouts while preserving spatial coherence and adhering to architectural constraints.

To construct this graph, each room within the vectorized floorplan is transmuted into a

node, serving as a foundational element [8]. Leveraging the wealth of information extracted from the floorplan, including room details and types, we meticulously populate these nodes with attributes, enabling a comprehensive representation of the floorplan's layout and organization.

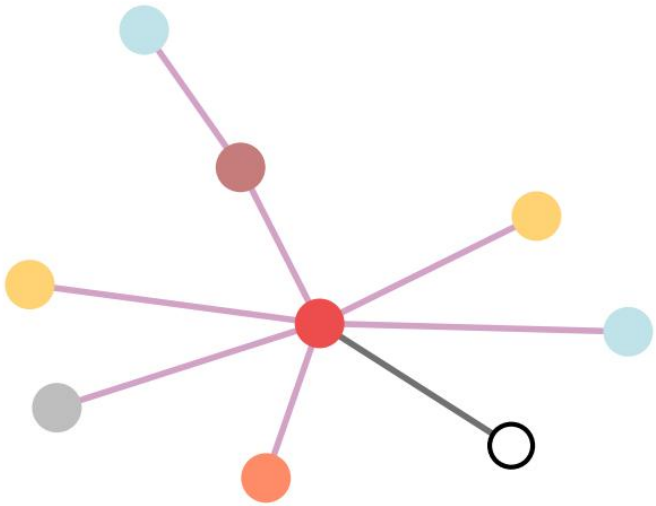

Figure 3.5: Graph representation.

Yet, it is the establishment of edges within this graph that truly breathes life into our representation. By carefully analyzing the positions of doors in each room, we discover how different spaces within the floorplan interact with each other [8]. We do this by observing if a door in one room is located on the same wall as a door in another room [6]. These shared walls and close proximity between doors act as signals, guiding us to create edges or connections between the corresponding rooms (Nodes) in the graph representation [6].

3.2.3 Modeling

The modeling approach presented in this work combines two powerful methods: Generative Adversarial Networks (GANs) [23] and diffusion models [49], as depicted in Figure 3.6. In the forward process, diffusion occurs across time steps, introducing noise along the way. Conversely, the reverse process leverages GANs in a multimodal [50] framework. The GAN architecture consists of a generator utilizing convolutional message passing [116] with attention network [117] and upsampling techniques, while the discriminator employs convolutional message passing [116] with attention network [117] and downsampling operations. Additionally, to control the temporal dynamics, sinusoidal time embedding is employed at each up or downscale operation, enabling fine-grained manipulation of time representations within the model.

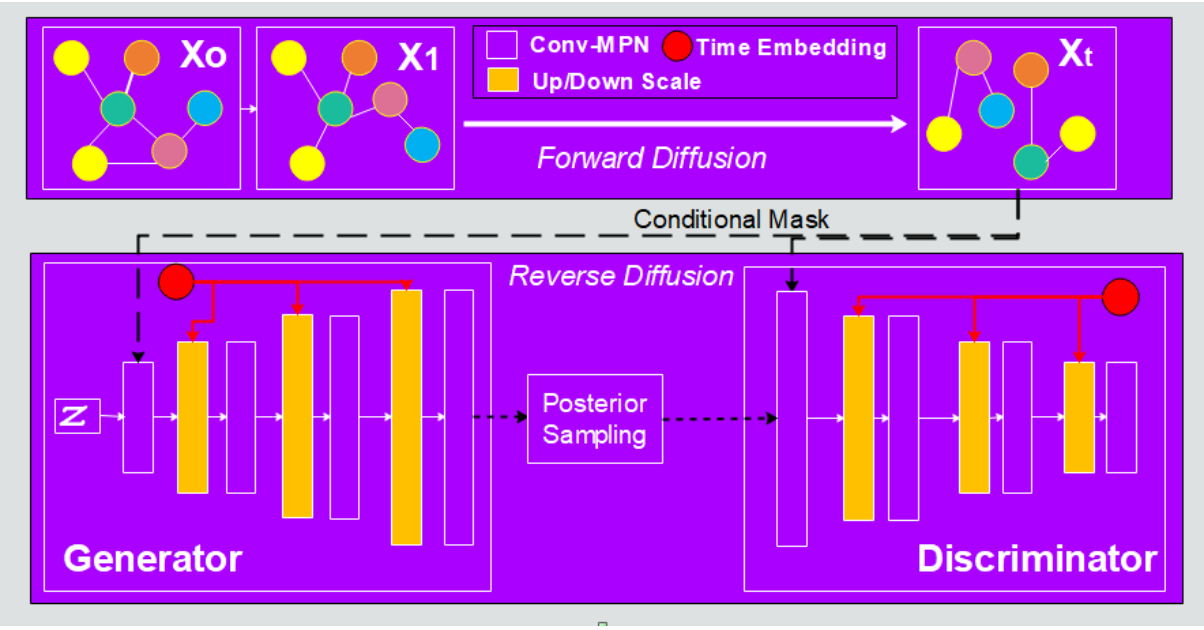

Figure 3.6: Floorplan Generation Model.

3.2.3.1 Forward Diffuision Process

The preprocessed graph data is subsequently fed into the forward diffusion process, allowing for an exploration of the underlying data distribution [180]. The previous approaches [181, 182] perform a forward diffusion process by embedding the graphs in a continuous space and adding Gaussian noise [21] to the node features and graph adjacency matrix. However, when the graphs are embedded in a continuous space, there is a risk of losing important discrete structural information, such as connectivity or cycle counts. Additionally, the Gaussian noise added to the node features and adjacency matrix can further obscure the underlying structural properties of the graph. Consequently, the denoising network may face challenges in accurately capturing and restoring the original graph's structural properties from the noisy inputs [180]. This limitation can potentially result in suboptimal performance in tasks that heavily rely on the graph's structural information.

To address this concern, we deviate from the conventional approach of treating the graph as a continuous space. Instead, inspired by the motive of DiGrees [180], we choose to treat the graph as a discrete space. Our model handles graphs with categorical node and edge attributes, represented by the spaces X and E, respectively, with cardinalities a and b. We use x_i to denote the attribute of node i and $x_i \in \mathbb{R}^a$ to denote its one-hot encoding. These encodings are organized in a matrix $X \in \mathbb{R}^{n \times a}$ where n is the number of nodes. Similarly, a tensor $E \in \mathbb{R}^{n \times n \times b}$

groups the one-hot encoding e_{ij} of each edge, treating the absence of an edge as a particular edge type.

Similarly to diffusion models for images, which apply noise independently on each pixel, we diffuse separately on each node and edge feature. As a result, the state-space that we consider is not that of graphs (which would be too large to build a transition matrix) [180], but only the node types X and edge types E. For any node (resp. edge), the transition probabilities [183] are defined by the matrices $[Q_t^X]_{ij} = q(x_t = j|x_{t-1} = i)$ and $[Q_t^E]_{ij} = q(e_t = j|e_{t-1} = i)$. Adding noise to form $G_t = (X_t, E_t)$ simply means sampling each node and edge type from a categorical distribution defined by :

$$q(G_t|G_{t-1}) = (X_{t-1}Q_t^X, E_{t-1}Q_t^E) \quad \text{and} \quad q(G_t|G) = (X\bar{Q}_t^X, \bar{Q}_t^E) \tag{3.1}$$

for $\bar{Q}_t^X = Q_1^X ... Q_t^X$ and $\bar{Q}_t^E = Q_1^E ... Q_t^E$.

In our discrete denoising diffusion model for graphs, we incorporate a cosine noise schedule [49] to control noise levels during the diffusion process. As shown in figure 3.7, the cosine noise schedule offers a smooth transition, avoiding sudden fluctuations and enabling a controlled evolution of the graph structure. This scheduling strategy addresses two key considerations: introducing sufficient noise for exploration and diversification, and preserving the underlying structural integrity [49]. By gradually increasing the noise level using the cosine function, we encourage progressive perturbation while avoiding excessive disruption [84]. This ensures that important structural properties, such as connectivity and cycle counts, are retained in the generated graphs. The inclusion of the cosine noise schedule strikes a balance between exploration and preservation, allowing for diverse graph generation while maintaining meaningful and interpretable structural properties.

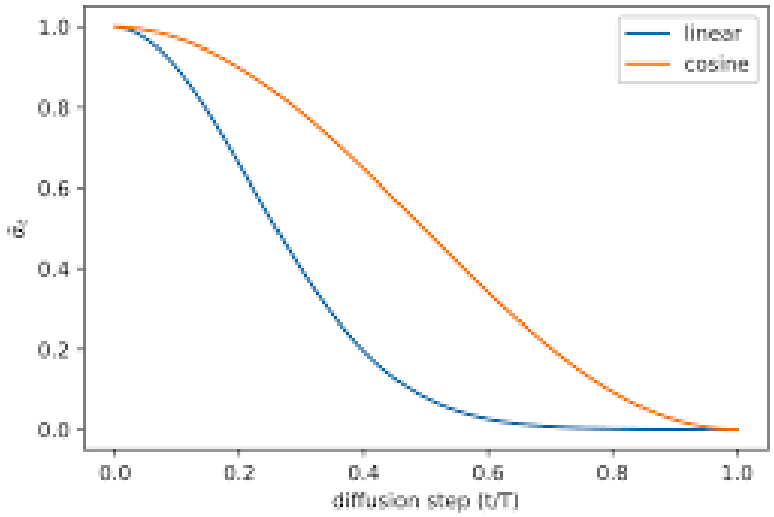

Figure 3.7: Noise Scheduler. Source: Lin et al. [2]

3.2.3.2 Architecture

Unlike Convolutional Neural Networks (CNNs) [13], which may struggle to effectively capture spatial relationships [100], our architecture harnesses the power of Graph Neural Networks (GNNs) [51] to address this challenge. GNNs are uniquely equipped to capture intricate spatial dependencies between rooms in floorplan data by explicitly modeling graph structures and leveraging message-passing techniques [51]. By leveraging GNNs, we aim to overcome the limitations of CNNs and achieve a more comprehensive understanding of the spatial layout inherent in floorplan data.

Among the various variants of Graph Neural Networks (GNNs) [51], our architecture is constructed upon an attention-based Convolutional Message Passing Network. This pioneering framework integrates the merits of attention-based message-passing networks with convolutional layers.

1. Message Passing Network (MPN)

Message passing is a fundamental concept in graph-based machine learning, where information is exchanged between nodes to enhance understanding and representation [128]. Existing message passing techniques like GCN [97] and GAT [117] often operate in single vector form, which may lack the capacity to comprehensively capture all feature volume information [116]. Conversely, if they can handle feature volume information like CMPN [116], they may not conduct adequate message passing, thus affording equal opportunity to all nodes [117]. Consequently, in

response to these limitations, we have developed weight-based feature message passings employing attention mechanisms. This approach enables the model to dynamically assign importance to different nodes based on their relevance, allowing for more effective information propagation and improved adaptation to the underlying graph structure [117].

"Attention is all you need." *Ashish Vaswani, et al.*

Unlike Conventional attention mechanisms [89] primarily operate on sequence data, employing fixed-length vectors for input representation and computing attention through dot product or similarity functions between elements, our graph attention mechanisms specialize in graph-structured data, utilizing the graph structure with node features for input representation and performing attention computation through graph convolution operations. While conventional attention mechanisms are limited to modeling relationships between adjacent sequence elements and contextualizing elements based on neighboring elements, graph attention mechanisms excel in capturing relationships between arbitrary graph nodes and contextualizing nodes based on their interactions with neighboring nodes. Moreover, graph attention mechanisms provide an explicit measure of node importance by assigning importance scores to nodes based on neighborhood relationships, whereas conventional attention mechanisms lack this feature [117].

The key idea in attention-based convolutional message passing is to compute attention coefficients that capture the importance of each neighbor for a given node. This approach allows the model to selectively focus on relevant information during the aggregation process [117]. As shown mathematically in equation 3.3, attention scores between pairs of nodes are computed using a shared weight matrix. These scores are then normalized using the softmax function, resulting in attention coefficients that represent the relative importance of each neighbor. The attention coefficient measures the importance of current node v for connected node w or non-connected node r, employing a shared attention mechanism across all nodes.

$$\alpha_{wv}^{k} = \text{softmax}(\text{LeakyReLU}(a^{k\top} \cdot [W^k f_v \| W^k f_w])) \tag{3.2}$$

The operation starts by linearly transforming the feature representations of the current node v and connected node w using the respective learnable weight matrices W^k. Next, these transformed features are concatenated together. The resulting concatenation is then subjected to a dot product operation with a learnable weight vector a^k. The dot product result is passed

through a LeakyReLU activation function to introduce non-linearity. This value represents the initial importance or relevance of the connected node w to the current node v within the k-th attention head.

To obtain the final attention coefficients, the softmax function is applied to the dot product result. The softmax function normalizes the values across all connected nodes w for a given current node v, producing a probability distribution. The resulting attention coefficients indicate the relative importance of each connected node w in influencing the current node v during information aggregation and processing.

Then, through the multiplication of the attention coefficients with the corresponding neighbor node features and their subsequent pooling operation, a weighted aggregation of information is attained, as illustrated in equation 3.3. This aggregation step allows the node to incorporate information from its neighbors based on their importance, capturing the most relevant aspects of the graph structure.

$$f_v \leftarrow \mathrm{CNN}\left(f_v; \mathrm{Pool}_{w \in N(v)} \alpha^k_{wv} f_w; \mathrm{Pool}_{r \in N(v)} \alpha^k_{rv} f_r\right) \tag{3.3}$$

To aggregate information and capture global insights [8], our approach involves performing message passings across both connected and non-connected nodes. Specifically, we employ attention-based message passing to update the current feature vector f_v. This process includes pooling the features of connected nodes (Pool$w \in N(v)\alpha^k wv f_w$) and non-connected nodes (Pool$r \in N(v)\alpha^k rv f_r$) using attention weights α^k_{wv} and α^k_{rv}, respectively. By incorporating attention mechanisms, our model dynamically assigns importance to neighboring nodes, facilitating effective information propagation and enabling the model to capture both local and global dependencies within the graph structure.

In our experiments, we explored different pooling methods, namely max, sum, and mean poolings, to encode the messages effectively. Among these methods, the max pooling operation yielded the best results. By employing max pooling, we were able to capture the most salient features from both connected and non-connected nodes, thereby enhancing the overall performance of our model. It allowed us to effectively extract and incorporate the most relevant information during the message passing process, contributing to improved representation learning and enhanced model capabilities.

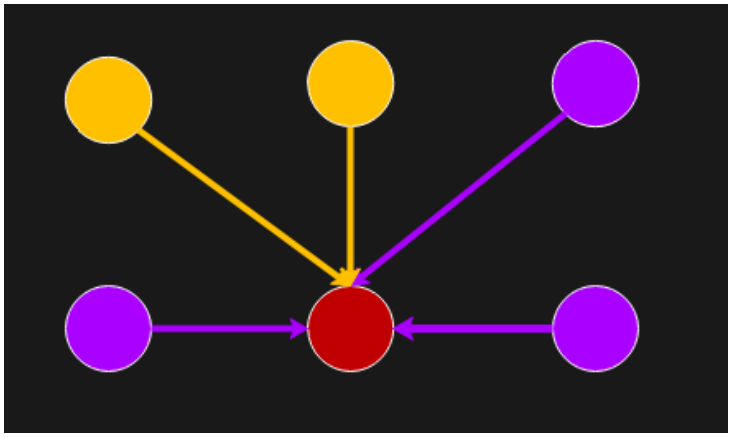

Figure 3.8: Message passing operation across connected and nonconnected nodes

Hint: The diagram shown in Figure 3.8 illustrates the process of Message Passings. In this diagram, there are different types of nodes represented by various colors. The red node represents the current node, denoted as f_v, which is the focal node we are focusing on. The non-connected yellow nodes represent the neighboring nodes of the current node, specifically $Pool_{w \in N(v)} f_w$. These yellow nodes represent the pooled information from the nonconnected nodes to the current node. The connected purple nodes represent the other nodes that are directly connected to the current node, denoted as $Pool_{r \in N(v)} f_r$. These purple nodes represent the pooled information from the connected nodes to the current node.

