

**ADAPTIVE MODULARITY: DEEP REINFORCEMENT
LEARNING FOR OPTIMIZED MODULAR HOUSING MASSING**

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ARCHITECTURE

By
Betül Keleş Ekici

July 2025

To my family

ADAPTIVE MODULARITY: DEEP REINFORCEMENT LEARNING FOR OPTIMIZED MODULAR HOUSING MASSING

By Betül Keleş Ekici

July 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Burcu Şenyapılı Özcan (Advisor)

Nuri Cihan Kayaçetin

Başak Uçar

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan

Director of the Graduate School

ABSTRACT

ADAPTIVE MODULARITY: DEEP REINFORCEMENT LEARNING FOR OPTIMIZED MODULAR HOUSING MASSING

Betül Keleş Ekici

M.S. in Architecture

Advisor: Burcu Şenyapılı Özcan

July 2025

The advancement of prefabrication technology has positioned modular housing as a precise, sustainable construction method that reduces greenhouse gas emissions and resource intensiveness while enabling a circular economy and fostering energy efficiency in meeting increasing housing demands. The design of massing configurations from modular units presents challenges due to the complexity of optimizing spatial arrangements to simultaneously balance architectural, structural, and environmental performance criteria and constraints. This thesis addresses the challenge of modular housing design, tackling the considerable complexity of optimizing spatial arrangements. The aim is to improve the design process through Deep Reinforcement Learning (DRL) to balance architectural criteria and constraints. The thesis investigates how DRL can enhance modular housing design by optimizing spatial configurations, overcoming the limitations of human-driven iterations, and

leveraging the benefits of a feedback-driven system. It proposes a simulation environment based on a Cellular Automata (CA) voxel space, where an AI agent, trained using the Proximal Policy Optimization (PPO) algorithm, autonomously places massing cells to generate optimized housing configurations. These configurations are evaluated based on multiple performance objectives, such as ensuring efficient space utilization by maximizing the number of housing units within the buildable area, minimizing circulation space while maintaining accessibility to all units, and promoting the creation of housing units that are more compact in shape. This thesis focuses on the effort to optimize modular mass configuration for housing design. It outlines the steps taken in the process, the challenges faced along the way, and evaluates the success of the final configurations that were developed.

Keywords: Modular Housing Design, Deep Reinforcement Learning, Cellular Automata, Agent-Based Systems, Multi-Objective Optimization

ÖZET

UYARLANABİLİR MODÜLERLİK: OPTİMUM MODÜLER KONUT KÜTLE YERLEŞİMİ İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME

Betül Keleş Ekici

Mimarlık, Yüksek Lisans

Tez Danışmanı: Burcu Şenyapılı Özcan

Temmuz 2025

Prefabrikasyon teknolojilerindeki ilerlemeler, modüler konutu, artan konut talebini karşılarken sera gazı emisyonlarını ve kaynak kullanımını azaltan, döngüsel ekonomiyi destekleyen ve enerji verimliliğini artıran hassas ve sürdürülebilir bir inşaat yöntemi olarak konumlandırmıştır. Ancak, modüler birimlerden kütle yerleşimlerinin tasarımı, mekânsal düzenlemelerin mimari, yapısal ve çevresel kriterler ile kısıtlamaları eşzamanlı dengeleyecek şekilde optimize edilmesindeki karmaşıklık nedeniyle zorluklar barındırmaktadır. Bu tez, modüler konut tasarımındaki bu zorluğa odaklanarak mekânsal düzenlemelerin optimizasyonundaki karmaşıklığı ele almaktadır. Amaç, mimari kriterler ve kısıtlamalar arasında denge kuran bir tasarım sürecini Derin Pekiştirmeli Öğrenme (DRL) ile geliştirmektir. Tez, DRL'in mekânsal konfigürasyonları optimize ederek, insan tarafından yürütülen yinelemeli süreçlerin sınırlılıklarını aşma ve geri bildirim temelli sistemlerden

yararlanma potansiyelini arařtırmaktadır. Bu kapsamda, Hücresel Özdevinim (CA) tabanlı voksel bir simölasyon ortamı önerilmiřtir. Bu ortamda, Proximal Policy Optimization (PPO) algoritmasıyla eęitilen bir yapay zekâ ajanı, kütle hücrelerini özerk biçimde yerleřtirerek optimize edilmiř konut konfigürasyonları üretmektedir. Bu konfigürasyonlar; inşa edilebilir alan içinde birim sayısını maksimize etme, sirkölasyon alanlarını minimize ederken tüm birimlere eriřimi saęlama ve daha kompakt formlara sahip konut birimleri oluřturmayı teřvik etmek gibi çoklu performans kriterlerine göre deęerlendirilmektedir. Bu tez, konut tasarımında modüler kütle konfigürasyonunun optimizasyonuna yönelik çabayı merkezine almakta; süreçte izlenen adımları, karřılařılan zorlukları ve elde edilen nihai konfigürasyonların başarısını deęerlendirmektedir.

Anahtar Kelimeler: Modüler Konut Tasarımı, Derin Pekiřtirmeli Öğrenme, Hücresel Özdevinim, Ajan Tabanlı Modeller, Çok Amaçlı Optimizasyon

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful to my thesis advisor, Assoc. Prof. Burcu Şenyapılı Özcan, for her unwavering support, guidance, and trust throughout this journey. Beyond being an exceptional advisor, she has been a role model whose dedication continuously motivated me to push beyond my limits. I feel truly fortunate to have had the opportunity to work with and learn from her.

I would like to sincerely thank Houssame Eddine H'sain for his invaluable contribution to every stage of this thesis. His guidance in shaping my research topic, introducing me to reinforcement learning, and supporting me in developing my technical skills was essential. As a peer who has recently walked a similar path, his encouragement has been a constant source of motivation.

I am also thankful to my thesis committee members, Assist. Prof. Nuri Cihan Kayaçetin and Assoc. Prof. Başak Uçar, for their valuable feedback and insightful suggestions, which significantly strengthened this thesis and broadened my perspective for future research.

I feel indebted to Bilkent University and TUBITAK for granting me the scholarship that has enabled me to pursue my academic studies. This study was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the 2211-National Graduate Scholarship Program.

My gratitude extends to my dear friends, Melek Kaynar, Mohit Purshotam, Didem Baykal, Zeynep Uzun and Mustafa Seyyid Çetin for always being there for me,

keeping me motivated, and brightening this process with their friendship. I would also like to express my sincere appreciation to Melek and Mustafa for their valuable contributions to this thesis. Their insights and support went beyond friendship, and I am truly grateful for the time and efforts they contributed to this work.

I owe my deepest thanks to my family, to my parents, Şenay Keleş and Hüseyin Keleş, for teaching me resilience and perseverance; to my sisters, Beyza Keleş, Hatice Koç; to my brother, Mustafa Koç, and my beloved nieces, Asaf and Damla; and to my other family, Nurcan Ekici, İsmail Ekici, and Feyza Ekici for their unwavering support.

Finally, I want to express my heartfelt gratitude to my better half, Muaz, my closest companion and greatest supporter, for his endless support, love, and encouragement; *“I could not have done this without you”*. And to *Kimyon*, the furry member of our little family, for bringing joy to this journey.

TABLE OF CONTENTS

ABSTRACT.....	iv
ÖZET.....	vi
ACKNOWLEDGEMENTS.....	viii
TABLE OF CONTENTS.....	x
LIST OF TABLES.....	xii
LIST OF FIGURES.....	xiii
LIST OF ABBREVIATIONS.....	xviii
CHAPTER 1:	
INTRODUCTION.....	1
1.1 Research Problem Statement.....	2
1.2 Aim of the Study and Research Questions.....	4
1.3 Methodology of the Research.....	7
1.4 Structure of the Thesis.....	8
CHAPTER 2:	
MODULAR ARCHITECTURE FOR HOUSING.....	10
2.1 Modular Construction.....	13
2.2 Advantages of Modular Construction.....	14
2.3 Modular Housing.....	17
2.3.1 Modular Housing as a Response to Housing Shortage.....	17
2.3.2 Challenges in Modular Housing.....	19
2.3.3 The Paradigm Shift in Design for Modular Housing.....	22
CHAPTER 3:	
REINFORCEMENT LEARNING IN GENERATIVE DESIGN.....	25
3.1 Generative Design and Applications in Architecture.....	27
3.1.1 Generative design applications in modular housing design.....	29
3.2 Reinforcement Learning.....	40

3.2.1. Artificial Intelligence, Machine Learning, and Reinforcement Learning	41
3.2.2 Reinforcement Learning and its Applications in Design.....	45
CHAPTER 4:	
DESIGN OF HOUSING MASSING THROUGH MACHINE LEARNING.....	54
4.1 Cell Design and Spatial Configuration.....	56
Initial state of the environment:.....	58
Cellular Automata Constraints:.....	58
4.2 Experiment Setting as Markov Decision Processes (MDPs).....	62
4.3 Description and Results of Experimental Settings.....	65
4.3.1 Experimental Setting I.....	66
4.3.1.1 RL Environment Setup for Experimental Setting I.....	66
4.3.1.2 Potential-Based Reward Shaping (PBRs).....	71
4.3.1.3 Results and Discussion for Experimental Setting I.....	72
4.3.2 Experimental Setting II.....	75
4.3.2.1 RL Environment Setup for Experimental Setting II.....	75
4.3.2.2 Potential-Based Reward Shaping (PBRs): Special Case of $\gamma=1$	78
4.3.2.3 Results and Discussion for Experimental Setting II.....	78
CHAPTER 5:	
CONCLUSION AND FUTURE IMPLICATIONS.....	90
REFERENCES.....	96
APPENDICES.....	108
APPENDIX A: RL ENVIRONMENT SETUP.....	108
APPENDIX B: HYPERPARAMETER CONFIGURATION.....	112
APPENDIX C: COMPARATIVE TRAINING DATA.....	114
APPENDIX D: REWARD SHAPING EXAMPLE.....	123

LIST OF TABLES

- Table A.1. Tuned hyperparameters used in the final model, grouped by function. Values were selected via Optuna and rounded for clarity. Produced by author.
- Table A.2. Comparison of design outputs generated by an untrained agent (random actions) and a trained agent with a fixed core location at 800 steps. The top row displays the 3D voxel configuration, while the subsequent rows show 2D floor slices. Produced by author.
- Table A.3. Comparison of untrained and trained model outputs at 1200 steps. Format and interpretation follow Table A.2. Produced by author.
- Table A.4. Comparison of untrained and trained model outputs at 1600 steps. Format and interpretation follow Table A.2. Produced by author.
- Table A.5. Comparison of untrained and trained model outputs at 2400 steps. Format and interpretation follow Table A.2. Produced by author.
- Table A.6. Design output generated by a trained agent using randomly placed core locations at 1600 steps. The top row shows the 3D voxel configuration, while the subsequent rows present corresponding 2D floor slices. Produced by author.
- Table A.7. Design output at 2000 steps with randomly placed core locations. Format and interpretation follow Table A.6. Produced by author.
- Table A.8. Design output at 2400 steps with randomly placed core locations. Format and interpretation follow Table A.6. Produced by author.
- Table A.9. Design output at 8000 steps with randomly placed core locations. Format and interpretation follow Table A.6. Produced by author.

LIST OF FIGURES

Figure 2.1 CONSTRUCTION of the Crystal Palace took only 17 weeks largely because the structure was built by highly organized crews assembling a variety of prefabricated modules. In the foreground crews lift trusses to be connected to hollow cast-iron columns. On the roof are six carts called traveling stages, which were used to install the glass panes. Reproduced for academic purposes from Kihlstedt (1984, p.10).

Figure 2.2 Habitat 67, designed by Safdie Architects, exemplifies an iconic project where volumetric and modular systems were used. Reproduced from Safdie Architects (n.d.), Habitat 67. Retrieved from <https://www.safdiearchitects.com/projects/habitat-67>

Figure 2.3 Noritaka Minami, Facade I, 2011, Tokyo, Japan. Courtesy of the artist. From the 2015 Individual Grant to Noritaka Minami and Ken Yoshida for 1972-Nakagin Capsule Tower. (Reproduced from Graham Foundation, 2015, p.1).

Figure 3.1. Types of dwellings including studio, one- and two-bedroom apartments. Reproduced for academic and illustrative purposes from Araghi and Stouffs (2015, p. 4), Exploring cellular automata for high density residential building form generation, Automation in Construction.

Figure 3.2. Figure 3.2.3D illustration of neighborhood patterns in a 5×25 grid of cells. Reproduced for academic and illustrative purposes from Araghi and Stouffs (2015, p. 5), Exploring cellular automata for high density residential building form generation, Automation in Construction.

Figure 3.3. Three optimal solutions for the spatial design of the case study. Reproduced for academic and illustrative purposes from Sharafi et al. (2017, p. 11). Automated spatial design of multi-story modular buildings using a unified matrix method. Automation in Construction, 82, 31–42.

Figure 3.4. The overview of the research methodology. Reproduced from Ghannad & Lee (2022, p.5). Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN). *Automation in Construction*, 139, 104234.

Figure 3.5. Example showing the vectorization process in modular housing design. Reproduced for academic and illustrative purposes from Ghannad and Lee (2022, p.8). Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN). *Automation in Construction*, 139, 104234.

Figure 3.6. a) Considers the primary apartment outline, b) Divides the outline into equally-lengthed slices, c) Divides the resulting residential floorplan outlines located on both sides of the hallway, with a specified entrance slice. Reproduced from Samadi et al. (2025, p.7).

Figure 3.7. a) Categories created from 173 designed Slices based on length and common combinations of functional spaces. Reproduced from Samadi et al. (2025, p.8).

Figure 3.8. WFC generates a design, by filling 3D empty slots (1), with predefined tiles (2), following adjacency rules (3) to generate a combination (4).

Figure 3.9. a) Sample images prepared for training data and corresponding knowledge graphs. Reproduced for academic and illustrative purposes from Liu et al. (2024, p.7), Intelligent floor plan design of modular high-rise residential building based on graph-constrained generative adversarial networks, *Automation in Construction*, 159, 105264.

Figure 3.10. Diagram of three typical types of machine learning: Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL). Reproduced from Wang and Snooks (2021), *Artificial Intuitions of Generative Design*, in *DigitalFUTURES*, Springer. Licensed under CC BY 4.0.

Figure 3.11. Selected training outcome samples from episode 0 to 9.4 k. Reproduced for academic and illustrative purposes from Wang and Snooks (2021),

Artificial Intuitions of Generative Design, in DigitalFUTURES, Springer. Licensed under CC BY 4.0.

Figure 3.12. Overall method of structuring the RL-based intuitive generative approach. Reproduced for academic and illustrative purposes from Wang and Snooks (2021, p. 4), Intuitive Behavior: The Operation of Reinforcement Learning in Generative Design Processes, CAADRIA 2021 Proceedings.

Figure 3.13. Object generation using a LEGO block-based model. Reproduced for academic and illustrative purposes from Akizuki et al. (2020), Generative modeling with design constraints: Reinforcement learning for object generation, presented at the 25th CAADRIA Conference.

Figure 3.14. Agent's action trajectory from initial random layout to final optimal layout.. Reproduced for academic and illustrative purposes from Kakooee and Dillenburger (2025), Illuminating Spaces: Deep Reinforcement Learning and Laser-Wall Partitioning for Architectural Layout Generation, ETH Zurich, arXiv preprint.

Figure 3.15. Overview of the construction simulations. Reproduced for academic and illustrative purposes from Zhu et al. (2023), Deep Reinforcement Learning for Real-Time Assembly Planning in Robot-Based Prefabricated Construction, IEEE Transactions on Automation Science and Engineering, 20(3), 1515–1526.

Figure 3.16. a) Spatial parts library (NoMAS 2019) b) Spatial assembly sequence (NoMAS 2019). Reproduced for academic and illustrative purposes from Hosmer et al. (2020), DSpatial Assembly with Self-Play Reinforcement Learning, ACADIA 2020: Volume I: Technical Papers, pp. 384.

Figure 4.1. Left: Moore neighborhood. Right: Von Neumann neighborhood. Produced by author.

Figure 4.2. Representation of the building as a three-dimensional array. Produced by author.

Figure 4.3. Cell types, from left to right: Empty Cell, Built-up Cell, Core Cell, and Hall Cell. Produced by author.

Figure 4.4. Static Rule Layer. Produced by author.

Figure 4.5. An instance of the rules for cell availability for placement. Produced by author.

Figure 4.6. Instances of IoU calculations. Produced by author.

Figure 4.7. Reinforcement Learning: An Introduction 2nd Edition, Richard S. Sutton and Andrew G. Barto. Reproduced from Sutton & Barto (2018, p.54).

Figure 4.8. Dynamic, decision layer. Design problem within MDP framework. Produced by author.

Figure 4.9. RL State (Observation) Space (S). Produced by author.

Figure 4.10. RL Action (A) Space. Produced by author.

Figure 4.11. Reward progress graph showing minimum, maximum, and mean rewards during training, with every fifth checkpoint indicated. Produced by author.

Figure 4.12. Outputs of the model from a) checkpoint 0, b) checkpoint 5, c) checkpoint 10, d) checkpoint 15. From left to right: Outputs from steps 200, 400, 600, 800, 1000. Produced by author.

Figure 4.13. Six different outputs generated by the trained model: single-floor building with a fixed core location. Produced by author.

Figure 4.14. Training graphs: left – mean episode length, right – mean episode reward during training for the fixed core location. Produced by author.

Figure 4.15. Three example training cases illustrating the randomly placed core cell conditions that the agent encountered during learning. Produced by author.

Figure 4.16. Six different outputs generated by the model trained with random core placement, evaluated here in a fixed core scenario. Produced by author.

Figure 4.17. Training graphs: left – mean episode length, right – mean episode reward during training, comparing fixed core (blue) and random core (red) location. Produced by author.

Figure 4.18. Training graph: Mean Episode Reward over 1.5M Episodes, eight-floor building with a fixed core location trained for 1200 steps. Produced by author.

Figure 4.19. Outputs generated every 10k episodes by the trained model in order to demonstrate the learning process of the model: eight-floor building with a fixed core location trained for 1200 steps. Produced by author.

Figure 4.20. 3D Isometric Visualization of Building Output at (left) Episode 40,000 | (right) Episode 1,100,000. Produced by author.

Figure 4.21. Floor-by-Floor Plan Views at (left) Step 40,000 | (right) Step 1,100,000. Produced by author.

Figure A.1. Training graphs: left – mean episode length, right – mean episode reward during hyperparameter tuning. Produced by author.

Figure A.2. Training diagnostics for all Optuna trials, shown from left to right, top to bottom: approximate KL divergence, clip fraction, clip range, entropy loss, explained variance, learning rate, total loss, policy gradient loss, and value loss. Produced by author.

Figure A.3. Additional training metrics for the PPO agent, extending the analysis presented in Figure 4.13 and Section 4.3.2.2. The graphs show the progression of key indicators such as KL divergence, entropy loss, explained variance, value loss, and policy gradients throughout training. Produced by author.

LIST OF ABBREVIATIONS

Abbreviation	Definition
AEC	Architecture, Engineering, and Construction
AI	Artificial Intelligence
AIA	American Institute of Architects
ANN	Artificial Neural Network
BIM	Building Information Modeling
Box	Continuous action/observation space type in Gymnasium RL library
CA	Cellular Automata
CD	Computational Design
CoGAN	Coupled Generative Adversarial Network
CumR	Cumulative Reward
DSM	Design Structure Matrix
DRL	Deep Reinforcement Learning
GC-GAN	Graph-Constrained Generative Adversarial Network
GD	Generative Design
GAN	Generative Adversarial Network
GNN	Graph Neural Network

GPU	Graphics Processing Unit
IFC	Industry Foundation Classes
IoU	Intersection over Union
MC	Modular Construction
MDP	Markov Decision Process
MHRB	Modular High-Rise Residential Building
ML	Machine Learning
MLP	Multi-Layer Perceptron
MMC	Modern Methods of Construction
PBRs	Potential-Based Reward Shaping
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SL	Supervised Learning
UL	Unsupervised Learning
USL	Unsupervised Learning
UMM	Unified Matrix Method
WFC	Wave Function Collapse

CHAPTER 1

INTRODUCTION

Machine learning methodologies are increasingly adopted by architecture and there is a quest for useful applications of these algorithms to design (Wang & Snooks, 2021). Among them, Reinforcement Learning (RL) stands out since it has a similar procedural nature to the design process. Both design and RL consist of repetitive processes in which an agent (or designer) engages with the environment to enhance a specific objective (Kakooee & Dillenburger, 2022). Recent studies (Wang & Snooks, 2021; Akizuki et al. 2020 ; Zhu et al., 2023; Kakooee & Dillenburger, 2025; Hosmer et al., 2020) show that RL-based generative design (GD) systems can tackle complex, multi-constraint design tasks yet its application in the design process remains relatively unexplored.

One such challenging design problem is modular housing, as it includes numerous constraints related to the nature of modular construction (*mc*). *Mc* requires early design commitment, integration between off-site and on-site workflows, and involves multiple stakeholders and often conflicting constraints. In addition to this complexity, modular housing design brings a paradigm shift in design that many architects are not yet familiar with. Modular housing design is better described as a *configure-to-order* problem, where designers work with predefined modules and must explore a wide range of spatial arrangements within fixed constraints. Since the

evaluation of design options for modular housing is a highly iterative process, automation becomes necessary to develop design alternatives (Wei et al., 2021).

This thesis offers a novel generative approach that augments RL with a Cellular Automata (CA) framework for the GD of modular housing massing configuration. The study investigates how DRL can enhance modular housing design by optimizing spatial configurations, overcoming the limitations of human-driven iterations, and leveraging the benefits of a feedback-driven system.

1.1 Research Problem Statement

The *mc* concept has been brought back into focus by the latest technological developments, allowing the construction of a variety of building forms with modular units (Sharafi et al., 2017). *Mc* is promising due to its potential to address the increasing housing demands through enhanced construction speed, sustainability, and environmental benefits (Muradov & Grabovyy, 2024). Sajid et al. (2024) indicate that *mc* can alleviate the climate crisis by reducing greenhouse gas emissions, enabling a circular economy, decreasing resource intensity, promoting energy efficiency, and supporting resource-based land use and management in the construction industry. However, the design of massing configurations from modular units presents challenges due to the considerable complexity of optimizing spatial arrangements to simultaneously balance architectural, structural, and environmental performance criteria along with contextual constraints.

The emergence of artificial intelligence (AI) applications in architecture introduces considerable potential in addressing complex design problems. GD, in particular, allows for design exploration and increases productivity compared to traditional

methods by rapidly producing large numbers of customized designs. GD in architecture utilizes computational techniques to automate and optimize the creation of architectural forms, enabling more efficient and innovative design processes. Recent studies emphasize the use of algorithms, deep learning, and evolutionary strategies to support the exploration of diverse design alternatives and facilitate more informed decision-making (Ashour & Gogo, 2024). Although it holds considerable potential, GD has not yet achieved widespread adoption in practical applications (Mukkavaara & Sandberg, 2020). This is explained by the limited integration of GD solutions into the iterative nature of established design processes, particularly regarding the role of human experts in the design loop (Zhao et al., 2023).

Attempts to address the modular housing design problem through GD methods have been explored in recent literature (El-Zanfaly, 2009; Araghi & Stouffs, 2015; Almeida et al., 2016; Mekawy & Petzold, 2017; Sharafi et al., 2017; Ghannad & Lee, 2022; Bonafede & Erioli, 2022; Zhao et al., 2023; Liu et al., 2024). While some studies focus on the automatic generation of 2D floor plan layouts or apartment unit arrangements (Almeida et al., 2016; Liu et al., 2024), others investigate full 3D modular building assemblies and multi-story massing configurations (El-Zanfaly, 2009; Araghi & Stouffs, 2015; Mekawy & Petzold, 2017; Sharafi et al., 2017; Ghannad & Lee, 2022; Bonafede & Erioli, 2022; Zhao et al., 2023). Despite these advancements, existing generative approaches present several limitations. Many rely on fully predefined rule sets or large training datasets derived from existing architectural solutions, which can limit their generalizability and variation. Given the importance of a feedback-driven system in design, Deep Reinforcement Learning (DRL) presents a promising approach to bridge this gap. Unlike other machine learning methods that depend on existing data or human input, RL helps

machines learn by doing and gaining experience through trial and error (Wang & Snooks, 2021).

Particularly in housing design, balancing numerous conflicting objectives can render the manual processes tedious and time-consuming. DRL offers a solution by effectively managing complex design solution spaces while dynamically refining spatial arrangements through iterative feedback loops. This study investigates how DRL can improve modular housing design by optimizing spatial configurations effectively, addressing the limitations inherent in traditional design processes while integrating human expert feedback within a feedback-driven system.

1.2 Aim of the Study and Research Questions

The thesis aims to investigate how computational tools, particularly DRL, can be utilized to solve complex design problems by optimizing spatial mass configurations of modular housing. It offers a GD approach that augments RL with the CA framework. The study explores how DRL can enhance modular housing design by overcoming the limitations of human-driven iterations, and leveraging the benefits of a feedback-driven system.

The aim of this thesis is not to achieve a perfect modular housing design, nor to define the ideal set of parameters, constraints, or reward structures. Instead, the main objective is to provide a fundamental example for implementation of how DRL can be applied to modular mass configuration problems. This thesis intends to examine the limitations, challenges, and potentials of DRL in this context and offer a baseline for future research. More broadly, it attempts to contribute to the expansion of the range and impact of Artificial Intelligence applications in architecture. For

architectural practice, this means that RL can be considered not only as an optimization tool but as a generative partner, capable of expanding the design space and challenging predefined modular logics.

The first question of this thesis is “**To what extent DRL offers potential to enhance modular housing design?**” To answer this question, this thesis first provides a critical review of existing GD approaches in modular housing. This analysis helps to frame the design space and identify the gap where alternative computational strategies may offer added value. In parallel, the thesis examines the nature of RL as a design method, and provides a theoretical justification. This theoretical foundation is further supported by the practical implementation to improve modular housing design. In doing so, the thesis contributes not only a working implementation but also a grounded understanding of where and how DRL can be meaningfully applied in modular housing design and similar GD problems.

The second question is “**How can the design problem of modular massing be reformulated into a machine-readable format suitable for the RL environment by utilizing CA?**” This question is addressed in Section 4, where the design problem is systematically translated into a computable structure using a CA framework. The section explains how CA is employed not only to define the spatial logic of the environment, but also to structure the constraints and formulate certain components of the reward system. The reasons behind selecting CA for this specific problem are also mentioned. By illustrating how a complex architectural problem can be made compatible with an RL environment, the thesis offers a reference point for future

studies aiming to develop machine-readable design models in similarly constrained generative contexts.