2. Convolution Layer

As shown in equation 3.3, the concatenated features obtained from the previous message passing steps are processed by a convolutional neural network (CNN) [13]. Convolutional operations are crucial in capturing local patterns and relationships within the graph. By applying filters to the feature vectors of the graph nodes, the CNN extracts and aggregates information from the immediate neighborhood of each node, capturing essential local dependencies and patterns [116]. This allows the network to understand the underlying structure of the graph. The use of convolutional operations also enables parameter sharing, reducing the number of parameters and enhancing generalization [13]. Through the iterative application of convolutional operations and aggregation of information from neighboring nodes, Our model propagates and integrates local information to capture global patterns and dependencies, resulting in informative representations of the room-wise feature volumes.

3.2.3.3 Generator

As we know, the generator in a Generative Adversarial Network (GAN) is a vital component responsible for synthesizing new data samples [23]. In our approach, we employ the generator in a reverse diffusion process, utilizing it in a multimodal fashion. Its main objective is to learn a mapping function from a latent space to the data space, enabling the generation of diverse and high-quality samples that resemble the distribution of the training data [33].

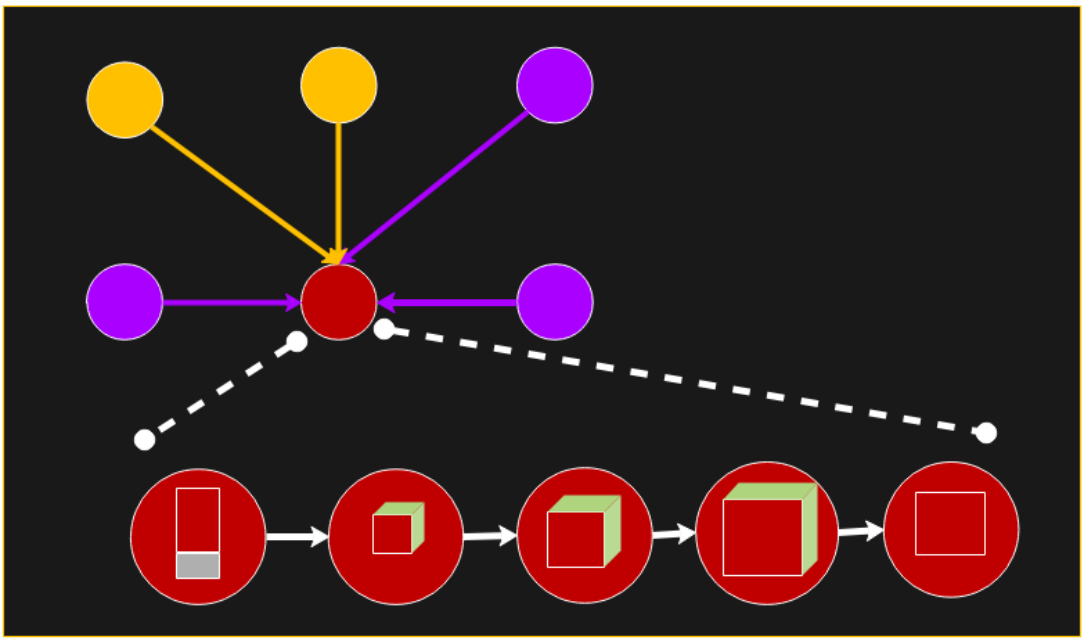

Figure 3.9: A schematic overview for generators operation.

In our proposed approach, the generator in the GAN operates using a graph format. To begin, we utilize a given bubble graph as the input graph representation. As shown in figure 3.9, these nodes are initially initialized with 128-dimensional noise vectors, which are sampled from a normal distribution. To incorporate additional room-specific information, an 18-dimensional room-type vector is concatenated with the noise vector for each node. This room-type vector is encoded in the one-hot format, where each dimension corresponds to a specific room index. By combining the noise vector and the room type vector, we obtain a 146-dimensional vector. This comprehensive vector serves as the representation of the input bubble graph within the graph, encompassing both the random noise and the room type information. To expand this vector, a linear reshape operation is performed, resulting in an (8x8x16) feature volume.

In the subsequent steps, the feature volumes undergo a series of operations to enhance their resolution and capture more detailed information [8]. Initially, the feature volumes are

fed into a message passing network, where information is aggregated and updated. Following this, an upsampling operation is performed, resulting in a feature volume of size (16x16x16). The upsampled feature volume is then fed back into the message-passing network. Another round of upsampling is conducted, yielding a feature volume of size (32x32x16). Finally, the last message passing and upsampling operation is performed, resulting in a feature volume of size (64x64x16). These operations, including upsampling and utilization of the Convolutional Message Passing Network, aim to increase the resolution and enrich the feature representation of the volumes, facilitating more detailed analysis and processing [37].

To ensure compatibility with the discriminator's input dimensions [23], we employ a three-layer Convolutional Neural Network (CNN) to convert the feature volume into the desired segmentation mask of (64x64x1) dimensions. This CNN architecture applies a series of convolutional operations to learn and extract features from the input, resulting in a mask that accurately delineates the segmentation of the floorplan at the desired resolution. It should be noted that the output of the generator typically takes the form of a graph rather than a raster format.

3.2.3.4 Discriminator

Another crucial component is the discriminator, which takes graph-formatted inputs and is trained on both generated graphs and preprocessed real graphs. The discriminator's main objective is to distinguish between these two types of inputs and determine their authenticity [23]. It receives graph representations, either generated by the generator or from real data, as its input. Through training, the discriminator learns to differentiate between the features and patterns present in the generated graphs and the real graphs. This adversarial training process encourages the generator to produce more realistic and coherent graphs that closely resemble the true underlying graph distribution.

The discriminator takes as input a graph representation of room segmentation masks (64x64x1). To incorporate room type information into this segmentation mask, a one-hot encoded room type vector is passed through a linear layer, and reshaped to a (64x64x8) format. This process ensures that the discriminator has access to both the spatial information of the segmentation mask and the categorical information of the room types [6].

Next, a three-layer CNN is applied to the concatenated features, resulting in a (64x64x16)

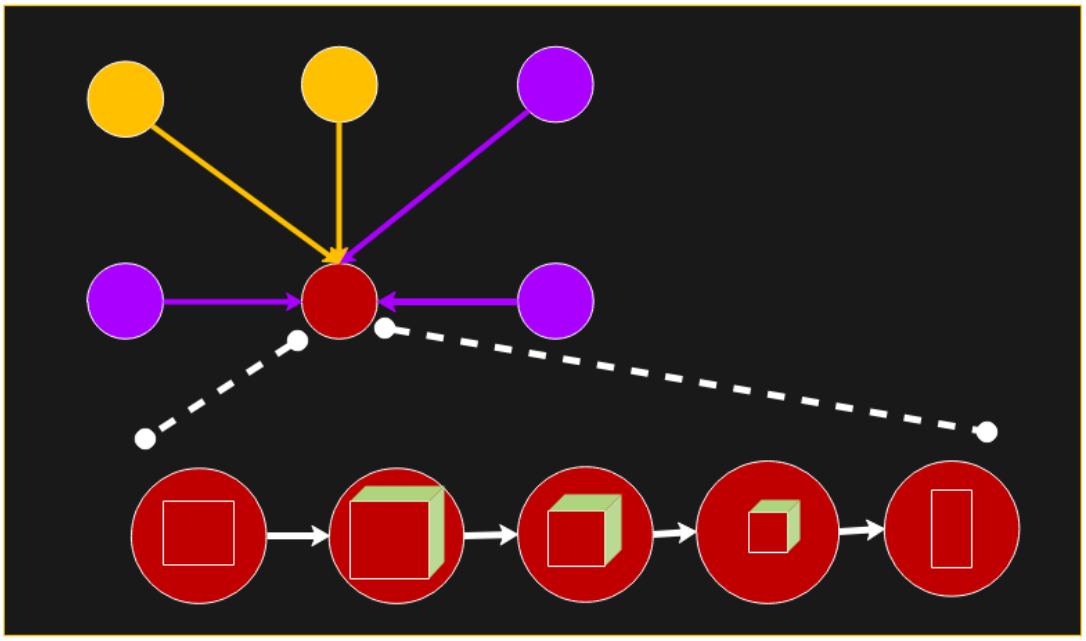

Figure 3.10: A schematic overview for discriminators operation.

feature volume. Subsequently, three downsampling and four ConvMPN modules are applied. These steps can be considered as the reverse of the upsampling and ConvMPN modules applied in the generator. The downsample operation reduces the spatial resolution of the feature volume while capturing more abstract information [37], and the ConvMPN module performs message passing and aggregation operations, incorporating information from neighboring features.

Lastly, a three-layer CNN is applied again to the processed feature volume, resulting in a 128-dimensional vector. This final CNN module extracts discriminative features from the feature volume and maps them into a lower-dimensional space, which serves as the output representation of the discriminator. By applying these operations, the discriminator learns to distinguish between real and generated room segmentation masks by capturing both spatial and categorical information and extracting meaningful features for discrimination.

To enhance the diversity of the generated floorplans, our discriminator incorporates the technique of mini-batch discrimination during training [39]. Mini-batch discrimination introduces additional information into the discriminator's computation by considering multiple samples within a mini-batch. This technique aims to encourage the generator to produce a broader range of unique and distinct floorplan outputs [42]. During training, the discriminator plays a crucial role in distinguishing between the generated floorplans and real floorplans. By incorporating mini-batch discrimination, the discriminator not only focuses on individual samples

but also considers the relationships between samples within the mini-batch. This allows the discriminator to capture patterns and variations that may arise from the generator producing similar or repetitive floorplans [39].

3.2.3.5 Spectral Normalization

In our methodology, we leverage key components for improved training and diverse generation of floorplans: spectral normalization in the convolution layers [41]. Spectral normalization is applied to the weight matrices of the convolution layers in the discriminator network. This technique stabilizes the training process by constraining the magnitude of the weight parameters and controlling the Lipschitz constant [145]. By doing so, spectral normalization prevents the discriminator from dominating the training dynamics and helps maintain a balance between the discriminator and the generator. This regularization method enhances training stability, reduces mode collapse, and promotes diversity in the generated floorplans [77].

$$W_{\text{spectral}} = \frac{W}{\sigma(W)} \tag{3.4}$$

The spectral norm, $\sigma(W)$, is calculated using the power iteration method. It involves iteratively multiplying the weight matrix with its transpose until convergence to approximate the largest singular value, which corresponds to the spectral norm.

By dividing the weight matrix by its spectral norm, we normalize the weight parameters. This ensures that the discriminator does not dominate the training dynamics and maintains a balanced interaction with the generator. The regularization effect of spectral normalization improves training stability, reduces mode collapse, and promotes diversity in the generated floorplans. This allows the generator to explore a wider range of design possibilities, resulting in more varied and diverse floorplan outputs [39].

3.2.3.6 Training Algorithms

In conventional diffusion models [24, 49, 84], noise is typically introduced to the training data during the forward process, while the reverse process aims to gradually remove the added noise through iterative unimodal denoising steps, as shown in figure 2.3. However, this unimodal denoising process often involves performing a large number of denoising steps, which can significantly impact the time required for sampling [33]. The iterative nature of the denoising steps,

although effective in restoring the noise-free data, can result in slower sampling times due to the prolonged computations involved in each step [45].

However, our model, as depicted in Figure 3.11, introduces a significant departure from the conventional methodology by implementing the reverse process in a multimodal fashion, integrating a GAN model through adversarial training. This incorporation allows our model to perform multiple denoising steps in the reverse process [33]. The advantage of this approach is that it can lead to reduced denoising steps, resulting in optimized sampling time. By performing multiple denoising steps, our model gains the ability to explore and capture diverse modes of the noise-free data distribution. This flexibility enables our model to effectively handle complex data distributions and capture different plausible denoised outputs. Consequently, it enhances the overall performance of our model by providing a more comprehensive representation of the underlying data distribution.

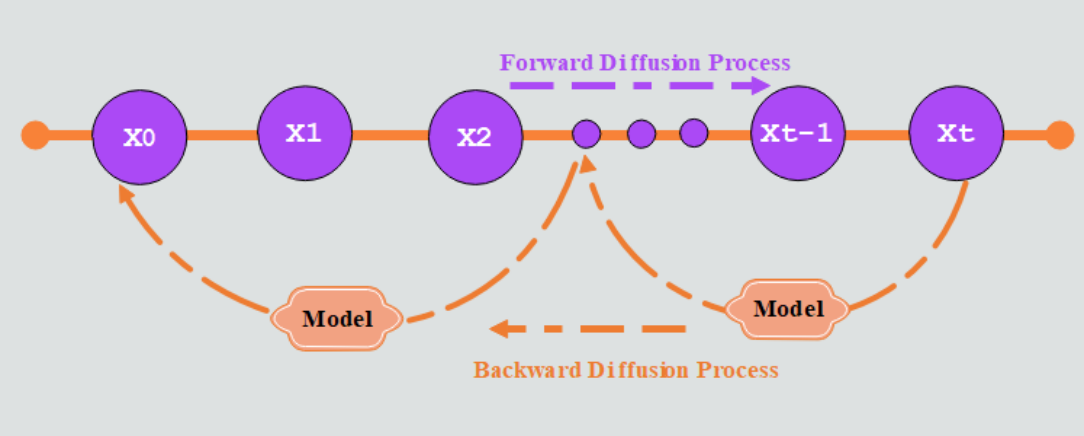

Figure 3.11: Forward and backward of or model training.

The training algorithm (algorithm 4) begins by sampling a batch of real data samples, denoted as X, from the dataset. These real data samples will be used to train our model. Additionally, noise is applied to the real data batch at each time step T using the noise function $Q(X_t|X_{t-1})$. This noise injection is performed to introduce variability and help the model learn robust representations [84].

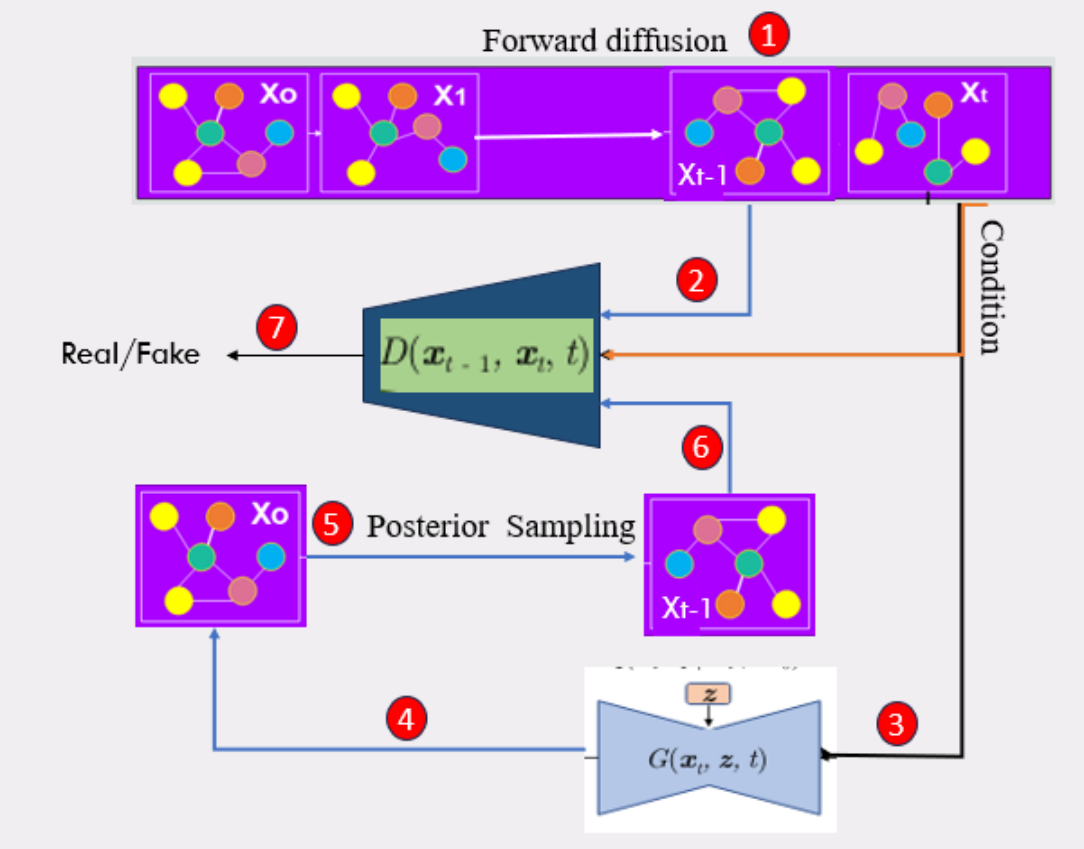

Figure 3.12: Pipeline for training algorithms

Algorithm 4 Training Algorithm

1: Sample a batch of real data examples X from the dataset.
2: Apply noise $Q(X_t|X_{t-1})$ to the real data batch at time step T.
3: **for** each training iteration **do**
4: **for** each denoising step t **do**
5: Train the Discriminator on real data : $D(X_{t-1}, X_t, t)$.
6: Train the Generator on random noise: $G(X_t, Z, t)$.
7: Posterior Sampling $Q(X_{t-1}|X_t, X_0)$.
8: Calculate the Loss of Generator (L_G).
9: Train Discriminator on fake data (Generated) : $D(X_{t-1}, X_t, t)$.
10: Calculate the Loss of Discriminator (L_D).
11: **end for**
12: Update the parameters of the generator G and discriminator D networks by the following losses:
13: Generator loss: $L_G = -\mathbb{E}_{z\sim P_z}[D(G(X_t, z, t))]$
14: Discriminator loss: $L_D = -\mathbb{E}_{x\sim P_r}[D(X_{t-1}, X_t, t)] + \mathbb{E}_{z\sim P_z}[D(G(X_t, z, t))] + \lambda \cdot \text{GP}$
15: **end for**
16: Perform denoising by minimizing the difference between X_t and X_{t-1} using a multi-modal conditional GAN approach.
17: Repeat the training loop until convergence or for a fixed number of iterations.