The third research question is **“How should the components of a RL environment be structured to optimize mass configurations of modular housing by balancing multiple design criteria and constraints?”** This question is addressed by explaining how the design problem is formulated as a Markov Decision Process (MDP), and how the main components of the RL environment (state space, action space, transition dynamics and reward structure) is defined in response to the specific requirements of the modular housing context. The working logic of these components is explained, reflecting how their configuration is closely tied to a clear understanding of the problem structure itself. By illustrating how the RL framework can be adapted to a domain-specific architectural task, the thesis offers a practical example that may guide the development of RL environments in future GD studies on modular housing.

The last research question is **“What are the limitations and generalization capabilities of a RL-based system when applied to different modular housing scenarios or varying design environments?”** This thesis addresses this question by providing a detailed account of the development process, including the challenges encountered during training and how they were addressed. By examining the causes of these issues and reflecting on the system’s performance throughout the experiments, the study offers insights into the scalability and adaptability of the proposed RL-based approach. The step-by-step evaluation of the model’s progress helps to understand to what extent the method can be applied to different design

contexts. In particular, errors and limitations observed during the process are discussed explicitly, aiming to inform and guide future research that aims to apply RL to other architectural design problems.

1.3 Methodology of the Research

This thesis follows an applied, experiment-based methodology that combines CA with DRL. A modular housing massing problem is first abstracted into a three-dimensional CA voxel space to ensure a structured and machine-readable design environment. Within this environment, a single-agent DRL model, trained using the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm, is developed to generate and optimize spatial configurations. The agent interacts with the environment step by step, receiving feedback through a shaped reward signal aligned with architectural design goals. The RL environment is formally defined as a Markov Decision Process (MDP), including definitions of state, action, transition, and reward components. Even though both settings address the same design problem under identical spatial constraints and environment structure, they differ in terms of certain technical implementations. The main differences include changes in the action space representation and modifications to the reward policy related to Potential-Based Reward Shaping (PBRs). These adjustments aim to improve agent behavior and training stability by addressing issues observed in the initial setting. Training is conducted within a customized Gymnasium environment. Hyperparameter tuning is performed with Optuna, and learning progress is tracked through TensorBoard metrics. The results are evaluated based on defined design criterias. The evaluation also includes an analysis of training metrics recorded in

TensorBoard, such as episodic rewards, episode length to better understand the agent's learning process. The methodology also includes reflections on observed challenges and model limitations to inform future studies.

1.4 Structure of the Thesis

This thesis is composed of five chapters:

The first chapter introduces the research topic, outlines the aim and scope of the thesis, presents the research questions, and explains the methodological approach. It also provides an overview of how the thesis is structured.

The second chapter focuses on the concept of modular architecture for housing, examining its growing relevance in response to the global housing demand. It reviews the advantages of *mc* and discusses the key challenges associated with modular housing design, including the need for early design commitment, the complexity of on-site and off-site integration, and the tension between standardization and spatial flexibility. The chapter also highlights the paradigm shift from conventional building logic to configure-to-order systems in modular housing. In this context, the chapter discusses how the challenges and the paradigm shift in modular housing design have led to the need for new design approaches that can operate within complex constraints while preserving spatial quality. As a response to this, GD systems are briefly introduced as a promising direction for enhancing variety and efficiency in modular housing design, setting the stage for a deeper exploration in the following chapter.

The third chapter provides the theoretical background of the study. It defines GD within the architectural context and presents a focused overview of RL. RL has been introduced as a response to the existing challenges of GD, offering a new approach to overcome its limitations. The chapter explains how RL operates, and reviews existing applications of RL in architectural, spatial and object design domains.

The fourth chapter presents the proposed DRL-based design framework developed in this research. It introduces the voxel-based simulation environment and explains the role of the CA system in structuring spatial rules and constraints. The design task is formulated as a Markov Decision Process, and the core components of the RL environment (state, action, reward, and transition) are defined. The chapter includes two experimental settings, highlighting key differences in action representation and reward shaping (PBRs). The training outcomes and generated spatial configurations are discussed based on both visual outputs and training metrics.

The fifth chapter concludes the thesis by summarizing the research findings and discussing the strengths and limitations of the proposed method. It reflects on the performance of the trained DRL agent in generating modular housing configurations and offers suggestions for future research, including potential extensions of the framework and further integration of architectural performance criteria.

CHAPTER 2

MODULAR ARCHITECTURE FOR HOUSING

The concept of assembling buildings from prefabricated parts has existed for over a century, but modern *mc* has evolved through several key phases. Early examples of prefabrication emerged in the 19th century (Goulding et al., 2014). The Crystal Palace, designed by Joseph Paxton for the 1851 Great Exhibition in London, can be shown as one of the most prominent examples. This structure, composed of mass-produced iron and glass components, was fabricated off-site and rapidly assembled on-site. The entire construction process was accomplished in about six months, a timeline that would still be ambitious today (Chilton, 2000; Young et al., 2020). The building was also later dismantled and relocated, which proves the flexibility and efficiency of early industrialized construction techniques.

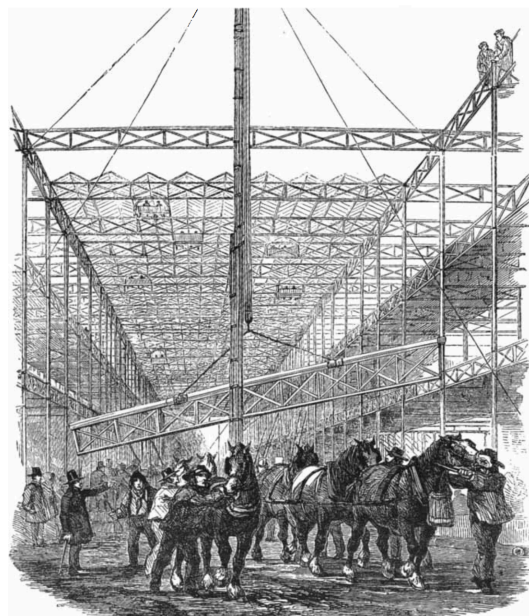


Figure 2.1 CONSTRUCTION of the Crystal Palace took only 17 weeks largely because the structure was built by highly organized crews assembling a variety of prefabricated modules. In the foreground crews lift trusses to be connected to hollow cast-iron columns. On the roof are six carts called traveling stages, which were used to install the glass panes. Reproduced for academic purposes from Kihlstedt (1984, p.10).

In the early 20th century, the Bauhaus movement further advanced the idea of standardization in design. Walter Gropius (the founder of Bauhaus) and others emphasized mass production and modular coordination as keys to addressing housing shortages. Le Corbusier's "Modulor" system (Le Corbusier, 1954) exemplified efforts to standardize building components based on human proportions, anticipating future applications in *mc* (Günay, 2025).

In the 20th century, wars and social needs accelerated the development of off-site construction. After World War II, for instance, the UK and other countries faced housing shortages and deployed prefabricated housing programs, factories turned out standardized housing units that could be transported and erected quickly to shelter populations (Young *et al.*, 2020). After World War II, the UK's housing crisis led to the mass production of temporary bungalows and portal houses (O'Neill & Organ, 2016, as cited in Young *et al.*, 2020). While these post-war "prefabs" met urgent needs, they were often rudimentary in quality and durability. Many early prefabricated homes had issues such as poor thermal performance or short lifespans, contributing to a public perception during the mid-20th century that factory-made buildings were inferior to traditional masonry or timber homes (Young *et al.*, 2020). This stigma slowed the acceptance of modular approaches for a time.

In the 1960s, architects experimented with radical modular designs, a well-known example is Habitat 67 in Montreal (built for the 1967 World Exposition), which stacked 354 identical prefabricated concrete modules into a striking multi-story

housing complex. That project gained international attention and showed that modular units could create iconic architecture.



Figure 2.2 Habitat 67, designed by Safdie Architects, exemplifies an iconic project where volumetric and modular systems were used. Reproduced from Safdie Architects (n.d.), Habitat 67. Retrieved from <https://www.safdiearchitects.com/projects/habitat-67>

Figure 2.3 Noritaka Minami, Facade I, 2011, Tokyo, Japan. Courtesy of the artist. From the 2015 Individual Grant to Noritaka Minami and Ken Yoshida for 1972-Nakagin Capsule Tower. (Reproduced from Graham Foundation, 2015, p.1)

From the 1960s to 1980s, volumetric and modular systems were used in other iconic projects as well like the Nakagin Capsule Tower and other Metabolist experiments, which aimed to explore modularity in dense urban contexts (Lawson *et al.*, 2014; El Messeidy, 2018). Although these projects were often expensive and complex, they paved the way for future applications of modular design. By the late 20th century, technology and materials had advanced, successful examples began to change perceptions, and *mc* had moved beyond temporary or low-rise buildings. Since the 1990s, load-bearing modular units have been employed in mid- and high-rise structures, including student housing and residential towers (Lawson *et al.*, 2014; Sharafi *et al.*, 2017; Ferdous *et al.*, 2019). Technological developments improved *mc* by enabling flexible design, high precision, and better integration with modern building systems (Sharafi *et al.*, 2017; Young *et al.*, 2020). These developments also supported the wider use of *mc* in various sectors such as healthcare, education, retail, and government (AIA, 2019).

Today, *mc* is seen as a strong alternative to traditional methods. It offers advantages in speed, quality, and sustainability. Kamali and Hewage (2016) note that it is widely implemented in North America, Japan, and parts of Europe. Volumetric prefabrication is now utilized in various building types, from schools and hospitals to offices, supermarkets, and high-rise housing (Lawson et al., 2014). The growing demand is largely driven by the off-site nature of production, which offers measurable economic and environmental advantages. In 1998, the UK government report *Re-thinking Construction* called for a change in the thinking of clients and the supply chain toward a partnership and less adversarial relationship, which has encouraged longer-term investment in manufacturing facilities and in developing new ways of building, which has led to an interest in the use of prefabricated construction technologies. (Lawson *et al.*, 2014).

2.1 Modular Construction

Mc is a building method in which major components of a structure are manufactured in an off-site factory and then assembled on-site. *Mc* typically employs large, pre-engineered, volumetric building modules that are delivered to and assembled at the construction site, forming integral elements of the completed building (Sharafi et al., 2017; Sajid et al., 2024; Lawson et al., 2014). This process differs from traditional methods, where construction occurs entirely on-site from the ground up. Typically, *mc* involves the production of large, three-dimensional volumetric units, also known as modules. These modules are either fully or partially completed in the factory and are designed to meet the same codes, standards, and design intent as the on-site constructed buildings. Once transported to the site, they are assembled into full structures, while not compromising from quality or aesthetics (Lawson et al., 2014; Sajid et al., 2024; AIA, 2019).

Almashaqbeh & El-Rayes (2021) state that the *mc* process integrates both off-site and on-site activities, combining prefabricated modules produced in a factory environment with structural elements constructed on-site, such as steel frames, elevator shafts, and staircases. Modules are generally finished or nearly finished within factory settings and subsequently transported to sites as primary structural components. Depending on the design, modules can be either fully enclosed or partially open-sided, allowing combinations to create larger, cohesive spaces upon assembly (Lawson et al., 2014). Nam et al. (2020) support the idea that *mc* involves minimal on-site effort to assemble the modules. Sajid et al. (2024) claim that up to 75% of the construction can occur off-site. By doing so, *mc* substantially reduces the construction time by approximately half that of on-site construction methods. In essence, *mc* relocates much of the labor and fabrication from the field to a factory, leveraging prefabrication and assembly-line techniques. It is often considered a subset of off-site construction or part of the so-called modern methods of construction (MMC) that aim to improve efficiency through industrialization (Lawson *et al.*, 2014).

2.2 Advantages of Modular Construction

Mc offers several well-recognized benefits that position it as an increasingly considered method in the building industry as an alternative to conventional on-site building methods. The modular building promises improved speed, quality, and resource efficiency compared to traditional on-site building approaches by shifting construction into a controlled environment. By producing building components off-site under controlled factory conditions, MC offers reduced project timelines, improved quality, increased control of costs, enhanced safety, and substantial

environmental benefits (Ghannad & Lee, 2022; Nam et al., 2020; Almashaqbeh & El-Rayes, 2021; Choi et al., 2017; Shahi et al., 2021; AIA, 2019).

One of the main advantages of *mc* is its potential to accelerate project completion significantly. Overall construction periods are compressed since site preparation, and module fabrication can be done in parallel. Moreover, completing most of the production in a factory environment helps avoid delays caused by weather and site conditions or labor shortages (Adeyemi et al., 2024; Almashaqbeh & El-Rayes, 2021; Gibb & Isack, 2003). As reported in many studies, the time saving can reach up to 30-50% depending on the project type and complexity. (Lawson et al., 2014; AIA, 2019; Gibb & Isack, 2003). Sajid et al. (2024) claim that in order to reduce the project completion time by up to %50, planning and execution of the work packages play a crucial role. Off-site production processes reduce site-related disruptions such as noise, theft, or material loss while speeding up construction (Kamali & Hewage, 2016). In addition to that, the need for on-site labor, equipment transport, and site overhead is significantly lower in off-site construction compared to traditional construction (Lawson et al., 2014; Choi et al., 2017).

The overall project cost of *mc* is generally more predictable. However, it may not necessarily be lower than that of traditional methods. As with conventional construction, the cost of a modular project depends on numerous variables. Although initial costs can vary, modular methods generally offer better cost-efficiency to create value in the long term if implemented properly (AIA, 2019).