Now it's a time in which learning performs via a backward diffusion process [49]. The

algorithm then enters a training iteration loop. Within each iteration, a series of denoising steps are performed. At each denoising step t, the Discriminator is trained on the real data samples X_{t-1} while conditioning on X_t, using the loss function $D(X_{t-1}, X_t, t)$. This step aims to improve the Discriminator's ability to distinguish between real and generated data [23]. Simultaneously, the Generator is trained on random noise Z and the denoised sample X_t as conditioning, using the function $G(X_t, Z, t)$. This step encourages the Generator to generate samples that resemble the real data distribution [23].

Following the training of the Discriminator and Generator, posterior sampling is performed to estimate the distribution $Q(X_{t-1}|X_t, X_0)$. This step helps refine the generator's output by considering the generated sample X_t and the initial sample X_0 [49]. The Loss of the Generator (L_G) is then calculated, representing how well the Generator is generating realistic samples. Subsequently, the Discriminator is trained on the generated (fake) data samples X_{t-1} and X_t using the loss function $D(X_{t-1}, X_t, t)$. This step aims to improve the Discriminator's ability to accurately classify real and fake samples [23]. Finally, the parameters of the Generator and Discriminator networks are updated according to their respective losses, leading to improved performance over successive iterations.

Generator Loss:

The generator loss WGAN-GP [132] aims to minimize the negative Wasserstein distance (-W(P_r, P_g)), where P_r represents the distribution of real samples and P_g represents the distribution of generated samples. As shown in equation 3.5 the generator loss is given by:

$$L_G = -\mathbb{E}_{z\sim P_z}[D(G(X_t, z, t))] \tag{3.5}$$

where z is a noise sample, X_t is input condition, t is the number of denoising time steps , G is the generator function, and D is the discriminator function.

Discriminator Loss:

The discriminator loss consists of two parts: the Wasserstein distance term [140] and the gradient penalty term [142]. As shown in equation 3.6 the discriminator loss is defined as:

$$L_D = -\mathbb{E}_{x\sim P_r}[D(X_{t-1}, X_t, t)] + \mathbb{E}_{z\sim P_z}[D(G(X_t, z, t))] + \lambda \cdot \text{GP} \tag{3.6}$$

where x represents a real sample, z represents a noise sample,X_t is input condition, X_{t-1} is fake or real sample, t is the number of denoising time steps, G is the generator function, D is the discriminator function, and GP denotes the gradient penalty term.

The Wasserstein distance, $W(P_r, P_g)$, is calculated as the difference between the expected values of the discriminator's outputs on real and generated samples:

$$W(P_r, P_g) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{z\sim P_z}[D(G(z))] \tag{3.7}$$

The gradient penalty term, GP, is used to enforce the Lipschitz constraint on the discriminator. It penalizes the norm of the discriminator's gradients with respect to interpolated samples between real and generated samples. The gradient penalty is formulated as:

$$\text{GP} = \lambda \cdot \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(|\nabla_{\hat{x}} D(\hat{x})|_2 - 1)^2] \tag{3.8}$$

where λ is a hyperparameter that controls the weight of the penalty, $\hat{x}$ represents interpolated samples between real and generated samples, and $D(\hat{x})$ represents the discriminator's output for $\hat{x}$.

In conclusion, the training algorithm aimed at optimizing the balance between diversity and sampling time incorporates a Generative Adversarial Network (GAN) into the diffusion model. This injection of GAN introduces a multimodal denoising process, which not only improves sampling time but also enables the model to effectively capture the underlying data distribution. By performing denoising in a multimodal fashion, the model can explore and generate diverse denoised samples, thereby enhancing the diversity of the output. Additionally, the introduction of a gradient penalty helps maintain training stability in the GAN framework. This penalty term encourages smoothness and prevents the generator from producing unrealistic samples during the denoising process, contributing to the overall stability and quality of the generated outputs.

3.2.3.7 Postprocessing

Floorplan construction from generator output involves the utilization of room segmentation masks generated for each room. These masks serve as a representation of the boundaries of individual rooms within a floorplan. However, to achieve a more refined and visually coherent

floorplan, the technique of rectangle fitting is employed as part of the data post-processing stage.

In the process of rectangle fitting, the room segmentation masks are first transformed to binary images by thresholding the output of the tanh activation function at 0.0. This conversion allows for a clear distinction between the room and non-room areas within the floorplan [8]. Once the binary masks are obtained, the next step is to generate the tightest axis-aligned rectangle for each room. This rectangular tight fitting offers a simpler and more efficient approach for aligning room segmentation masks with the walls and corners, ensuring a visually appealing and grid-aligned floorplan representation [184]. The process of generating the tightest rectangle for each room involves finding the minimum bounding box that tightly encompasses the foreground pixels of the room mask. By aligning the rectangle with the walls and corners of the room, a more accurate and visually appealing floorplan representation is achieved.

Furthermore, the concept of spatial adjacency plays a significant role in the rectangle fitting process. Two rooms with a shared edge must be spatially adjacent, indicating a connection between them, often represented by a door [6]. This adjacency constraint ensures that the generated floorplan maintains logical connectivity between rooms, maintaining functional relationships and promoting efficient space utilization.

In summary, floorplan construction from generator output involves the use of room segmentation masks to delineate individual rooms. The technique of rectangle fitting is then applied in the data post-processing stage to generate tight-fitting, axis-aligned rectangles for each room. These rectangles align with the walls and corners of the rooms, enhancing the visual coherence of the floorplan. Additionally, the adjacency constraint ensures spatial connectivity between rooms, often represented by doors. By incorporating these techniques, the generated floorplans become more realistic, functional, and visually appealing, facilitating effective architectural design and space planning.

Chapter 4

Experiments

4.1 Implementation Details

In order to assess the performance of our model, we initially implemented baseline static House-GAN++ [6] and HouseDiffusion [7] models on the RPLAN dataset. A comprehensive compilation of the hyperparameters employed is presented in Table 4.1. Multiple models were then trained utilizing these pre-defined hyperparameters. Subsequently, the most effective model was chosen for each size category based on the specified hyperparameters, wherein the final selected model hyperparameters were distinctly highlighted within Table 4.1.

The determination of hyperparameter ranges was influenced by the limitations imposed by available computational resources. However, it is crucial to note that extensive experimentation was undertaken to meticulously explore and identify the optimal settings. This process involved a thorough evaluation of various hyperparameter combinations to ascertain the configurations that yielded the best performance.

The proposed architecture was implemented in PyTorch and trained using a workstation equipped with Google Colab A100 GPUs. To provide further insight into the choices made, the following list offers a detailed description of the specific hyperparameter selections. These choices were made with careful consideration of their impact on the model's behavior and performance during the training phase.

- **Batch size:** refers to the number of samples processed simultaneously before updating the model's weights during training. In our experiments, we varied the batch size to observe its impact on model performance. Larger batch sizes, such as 32, tend to result in faster convergence, whereas smaller batch sizes, like 8, require longer training times. However, smaller batch sizes introduce more randomness and noise into the gradient estimation process, which promotes diversity in the generated samples. This increased stochasticity allows the generator to explore a wider range of variations, enhancing the uniqueness and variety of the outputs produced by the model.

Table 4.1: Model hyperparameters.

Parameters	**Labels**
g_lr	0.0001
d_lr	0.0001
b1	0.5
b2	0.999
Epoch	200
CMPN_G	4
CMPN_D	4
Batch size	**8**
Upsample	3
Downsample	3
Gradient penalty	10
Kernel size	3
Stride and padding	1
Optimizer	**Adam**
Activation Function	**Leaky ReLU**, and **Tanh**

- **Epoch:** an epoch refers to a single iteration through the entire training dataset. During each epoch, the generator and discriminator networks are updated based on the defined loss functions and optimization algorithms. The number of epochs is a hyperparameter that determines how many times the training algorithm will iterate over the dataset. Increasing the number of epochs allows the model to learn from the data more extensively, potentially improving the quality of generated samples. By training the model over 200 epochs and implementing some stopping criteria based on the absence of changes between epochs, we assess the stability and convergence of the model.

- **CMP-ATT:** We adopt a 4 Convolutional Message Passing Network (CMPN) with attention to both the generator(CMPN_G) and discriminator (CMPN_D), as depicted in Figure 3.6. This architecture facilitates information exchange and collaboration between layers, leveraging convolutional layers to capture spatial dependencies and enhance the quality of generated outputs.

- **Learning rates:** Selecting the optimal learning rate is crucial, requiring experimentation and fine-tuning. A high learning rate can cause instability, while a low rate leads to slow convergence. Through empirical testing, a learning rate of 0.0001 for both discriminator (d_lr) and generator (g_lr) was found to strike a balance, enabling efficient training and meaningful representation learning.

- **Convolutions:** Considering that the architecture conducts convolution subsequent to message passing, a kernel size of 3, stride of 1, and padding of 1 are employed. This choice is strategic as it ensures that the convolutional operation preserves spatial dimensions while efficiently capturing local features within the data. The kernel size of 3 allows the filter to encompass neighboring information, while the stride of 1 ensures a comprehensive exploration of the input space without downsampling. Furthermore, the padding of 1 maintains the spatial integrity of the input, facilitating seamless information flow throughout the network. Overall, these parameters contribute to an effective feature extraction process, enhancing the model's capacity to capture intricate patterns and representations within the data.

4.2 Training stability

Training instability is a well-known challenge in most deep learning models, especially Generative Adversarial Networks (GANs) [23]. Several factors contribute to this issue. Firstly, the GAN training process involves a delicate balance between the generator and discriminator networks [34]. As the generator learns to generate more realistic samples, it becomes increasingly difficult for the discriminator to distinguish between real and fake samples. This delicate equilibrium can easily be disrupted, leading to training instability. Secondly, GANs' sensitivity to hyperparameters like learning rate and batch size can lead to issues such as oscillating losses, vanishing or exploding gradients, and mode collapse, hindering stable convergence by disrupting the delicate balance between generator and discriminator [139]. Furthermore, the non-convex nature of the GAN objective function exacerbates training complexity as the generator and discriminator engage in an adversarial game, often resulting in oscillations and challenges in reaching a global optimum due to conflicting objectives [185].

During the GAN training process, the generator (G) and discriminator (D) models undergo multiple epochs and batches, with each batch consisting of a certain number of training samples. The loss values reported in the provided figure 4.1 indicate the performance and progress of our model during training.

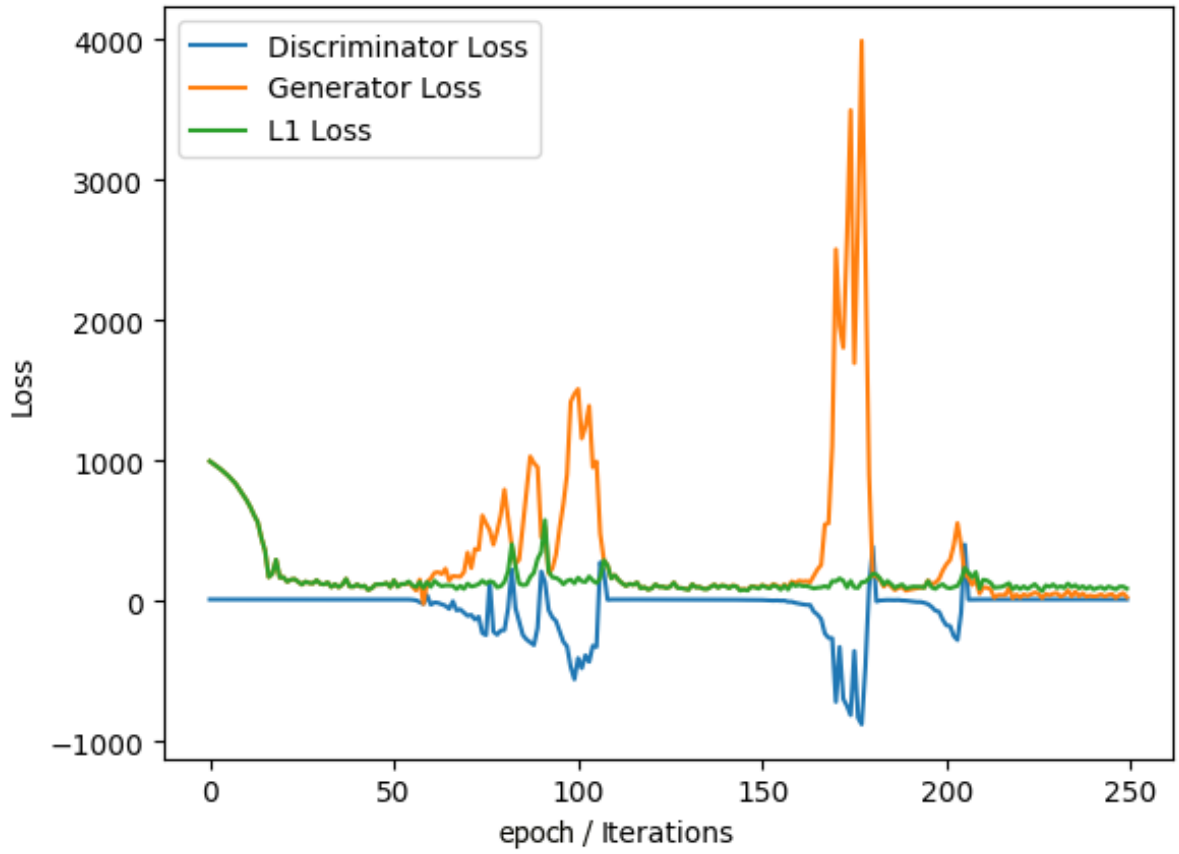

Figure 4.1: Adversarial training

As we know the primary objective of the discriminator is to maximize its loss, while the generator aims to minimize it [23]. This adversarial dynamic creates a competitive environment where both networks continuously improve their performance. As the training progresses, the discriminator becomes more skilled at differentiating between real and generated samples [23]. As shown in 4.1 the decreasing discriminator loss signifies that the discriminator is enhancing its ability to distinguish between the distributions of real and generated data. Consequently, this drives the model towards generating samples that closely resemble real data, leading to a more accurate and sophisticated generator [186].

Simultaneously, as the training advances, the generator's objective is to produce samples that the discriminator perceives as authentic. By minimizing the generator loss, the generator becomes progressively proficient in generating realistic samples that exhibit characteristics similar to real data [23]. The estimated training graph 4.1, provides insights into the generator's progress by demonstrating a decreasing trend in the generator loss over time. This decline in the generator loss indicates that the generator is continuously improving its capability to

generate samples that closely resemble real data, both in terms of overall quality and specific details [186]. As a result, the generator becomes increasingly adept at generating samples that are indistinguishable from real data, aligning with the ultimate goal of model training.

The L1 loss encourages the generator to generate samples that are visually similar to the real samples by penalizing large differences in pixel values [78]. It helps to ensure that the generated samples capture the overall structure and details present in the real data [76]. In the estimated training graph 4.1, we may see the L1 loss decreasing over time. This reduction indicates that the generator is progressively improving its ability to generate samples that closely resemble the real samples [78].