In principle, modules can be disassembled and reused, allowing buildings to be relocated, extended, or reconfigured in different contexts (Lawson et al., 2014; The American Institute of Architects, 2019). The dismantle and reuse ability of the

modular components helps to maintain the asset value and allows for better long-term planning, especially in urban or rapidly changing areas (Choi et al., 2017). Moreover, factory production strengthens this adaptability by optimizing material use, minimizing waste, and making surplus easier to recycle than on conventional sites (Adeyemi et al., 2024). When modular projects are deliberately designed for disassembly, the same modules and materials can cycle through multiple uses, cutting lifecycle impacts by up to 88% and supporting circular economy goals, while routine maintenance, repair, and future upgrades become simpler (Shahi et al., 2021). Standardized "plug-and-play" connections let buildings evolve in manageable increments, and even if modules are not reused, their controlled deconstruction generates far less environmental disruption than traditional demolition (AIA, 2019).

Besides minimizing waste, *mc* supports environmental sustainability by improving energy efficiency both during the construction phase and throughout the building's lifecycle, thanks to tighter envelopes and better thermal performance (Liu et al., 2022). Quality is another area in which *mc* offers strong advantages. Since modules are built in factories using advanced tools and machines, there is more control over the precision and consistency of the components (Razkenari et al., 2020; Adeyemi et al., 2024). Repetitive processes, standardization, and the use of specialized workers help improve the overall quality of the buildings. (Lawson et al., 2014; AIA, 2019). Moreover, factory settings offer more organized operations and allow for better planning and supervision. This reduces the reliance on low-skilled on-site workers and increases output efficiency (Ghannad & Lee, 2022; Ferdous et al., 2019). In terms of safety, working in a controlled factory environment means less exposure to dangerous tasks, extreme weather, and on-site accidents (Choi et al., 2017; AIA, 2019).

In summary, *mc* can improve project outcomes in many ways. It shortens construction time, reduces costs, improves quality and safety, and offers significant environmental benefits. Its flexibility also allows buildings to be more adaptable and reusable over time. Because of these features, *mc* is considered a reliable and sustainable solution for modern building challenges, given that it is applied in the appropriate context, for the well-justified reasons, and with a clear understanding of its limitations (Adeyemi et al., 2024; Nam et al., 2020; Ghannad & Lee, 2022).

2.3 Modular Housing

Mc has evolved significantly over centuries from colonial shelters to high-rise towers. Its role in the residential sector remains particularly critical. As the global demand for sustainable, affordable, and rapidly deployable housing increases, modular approaches are increasingly seen as a viable solution to this crisis. While standardization and industrialization offer efficiency, they also introduce certain challenges. Those challenges are particularly related to design variation, adaptability, and spatial optimization. In this section, the focus shifts specifically to modular housing, its applications, and the design complexities it presents.

2.3.1 Modular Housing as a Response to Housing Shortage

Mc has become increasingly prominent in response to evolving social, environmental, and economic demands. One of the most urgent drivers of this renewed importance is the global housing shortage, which challenges both traditional and *mc* methods regarding speed, affordability, and adaptability.

The global shortage of affordable and adequate housing has become a growing concern across both developed and developing countries. This crisis is mainly shaped

by factors such as rapid urbanization, population growth, increasing land and construction costs, and the limited capacity of conventional building practices to meet current demands (Adeyemi et al., 2024; Li et al., 2023; Goodier & Gibb, 2007). In cities like Dhaka, for instance, where the population is projected to double by 2035 within an already dense metropolitan area, the pressure on housing infrastructure is becoming increasingly unsustainable (Aman et al., 2021). Similar trends are visible in many parts of the world. In the United Kingdom, persistent challenges such as unaffordability, supply deficits, and rising rents have kept the housing issue at the center of political and policy discussions (Li et al., 2023). In the United States, decades of underproduction have resulted in a shortfall of millions of affordable housing units, with some states like California annually falling behind by over 100,000 units (MBI, 2019).

The benefits of *mc* are particularly relevant to residential applications, where standardization and scalability are critical. Modular units are well-suited to multi-family housing developments such as apartment buildings, dormitories, and public housing projects. These systems enable repeatable configurations that can be efficiently produced and customized to local needs (Kamali & Hewage, 2016; Ferdous et al., 2019). Globally, countries like Japan, Germany, and Scandinavian nations have widely adopted *mc* in the residential sector. The UK and US have begun to integrate these methods into their affordable housing strategies (Young et al., 2020). Public initiatives such as London's investment in modular apartment developments and New York City's policy to require *mc* in new affordable housing illustrate a gradual shift. (AIA, 2019).

In addition to speed and productivity, *mc* supports broader systemic improvements

that extend across the housing supply chain. Modular systems provide resilience against labor shortages and scheduling delays by decreasing dependence on on-site labor. It allows more efficient project coordination. As land and labor costs continue to rise, modular methods also offer developers financial predictability and opportunities for early occupancy, which are the factors that significantly improve project applicability, particularly in dense urban settings (Bertram et al., 2019; MBI, 2019).

The growing integration of *mc* into residential development supports global sustainability goals. Its potential to rapidly deliver durable, safe, and cost-effective housing aligns with the United Nations Goal 11 for inclusive, safe, and sustainable cities (UN, 2015; Goodier & Gibb, 2007). Thus, modular housing emerges as a strategic approach that can help address deeper structural challenges within the construction and housing sectors. This growing interest in modular housing highlights a closer examination of the limitations that prevent its wider use in architectural practice, despite its potential role in addressing the housing crisis.

2.3.2 Challenges in Modular Housing

The wider adoption of *mc* faces several barriers. Ghannad and Lee (2022) note that one frequently cited reason for hesitation is the complexity and length of the design process. The design process for modular housing may be costly and time-intensive due to the repeated revisions needed to meet both project-specific and *mc* requirements. Achieving all functional goals within a limited budget remains a significant challenge (Ghannad and Lee, 2022). Below are the challenges faced in the design of modular housing.

Need for early commitment:

In *mc*, important design and planning decisions must be finalized at earlier stages compared to traditional methods. Effective implementation of *mc* therefore depends on committing to early planning from the very start of the project. If a modular strategy is adopted only after design development, many of its advantages can be lost. Modularization is more than cutting up a conventional design into parts. It calls for a modular mindset from the start, integrating geometry, materials, transportation logistics, shared scope, and coordination into a single process. Far more decisions must be made up-front in a modular approach when it compares to traditional ones (AIA, 2019 ; Bertram et al.,2019). As a result, *mc* requires a notably longer “lead-in time” during pre-planning and design. Goodier and Gibb (2007) report that for the contractors the long preparation time required for off-site construction can be seen as a disadvantage. Jaillon and Poon (2010) likewise identify that the longer design period is a major constraint for using prefabrication . If this extended lead-in period is not managed effectively, *mc* may increase early project costs and delay the start of construction. Such delays can eliminate the expected time savings and other benefits during the construction phase.

Another critical barrier that increases the difficulty and complexity of the design process is the lack of a systematic approach to module configuration. The design process requires a high level of design and production integration (AIA, 2019).

Although the planning and design stages of modular and traditional construction share similarities, project teams unfamiliar with *mc* must account for key differences. Unlike conventional design practices where flexibility and customization are relatively accessible, modular design demands early, critical decisions that impact

everything from fabrication to assembly. These decisions are tightly interlinked, and errors in the early design stages may lead to costly revisions, fabrication conflicts, or unbuildable modules (Bertram et al., 2019; Ghannad & Lee, 2022; Sharafi et al., 2017).

Need for on-site and off-site integration:

Since a substantial amount of work is completed off-site, it is important to consider the requirements of both on-site and off-site construction processes during the design stage. This also can create need for effective coordination among stakeholders essential, particularly regarding construction tolerances and project scheduling (AIA, 2019). Zhang et al. (2018) note that the need for early collaboration among architects, contractors, and suppliers in prefabrication also is a factor that lengthens the design process.

Ghannad and Lee (2022) claims that architects need a deep and specialized knowledge of the production and assembly process to design modular housing which meets multiple requirements, including user needs, building codes, and regulations. In addition, designing modular buildings requires managing a complex relationship among various objectives, such as functionality, uniqueness, standardization, manufacturability, and constructability, which can conflict to some extent (Sharafi et al., 2017). This complexity often leads to problems in *mc*, such as a lack of design flexibility and weak coordination between design and production processes.

Standardisation, flexibility and spatial complexity:

Modular design usually adheres to manufacturing constraints from the beginning. In addition to that, modules should comply with strict transport requirements, resulting

in standardized unit sizes. (Sharafi et al., 2017). It is argued that using standardized block designs can improve buildability and shortens construction time (Jaillon & Poon, 2010). On the other hand, this high degree of standardization, while beneficial for lowering production cost, may reduce flexibility in the design process (Ghannad & Lee, 2022). While standardization offers logistical advantages, it leads to an inherently rigid spatial logic and grid systems. This can limit design flexibility and may result in repetitive layouts that are unable to adapt to user needs, diverse site contexts, or evolving programmatic demands (El-Zanfaly, 2009). According to AIA (2019), some architects may assume that *mc* reduces design flexibility or limits their control. Li et al. (2023) argue that standardized construction production can further constrain design flexibility and, despite its efficiencies, may still demand substantial on-site labour. Studies have identified this as one of the main reasons behind the slow adoption of modular systems in the residential sector (Ghannad & Lee, 2022; El-Zanfaly, 2009). The configuration of functional spaces is constrained by volumetric modular units, transport requirements, and assembly logistics (Sharafi et al., 2017; Ghannad & Lee, 2022). These constraints may lead to limited variations of modular buildings beyond the loss of flexibility in design during the architectural workflow. El-Zanfaly (2009) argues that modular housing arrangements offer limited variety. The limited spatial variety in many modular housing projects can be shown as another reason which reduces their architectural appeal and weakens their ability to adapt to changing user needs over time.

2.3.3 The Paradigm Shift in Design for Modular Housing

Designing a house that meets all functional requirements while staying within a reasonable budget presents a significant challenge and entails a time-consuming

iterative process (Nauata et al. 2020). Sharafi et al. (2017) describe modular building design as a complex interplay among the intended space, the functionality of the building, and the efficient use of similarly sized modules. A fundamental challenge in modular housing lies in the required paradigm shift from traditional *engineer-to-order* approaches toward *configure-to-order* and *select-variant-to-order* systems. This change requires a modular mindset and a cultural break from the “one-off” design mentality typical of conventional architecture, adding complexity to the design process (Ghannad & Lee, 2022). In conventional construction, projects are unique and this uniqueness generally makes them integral which limits the feasibility of modularization. This presents a challenge for moving toward modular design within the traditional way of working in construction. Adeyemi et. al (2024) pointed out that there might be a need for a different approach to design and building for *mc*. For example, modular units should be tailored to the site’s specific constraints while also satisfying local building codes. This calls for considerable flexibility and creativity from the design team. Likewise, *mc* may require adjusting the site layout and foundation design to accommodate the prefabricated units. These examples show that being willing to adapt the design and construction processes to a project’s specific needs is often crucial for successfully applying modular building methods.

In this context, research-through-design provides a useful methodological lens:

Djukanović, Alegre & Bastos (2025) argue that treating design and research as an integrated, iterative loop where conceptualisation, experimentation, and refinement happen together helps designers tackle such “wicked” modular problems and generate knowledge while they create solutions.

Modular housing can offer fast, cost-effective, and higher-performance housing due

to its requiring less labor, fewer mistakes, and less waste of materials than traditional construction methods. At the same time, El- Zanfaly (2009) claims that many current applications show limited variety in their configurations. GD systems provide a promising way to address this by easing the design process, reducing time in early stages, and offering unexpected alternatives within well-defined constraints. Adeyemi et al. (2024) argue that further research and practical initiatives are essential to strengthen the use of modular and prefabricated construction in affordable housing. They stress that these studies should concentrate on streamlining design and manufacturing workflows to lower costs and improve overall efficiency.

In the next chapter, GD systems will be discussed as a potential solution to the challenges in modular housing design. The discussion will cover the aspects in which these systems provide effective solutions, as well as the areas where they remain limited. Applications of generative systems in the existing literature will be reviewed, and within the framework established by these studies, the positioning of this thesis will be clarified. The reasons for selecting DRL as a suitable approach for this study, along with the advantages it offers for addressing design problems, will be explained with reference to relevant works and debates in the literature. In this thesis, rather than emphasizing the shortcomings of modular housing alone, the focus is on exploring how RL within a generative framework can turn these limitations into opportunities for producing more diverse and adaptable design outcomes.

CHAPTER 3

REINFORCEMENT LEARNING IN GENERATIVE DESIGN

Mc offers advantages in terms of speed, cost, and performance. At the same time, modular outcomes sometimes face challenges of limited spatial and visual diversity. This is mostly due to the standardized nature of prefabricated modules, which can lead to repetitive housing typologies and limited adaptability to different user needs and site conditions (Araghi & Stouffs, 2015). While modularity introduces constraints related to uniform dimensions and standard components, it also creates new design opportunities through mass customization and flexible configurations (Laovisutthichai et al., 2020; Lee & Widrig, 2020). These opportunities are promising but not always fully realized, as the complexity of modular housing design can restrict the exploration of diverse alternatives.

Designing modular massing requires balancing architectural, structural, environmental, and contextual criteria simultaneously. This complexity can increase as more realistic constraints are introduced. Due to constraints in time and resources, architects are often compelled to narrow down solutions prematurely, limiting their ability to explore a broader spectrum of viable alternatives (Ghannad & Lee, 2022; Mukkavaara & Sandberg, 2020). In this sense, automation in the early design process becomes valuable as an augmentation of architectural decision-making. The evaluation of modular design options is inherently iterative, and manual methods are not sufficient to explore the full design space efficiently (Wei et al., 2021). Errors and

inefficiencies in the design phase may occur, especially in complex scenarios such as modular high-rise residential buildings. Automated systems, including floor plan generation tools, can reduce these errors and improve productivity across the architecture, engineering, and construction sectors (Liu et al., 2024).

GD systems can offer a promising direction for addressing these challenges. These systems use algorithms to generate a wide range of design alternatives within predefined constraints. This enables designers to explore unexpected yet valid solutions early in the process, without being limited to a narrow set of manual decisions (El-Zanfaly, 2009). Even simple generative approaches can support decision-making by integrating quantifiable parameters into the design workflow (Araghi & Stouffs, 2015). As computational design continues to evolve, scripting and algorithmic thinking have become essential tools for navigating both well-defined and ambiguous design problems. These approaches expand the role of the designer by combining computational logic with creative decision-making (Terzidis, 2006; Song et al., 2016).

Due to its repetitive and parametric nature, *mc* is especially suitable for the integration of generative methods. The ability to define flexible parameters and automatically generate varied yet buildable solutions makes GD a natural fit for prefabricated systems (Banihashemi et al., 2018; Pasetti Monizza et al., 2018; Wei et al., 2021). Yet many existing digital tools remain limited in scope. Several studies note that they may overlook structural and constructional aspects or struggle to integrate stakeholder input from the early stages (Ghannad & Lee, 2022; Mekawy & Petzold, 2017). To improve the design-to-production pipeline, further research is

needed to develop more integrated, informed, and adaptable computational tools for *mc* (Adeyemi et al., 2024).

3.1 Generative Design and Applications in Architecture

In this section, the concept of GD will be reintroduced, along with examples of its applications. The discussion will highlight the limitations identified in the literature, particularly when workflows are data-driven or heavily based on predefined constraints. The limitations in terms of restricted variation and generalizability will be addressed, and in cases where AI is not involved, the absence of learning capability has been discussed as an important limitation. Following this critique, the use of DRL in this study will be presented as a potential response to these limitations.

The concept of GD can be traced back to the 1870s, when it was initially described as systems designed to produce possible answers to a specific problem. (Zhang et al., 2021). Since the late 20th century, computational methods have been extensively applied to address design challenges within the Architecture, Engineering, and Construction (AEC) industries. The adoption of programming within architectural workflows has positioned Computational Design (CD) as a crucial tool in tackling modern architectural issues. This advancement has led to the rise of related concepts such as parametric design, GD, and algorithmic design. (Samadi et al., 2025)

GD is defined as a method where a designer collaborates with an autonomous system to develop solutions to design problems. It allows designers to utilize computational power to explore, generate, and evaluate a wide range of alternatives within predefined constraints. Compared to traditional methods, it enables rapid iteration,

expands the design space, and increases overall productivity (Mukkavaara & Sandberg, 2020; Samadi et al., 2025).

In recent years, it has been particularly relevant in architectural design. GD is described as a process where architects work through generative systems instead of directly shaping materials or forms. It enables the automatic production of diverse design options through iterative algorithms under predefined constraints. They are commonly used for form generation, spatial layout strategies, and early-stage exploration (Zhang et al., 2021; Ghannad & Lee, 2022).

Technological enhancement has transformed how architects operate, demanding updates in knowledge and workflows. In response, digital GD has become a prominent research focus. CA, genetic algorithms, L-systems, shape grammars, and swarm intelligence have been categorized as key generative strategies within this field (Zhang et al., 2021). These systems support designers in rapidly exploring potential solutions, reducing time and effort during the process, and assessing alternatives based on specific design requirements. Such methods play a crucial role in enhancing decision-making in design, where continuous evaluation of information and relationships is essential. GD is commonly used in contemporary design practices, particularly for form generation, spatial layout strategies, and exploring architectural alternatives.

The growth of artificial intelligence (AI) applications in architecture introduces considerable potential in addressing complex design problems (Ashour & gogo, 2024). (Mukkavaara & Sandberg, 2020; Samadi et al., 2025). By leveraging the superior processing capabilities of computers, this approach allows for a deeper exploration of complex design scenarios. Traditionally, it has been applied to

optimize models for specific performance goals. Furthermore, generative systems can reveal novel or unexpected design solutions and shed light on underexplored areas of the design space. (Samadi et al., 2025)

Contemporary research showcases how computational techniques, such as deep learning, algorithms, and evolutionary strategies are being used to facilitate architectural design processes and expand the exploration of design options (Ashour & Gogo, 2024). GD has found diverse applications across multiple fields such as construction, urban planning, and manufacturing (Wei et al., 2021).

In recent decades, numerous studies have investigated automated building design and spatial layout generation, demonstrating their considerable potential to enhance architectural design processes. (Ghannad & Lee, 2022). Although it holds considerable potential, GD has not yet achieved widespread adoption in practical applications (Mukkavaara & Sandberg, 2020). This is explained by the limited integration of GD solutions into the iterative nature of established design processes, particularly regarding the role of human experts in the design loop. (Zhao et al., 2023).

3.1.1 Generative design applications in modular housing design

Attempts to solve the modular housing design problem through GD have been documented in the literature (Araghi & Stouffs, 2015; Almeida et al., 2016; Mekawy & Petzold, 2017; Sharafi *et al.*, 2017; Ghannad & Lee, 2022; Zhao et al., 2023 ; Liu et al., 2024). Some of these studies focus on automatically generating 2D floor plan layouts or apartment unit configurations (Almeida et al., 2016; Liu et al., 2024), while others explore full 3D modular building assemblies and multi-story

configurations (Araghi & Stouffs, 2015; Mekawy & Petzold, 2017; Sharafi et al., 2017; Ghannad & Lee, 2022; Zhao et al., 2023). Together, these contributions show how generative approaches can broaden the early search space of modular design. However, they also reveal challenges.

Several studies note that existing generative methods often remain tied to predefined constraints or large training datasets, which can reduce their adaptability to new design criteria (Araghi & Stouffs, 2015; Ghannad & Lee, 2022 ; Zhao et al., 2023). Approaches that generate 3D layouts tend to converge on a narrow set of solutions, producing limited variation and focusing on optimizing a single outcome (Sharafi et al., 2017; Ghannad & Lee, 2022). Many frameworks also assume standardized module sizes and shapes, which makes it difficult to capture irregular geometries or more diverse unit dimensions.

Beyond these geometric limitations, some frameworks also treat modules as generic blocks rather than differentiated dwelling units. As a result, issues of program richness, spatial diversity, and architectural meaning remain underexplored. Unlike these static approaches, DRL introduces a feedback-driven mechanism that can iteratively adapt to new contexts, improve performance, and balance multiple design goals without manual reprogramming.

Araghi and Stouffs (2015) investigate the use of CA as a GD method for high-density residential building form generation. Their aim is to develop a computational tool that translates architectural requirements, such as accessibility and natural light, into CA rules, enabling the creation of diverse and complex spatial configurations at the early stages of the design process. The method relies on diagrammatic rule definition and rule-based iteration within a 3D voxel lattice, using

fixed housing types (e.g., studio, one-bedroom, two-bedroom units) and neighborhood-based criteria.

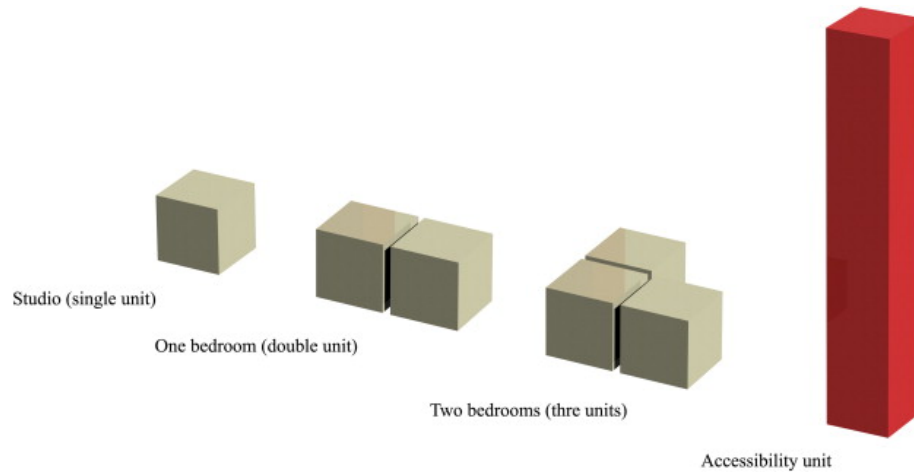


Figure 3.1. Types of dwellings including studio, one- and two-bedroom apartments. Reproduced for academic and illustrative purposes from Araghi and Stouffs (2015, p. 4), *Exploring cellular automata for high density residential building form generation*, Automation in Construction.

While this approach aligns conceptually with modular massing, particularly in its use of voxel grids and modular logic, it operates deterministically and depends on pre-defined rules. As a result, the method can produce meaningful patterns but lacks the adaptability and iterative learning capacity that more advanced AI-based systems allow. In this sense, its contribution is in showing how architectural requirements can be encoded into computational rules, though its scope remains limited to behaviors explicitly modeled by the designer.

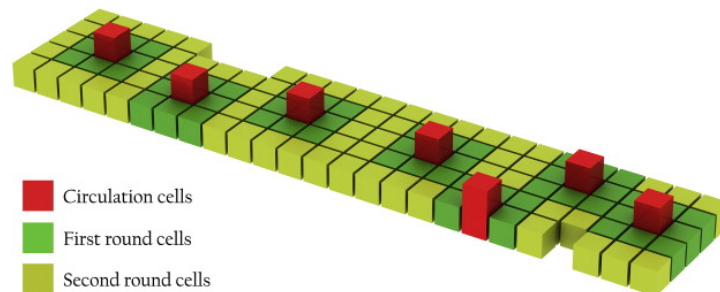


Figure 3.2. 3D illustration of neighborhood patterns in a 5×25 grid of cells. Reproduced for academic and illustrative purposes from Araghi and Stouffs (2015, p. 5), *Exploring cellular automata for high density residential building form generation*, Automation in Construction.

Sharafi et al. (2017) introduce a unified matrix-based framework for the automated spatial design of multi-story modular buildings. This approach addresses a critical need in *mc*: optimizing layout configurations early in the design process while accounting for structural, architectural, and constructional constraints. UMM tackles the harder question of how to place those modules floor-by-floor under multiple, often conflicting, criteria. The proposed method models modular design as a three-dimensional assignment problem, using a design structure matrix (DSM) to represent module placements and their interconnections. Their framework integrates architectural, structural, and constructional aspects early in the design process, to represent spatial configurations and interconnections between modular units. The aim is to generate cost-effective, structurally stable, and energy-efficient building forms by minimizing objective functions such as plan irregularity, energy inefficiency, and height-dependent construction cost. The method assumes a discrete grid space where module placements are optimized using combinatorial logic and constraint-based heuristics (e.g., Ant Colony Optimization). A case study on a three-story modular tourist accommodation demonstrates the method's effectiveness in producing Pareto-optimal configurations that reduce construction costs by approximately 30%, improve plan regularity, and minimize façade area to enhance energy efficiency. This integrated design approach proves especially beneficial during early-stage decision-making, where design choices significantly impact the project's lifecycle performance and feasibility.

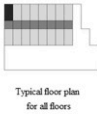
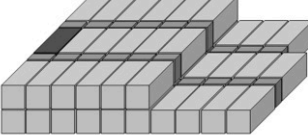
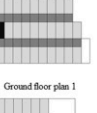
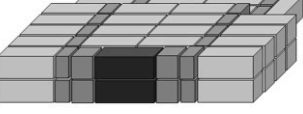
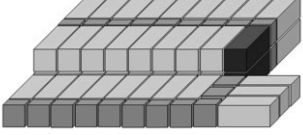
Spatial Design	Floor Plans	Total Cost
 The existing spatial design	 Typical floor plan for all floors	1681 units
 Spatial Design 1 by the Unified Matrix Method	 Ground floor plan 1 First floor plan 1	1222 units
 Spatial Design 2 by the Unified Matrix Method	 Ground floor plan 2 First floor plan 2	1241 units
 Spatial Design 3 by the Unified Matrix Method	 Ground floor plan 3 First floor plan 3	1250 units

Figure 3.3. Three optimal solutions for the spatial design of the case study. Reproduced for academic and illustrative purposes from Sharafi et al. (2017, p. 11). Automated spatial design of multi-story modular buildings using a unified matrix method. *Automation in Construction*, 82, 31–42.

Sharafi et al. (2017)’s study is a valuable reference for modular housing design and performance optimization discussed in this thesis. However, its reliance on pre-defined cost functions, heuristics, and deterministic optimization means that while it integrates multiple criteria effectively, it does not evolve or generalize beyond the scenarios encoded at the start. Their method uses deterministic optimization through matrix manipulation and metaheuristics (e.g., ant colony algorithm). In this sense, it not only demonstrates the benefits of multi-objective modeling, but also the limits of non-learning approaches.

Ghannad and Lee (2022) propose an end-to-end workflow for automatically generating and modularising single-storey house plans. The study aims to develop an

integrated, automated framework that generates constructible and feasible modular housing layouts. It addresses key challenges in *mc*, such as the reduced flexibility of design, lack of automation, and the need to incorporate manufacturing, transportation, and assembly constraints early in the design process. A rule-based algorithm decomposes generated layouts into viable module configurations based on geometric and constructability constraints (e.g., max dimensions, weight, ratio). A pixel-wise algorithm translates layouts into coordination-based vector models. Designs are filtered for functionality (connectivity, circulation, compactness). Feasible configurations are converted into IFC-based 3D BIM models for practical implementation.