4.3 Resuls

The subsequent section is dedicated to discussing the results obtained from our research. In the context of our thesis, we have specifically selected the houseGAN++ [6] and HouseDiffusion [7] models as baselines due to several compelling reasons. Firstly, these models are directly relevant to our research objective, which revolves around generating diverse and realistic house related data. By choosing these baselines, we can assess the effectiveness and novelty of our proposed models in comparison to existing methods specifically designed for house-related tasks. Additionally, both represent state-of-the-art approaches in their respective domains. They have demonstrated impressive performance on benchmark datasets and tasks similar to ours, making them suitable for meaningful comparisons. Moreover, the availability of code, and, datasets for these baselines ensures reproducibility and allows to conduct of fair evaluations. Lastly, both models have been published in reputable conferences, indicating their impact and credibility within the field. By selecting houseGAN++ [6] and houseDiffusion [7] as baselines, we aim to showcase the advancements and contributions of our research in the context of existing cutting-edge techniques.

4.3.1 Diversity

Evaluating the performance of floorplan generation models requires considering the diversity of the generated floorplans. It is crucial to measure how varied and distinct the generated floorplans are. This assessment helps understand the model's ability to explore different design possibilities and create a wide range of viable floorplan layouts. By examining diversity, we can

ensure that the model does not produce repetitive or biased results and identify any limitations or biases in its training data or architecture. To measure this diversity, the FID (Fréchet Inception Distance) [166] score is commonly utilized, which compares the distribution of rasterized layout images generated by the model against a reference distribution.

Table 4.2: Diversity in the FID scores. (↓) indicate the lower the better metrics.

Model	Diversity (↓)			
	5	6	7	8
HouseGAN++ [6]	30.4±4.4	37.6±3.0	27.3±4.9	32.9±4.9
Exp1	20.3±3.2	22.4±1.7	18.2±1.4	18.7±2.2
HouseDiffusion [T=1000] [7]	11.2±0.2	10.3±0.2	10.4±0.4	9.5±0.1
Exp2 [T=20]	9.1±1.4	8.1±1.2	9.3±0.3	8.4±0.4

Note: The FID scores in the table reflect the level of diversity observed in the generated floorplans. Lower FID scores indicate a greater degree of diversity. It is worth noting that Exp1 utilizes only a GAN, while Exp2 incorporates both a GAN and a diffusion model. Additionally, the numerical labels 5, 6, 7, and 8 correspond to the number of rooms in each floorplan.

Table 4.2 shows the FID score of generated floorplans for comparative analysis. In order to enhance the diversity of generated floorplans, HouseGAN++ [6] utilizes the Wasserstein loss [140], a popular optimization technique in generative models. By incorporating this loss function and conditioning [76], HouseGAN++ [6] is able to produce a wider range of unique floorplan designs compared to their baseline HouseGAN [8]. As shown in Table 4.2 the FID scores for floorplans with 5, 6, 7, and 8 rooms are 30.4±4.4, 37.6±3.0, 27.3±4.9, and 32.9±4.9, respectively.These scores suggest that the floorplans produced by HouseGAN++ exhibit relatively higher FID scores, indicating a lower level of diversity compared to the other models [7]. However, when compared to its baseline model HouseGAN [8], HouseGAN++ [6] showcases remarkable improvements in diversity. This enhancement can be attributed to the specific techniques and architectural characteristics employed in HouseGAN++, such as the incorporation of the Wasserstein loss [140] and conditioning [76] mechanisms.

The Wasserstein loss [132] promotes a smoother optimization process, enabling HouseGAN++ to explore a broader solution space and produce floorplans with greater variation. Additionally, the conditioning mechanism, implemented through Conditional Generative Adversarial Networks (CGAN) [76], provides fine-grained control over the generated floorplans, allowing HouseGAN++ to adhere to specific constraints or exhibit desired architectural char-

acteristics. These techniques and architectural characteristics collectively contribute to the significant improvements in diversity observed in HouseGAN++ compared to its baseline.

Based on the motive of HouseGAN++ [6], our experiment 1 focuses on utilizing only a GAN [23] to generate floorplans. Inspired by the successful techniques employed in HouseGAN++, such as the Wasserstein loss [132] and conditioning [76], we aim to leverage these mechanisms to enhance the diversity of the generated floorplans. By incorporating the Wasserstein loss, we encourage a smoother optimization process that minimizes the discrepancy between the generated and real floorplans. This optimization technique allows for a wider exploration of the solution space and contributes to the generation of more diverse floorplan designs.

To further enhance the diversity in our experiment, we introduce additional regularization techniques. These techniques include minibatch discriminations [42], spectral normalization [41], and architectural modifications on convolutional message passing networks. Minibatch discriminations [42] enable the model to consider information from multiple samples simultaneously, promoting diversity by incorporating diversity-aware features in the generation process. Spectral normalization helps stabilize the training process and encourages the generator to produce a wider range of floorplan variations. Lastly, architectural modifications on convolutional message passing networks allow for more complex and intricate spatial relationships to be captured, leading to the generation of diverse and visually appealing floorplans.

While our Experiment 1 demonstrates an average improvement of 12.5 units compared to HouseGAN++, it still falls short of outperforming diffusion-based models. HouseDiffusion [T=1000] [7], for example, achieves lower FID scores of 11.2±0.2, 10.3±0.2, 10.4±0.4, and 9.5±0.1 for floorplans with 5, 6, 7, and 8 rooms, respectively. These scores are notably lower than both HouseGAN++ and Experiment 1, indicating that HouseDiffusion generates highly diverse floorplans. The results obtained in Experiment 1 demonstrate that the techniques and architectural modifications employed successfully contribute to improving the diversity of the generated floorplans compared to HouseGAN++. However, the performance of Experiment 1 still falls behind that of diffusion-based models like HouseDiffusion. The diffusion model, with its ability to capture long-range dependencies and explore a wider solution space, leads to even greater diversity in the generated floorplans [43]. The lower FID scores obtained by HouseDiffusion indicate a higher level of dissimilarity between the generated and real floorplans, reflecting the model's capability to produce a diverse range of unique floorplan designs.

From the above three results, we can observe a pattern: the incorporation of different regularization techniques and the utilization of a diffusion model to capture long-range dependencies significantly enhance the diversity of the generated floorplans. Building upon this pattern, in our main experiment (Exp2), we integrate the findings from Exp1, which includes five regularization techniques, along with a minimized denoising step (T=20) in the diffusion model. This leads to improved FID scores of 9.1±1.4, 8.1±1.2, 9.3±0.3, and 8.4±0.4 for floorplans with 5, 6, 7, and 8 rooms, respectively, compared to the HouseDiffusion model that employs 1000 denoising steps.

Unlike the Housediffusion model [7], which is trained on 1000 denoising steps, our Exp2 model utilizes 20 denoising steps. Due to the incorporation of minibatch discriminations [39] and attention mechanism [39] abilities, our experimental results, as illustrated in the result table, unequivocally demonstrate the remarkable superiority of our Exp2 model over the extensively acknowledged Housediffusion model [7]. Moreover, it is noteworthy that the synergistic integration of adversarial training with likelihood mechanisms significantly enhances the diversity of our output [33]. These findings highlight the effectiveness of our approach in capturing and representing the intricate data distribution, even with a significantly reduced number of denoising steps. Our Exp2 model successfully achieves a higher level of diversity and demonstrates its capability to generate floorplans that exhibit a wider range of architectural variations and design possibilities.

4.3.2 Sampling Time Dynamics Across Models

The field of floorplan generation has witnessed significant advancements with the emergence of powerful generative models and those models have revolutionized the way floorplans are created by offering the ability to generate diverse and visually appealing layouts. However, understanding the temporal dynamics of sample generation is a crucial aspect that demands attention. Exploring the sampling time dynamics across these generative models allows us to delve deeper into the evolution of generated floorplans over the number of samples. By examining the coherence of the samples at different stages of the sampling process, we gain valuable insights into the behavior and performance of these models.

Based on the results obtained from the comparison of sampling times per number of samples for four different generative models, as shown in figure 4.2, several observations can be made.

Firstly, it's evident that each model exhibits a unique sampling time pattern as the number of samples increases. For instance, the HouseGAN++ [6] model consistently demonstrates the shortest sampling times across all tested sample sizes. The shorter sampling times in HouseGAN++ [6] could be attributed to its utilization of single-shot noise injection [33], unlike diffusion models [24] that involve varying numbers of diffusion steps. This streamlined approach to noise injection potentially contributes to the faster generation process, as it involves fewer computational steps and iterations compared to the iterative nature of diffusion models.

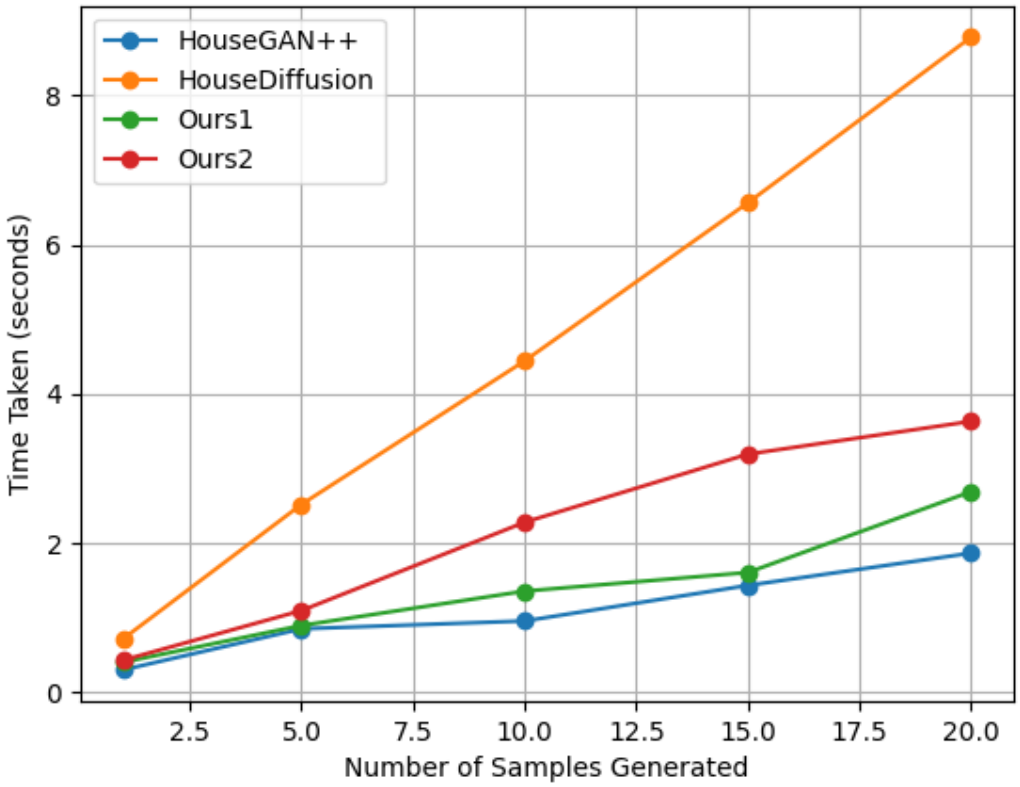

Figure 4.2: Time Taken vs Number of Samples Generated

Table 4.3: Tabular comparison for the time taken vs number of samples generated.

Samples Generated	HouseGAN++	HouseDiffusion	Ours1	Ours2
1	0.302 sec	0.717 sec	0.404 sec	0.432 sec
5	0.853 sec	2.521 sec	0.898 sec	1.098 sec
10	0.958 sec	4.445 sec	1.358 sec	2.282 sec
15	1.435 sec	6.564 sec	1.608 sec	3.194 sec
20	1.868 sec	8.784 sec	2.692 sec	3.634 sec

Hint: *Experiment 1 (Exp1) is referred to as Ours1, while Experiment 2 (Exp2) is denoted*

as Ours2.

In the context of the provided result, Exp1, which incorporates advanced techniques such as minibatch discrimination [42], spectral normalization [155], and attention mechanisms [117] within its GAN-based architecture, exhibits slower sampling times compared to HouseGAN++ [6] that does not include these techniques. This difference in sampling times can be attributed to the additional computational complexity introduced by the integration of these sophisticated techniques. Despite the slower sampling rates, Exp1 exhibits superior performance relative to other models, indicating the efficacy of minibatch discrimination and attention mechanisms in enhancing the diversity of generated floor plan images. This highlights the nuanced balance between sampling time and model performance, where the incorporation of advanced techniques can lead to improved overall efficiency and effectiveness in generating high-diversity floor plans, ultimately surpassing models with more straightforward architectures but faster sampling times.

By integrating Exp1's architecture as its core framework, Exp2 incorporates diffusion models with shorter diffusion steps, resulting in significantly reduced sampling times compared to the HouseDiffusion model [7]. Whereas HouseDiffusion [7] relies on large denoising steps and employs intricate mechanisms such as transformer-based architectures and both discrete and continuous denoising processes, contributing to its slower sampling times [43], Exp2 opts for smaller diffusion steps. This strategic adaptation enables Exp2 to streamline the generation process, prioritizing efficiency without compromising on the diversity of generated floor plan images. By leveraging Exp1's architecture and tailoring diffusion methods [49] to incorporate smaller steps, Exp2 demonstrates a nuanced approach to balancing sampling efficiency and diversity, ultimately presenting a compelling alternative to traditional diffusion models like HouseDiffusion [7].

Comparing the generation times of the different models, it's evident that HouseDiffusion [7] takes the longest time per sample at approximately 0.717 seconds. This is followed by Exp2 with a generation time of approximately 0.432 seconds per sample. Exp2 is notably faster than HouseDiffusion, showcasing an improvement in efficiency. However, it is slower than HouseGAN++ [6], which has the shortest generation time of approximately 0.302 seconds per sample. While "Exp2" demonstrates enhanced speed compared to "HouseDiffusion," it still falls short of the efficiency achieved by HouseGAN++ [6]. This indicates that although Exp2 presents a notable advancement in terms of generation time compared to HouseDiffusion [7], there is room

for further optimization to match the speed of GAN based models like HouseGAN++ [6].

Analyzing the performance of Exp2 in relation to both HouseDiffusion [7] and HouseGAN++ [6] provides valuable insights into its efficiency. Exp2 showcases a significant improvement over HouseDiffusion [7], with a generation time approximately 41.04 % faster and a competitive sampling time with houseGAN++ [6].

4.3.3 Realsim of Generated floorpans

Figure 4.3 illustrates the sample of floorplan outputs of three models, HouseGAN++ [6], HouseDiffusion [7], and our custom model, which incorporates the bubble diagram as a constraint. The bubble diagram serves as an input modality that guides the generation of floorplans by specifying node and edge connectivity.

To conduct a comprehensive evaluation of the generated floorplan from a realism perspective, a collaborative approach was adopted with professional advisors, specifically architects. To begin the process, the initial step entails crafting a questionnaire specifically designed to assess the realism of generated floorplans. This questionnaire serves as a tool for evaluating the authenticity and believability of the floorplan outputs. To ensure a comprehensive evaluation, two distinct architectures are employed to independently develop their own sets of questions within the questionnaire. This approach allows for diverse perspectives and insights from each architecture. To validate the effectiveness and reliability of the generated questions, the questionnaires are subsequently evaluated by a panel of three experienced architects. This collective assessment further enhances the robustness and accuracy of the evaluation process.

The questionnaires encompass two perspectives: visual realism, which measures the visual appeal and authenticity of the generated floorplans, and functional realism, which evaluates how well the floorplans fulfill functional constraints. Each question is thoughtfully labeled, presenting respondents with a range of choices, including -1 for worse, 0 for moderate, and 1 for better, aligning with the intended criteria. To ensure a thorough and diverse evaluation, we have enlisted the participation of nine architecture students, whose unique perspectives offer valuable insights. Additionally, we have sought input from three civil and architecture university teachers who possess expertise and knowledge in relevant fields, thereby leveraging their valuable contributions. Furthermore, we have actively engaged three professional architects currently working in the industry, whose practical experience adds substantial depth and real-

world context to the evaluation process.