The framework couples a generative adversarial network (CoGAN) with a rule-based module configurator and BIM export. While this creates a strong pipeline from GD to production, its limitation lies in being heavily data-driven and lacking multi-objective optimization. It only filters out infeasible options, without comparing or improving alternatives. In this sense, it is a useful step toward automation, but it still carries typical GAN limitations such as data dependency and lack of flexibility.

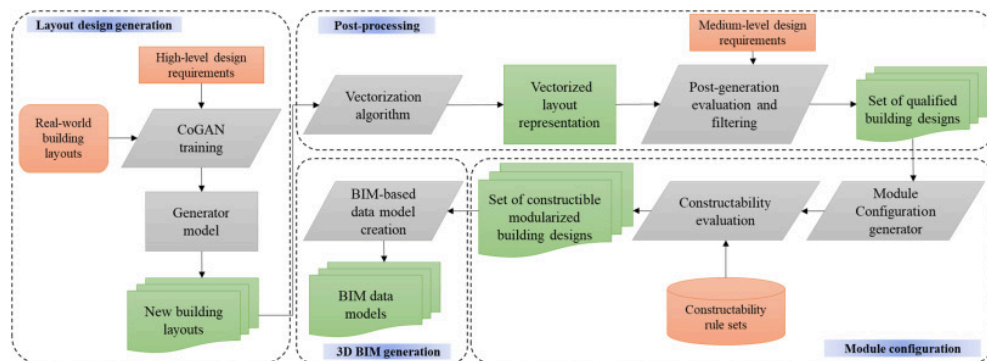


Figure 3.4. The overview of the research methodology. Reproduced from Ghannad & Lee (2022, p.5). Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN). *Automation in Construction*, 139, 104234.

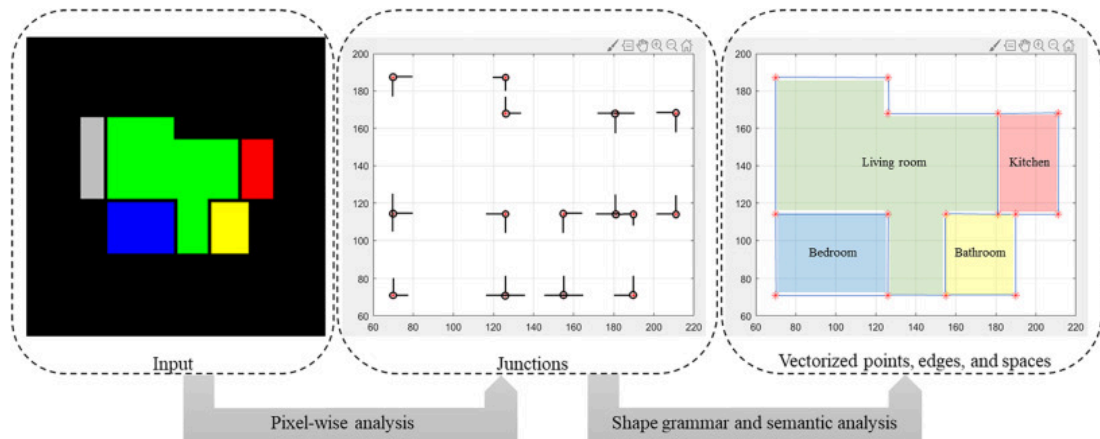


Figure 3.5. Example showing the vectorization process in modular housing design. Reproduced for academic and illustrative purposes from Ghannad and Lee (2022, p.8). Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN). *Automation in Construction*, 139, 104234.

Samadi et al. (2025) introduce a generative framework for producing floorplan layouts within fixed linear boundaries by utilizing modular “Slices” that correspond to functionally complete parts of housing units (e.g., bathroom, kitchen, bedroom). This framework addresses the internal spatial configuration of the housing unit itself. Implemented using GhPython and Grasshopper, the system allows users to define space requirements and entrance positions, which the algorithm uses to sequentially fill the apartment layout based on adjacency rules and spatial ratios. Each generated solution is then automatically transformed into a 3D layout with furniture, walls, and doors, providing not only design alternatives but also spatially legible outputs suitable for end-user evaluation and early-stage BIM integration. The design space is expanded through a catalog of 173 predefined modular Slices, enabling flexibility despite the absence of data-driven learning or optimization routines. The slice-based system creates interior layouts from modular components. The system offers variety and clarity, but its outcomes remain fixed to a predefined library. Without optimization or adaptive feedback, the process generates alternatives that are limited in scope.

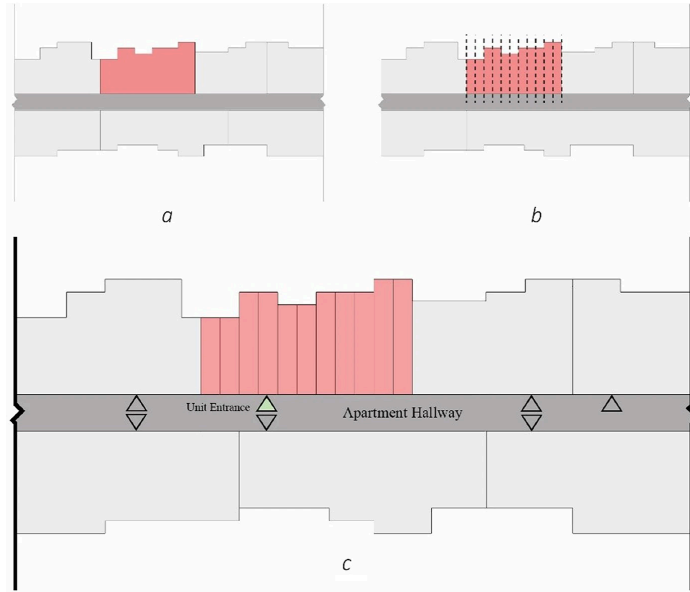


Figure 3.6. a) Considers the primary apartment outline, b) Divides the outline into equally-lengthed slices, c) Divides the resulting residential floorplan outlines located on both sides of the hallway, with a specified entrance slice .Reproduced from Samadi et al. (2025, p.7)

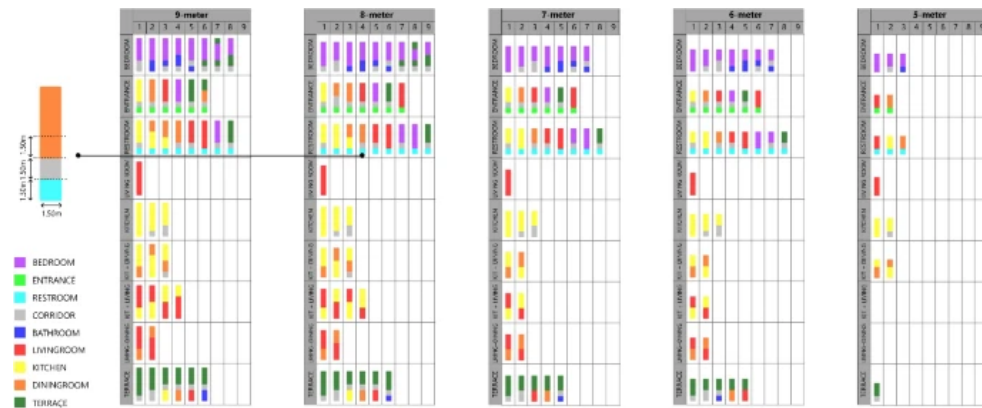


Figure 3.7. a) CCategories created from 173 designed Slices based on length and common combinations of functional spaces. Reproduced from Samadi et al. (2025, p.8)

Zhao et al. (2023) present a novel framework for designing and evaluating architectural tilesets within the context of GD, increasingly popular in architecture for its efficiency and potential for creativity compared to traditional methods. This study addresses the gap in research focusing primarily on discovering designs rather than fine-tuning tilesets themselves. By introducing metrics for evaluating generated layouts and the tileset design space, the research facilitates practical applications and allows architects to customize tilesets based on specific objectives. The methodology

exemplified through double-floor single-family house tilesets utilizes the Wave Function Collapse (WFC) algorithm for generating 3D designs and 2D layouts. This process enables the creation of diverse design options from minimal tileset specifications and demonstrates how fine-tuning can overcome the common issue of grid-like monotonicity in tile-based GD methods. The framework not only simplifies the design process by enabling the creation of tilesets with minimal specifications but also introduces a novel approach to evaluating the quality of generated layouts and the overall design space of the tileset. This comprehensive method supports iterative design processes, offering a significant contribution to the field of generative architectural design and providing a basis for further research and development in the area.

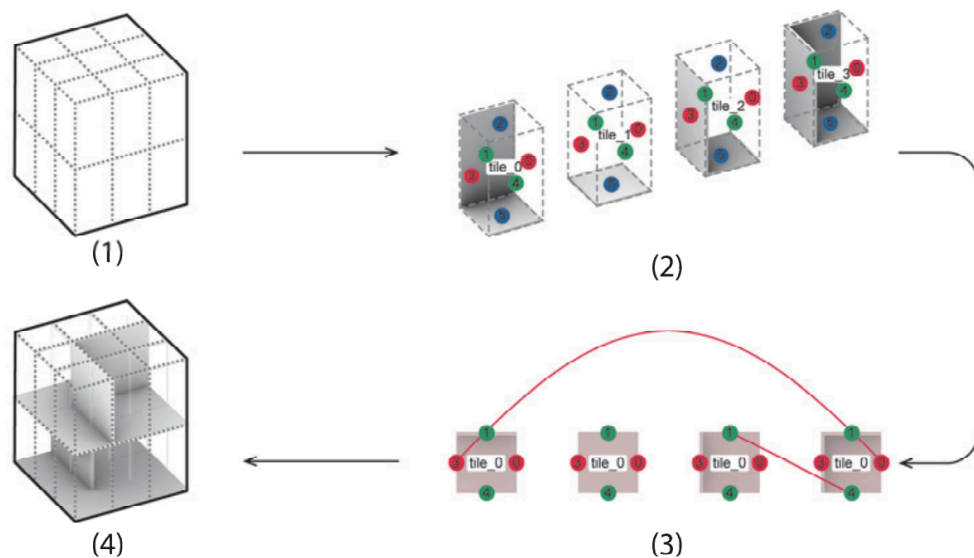


Figure 3.8. WFC generates a design, by filling 3D empty slots (1), with predefined tiles (2), following adjacency rules (3) to generate a combination (4)

Zhao et al. (2023)'s study serves as a valuable example for the modular approach and iteration optimization in the thesis, yet it diverges in both scale and methodology. In the article, iterations are conducted parametrically using the WFC algorithm. WFC

is an algorithm that can procedurally generate images, text, audio and almost anything that follows a pattern of a simple set of initial parameters and a set of rules. However, in this thesis, these iterations are performed using RL, employing a reward-punishment logic to define parameters. This distinction highlights the innovative methodological shift in the work towards leveraging advanced computational strategies to achieve modular iterative optimization in architectural design to serve environmental sustainability.

Liu et al. (2024) proposes a graph-constrained generative adversarial network (GC-GAN) to automate the floor plan design process of modular high-rise residential buildings (MHRBs). In this approach, domain-specific architectural knowledge, such as spatial connectivity, circulation, compactness, and modular production requirements, is first represented as a knowledge graph. This graph then guides the GAN model to generate realistic and functional floor plans that comply with *mc* standards. The generator creates segmentation-based layout outputs, while the discriminator evaluates their validity based on both real samples and design constraints. The generated floor plans are automatically matched with a standard modular flat design library, and then transformed into BIM models using Revit, making the system directly applicable to production workflows.

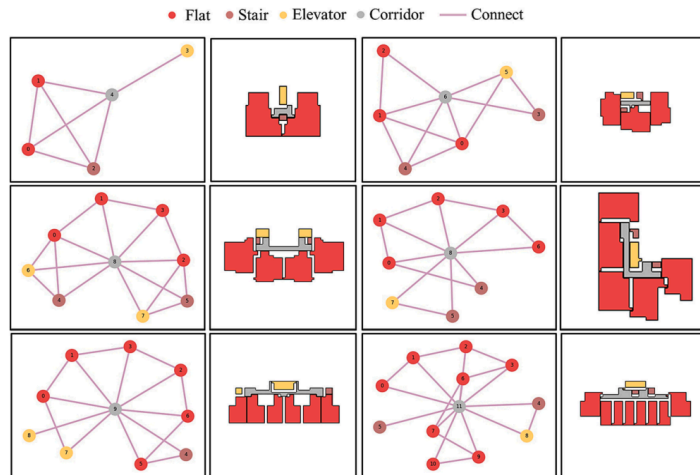


Figure 3.9. a) Sample images prepared for training data and corresponding knowledge graphs. Reproduced for academic and illustrative purposes from Liu et al. (2024, p.7), *Intelligent floor plan design of modular high-rise residential building based on graph-constrained generative adversarial networks*, *Automation in Construction*, 159, 105264.

To evaluate the method, the authors use a combination of expert scoring (for realism), Frechet Inception Distance (for diversity), and graph edit distance (for topological accuracy). Results show that the system can produce diverse and feasible layouts that align with key design goals, and significantly reduce the design-to-production time compared to traditional methods. This GC-GAN framework combines architectural knowledge with generative models for modular floor plans. It reduces design time and ensures feasibility. Its limitation lies in dependence on large training datasets and a lack of adaptability to objectives not encoded in the graph.