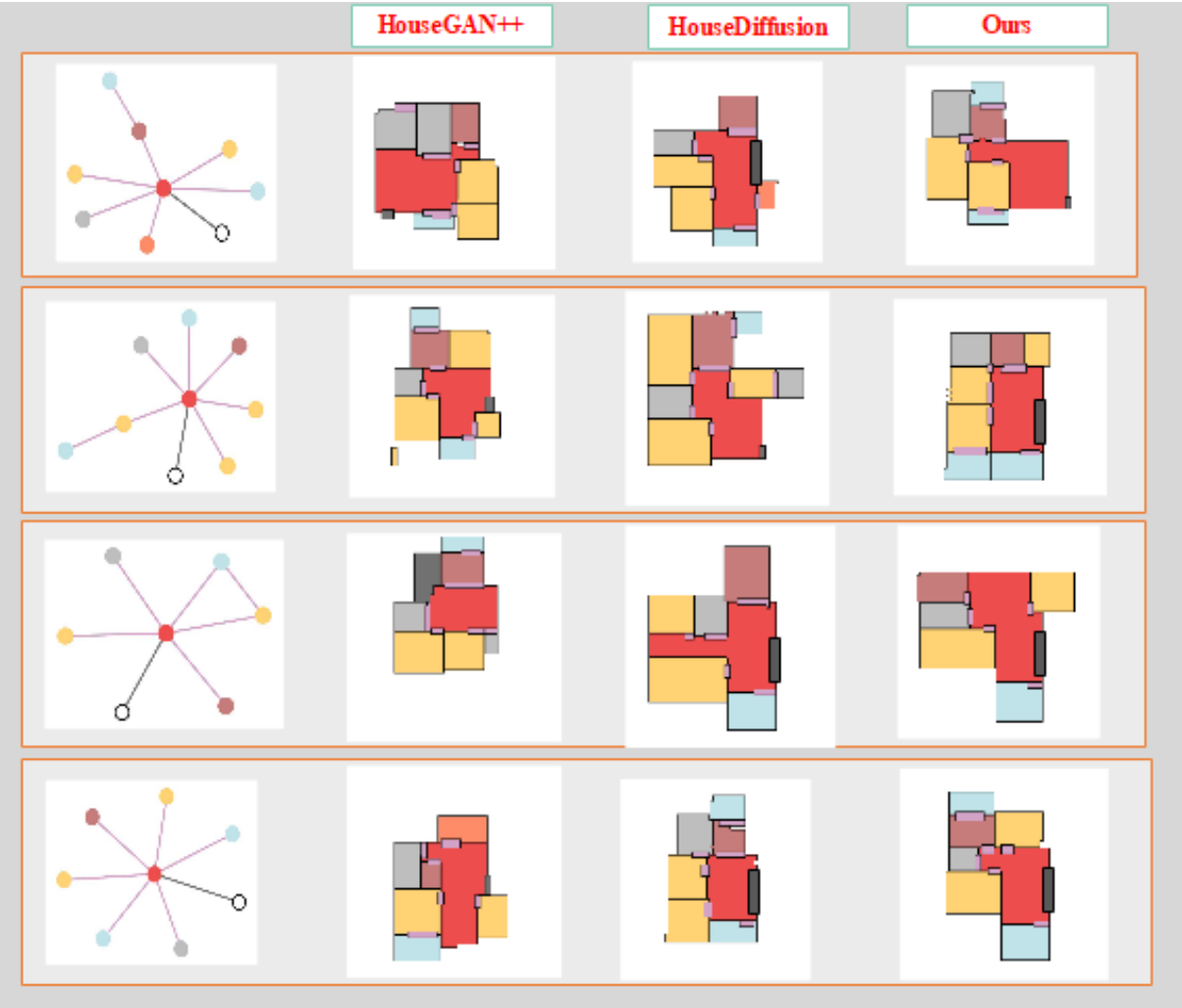

Figure 4.3: Generated floorplans for three models with the same bubble diagram as input constraint.

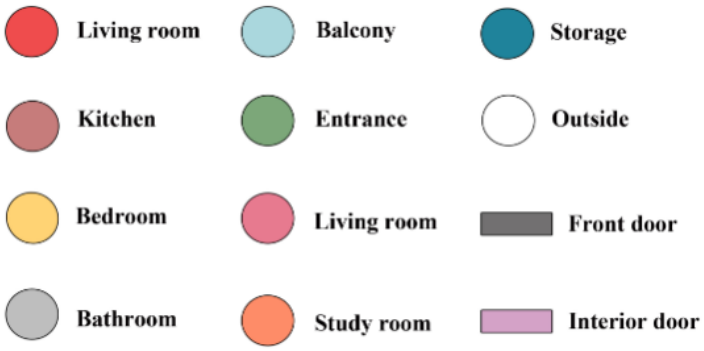

Figure 4.4: Color coding for each room.

The table 4.4 presents the realism scores of three methods (HouseGAN++ [6], HouseDiffusion [7], and Exp2) based on six specific questions. To calculate the scores, the average rating

Table 4.4: Realism scores among House-GAN++, HouseDiffusion, and ours.

	Questions	HouseGAN-++[6]	HouseDiffusion [7]	Exp2
Visual Realism	How well do the shapes of the rooms contribute to a visually appealing and cohesive design?	0.06	0.2	0.2
	Does the method address visual problems like disjointed lines, unconnected rooms, and other relevant factors to make the design look real and fit together well?	0	-0.06	0.13
Functional Realism	Does the placement of key features, such as bathrooms, kitchens, entrances and others, align with common architectural principles?	0.26	0.2	0.33
	Are the necessary functional spaces, such as bedrooms, bathrooms, kitchens, living areas, and others appropriately sized?	0.06	0	-0.13
	How effectively do the spatial relationships between rooms contribute to the functionality and coherence of the floorplan?	0.06	0.26	0.26
	Is there efficient flow between rooms, allowing for easy movement throughout the space?	-0.06	0.26	0.33
Avg	**Total Realism Results**	**0.063**	**0.143**	**0.186**

given by all respondents for each question is determined. This process ensures that the results reflect the collective perspective of the evaluators across all questions [8]. By employing this approach, we obtain a comprehensive assessment of the realism of the three methods based on the specific criteria outlined in the questions.

Upon analyzing the table, it becomes evident that HouseDiffusion and Exp2 yield comparable results in terms of their ability to generate visually appealing room shapes. However, in terms of addressing visual problems such as disjointed lines and unconnected rooms, Exp2 surpasses HouseDiffusion. This superiority can be attributed to our attention mechanism [117], which plays a crucial role in achieving a realistic and cohesive design. Despite this advantage, HouseDiffusion may still have drawbacks in other relevant aspects that contribute to the overall realism of the design. On the other hand, Exp2 demonstrates a strong performance in visual realism, showcasing its ability to create room shapes that visually fit together well. While it may have room for improvement in addressing visual issues, Exp2 presents a commendable level of visual realism when compared to the other two methods.

Moving on to functional realism, the questions evaluate different aspects related to the functionality and coherence of the floorplan. They consider factors such as the placement of key features (e.g., bathrooms, kitchens, entrances) aligning with architectural principles, appropriate sizing of functional spaces (e.g., bedrooms, bathrooms, kitchens, living areas), effectiveness of spatial relationships between rooms, and the presence of efficient flow for easy movement throughout the space.

The placement of key features is an essential aspect of functional realism. It involves assessing whether features such as bathrooms, kitchens, entrances, and others align with common architectural principles. Exp2 demonstrated the highest score in this category, indicating that it excelled in placing these key features in a manner that adheres to architectural guidelines. HouseGAN++ and HouseDiffusion also obtained relatively high scores, suggesting a reasonable alignment with architectural principles, although not as strong as Exp2.

Appropriately sized functional spaces are another important consideration in functional realism. It involves evaluating whether bedrooms, bathrooms, kitchens, living areas, and other functional spaces are sized adequately. HouseDiffusion achieved the highest score in this aspect, indicating that it successfully addressed the sizing of functional spaces. HouseGAN++, on the other hand, received a lower score, suggesting a potential mismatch in the sizing of these spaces. Exp2 also scored lower in this category, indicating room for improvement in terms of appropriately sizing functional spaces.

The effectiveness of spatial relationships between rooms is a key factor in determining the functionality and coherence of a floorplan. Exp2 demonstrated the highest score in this regard, implying that its spatial relationships between rooms effectively contribute to the overall functionality of the design. HouseDiffusion also performed relatively well, indicating that its room arrangements are effective in promoting functionality. However, HouseGAN++ received the lowest score, suggesting that its spatial relationships between rooms may need improvement to enhance functional realism.

In conclusion, analyzing the functional realism perspective based on the provided table, Exp2 displayed strengths in the placement of key features and the effectiveness of spatial relationships between rooms. It excelled in aligning with architectural principles and creating functional spatial arrangements. However, there is room for improvement in appropriately sizing functional spaces. HouseDiffusion [7] showcased good performance in the placement of key

features and appropriately sizing functional spaces but lagged behind in the effectiveness of spatial relationships. HouseGAN++ [6] demonstrated potential for enhancement in all aspects of functional realism. These findings highlight the importance of considering visual and functional realism in floorplan design and suggest directions for further refinement and development of these methods.

4.4 Statistical Significance

Statistical significance indeed plays a pivotal role, not only in assessing model performance but also in gauging the attainment of research objectives. Our research aim is to enhance the diversity of the generated floorplan while optimizing or balancing the sampling time and realism. Through rigorous statistical method T-test we can comprehensively assess the effectiveness of our approaches. These analyses allow for a quantitative evaluation of various factors, such as diversity, sampling time, and realism.

4.4.1 Diversity

The study investigated the diversity of generated floorplans by comparing the Fréchet Inception Distance (FID) scores between a baseline mode houseDiffusion model and the generated floorplans. An independent samples t-test was conducted to statistically evaluate the significance of the difference in FID scores. The resulting t-statistic was 3.72, with a corresponding p-value of 0.034. Considering a significance level of $\alpha = 0.05$, the p-value was found to be less than the significance level ($p < \alpha$). These results indicate that the difference in FID scores between the baseline and the generated floorplans is statistically significant. This suggests that the proposed floorplan generation approach has produced outputs that are significantly more diverse compared to the baseline, as measured by the FID metric. The statistical significance of the FID score difference provides confidence that the observed improvement in diversity is not due to chance, but rather reflects a genuine and measurable enhancement in the ability of the proposed method to generate diverse floorplans. This information is valuable when evaluating the efficacy of the floorplan generation technique in preserving and enhancing the diversity of the generated outputs.

4.4.2 Sampling time

The study also examined the sampling time performance of the floorplan generation models. A statistical significance test, specifically a t-test, was conducted to compare the sampling times. The results showed a statistically significant difference (p=0.04) in the sampling time between the proposed model and the House Diffusion model, indicating that the proposed model had a significantly faster sampling time compared to the House Diffusion model. However, the analysis revealed no statistically significant difference in sampling time between the proposed model and the HouseGAN++ model. From the study's objective perspective, which aims to optimize or balance the sampling time performance, these findings suggest that the proposed model achieves a desirable balance, performing on par with the faster HouseGAN++ model, while still maintaining the statistically significant diversity improvements observed through the FID score analysis. This balanced sampling time characteristic is a valuable attribute, as it allows the proposed model to generate diverse floorplans efficiently, addressing the practical considerations of computational cost and real-world deployment.

4.4.3 Realism

From the perspective of realism, the study performed a t-test to compare the generated floorplans across the two models. The results showed no statistically significant difference in the realism of the generated floorplans between all the models. However, from the study's objective perspective, which aims to maintain the realism of the generated floorplans while also enhancing the diversity, the proposed model was able to achieve this balance. Despite not showing a significant improvement in realism over the other models, the proposed model was able to generate diverse floorplans without compromising their realistic attributes. This balanced approach is a valuable contribution, as it demonstrates the ability to enhance diversity without sacrificing the practical viability and real-world applicability of the generated floorplans. By maintaining realism, the proposed model ensures that the diverse floorplan designs remain grounded in practical constraints and user preferences, making them more suitable for real-world architectural design and space planning applications.

From all the discussions above, it becomes evident that statistical significance serves as a crucial tool in assessing various aspects of floorplan generation. The statistical significance observed in diversity indicates an enhanced variation, signifying the effectiveness of our efforts

in diversifying floorplans. Additionally, the comparable statistical significance found in other aspects, such as realism and sampling time, suggests optimization and balance in these critical factors. Together, these findings highlight the multifaceted nature of our research approach, which aims not only to enhance diversity but also to optimize sampling time by maintaining realism. By leveraging statistical significance as a guiding metric, we gain valuable insights into the performance of our models and their alignment with our research objectives.

4.5 Ablation Studies

4.5.1 Number of denoising steps

In this study, we investigated the effect of the number of denoising steps on floorplan generation. The number of denoising steps plays a crucial role in determining the quality and diversity of the generated floorplans [33]. To explore this, we varied the diffusion steps and evaluated the results. An important aspect to consider is the impact of diffusion steps on the sampling time during inference. As we know, in conventional diffusion models [49], the diversity of the generated floorplan increases as the number of denoising steps increases [49]. Additionally, the time required for sampling also increases. Therefore, it becomes essential to strike a balance between denoising steps and the desired level of diversity in the generated floorplans.

Table 4.5: Effect of diffusion steps in diversity. Diversity is measured by the FID scores. (↓) indicate the-lower the-better metrics.

Steps	Diversity (↓)			
	5	6	7	8
T = 1	21.7±2.6	27.1±2.1	25.2±1.4	25.6±2.3
T = 2	19.2±0.5	23.4±0.6	20.3±1.2	24.1±21.1
T = 4	17.6± 1.2	20.3±1.4	18.9±2.1	22.8±1.4
T = 10	15.9±2.1	18.8±0.8	16.6±1.6	17.3±0.3
T = 20	**9.1±1.4**	**8.1±1.2**	**9.3±0.3**	**8.4±0.4**
T = 100	15.4±1.3	19.5±0.6	18.7±0.9	16.2±2.2

In our study, we varied the number of diffusion steps from T=1 to T=100 and analyzed the resulting floorplan outputs. Upon analyzing the results, several key observations can be made. Initially, when comparing the FID scores at T=1, T=2, T=4, and T=10, we notice a gradual decrease in diversity as the number of diffusion steps increases. This suggests that a limited number of diffusion steps may not be sufficient to introduce significant diversity into

the generated outputs. It is worth noting that T = 1 corresponds to training an unconditional GAN [23], where the conditioning information (xt) contains minimal details about the initial state (x0). Our findings revealed that utilizing T = 1 resulted in significantly poorer outcomes, characterized by low sample diversity as indicated by the high FID score. This confirms the advantages of dividing the generation process into multiple denoising steps, particularly in enhancing sample diversity.

However, as we move to T=20, a notable shift occurs, with the FID scores experiencing a sharp decline. This indicates that a higher number of diffusion steps (in this case, 20) leads to a substantial improvement in diversity. The FID scores at T=20 (9.1±1.4, 8.1±1.2, 9.3±0.3, and 8.4±0.4) suggest a considerable increase in the variety of generated outputs across different categories. This finding suggests that an increase in diffusion steps in a multimodal way positively impacts the realism and diversity of the generated floorplans.

From a technical standpoint, the longer diffusion process with a higher number of denoising steps contributes to improved performance by allowing for a more thorough exploration of the solution space [43]. The diffusion process helps to gradually refine and enhance the generated floorplans, reducing noise and inconsistencies in the layouts. By increasing the number of diffusion steps, the model has more opportunities for interaction and adjustment, leading to higher-diversity and more realistic floorplan outputs. However, as shown in Table 4.5, there was a slight decline in performance for larger T values (T=100) compared to T=20. We hypothesize that accommodating larger T values may require a significantly higher model capacity due to the need for a conditional GAN for each denoising step. The increased complexity associated with a larger number of denoising steps might pose challenges in maintaining optimal performance, potentially leading to a degradation in the diversity of the generated floorplans.

The results of this study have important implications for the application of diffusion-based models in generating diverse outputs. By understanding the relationship between diffusion steps and diversity, researchers can make informed decisions when designing models for specific tasks. Optimizing the number of diffusion steps based on the desired level of diversity is crucial to strike a balance between computational efficiency and output quality. The findings suggest that an excessive number of diffusion steps may not necessarily lead to a proportional increase in diversity. Therefore, researchers should carefully consider the trade-off between computational resources and the diversity requirements of their specific application.

4.5.2 Effect of Used Techniques

Table 4.6: Effect of Different Techniques on Diversity at T=20.

Name	AM	MDB	SN	Loss	Diversity (↓)			
					5	6	7	8
Expt1	✓	-	-	WGAN-GP	33.6±2.4	36.4±1.7	34.5±1.9	38.6±2.1
Expt2	-	✓	-	WGAN-GP	31.4±1.3	33.5±1.6	24.4±2.2	30.2±1.2
Expt3	-	-	✓	WGAN-GP	37.3±1.3	40.5±2.1	44.3±2.1	43.4±1.9
Expt4	✓	✓	-	WGAN-GP	17.6±1.1	20.1±1.3	20.6±2.2	17.2±1.8
Expt5	✓	-	✓	WGAN-GP	26.1±1.1	21.2±1.8	23.7±1.6	20.4±1.3
Expt6	✓	✓	✓	WGAN-GP	**9.1±1.4**	**8.1±1.2**	**9.3±0.3**	**8.4±0.4**
Expt7	✓	-	-	Hinge	35.6±2.1	39.6±2.5	30.3±3.3	36.7±1.3
Expt8	-	✓	-	Hinge	33.6±1.4	36.1±2.3	31.3±3.2	34.6±1.7
Expt9	-	-	✓	Hinge	40.3±2.1	43.5±3.2	38.4±2.5	45.4±1.8
Expt10	✓	✓	-	Hinge	30.5±2.2	28.4±3.1	28.7±1.2	31±2.6
Expt11	✓	-	✓	Hinge	50.5±3.3	54.7±2.6	61.2±3.4	55.5±2.1
Expt12	✓	✓	✓	Hinge	26.5±2.1	21.6±2.9	20.1±1.7	21.2±2.3

Hint: SN: Spectral Norm; MBD: Minibatch Std Dev AM: Attention Mechanism. The lower the value, the higher the diversity, which is measured in FID score.

The experimental results presented in Table 4.6 offer valuable insights into the influence of different techniques, such as Wasserstein [132] and hinge loss [135] with attention mechanisms [39], minibatch discriminations [42], and spectral normalization [41], on the diversity of the generated floorplans. The table showcases the presence or absence of these properties, indicated by a checkmark (✓) or a dash (-), respectively. By performing nearly 12 experiments (Expt1-12) at T=20, we were able to analyze the impact of incorporating these techniques on the diversity of the generated floorplans. These results shed light on the effectiveness of each technique and provide guidance for selecting the appropriate combination of techniques to enhance the diversity of the generated floorplans in our approach. Analyzing the results, we observe the following trends:

Impact of Individual Techniques:

- In Experiment 1 (Expt1), the Attention Mechanism (AM) was utilized without Minibatch Std Dev (MDB) or Spectral Norm (SN), with WGAN-GP as the loss function. The diversity scores for Expt1 are relatively high compared to other configurations, indicating lower diversity in the generated outputs. This suggests that using AM alone with WGAN-GP

does not significantly enhance diversity. The moderate performance could be attributed to the inability of the Attention Mechanism alone to capture sufficient variability in the data. The absence of MDB and SN might be limiting the model's capacity to generalize and introduce diverse features effectively. This highlights the need for additional techniques alongside AM to improve diversity, and incorporating other methods like MDB or SN might help in capturing more diverse features.

- In Experiment 2 (Expt2), Minibatch Std Dev (MDB) was incorporated without the Attention Mechanism (AM) or Spectral Norm (SN), using WGAN-GP as the loss function. The diversity scores show an improvement compared to Expt1. The lower scores suggest higher diversity. MDB appears to contribute positively to diversity; by reducing batch variability, MDB allows the model to better capture subtle variations within the data, leading to more diverse outputs. These findings support the use of MDB in enhancing model diversity, especially when used in conjunction with effective loss functions like WGAN-GP.

- In Experiment 3 (Expt3), Spectral Norm (SN) was used without the Attention Mechanism (AM) or Minibatch Std Dev (MDB), with WGAN-GP as the loss function. The FID scores for Expt3 were the highest among the first three experiments, ranging from 37.3 ± 1.3 to 44.3 ± 2.1, indicating the lowest diversity. The application of SN alone does not seem to significantly enhance diversity. The high scores suggest that SN might be overly regularizing the model, thus reducing variability in the generated outputs. These findings imply that while SN is useful for regularization, it should be combined with other techniques to effectively improve diversity.

Impact of Combined techniques:

- In Experiment 4 (Expt4), both the Attention Mechanism (AM) and Minibatch Std Dev (MDB) were utilized without Spectral Norm (SN), employing WGAN-GP as the loss function. The diversity scores significantly decreased indicating a marked improvement in diversity compared to previous experiments. The combination of AM and MDB proved to be effective in enhancing diversity, where AM focused on relevant features while MDB ensured variability across batches, collectively promoting diverse outputs. These findings underscore the efficacy of combining AM and MDB for applications requiring high di-

versity, as it effectively leverages the strengths of both techniques. This highlights the importance of thoughtful technique selection and integration to achieve optimal outcomes in model performance.

- In Experiment 5 (Expt5), both the Attention Mechanism (AM) and Spectral Norm (SN) were employed without Minibatch Std Dev (MDB), using WGAN-GP as the loss function. The diversity scores ranged from 20.4±1.3 to 26.1±1.1, representing an improvement compared to experiments using AM or SN alone but not reaching the level achieved when combined with MDB (Expt4). The use of AM and SN together provides a balanced approach that enhances diversity, albeit not optimally. While this combination helps capture relevant features and regularizes the model, it might miss out on some variability aspects covered by MDB. Therefore, while the combination of AM and SN is beneficial, incorporating MDB could further enhance diversity and should be considered for future experiments aiming to maximize model performance.

- In Experiment 6 (Expt6), incorporating all three techniques - Attention Mechanism (AM), Minibatch Std Dev (MDB), and Spectral Norm (SN) with WGAN-GP as the loss function, resulted in the lowest diversity scores, ranging from 8.1±1.2 to 9.3±0.3, indicating the highest diversity among all experiments. The synergistic effect of combining AM, MDB, and SN with WGAN-GP is evident, as each technique contributes uniquely to maximizing diversity. AM focuses on capturing relevant features, MDB maintains batch variability, and SN ensures robust regularization. This comprehensive approach effectively enhances diversity, making it the most effective for applications requiring high diversity. Therefore, incorporating AM, MDB, and SN with WGAN-GP should be considered a best practice for similar scenarios, emphasizing the importance of integrating multiple techniques to achieve optimal results in model performance.

- Experiments 7 to 12 offer valuable insights into the impact of different combinations of techniques on diversity in generated outputs. Experiment 7 employs the Attention Mechanism (AM) alone with Hinge loss, resulting in moderate diversity scores ranging from 30.3±3.3 to 39.6±2.5. This indicates that while AM contributes to diversity, its effectiveness is limited without additional techniques such as Minibatch Std Dev (MDB) or Spectral Norm (SN). Experiment 8 combines MDB with Hinge loss, resulting in slightly

improved diversity scores ranging from 31.3±3.2 to 36.1±2.3, suggesting that MDB enhances diversity, albeit modestly.

- Experiment 9 introduces SN alone with Hinge loss, yielding high diversity scores ranging from 38.4±2.5 to 45.4±1.8. However, the high scores indicate potential over-regularization, emphasizing the need for complementary techniques to balance SN's effect. Experiment 10 combines AM and MDB with Hinge loss, resulting in improved but suboptimal diversity scores ranging from 28.4±3.1 to 31±2.6, suggesting that the choice of loss function plays a crucial role in diversity enhancement. Experiments 11 and 12 explore various combinations of AM, MDB, and SN with Hinge loss, with Experiment 12 achieving the lowest diversity scores ranging from 20.1±1.7 to 26.5±2.1. This highlights the importance of selecting an appropriate combination of techniques and loss functions to maximize diversity in generated outputs.

Impact of Loss Functions:

The choice of loss function has a significant impact on the diversity of generated samples, as evidenced by the comparison between WGAN-GP and Hinge loss functions. Experiments 1 through 6, which utilized WGAN-GP, generally exhibited lower FID scores, indicating higher diversity compared to their counterparts that employed Hinge loss. Notably, Experiment 6 achieved the best diversity results with FID scores ranging from 8.1 to 9.3, showcasing the effectiveness of WGAN-GP in promoting diverse sample generation. WGAN-GP's success can be attributed to its ability to enforce Lipschitz continuity through gradient penalty, thereby stabilizing the training process and encouraging the model to generate more diverse outputs.

In contrast, experiments employing the Hinge loss function (Experiments 7 through 12) generally yielded higher FID scores, indicating lower diversity compared to WGAN-GP experiments. For instance, Experiment 11, which utilized Hinge loss, AM, and SN, had the highest FID scores ranging from 50.5 to 61.2, reflecting the lowest diversity. The reduced performance of Hinge loss compared to WGAN-GP can be attributed to its inherent limitations in encouraging diverse sample generation. Hinge loss focuses primarily on maximizing the margin between different classes, which may lead to overly restrictive optimization and limited exploration of the sample space. Consequently, while Hinge loss may excel in certain tasks such as classification, it may not be as effective as WGAN-GP in promoting diversity in sample generation tasks.

These findings highlight the significance of selecting and combining appropriate techniques

to enhance the diversity of generated floorplans. It showcases the importance of selecting the right combination of techniques and loss functions to maximize diversity. The results highlight the effectiveness of incorporating attention mechanisms, spectral normalization, and minibatch discrimination with Wasserstein loss function in producing diverse floorplans.

4.6 Discussion

In our research, we aimed to strike a balance between sampling time and diversity while maintaining realism in the generation of floorplans. To achieve this, we employed the strengths of two models, namely GAN (Generative Adversarial Network) [23] and diffusion models [24], and explored different techniques. Our study consisted of three main experiments, each of which contained several sub-experiments.

In the first experiment, we investigated the effects of various techniques on the GAN [23] model alone. Specifically, we incorporated attention mechanism [117], minibatch discrimination [42], spectral normalization [155], and two different loss functions [132, 135]. Through this experiment, we discovered that the combination of the Wasserstein loss function with the three aforementioned techniques significantly enhanced the diversity of our generated floorplans, surpassing the baseline model HouseGAN++ [6]. In this experiment, our principal aim was to enhance the diversity while disregarding sampling time considerations.

Building upon the results of the first experiment, we designed a second experiment by integrating the GAN [23] and diffusion models [24]. We evaluated the impact of denoising time steps on the generation process. Our aim here was to reduce the sampling time while still improving diversity. We observed that employing unimodal time steps (using only one denoising step) did not significantly enhance diversity, and it even reduced the sampling efficiency. This unimodal approach resembled the direct GAN method but with differences in the posterior sampling and noise distribution used compared to conventional GAN-based methods. However, when we experimented with different time steps in a multimodal manner, we found that increasing the number of time steps positively affected diversity but also had an impact on the sampling time. After experimenting with various time steps, we determined that an optimal balance between sampling time and diversity was achieved at T=20.

Finally, we conducted a comparative analysis on the number of sampling time steps with the number of generated floorplans. We observed that GAN-based models like HouseGAN++

[6] and our proposed model Exp1 exhibited a proportional relationship between the number of floorplans generated and the required time for sampling. On the other hand, diffusion-based models like HouseDiffusion [7] and Exp2 demonstrated a non-proportional increase in sampling time as the number of samples grew (shown in figure 4.2). Overall, our experiments showed that by leveraging the advantages of GAN and diffusion models and exploring techniques such as attention mechanism, minibatch discrimination, spectral normalization, and the Wasserstein loss function, we were able to strike a balance between sampling time and diversity in floorplan generation tasks. Compared to our baseline model, HouseGAN++ [6], we achieved superior diversity with competitive sampling time, outperforming HouseDiffusion [7] in both diversity and sampling time.

Based on these observations, we can confidently answer the research questions posed in this study. In our research, we have addressed the following key research questions to overcome the challenges of sampling time and diversity in generative models:

- **How do various regularization techniques affect the diversity of floorplan layouts generated?**

By incorporating techniques such as minibatch discrimination, spectral normalization, and the Wasserstein loss function, the models achieve enhanced diversity in the generated outputs. Architectural modifications, specifically the integration of a convolutional message passing network with an attention mechanism, further contribute to enhancing the diversity of the generated floorplan. The experiments conducted on the ablation study (section 4.5.2) demonstrate that these techniques and modifications play a significant role in capturing important features, encouraging discrimination at the batch level, stabilizing training, and generating various outputs that closely resemble the target distribution. Overall, the findings highlight the effectiveness of these techniques and modifications in influencing the diversity of floorplans produced by our model.

- **How can we achieve optimized sampling time and enhanced diversity through the combination of two generative models?**

To leverage the advantages of capturing large data distributions in diffusion models, our research proposes an approach that integrates a GAN in a multimodal fashion. By combining

the strengths of both models, we overcome the limitations of individual GAN generation and achieve a more comprehensive and efficient generative framework. In our proposed approach, the multimodality of the model plays a crucial role. This multimodal training strategy allows us to reduce the denoising steps and optimize the overall sampling efficiency. By incorporating the diffusion model, we capture long-range distributions and intricate patterns that are challenging for individual GAN models. This integration enables us to generate diverse samples while maintaining efficient sampling time. The combination of the diffusion model and GAN in a multimodal fashion holds great potential for achieving optimized sampling time and enhanced diversity.

4.6.1 Key finidngs

- The integration of self-attention mechanisms into Convolutional Message Passing Networks (MPNs) emerges as a crucial breakthrough, enabling dynamic aggregation of information across entire graphs. This adaptive weight assignment enhances MPNs' capability to capture explicit and nuanced dependencies, fostering heightened expressiveness and realism in graph analysis.

- Among the different techniques investigated, minibatch discrimination and the Wasserstein loss function have a significant impact on enhancing diversity in the generated outputs. The inclusion of minibatch discrimination promotes diversity by considering relationships between samples within a minibatch. This technique encourages discrimination at the batch level, leading to a broader range of generated samples. Similarly, the adoption of the Wasserstein loss function encourages the generative model to closely match the target distribution, resulting in more diverse and realistic samples. These findings highlight the importance of incorporating minibatch discrimination and the Wasserstein loss function in generative models to achieve higher levels of diversity in the generated outputs.

- Another key finding of the research is that the number of time steps, such as increasing the denoising time, does not always guarantee a high level of diversity in the generated outputs. The research demonstrates that simply increasing the denoising time or the number of time steps does not automatically lead to a proportional increase in diversity. Instead, the study highlights the significance of selecting optimal parameters to achieve

high diversity in generative models.

- Another pivotal discovery of this research lies in the strategic amalgamation of GANs with Diffusion models, a synthesis that unveils a harmonious balance between two critical facets of generative modeling: sampling time and diversity. This fusion capitalizes on the distinctive strengths of each model archetype. GANs, renowned for their expeditious sample generation capabilities, empower the synthesis of outputs at a remarkable pace, owing to their discriminative and generative prowess. Conversely, Diffusion models offer a unique methodology for enriching diversity by traversing a broader distribution space, thereby facilitating the exploration of novel and varied samples. By seamlessly integrating these models, we orchestrate a harmonious synergy wherein the efficiency of GANs harmonizes with the diversity-enhancing capabilities of Diffusion models. This symbiotic alliance not only optimizes sample generation but also enriches the generative process with a diverse array of outputs, thereby exemplifying the transformative potential of hybrid generative frameworks in advancing the frontier of sample synthesis.

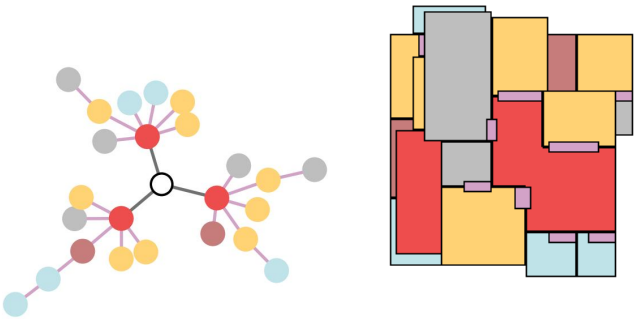

Figure 4.5: Input the complex graph along with its corresponding generated floorplan.