In summary, while these generative approaches represent valuable steps toward automating modular housing design, they often operate within predefined rules or depend heavily on existing datasets, which can make it difficult to adapt to new or evolving design goals. This reliance frequently results in a narrow solution space with limited variation, as noted in studies such as Sharafi et al. (2017) and Ghannad & Lee (2022). As emphasized by Mukkavaara and Sandberg (2020), the iterative act

of navigating between exploration and exploitation in design requires tools that can not only generate a range of alternatives but also refine them through feedback.

Within this context, RL does not replace these approaches but extends them, offering a stochastic, feedback-driven process that begins from random initializations and explores diverse alternatives before converging on refined solutions. By learning from interaction and balancing multiple objectives, RL provides an adaptive framework capable of incorporating variation, responding flexibly to constraints, and potentially addressing one of the central limitations of earlier GD methods. The next section will further explore how RL can serve as a GD strategy, capable of bridging exploration and exploitation, through its capacity to learn from interaction, adapt to constraints, and iteratively improve design outcomes over time.

3.2 Reinforcement Learning

Unlike data-driven generative models, an RL agent can iteratively improve layouts via feedback without requiring a pre-existing dataset or an exhaustive rule set of constraints. This feedback-driven approach enables more flexible adaptation to emerging design objectives and constraints, and offers more generalizable solutions. Early studies have shown the potential of deep RL in design contexts. Building on these advantages, this thesis proposes a RL-based GD approach that learns through trial-and-error interactions within a simulated environment. The RL-based system in this research dynamically learns how to configure housing massing while satisfying overall design requirements. In summary, by addressing the limitations of generative systems, the proposed RL generative approach aims to fill the identified research gap, enabling more adaptive, constraint-flexible, and variety-rich solutions for modular housing design.

3.2.1. Artificial Intelligence, Machine Learning, and Reinforcement Learning

Artificial intelligence (AI), in the literature, in its simplest definition, is defined as machines that can perform tasks that humans can accomplish by thinking, according to Dörfler (2022). Artificial intelligence has evolved and developed since the Dartmouth summer workshop, which is considered to be the founding event of artificial intelligence. In recent years, AI technologies have rapidly progressed, especially with the development of deep learning architectures and increased computational power. The rise of powerful processors, GPUs, and access to large-scale data has enabled AI systems to perform increasingly complex tasks across domains ranging from language processing to autonomous driving (Akizuki et al., 2020). In the field of architecture, these developments have facilitated the emergence of computational design workflows where AI supports tasks such as spatial optimization, energy analysis, and design automation (Akizuki et al., 2020).

The integration of AI into various domains has significantly accelerated in recent years, primarily due to the rapid advancements in computational hardware and algorithmic innovation (Akizuki et al., 2020). These technological improvements have not only expanded the capabilities of AI systems but also made them more accessible across disciplines, including architecture, where they assist in designing, analyzing, and optimizing spatial configurations and structures (Akizuki et al., 2020).

Machine Learning models in the AI ecosystem can approach a phenomenon through iterative exposure to large amounts of data, called ‘learning’. In this way, the model can be used to predict or simulate the same phenomenon under different parameters with its learning. Machine learning algorithms have already outperformed human experts in complex strategy games. Akizuki et al. (2020) claims that this

demonstrates the potential for ML to generate design solutions that may surpass human capabilities, as design itself involves abstract reasoning and strategic decision-making similar to these games. The field of architecture is increasingly engaging with machine learning techniques, exploring how these algorithms can be effectively applied within design processes (Wang & Snooks, 2021).

Machine learning is generally divided into three subcategories depending on the model's training strategies: supervised, unsupervised, and reinforcement:

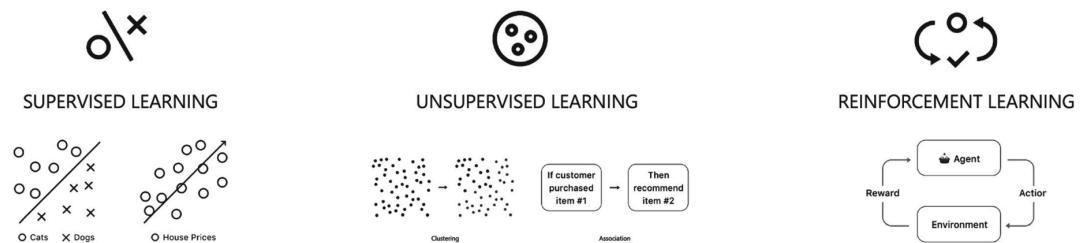


Figure 3.10. Diagram of three typical types of machine learning: Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL). Reproduced from Wang and Snooks (2021), *Artificial Intuitions of Generative Design*, in *DigitalFUTURES*, Springer. Licensed under CC BY 4.0.

In supervised learning, learning is provided by matching input-output data pairs (labeled data). It is powerful but relies on large, carefully curated datasets that encode the “correct” answers (Sutton & Barto, 2018). In recent years, supervised learning (SL) has achieved remarkable advancements across various domains such as data prediction, image generation, and natural language processing. Despite this progress, its application in three-dimensional GD remains limited according to Wang and Snooks (2021). A key limitation of SL is its dependency on large, labeled datasets, which is often inefficient and impractical in design contexts. Although some labeling tasks can be automated, the feedback process in artificial neural networks (ANNs) operates in a fixed linear manner, regardless of the variability or complexity

of the generated data. This rigid structure contrasts with the adaptive and exploratory nature of GD workflows. Moreover, representing and manipulating three-dimensional geometries within SL frameworks (3D GANs) poses significant computational challenges, which further limits their effectiveness in spatial design problems.

In contrast, in unsupervised learning, the data is not matched with any output value; that is, it is not labeled (Chaillou, 2022). The learning process is completely left to the model, depending on the observation of data. USL focuses on identifying patterns or associations within the unlabeled data rather than optimizing toward a defined objective. This lack of a clear goal makes it less suitable for GD tasks that require a directed design intention or targeted outcomes (Wang & Snooks, 2021).

In RL, the process is based on behavior update based on the reward system (Chaillou, 2022). RL is a robust framework with various successful applications in different fields like video games, dynamic systems control, finance, healthcare, and self-driving cars, thanks to its capacity to solve complex sequential decision processes. RL is a trial-and-error approach to machine learning based on a trade-off between exploration and exploitation within a simulation environment instead of learning from a curated dataset necessary for supervised machine learning algorithms (Sutton & Barto, 1998).

In RL, an autonomous agent learns by gaining experience through interaction with an environment. The agent sequentially performs actions that alter the state of that environment, and the environment in turn gives feedback in the form of a numeric reward signal to the agent to evaluate the quality of its policy of action given a set of objectives. The agent aims to develop a behavior or policy that maximizes its

long-term rewards (Goodfellow et al., 2016). So, RL enables machines to learn and make decisions autonomously in pursuit of defined objectives (Cao et al., 2021).

In GD, RL can offer some important advantages. Wang and Snooks (2021) classified contemporary GD approaches into two as parametric-based and behavioral-based. Parametric systems rely on manipulating parameters that follow predefined, linear relationships with geometric outcomes. In contrast, behavioral-based systems encode local design intentions, resulting in bottom-up, self-organizing processes. Although both approaches can yield complex and innovative results, each has inherent limitations. Parametric models are constrained by their predefined parameter-geometry relationships. Behavioral systems, on the other hand, emphasize local interactions but lack global awareness, making it difficult to embed overarching design intentions. Also, integrating real-time material and structural performance into these non-linear workflows remains a significant challenge due to the instability of such systems. (Wang & Snooks, 2021) RL stands between these two methods. It uses local actions like behavioral systems, but it also follows long-term goals through reward signals.

As noted by Kakoei and Dillenburger (2025), RL has structural similarities with architectural design processes, as both rely on iterative feedback. RL agents learn by taking actions in an environment and receiving rewards, mirroring the trial-and-error nature of design. In this way, RL also functions as a generative tool. Wang and Snooks demonstrate how RL can generate architectural forms based on performance goals. Unlike fixed parametric systems, RL allows agents to learn adaptive strategies, discover non-obvious solutions, and respond to evolving constraints. Akizuki et al. (2020) suggest that RL can balance between strict design constraints and creative

freedom. RL operates through an agent interacting with its environment, learning an optimal policy via feedback between observed states and reward signals formally modeled as a Markov Decision Process. Unlike other ML methods, RL relies on self-experience rather than predefined human knowledge. Its core structure includes key elements: agent, states, actions, rewards, and policy. (Wang & Snooks, 2021). RL does not only assist design, but takes an active role as a decision-maker. This is possible because of these three basic parts of RL: actions, states, and rewards. These elements allow both control and variation.

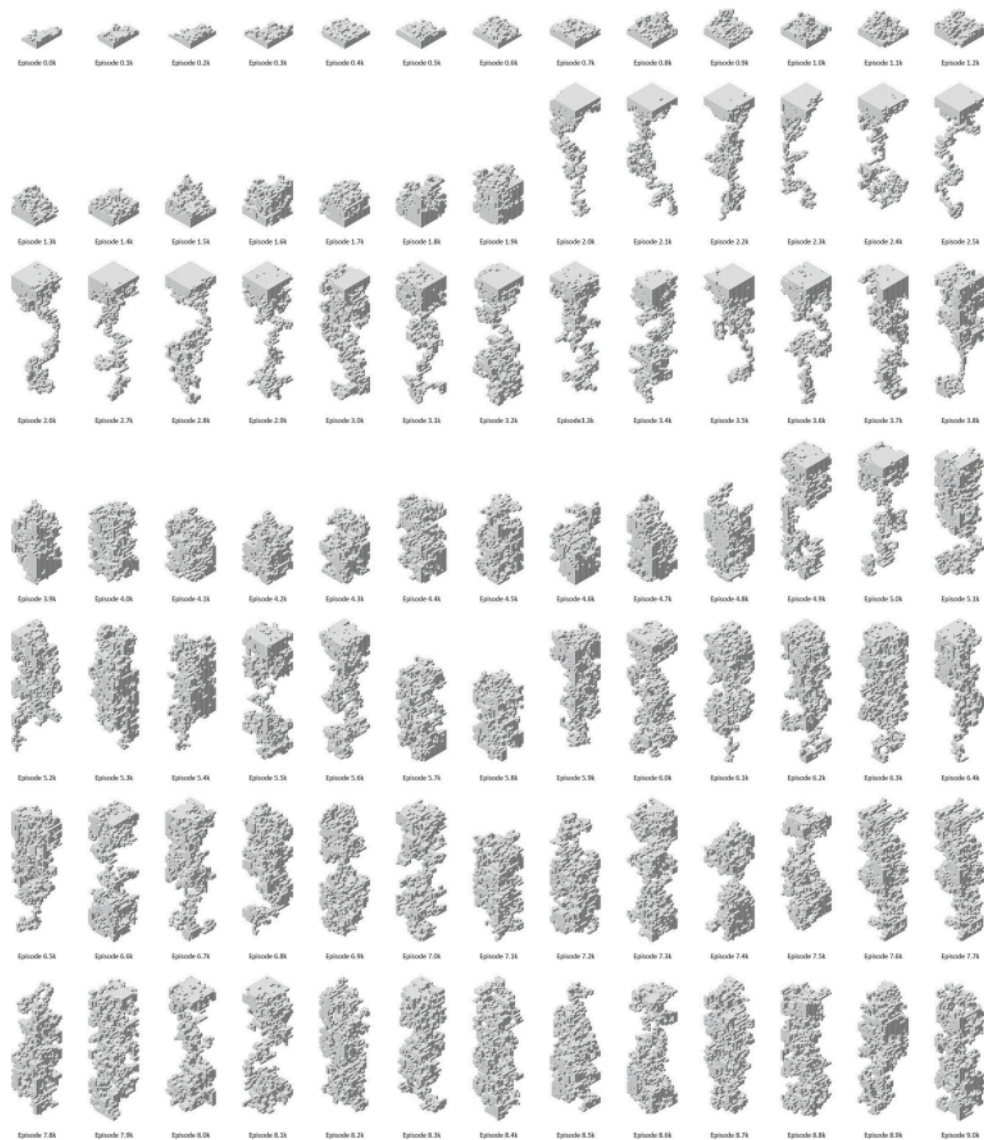
Kakoe and Dillenburger (2025) claims that many existing AI-based methods in design use pixel-based or black-box systems that don't clearly represent form, function, or spatial logic. They pointed out that this limits their use in real design tasks. They argue that there is a need for more intuitive and structured systems. RL is well suited for this because it can work with performance-based rewards and structured decision-making.

3.2.2 Reinforcement Learning and its Applications in Design

Several recent studies have explored the use of RL in architectural design, particularly within generative workflows.

One notable example is the work by Wang and Snooks (2021) explores how RL can be used to integrate design intentions into existing generative workflows. Instead of replacing current systems, RL is implemented as a control mechanism that guides the generative process through reward-based feedback. The authors compare different machine learning types and argue that RL is more aligned with GD due to its heuristic and sequential structure. They present a Random Walk-based experiment

where an RL agent generates voxel-based spatial forms by learning through predefined rewards. These rewards reflect architectural goals such as vertical growth, structural logic, spatial connectivity, and avoiding site collisions. The agent is trained using Deep Q-Learning, and the results show that the system gradually develops a sense of direction and begins to produce more coherent outcomes aligned with the design criteria.



*Figure 3.11. Selected training outcome samples from episode 0 to 9.4 k. Reproduced for academic and illustrative purposes from Wang and Snooks (2021), *Artificial Intuitions of Generative Design*, in *DigitalFUTURES*, Springer. Licensed under CC BY 4.0.*

While the setup is relatively simple, the experiment demonstrates how RL can be used to embed adaptive behavior and design intuition into generative systems.