4.6.2 Limitations

- One limitation of our research is the difficulty our model encounters in maintaining the realism of generated complex floorplans. Although our model demonstrates satisfactory performance in simpler and moderate floorplan generation tasks, it faces challenges when tasked with more intricate architectural designs, as shown in figure 4.5. This limitation, in our view, arises due to a deficiency in complex training data compared to simpler designs, as well as the limitations of the model's capabilities. Complex floorplans require

a higher level of detail and intricacy, which may not be adequately represented in the available training data. As a result, the model may struggle to accurately capture and reproduce the intricate features of complex floorplans, resulting in a loss of realism in the generated outputs.

- Another limitation of our research is the inability to generate non-Manhattan floor plans and the lack of room size information in the node property. Our current model is constrained to strictly producing floor plans that adhere to Manhattan-style layouts, characterized by orthogonal lines and right angles. This limitation restricts the diversity of floor plan designs that our model can generate, as it excludes curved or diagonal elements commonly found in non-Manhattan layouts. Addressing this limitation could involve exploring alternative modeling approaches or incorporating additional data sources to enable the generation of more varied floor plan styles.

Chapter 5

Conclusion and recommendation

5.1 Conclusion

Generative models have brought about a significant revolution in various industries, including floorplan generation. Despite the advancements made in this field using techniques such as GANs, diffusion models, and others, specific challenges such as mode collapse, training instability, and sampling time challenges have persisted. Although several methods have been proposed to address these challenges, they often fall short of achieving the desired effectiveness.

In this research, we have presented a novel hybrid approach to overcome the obstacles of diversity and sampling time in floorplan generation. Our proposed method, which combines the strengths of GANs and diffusion models, incorporates regularization techniques, training algorithms, and attention mechanisms. Notably, our work introduces a solution that not only balances sampling time and diversity but also enhances the realism of the generated floorplans.

To validate our approach, we trained our model on the RPLAN dataset and conducted experiments with different time steps. We found that a denoising step at T=20 achieved better diversity results. The diversity of the generated floorplans was evaluated using FID across the number of rooms, and our model demonstrated a higher variety compared to the two baselines we employed. Furthermore, our hybrid approach reduced the time required for generation by 41.04 % when compared to the housediffusion model.

These findings highlight the effectiveness and superiority of our proposed method in addressing the challenges of floorplan generation. By combining multiple techniques and leveraging the advantages of different generative models, we have achieved significant advancements in terms of diversity, sampling time, and realism. However, it is important to acknowledge that further research and experimentation are required to explore additional avenues for improvement. For instance, investigating alternative architectural modifications and exploring new regularization methods could potentially enhance the performance of our approach even further.

In future research, our aim is to expand the capabilities of floorplan generation by incorpo-

rating non-Manhattan layouts and complex furniture placement. Currently, our model is limited to generating floorplans adhering strictly to Manhattan-style layouts, characterized by orthogonal lines and right angles. By addressing this limitation, we envision enabling our model to generate floorplans with curved or diagonal elements commonly found in non-Manhattan layouts. Additionally, we aim to incorporate furniture placement in the generation process, allowing for more realistic and detailed floorplan designs. By pursuing these advancements, we strive to enhance the versatility and creativity of our floorplan generation approach, catering to a broader range of architectural styles and design requirements.

5.2 Recommendation

- **Complex floorplans:** Based on the limitation of maintaining the realism of generated complex floorplans, there are several recommendations for future research. Firstly, refining the model architecture should be a priority to better handle the intricacies of complex designs. This can involve exploring advanced techniques and architectures such as different attention mechanisms or hierarchical modeling, which would enhance the model's ability to capture finer details and maintain realism. Secondly, increasing the amount and diversity of training data, specifically focusing on complex floorplans, would provide the model with a broader range of examples to learn from and improve its understanding of complex designs. Additionally, considering an End-to-End approach could be beneficial, as it would enable the model to directly generate complex designs without the need for postprocessing, allowing for better preservation of realism throughout the entire generation process. By implementing these recommendations, future research endeavors can overcome the limitations and enhance the realism of generated complex floorplans.

- **Optimization algorithms:** To address the limitation of sampling time in floorplan generation, a recommendation is to leverage various optimization techniques such as Bayesian optimization, reinforcement learning (RL), or genetic algorithms. These optimization methods can effectively expedite the sampling process. By incorporating Bayesian optimization, researchers can intelligently explore the parameter space and identify optimal sample configurations, resulting in reduced sampling time. RL algorithms, on the other hand, can learn from past experiences and improve sample selection over time, accelerating the generation process. Genetic algorithms mimic natural selection to iteratively

refine the sampling strategy, leading to faster and more efficient floorplan generation. By utilizing these different optimization techniques, researchers can expedite the sampling process and overcome the limitation of time constraints, enabling quicker and more responsive floorplan generation.

- **Residual Network:** To enhance the diversity of generated floorplans, a recommendation for future work is to leverage residual networks. Residual networks are renowned for their ability to capture and propagate information effectively through residual connections, making them valuable for improving the quality and variety of generated floorplans. By incorporating residual networks into the generative model, the aim is to enable the model to capture finer details, intricate spatial relationships, and unique architectural features. This integration of residual networks will expand the modeling capability, resulting in a broader range of diverse floorplan designs that encompass different layouts, styles, and functional arrangements. By pursuing this recommendation, significant improvements can be expected in the diversity and creativity of the generated floorplans, thereby advancing generative floorplan generation techniques and their practical applications in architecture and design.

REFERENCES

[1] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," arXiv preprint arXiv:2303.04226, 2023.

[2] S. Lin, B. Liu, J. Li, and X. Yang, "Common diffusion noise schedules and sample steps are flawed," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 5404–5411, 2024.

[3] O. Heckmann and F. Schneider, "Floor plan manual: Housing," 1997.

[4] R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for findpath with rotation," IEEE Transactions on Systems, Man, and Cybernetics, no. 2, pp. 224–233, 1985.

[5] J. Martin, "Procedural house generation: A method for dynamically generating floor plans," in Proceedings of the Symposium on Interactive 3D Graphics and Games, vol. 2, 2006.

[6] N. Nauata, S. Hosseini, K. Chang, H. Chu, C. Cheng, and Y. Furukawa, "House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects.," in CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13627–13636, 2021.

[7] M. A. Shabani, S. Hosseini, and Y. Furukawa, "Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5466–5475, 2023.

[8] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation," in Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16, pp. 162–177, Springer, 2020.

[9] J. Liu, Y. Xue, J. Duarte, K. Shekhawat, Z. Zhou, and X. Huang, "End-to-end graph-constrained vectorized floorplan generation with panoptic refinement," in European Conference on Computer Vision, pp. 547–562, Springer, 2022.

[10] H. Tang, Z. Zhang, H. Shi, B. Li, L. Shao, N. Sebe, R. Timofte, and L. Van Gool, "Graph transformer gans for graph-constrained house generation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2173–2182, 2023.

[11] F. Q. Lauzon, "An introduction to deep learning," in 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA), pp. 1438–1439, IEEE, 2012.

[12] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," WSEAS Transactions on Circuits and Systems, vol. 8, no. 7, pp. 579–588, 2009.

[13] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," arXiv preprint arXiv:1511.08458, 2015.

[14] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in 2014 13th international conference on control automation robotics & vision (ICARCV), pp. 844–848, IEEE, 2014.

[15] S. Wang, L. Sun, W. Fan, J. Sun, S. Naoi, K. Shirahata, T. Fukagai, Y. Tomita, A. Ike, and T. Hashimoto, "An automated cnn recommendation system for image classification tasks," in 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 283–288, IEEE, 2017.

[16] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," Ecological informatics, vol. 48, pp. 257–268, 2018.

[17] R. Chauhan, K. K. Ghanshala, and R. Joshi, "Convolutional neural network (cnn) for image detection and recognition," in 2018 first international conference on secure cyber computing and communication (ICSCCC), pp. 278–282, IEEE, 2018.

[18] S. Hijazi, R. Kumar, C. Rowen, et al., "Using convolutional neural networks for image recognition," Cadence Design Systems Inc.: San Jose, CA, USA, vol. 9, no. 1, 2015.

[19] A. Singh, A. Bansal, N. Chauhan, S. P. Sahu, and D. K. Dewangan, "Image generation using gan and its classification using svm and cnn," in Proceedings of Emerging Trends and Technologies on Intelligent Systems: ETTIS 2021, pp. 89–100, Springer, 2022.

[20] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., "Conditional image generation with pixelcnn decoders," Advances in neural information processing systems, vol. 29, 2016.

[21] D. A. Reynolds et al., "Gaussian mixture models.," Encyclopedia of biometrics, vol. 741, no. 659-663, 2009.

[22] S. R. Eddy, "Hidden markov models," Current opinion in structural biology, vol. 6, no. 3, pp. 361–365, 1996.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," Advances in neural information processing systems, vol. 27, 2014.

[24] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in International conference on machine learning, pp. 2256–2265, PMLR, 2015.

[25] A. MODEL, "System identification and," in Instrumentation, Control and Automation of Water and Wastewater Treatment and Transport Systems: Proceedings of the 5th IAWPRC Workshop Held in Yokohama and Kyoto, Japan, 26 July–3 August 1990, p. 121, Elsevier, 2013.

[26] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," Advances in neural information processing systems, vol. 29, 2016.

[27] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Cvae-gan: fine-grained image generation through asymmetric training," in Proceedings of the IEEE international conference on computer vision, pp. 2745–2754, 2017.

[28] Z. Qin, Z. Liu, P. Zhu, and Y. Xue, "A gan-based image synthesis method for skin lesion classification," Computer Methods and Programs in Biomedicine, vol. 195, p. 105568, 2020.

[29] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "Gan-based synthetic brain mr image generation," in 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), pp. 734–738, IEEE, 2018.

[30] Z. Lin, Y. Gong, Y. Shen, T. Wu, Z. Fan, C. Lin, N. Duan, and W. Chen, "Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise," in International Conference on Machine Learning, pp. 21051–21064, PMLR, 2023.

[31] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," arXiv preprint arXiv:1711.04340, 2017.

[32] R. Salakhutdinov, "Learning deep generative models," Annual Review of Statistics and Its Application, vol. 2, pp. 361–385, 2015.

[33] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion gans," arXiv preprint arXiv:2112.07804, 2021.

[34] R. Bayat, "A study on sample diversity in generative models: Gans vs. diffusion models," 2023.

[35] Y. Kossale, M. Airaj, and A. Darouichi, "Mode collapse in generative adversarial networks: An overview," in 2022 8th International Conference on Optimization and Applications (ICOA), pp. 1–6, IEEE, 2022.

[36] D. Saxena and J. Cao, "Generative adversarial networks (gans) challenges, solutions, and future directions," ACM Computing Surveys (CSUR), vol. 54, no. 3, pp. 1–42, 2021.

[37] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," arXiv preprint arXiv:1710.10196, 2017.

[38] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "Mgan: Training generative adversarial nets with multiple generators," in International conference on learning representations, 2018.

[39] M. M. Saad, M. H. Rehmani, and R. O'Reilly, "A self-attention guided multi-scale gradient gan for diversified x-ray image synthesis," in Irish Conference on Artificial Intelligence and Cognitive Science, pp. 18–31, Springer, 2022.

[40] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," arXiv preprint arXiv:1803.02155, 2018.

[41] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," arXiv preprint arXiv:1802.05957, 2018.

[42] D. Bang and H. Shim, "Improved training of generative adversarial networks using representative features," in International conference on machine learning, pp. 433–442, PMLR, 2018.

[43] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," Advances in neural information processing systems, vol. 34, pp. 8780–8794, 2021.

[44] Z. Kong and W. Ping, "On fast sampling of diffusion probabilistic models," arXiv preprint arXiv:2106.00132, 2021.

[45] E. Luhman and T. Luhman, "Knowledge distillation in iterative generative models for improved sampling speed," arXiv preprint arXiv:2101.02388, 2021.

[46] B. Chandra and R. K. Sharma, "Adaptive noise schedule for denoising autoencoder," in Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I 21, pp. 535–542, Springer, 2014.

[47] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," arXiv preprint arXiv:2011.13456, 2020.

[48] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman, and I. Mitliagkas, "Gotta go fast when generating data with score-based models," arXiv preprint arXiv:2105.14080, 2021.

[49] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," Advances in neural information processing systems, vol. 33, pp. 6840–6851, 2020.

[50] A. Tewari, M. J. Giering, and A. Raghunathan, "Parametric characterization of multimodal distributions with non-gaussian modes," in 2011 IEEE 11th international conference on data mining workshops, pp. 286–292, IEEE, 2011.

[51] S. Scarselli, C. Bellazzi, L. Lagnado, and A. Tacchella, "The graph neural network model," Advances in neural information processing systems, vol. 22, pp. 1467–1474, 2009.

[52] F. Zhang, N. Nauata, and Y. Furukawa, "Conv-mpn: Convolutional message passing neural network for structured outdoor architecture reconstruction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2798–2807, 2020.

[53] R. J. A. Buhr, G. M. Karam, C. J. Hayes, and C. M. Woodside, "Software cad: A revolutionary approach," IEEE Transactions on Software Engineering, vol. 15, no. 3, pp. 235–249, 1989.

[54] L. Rinde and A. Dahl, "Procedural generation of indoor environments," Charmers University of Technology, 2008.

[55] C.-H. Peng, Y.-L. Yang, and P. Wonka, "Computing layouts with deformable templates," ACM Transactions on Graphics (TOG), vol. 33, no. 4, pp. 1–11, 2014.

[56] P. Merrell, E. Schkufza, and V. Koltun, "Computer-generated residential building layouts," in ACM SIGGRAPH Asia 2010 papers, pp. 1–12, 2010.

[57] Y. Shi, M. Shang, and Z. Qi, "Intelligent layout generation based on deep generative models: A comprehensive survey," Information Fusion, p. 101940, 2023.

[58] N. M. Abd El-Maksoud and E. B. Ahmed, "Artificial intelligence applications in green architecture," Fayoum University Journal of Engineering, vol. 7, no. 2, pp. 317–337, 2024.

[59] N. Rane, S. Choudhary, and J. Rane, "Leading-edge technologies for architectural design: a comprehensive review," Available at SSRN 4637891, 2023.

[60] T. Tutenel, R. Bidarra, R. M. Smelik, and K. J. De Kraker, "Rule-based layout solving and its application to procedural interior generation," in CASA workshop on 3D advanced media in gaming and simulation, 2009.

[61] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes, "A survey on procedural modelling for virtual worlds," in Computer graphics forum, vol. 33, pp. 31–50, Wiley Online Library, 2014.

[62] C. Borgelt, "An implementation of the fp-growth algorithm," in Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations, pp. 1–5, 2005.

[63] R. Zhao and Q. Ji, "An adversarial hierarchical hidden markov model for human pose modeling and generation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[64] Y. Cao, L. Sun, C. Han, and J. Guo, "Improved side information generation algorithm based on naive bayesian theory for distributed video coding," IET Image Processing, vol. 12, no. 3, pp. 354–360, 2018.

[65] B. Fernando, E. Fromont, D. Muselet, and M. Sebban, "Supervised learning of gaussian mixture models for visual vocabulary generation," Pattern Recognition, vol. 45, no. 2, pp. 897–907, 2012.

[66] C. Sutton, A. McCallum, et al., "An introduction to conditional random fields," Foundations and Trends® in Machine Learning, vol. 4, no. 4, pp. 267–373, 2012.

[67] M. P. LaValley, "Logistic regression," Circulation, vol. 117, no. 18, pp. 2395–2399, 2008.

[68] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in International Conference on Machine Learning, pp. 8821–8831, PMLR, 2021.

[69] C. Li, Z. Wang, and H. Qi, "Fast-converging conditional generative adversarial networks for image synthesis," in 2018 25th IEEE international conference on image processing (ICIP), pp. 2132–2136, IEEE, 2018.

[70] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan++: Realistic image synthesis with stacked generative adversarial networks," IEEE transactions on pattern analysis and machine intelligence, vol. 41, no. 8, pp. 1947–1962, 2018.

[71] X. Li, J. Luo, and R. Younes, "Activitygan: Generative adversarial networks for data augmentation in sensor-based human activity recognition," in Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, pp. 249–254, 2020.

[72] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., "Photo-realistic single image super-resolution using a generative adversarial network," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4681–4690, 2017.

[73] Z. Chen and X. Chen, "Tachiegan: Generative adversarial networks for tachie style transfer," in 2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–6, IEEE, 2022.

[74] M. Pektas, B. Gecer, and A. Ugur, "Efficient hair style transfer with generative adversarial networks," arXiv preprint arXiv:2210.12524, 2022.

[75] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.

[76] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784, 2014.

[77] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," arXiv preprint arXiv:1802.05957, 2018.

[78] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8798–8807, 2018.

[79] H. Lim, "Automatic generation of ai-powered architectural floor plans using grid data," International Journal of Applied Engineering Research, vol. 18, no. 2, pp. 97–102, 2023.

[80] S. Chaillou, "Archigan: a generative stack for apartment building design," NVIDIA Corporation, 2019.

[81] S. Wang, W. Zeng, X. Chen, Y. Ye, Y. Qiao, and C.-W. Fu, "Actfloor-gan: Activity-guided adversarial networks for human-centric floorplan design," IEEE Transactions on Visualization and Computer Graphics, pp. 1–1, 2021.

[82] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Research Report 9811, 1998.

[83] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in International conference on machine learning, pp. 2256–2265, PMLR, 2015.

[84] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in International Conference on Machine Learning, pp. 8162–8171, PMLR, 2021.

[85] W. Yu, S. Heber, and T. Pock, "Learning reaction-diffusion models for image inpainting," in Pattern Recognition: 37th German Conference, GCPR 2015, Aachen, Germany, October 7-10, 2015, Proceedings 37, pp. 356–367, Springer, 2015.

[86] Y. Li, H. Wang, Q. Jin, J. Hu, P. Chemerys, Y. Fu, Y. Wang, S. Tulyakov, and J. Ren, "Snapfusion: Text-to-image diffusion model on mobile devices within two seconds," arXiv preprint arXiv:2306.00980, 2023.

[87] L. He, Y. Lu, J. Corring, D. Florencio, and C. Zhang, "Diffusion-based document layout generation," arXiv preprint arXiv:2303.10787, 2023.

[88] S. Chai, L. Zhuang, and F. Yan, "Layoutdm: Transformer-based diffusion model for layout generation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18349–18358, 2023.

[89] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[90] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, et al., "Scaling autoregressive models for content-rich text-to-image generation," arXiv preprint arXiv:2206.10789, vol. 2, no. 3, p. 5, 2022.

[91] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, et al., "Scaling autoregressive models for content-rich text-to-image generation," arXiv preprint arXiv:2206.10789, vol. 2, no. 3, p. 5, 2022.

[92] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., "Conditional image generation with pixelcnn decoders," Advances in neural information processing systems, vol. 29, 2016.

[93] A. Kolesnikov and C. H. Lampert, "Pixelcnn models with auxiliary variables for natural image modeling," in International Conference on Machine Learning, pp. 1905–1914, PMLR, 2017.

[94] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in Eleventh Annual Conference of the International Speech Communication Association, 2010.

[95] Z. C. Lipton, J. Berkowitz, and C. Elkan, "Critical review of recurrent neural networks for sequence learning," arXiv preprint arXiv:1506.00019, 2015.

[96] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., "Conditional image generation with pixelcnn decoders," Advances in neural information processing systems, vol. 29, 2016.

[97] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in International Conference on Learning Representations (ICLR), 2017.

[98] C. Liu, J. Wu, and Y. Furukawa, "Floornet: A unified framework for floorplan reconstruction from 3d scans," in Proceedings of the European conference on computer vision (ECCV), pp. 201–217, 2018.

[99] Q. Chen, Q. Wu, R. Tang, Y. Wang, S. Wang, and M. Tan, "Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12625–12634, 2020.

[100] R. Hu, Z. Huang, Y. Tang, O. Van Kaick, H. Zhang, and H. Huang, "Graph2plan: Learning floorplan generation from layout graphs," ACM Transactions on Graphics (TOG), vol. 39, no. 4, pp. 118–1, 2020.

[101] A. Chen, "Generation of layouts for living spaces using conditional generative adversarial networks: Designing floor plans that respect both a boundary and high-level requirements," 2022.

[102] E. Schiller, Creating Novel Architectural Layouts With Generative Adversarial Networks. PhD thesis, Harvard University, 2018.

[103] Y. Liu, Y. Luo, Q. Deng, and X. Zhou, "Exploration of campus layout based on generative adversarial network: Discussing the significance of small amount sample learning for architecture," in Proceedings of the 2020 DigitalFUTURES: The 2nd International Conference on Computational Design and Robotic Fabrication (CDRF 2020), pp. 169–178, Springer, 2021.

[104] W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, and L. Liu, "Data-driven interior plan generation for residential buildings," ACM Transactions on Graphics (TOG), vol. 38, no. 6, pp. 1–12, 2019.

[105] M. Boroumand, M. Chen, and J. Fridrich, "Image processing operations identification via convolutional neural networks," IEEE Transactions on Information Forensics and Security, vol. 13, no. 1, pp. 67–82, 2017.

[106] A. Sufian, "Advancements in image classification using convolutional neural network," arXiv preprint arXiv:1905.03288, 2019.

[107] S. Ratnasingam, S. S. Kumar, and S. S. Kumar, "Deep camera: A fully convolutional neural network for image signal processing," arXiv preprint arXiv:1908.09191, 2019.

[108] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," Information Fusion, vol. 81, pp. 84–90, 2022.

[109] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 11, pp. 4883–4894, 2019.

[110] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), pp. 1234–1241, 2020.

[111] J. Gu, H. Hu, L. Wang, Y. Wei, and J. Dai, "Learning region features for object detection," in Proceedings of the european conference on computer vision (ECCV), pp. 381–395, 2018.

[112] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3588–3597, 2018.

[113] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," Advances in neural information processing systems, vol. 29, 2016.

[114] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7370–7377, 2019.

[115] Y. Wang, J. Luo, S. Zhang, X. Wang, and X. Yao, "Graph neural networks and their current applications in bioinformatics," Frontiers in Genetics, vol. 12, p. 690049, 2021.

[116] F. Zhang, N. Nauata, and Y. Furukawa, "Conv-mpn: Convolutional message passing neural network for structured outdoor architecture reconstruction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2798–2807, 2020.

[117] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in International Conference on Learning Representations (ICLR), 2018.

[118] A. Calissano, A. Feragen, and S. Vantini, "Graph-valued regression: Prediction of unlabelled networks in a non-euclidean graph space," Journal of Multivariate Analysis, vol. 190, p. 104950, 2022.

[119] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, and J. Huang, "Rumor detection on social media with bi-directional graph convolutional networks," in Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 549–556, 2020.

[120] J. Choi, T. Ko, Y. Choi, H. Byun, and C.-k. Kim, "Dynamic graph convolutional networks with attention mechanism for rumor detection on social media," Plos one, vol. 16, no. 8, p. e0256039, 2021.

[121] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," Advances in neural information processing systems, vol. 31, 2018.

[122] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 974–983, 2018.

[123] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," Advances in neural information processing systems, vol. 31, 2018.

[124] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, and G. Chen, "Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems," in The world wide web conference, pp. 2091–2102, 2019.

[125] B. Lai and J. Xu, "Accurate protein function prediction via graph attention networks with predicted structure information," Briefings in Bioinformatics, vol. 23, no. 1, p. bbab502, 2022.

[126] H. Wang, G. Zhou, S. Liu, J.-Y. Jiang, and W. Wang, "Drug-target interaction prediction with graph attention networks," arXiv preprint arXiv:2107.06099, 2021.

[127] T. Yang, L. Hu, C. Shi, H. Ji, X. Li, and L. Nie, "Hgat: Heterogeneous graph attention networks for semi-supervised short text classification," ACM Transactions on Information Systems (TOIS), vol. 39, no. 3, pp. 1–29, 2021.

[128] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction," in 2010 IEEE information theory workshop on information theory (ITW 2010, Cairo), pp. 1–5, IEEE, 2010.

[129] M. R. Greenstein, "Machine learning regularization explained with examples," SearchEnterpriseAI, 2019.

[130] Q. Xu, M. Zhang, Z. Gu, and G. Pan, "Overfitting remedy by sparsifying regularization on fully-connected layers of cnns," Neurocomputing, vol. 328, pp. 69–74, 2019.

[131] M. Lee and J. Seok, "Regularization methods for generative adversarial networks: An overview of recent studies," arXiv preprint arXiv:2005.09165, 2020.

[132] X. Gao, F. Deng, and X. Yue, "Data augmentation in fault diagnosis based on the wasserstein generative adversarial network with gradient penalty," Neurocomputing, vol. 396, pp. 487–494, 2020.

[133] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," Advances in neural information processing systems, vol. 31, 2018.

[134] S.-H. Yang and B.-G. Hu, "A stagewise least square loss function for classification," in Proceedings of the 2008 SIAM International Conference on Data Mining, pp. 120–131, SIAM, 2008.

[135] C. Gentile and M. K. Warmuth, "Linear hinge loss and average margin," Advances in neural information processing systems, vol. 11, 1998.

[136] Q. Zhu, Z. He, T. Zhang, and W. Cui, "Improving classification performance of softmax loss function based on scalable batch-normalization," Applied Sciences, vol. 10, no. 8, p. 2950, 2020.

[137] K. Das, J. Jiang, and J. Rao, "Mean squared error of empirical predictor," 2004.

[138] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "On mean absolute error for deep neural network based vector-to-vector regression," IEEE Signal Processing Letters, vol. 27, pp. 1485–1489, 2020.

[139] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in International conference on machine learning, pp. 214–223, PMLR, 2017.

[140] V. M. Panaretos and Y. Zemel, "Statistical aspects of wasserstein distances," Annual review of statistics and its application, vol. 6, pp. 405–431, 2019.

[141] M. Menéndez, J. Pardo, L. Pardo, and M. Pardo, "The jensen-shannon divergence," Journal of the Franklin Institute, vol. 334, no. 2, pp. 307–318, 1997.

[142] D. Baby and S. Verhulst, "Sergan: Speech enhancement using relativistic generative adversarial networks with gradient penalty," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 106–110, IEEE, 2019.

[143] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," Advances in neural information processing systems, vol. 29, 2016.

[144] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in International conference on machine learning, pp. 7354–7363, PMLR, 2019.

[145] G. R. Wood and B. Zhang, "Estimation of the lipschitz constant of a function," Journal of Global Optimization, vol. 8, pp. 91–103, 1996.

[146] J. Donahue, K. Simonyan, et al., "Large scale adversarial representation learning," arXiv preprint arXiv:1907.02544, 2019.

[147] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4401–4410, 2019.

[148] A. Karnewar and O. Wang, "Msg-gan: Multi-scale gradients for generative adversarial networks," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7799–7808, 2020.

[149] Y.-J. Yeo, M.-C. Sagong, S. Park, S.-J. Ko, and Y.-G. Shin, "Image generation with self pixel-wise normalization," Applied Intelligence, vol. 53, no. 8, pp. 9409–9423, 2023.

[150] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," arXiv preprint arXiv:2202.00512, 2022.

[151] R. Feng, D. Zhao, and Z.-J. Zha, "Understanding noise injection in gans," in international conference on machine learning, pp. 3284–3293, PMLR, 2021.

[152] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in Proceedings of the IEEE international conference on computer vision, pp. 1501–1510, 2017.

[153] T. Karras, S. Laine, T. Aila, et al., "Stylegan2: A style-based generator architecture for generative adversarial networks," arXiv preprint arXiv:1912.04948, 2019.

[154] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in Proceedings of the IEEE international conference on computer vision, pp. 2223–2232, 2017.

[155] J. Zhang, X. Yao, Y. Zhang, N. Xu, and X. Liu, "Spectral norm normalized network for facial expression generation," in 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), pp. 202–206, IEEE, 2020.

[156] J. Ren, M. Ren, L. Sun, L. Zhu, and X. Jiang, "Generative model-driven sampling strategy for the high-efficiency measurement of complex surfaces on coordinate measuring machines," IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1–11, 2021.

[157] K. Sankaran and S. P. Holmes, "Generative models: An interdisciplinary perspective," Annual Review of Statistics and Its Application, vol. 10, pp. 325–352, 2023.

[158] M. Boguná, L. F. Lafuerza, R. Toral, and M. Á. Serrano, "Simulating non-markovian stochastic processes," Physical Review E, vol. 90, no. 4, p. 042108, 2014.

[159] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[160] C. Andrieu and J. Thoms, "A tutorial on adaptive mcmc," Statistics and computing, vol. 18, pp. 343–373, 2008.

[161] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," arXiv preprint arXiv:2010.02502, 2020.

[162] R. San-Roman, E. Nachmani, and L. Wolf, "Noise estimation for generative diffusion models," arXiv preprint arXiv:2104.02600, 2021.

[163] A. F. Bastani and S. M. Hosseini, "A new adaptive runge–kutta method for stochastic differential equations," Journal of computational and applied mathematics, vol. 206, no. 2, pp. 631–644, 2007.

[164] C. Yuan and X. Mao, "Convergence of the euler–maruyama method for stochastic differential equations with markovian switching," Mathematics and Computers in Simulation, vol. 64, no. 2, pp. 223–235, 2004.

[165] C. Meng, L. Trinh, and N. Xu, "Interpretability and fairness evaluation of deep learning models on mimic-iv dataset," Scientific Reports, vol. 12, p. 7166, 2022.

[166] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," Advances in neural information processing systems, vol. 30, 2017.

[167] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in 2017 2nd international conference on image, vision and computing (ICIVC), pp. 783–787, IEEE, 2017.

[168] A. Driemel, I. van der Hoog, and E. Rotenberg, "On the discrete fr\'echet distance in a graph," arXiv preprint arXiv:2201.02121, 2022.

[169] P. Kordík, J. Koutník, J. Drchal, O. Kovářík, M. Čepek, and M. Šnorek, "Meta-learning approach to neural network optimization," Neural Networks, vol. 23, no. 4, pp. 568–582, 2010.

[170] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in International conference on machine learning, pp. 5156–5165, PMLR, 2020.

[171] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," Advances in neural information processing systems, vol. 32, 2019.

[172] K. Peffers, T. Tuunanen, C. E. Gengler, M. Rossi, W. Hui, V. Virtanen, and J. Bragge, "Design science research process: a model for producing and presenting information systems research," arXiv preprint arXiv:2006.02763, 2020.

[173] W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, and L. Liu, "Data-driven interior plan generation for residential buildings," ACM Transactions on Graphics (TOG), vol. 38, no. 6, pp. 1–12, 2019.

[174] C. Suresh, S. Singh, R. Saini, and A. K. Saini, "A comparative analysis of image scaling algorithms," International Journal of Image, Graphics and Signal Processing, vol. 5, no. 5, p. 55, 2013.

[175] C. Liu, J. Wu, P. Kohli, and Y. Furukawa, "Raster-to-vector: Revisiting floorplan transformation," in Proceedings of the IEEE International Conference on Computer Vision, pp. 2195–2203, 2017.

[176] D. Ziou and S. Tabbone, "Edge detection techniques-an overview," Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications, vol. 8, no. 4, pp. 537–559, 1998.

[177] R. Chandel and G. Gupta, "Image filtering algorithms and techniques: A review," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 10, 2013.

[178] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1, pp. 128–144, Springer, 2020.

[179] H. P. Williams, "Integer programming," in Logic and Integer Programming, pp. 25–70, Springer, 2009.

[180] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "Digress: Discrete denoising diffusion for graph generation," arXiv preprint arXiv:2209.14734, 2022.

[181] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon, "Permutation invariant graph generation via score-based generative modeling," in International Conference on Artificial Intelligence and Statistics, pp. 4474–4484, PMLR, 2020.

[182] J. Jo, S. Lee, and S. J. Hwang, "Score-based generative modeling of graphs via the system of stochastic differential equations," in International Conference on Machine Learning, pp. 10362–10383, PMLR, 2022.

[183] T. H. Do, D. M. Nguyen, G. Bekoulis, A. Munteanu, and N. Deligiannis, "Graph convolutional neural networks with node transition probability-based message passing and dropnode regularization," Expert Systems with Applications, vol. 174, p. 114711, 2021.

[184] T. Larsson and L. Källberg, Fast computation of tight-fitting oriented bounding boxes. AK Peters, Ltd., 2011.

[185] V. Nagarajan and J. Z. Kolter, "Gradient descent gan optimization is locally stable," Advances in neural information processing systems, vol. 30, 2017.

[186] Y. Li and G. Baciu, "Sg-gan: Adversarial self-attention gcn for point cloud topological parts generation," IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 10, pp. 3499–3512, 2021.