Another study by Wang and Snooks (2021), which presents a RL-based GD approach that combines top-down and bottom-up processes within a single framework. Their study proposes the integration of a RL agent with multi-agent behavioral systems to overcome the limitations of local interaction logics and enable a more global design awareness.

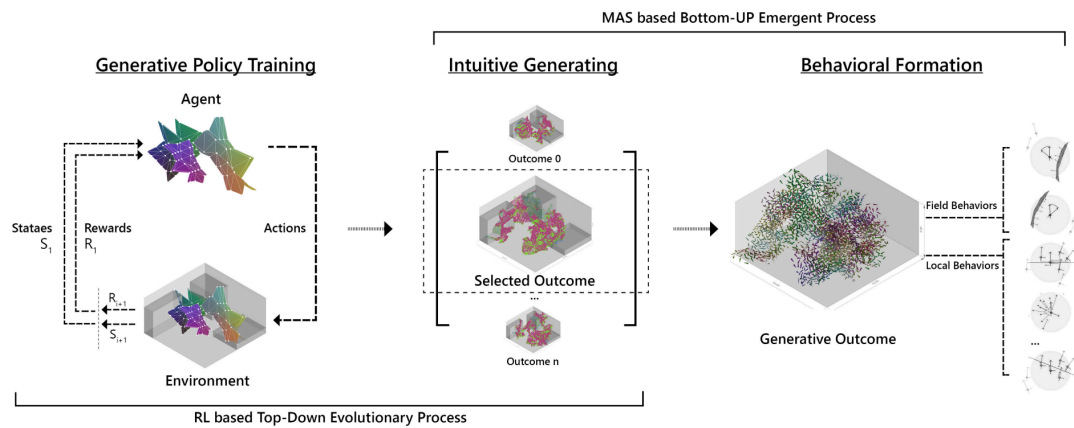


Figure 3.12. Overall method of structuring the RL-based intuitive generative approach. Reproduced for academic and illustrative purposes from Wang and Snooks (2021, p. 4), *Intuitive Behavior: The Operation of Reinforcement Learning in Generative Design Processes*, CAADRIA 2021 Proceedings.

The RL agent is trained using the PPO algorithm within Unity's ML-Agent toolkit to optimize a mesh-based geometry in response to pavilion-scale design criteria. The training process incorporates spatial, structural, and environmental performance goals into a set of global reward functions. The trained generative policy produces a topological field that is later used to guide a multi-agent formation process. While the approach shows promise in producing adaptive and intention-driven design outcomes, the authors also highlight computational challenges and the need for more advanced multi-agent RL methods. Overall, the study demonstrates how RL can be used not just to generate architectural forms, but also to embed design intention and performance logic into the generative process.

Akizuki et al. (2020) explores the potential of using RL within the realm of GD to produce objects that not only exhibit unique aesthetics but also adhere to specified design and fabrication constraints. The authors aim to address challenges in the state of the art, such as topological inconsistency, limited style variation, and design unpredictability, by developing a machine learning framework that acts as a sophisticated design interpreter. This framework is capable of transforming input geometries into new forms that meet both aesthetic and practical requirements. Through a series of experiments, including the generation of table designs, the study demonstrates that RL can successfully generate objects that maintain topological consistency under the given constraints. The methodology involves training an agent to navigate within the space of an input geometry, with the goal of shaping a new geometry that fulfills design objectives while staying within the confines of the input shape. This approach represents a significant advancement in GD, highlighting the power of RL as a tool for achieving design goals within specified constraints. Although this study differs significantly from the thesis in terms of design scale, its approach to using modular blocks and RL in a 3D environment serves as a methodological exemplar for the thesis. The "LEGO block model" in study illustrates the use of RL to navigate modularity in design, using a set of predefined LEGO blocks to generate table designs within specific constraints. This approach showcases the potential of algorithmically creating diverse and complex structures from a limited palette of modular components, emphasizing the integration of design and fabrication constraints within the GD process.

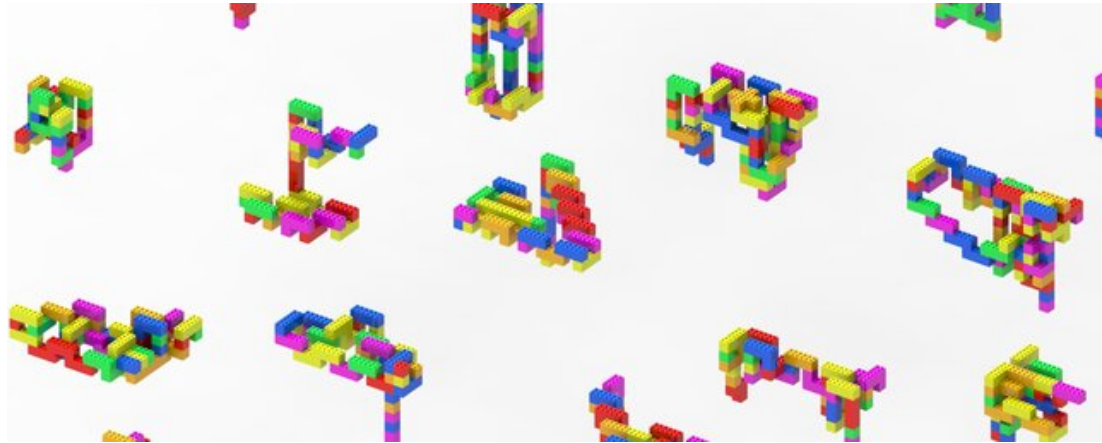


Figure 3.13. Object generation using a LEGO block-based model. Reproduced for academic and illustrative purposes from Akizuki et al. (2020), *Generative modeling with design constraints: Reinforcement learning for object generation*, presented at the 25th CAADRIA Conference.

Kakooee and Dillenburger (2025) present a RL-based approach for architectural space layout design (SLD), using a novel partitioning method called *laser-wall*. Their method conceptualizes walls as light-emitting elements that divide space through beam interactions, bridging pixel-based and vector-based partitioning strategies. The study introduces two RL-compatible planning strategies for generating layouts and focuses on iterative refinement techniques to improve spatial configurations. To support this, the authors develop a custom OpenAI Gym-compatible simulator for layout generation and evaluation. The RL agent, trained using the PPO algorithm, processes image-based layout states and improves them over time through wall movement and rotation actions. Rewards are based on both geometric constraints (such as area and aspect ratio) and topological requirements (such as room adjacency and entrance connection). The results show that the agent is capable of generating coherent and functional layouts that align well with design targets.

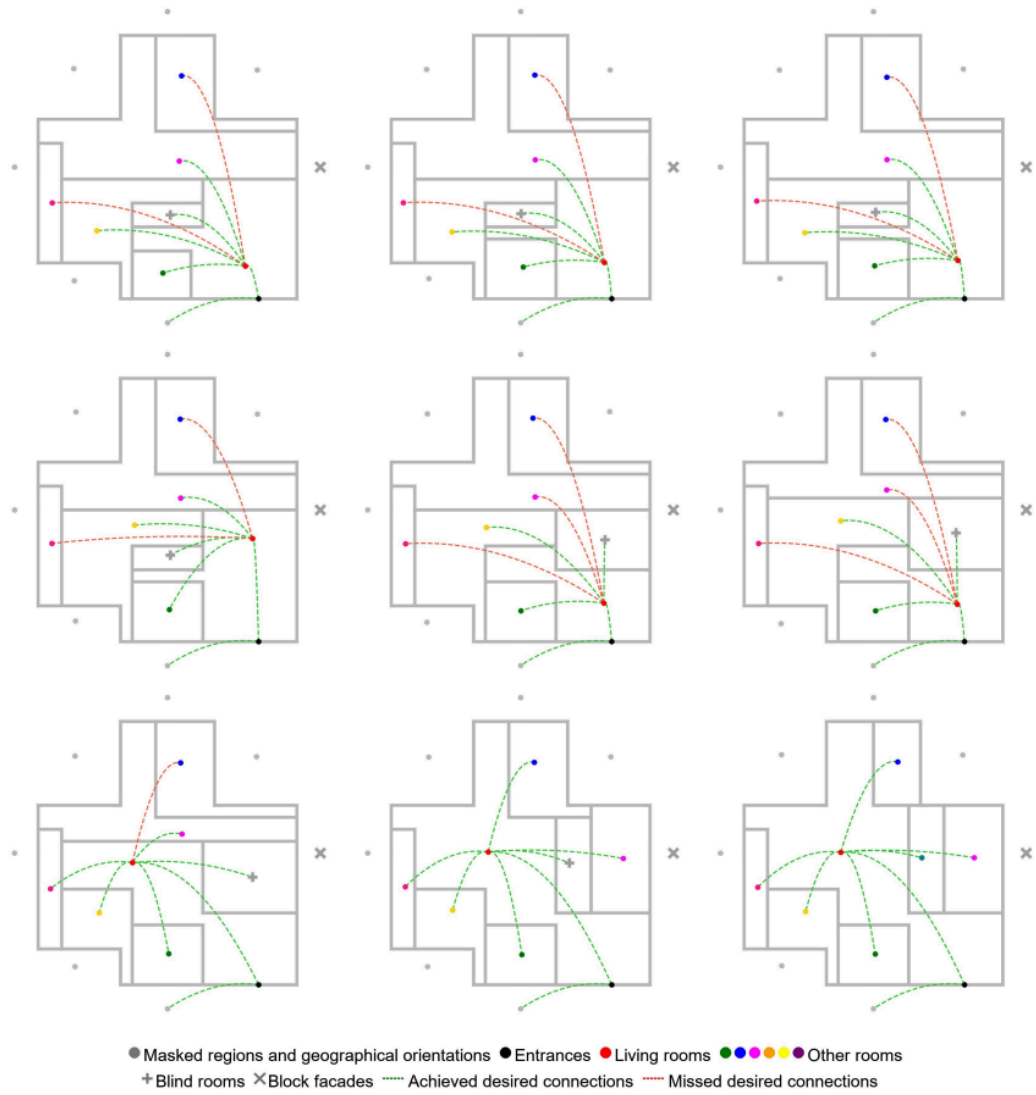


Figure 3.14. Agent's action trajectory from initial random layout to final optimal layout.. Reproduced for academic and illustrative purposes from Kakooee and Dillenburger (2025), *Illuminating Spaces: Deep Reinforcement Learning and Laser-Wall Partitioning for Architectural Layout Generation*, ETH Zurich, arXiv preprint.

This study demonstrates how RL can be used not only to explore spatial alternatives but also to refine and improve layout quality through procedural and adaptive learning strategies.

A recent study by Zhu et al. (2023) investigates the integration of DRL into real-time robotic assembly planning within prefabricated construction environments. The paper addresses the limitations of traditional decision-making methods, particularly

their inflexibility in the face of dynamic site conditions, such as changing layouts, component variability, and uncertainty in task sequences. Instead, the authors propose a DRL-based framework that enables robots to learn assembly strategies autonomously through continuous interaction with a simulated environment, modeled using a combination of BIM data and a voxel-based 3D simulator.

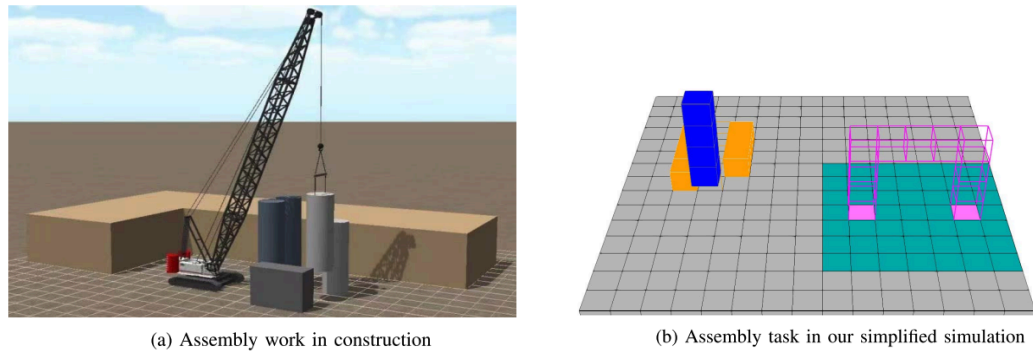


Figure 3.15. Overview of the construction simulations. Reproduced for academic and illustrative purposes from Zhu et al. (2023), Deep Reinforcement Learning for Real-Time Assembly Planning in Robot-Based Prefabricated Construction, IEEE Transactions on Automation Science and Engineering, 20(3), 1515–1526.

This method is particularly effective for planning workflows in complex, real-world construction settings, where the sequence and location of tasks are not fixed but must adapt to spatial constraints and operational safety. By framing the assembly process as a Markov Decision Process (MDP), the approach allows DRL agents to make context-aware decisions for moving and placing prefabricated components under constraints. The study showcases how DRL can move beyond form generation and be embedded in construction-level decision-making, offering new potential for adaptive, real-time robotic workflows in architectural assembly.

In addition to layout and form-focused studies, Hosmer et al. (2020) investigate RL in the context of construction-aware assembly. They frame building design as a constraint-based game, where an agent learns through self-play to select and

assemble predefined spatial parts. Each step must follow strict geometric rules while also optimizing for design goals such as structural stability, material efficiency, spatial connectivity, segmentation, and daylight access. The method is tested in three case studies where the agent responds to practical constraints such as user needs, site limitations, and robotic reconfigurability.

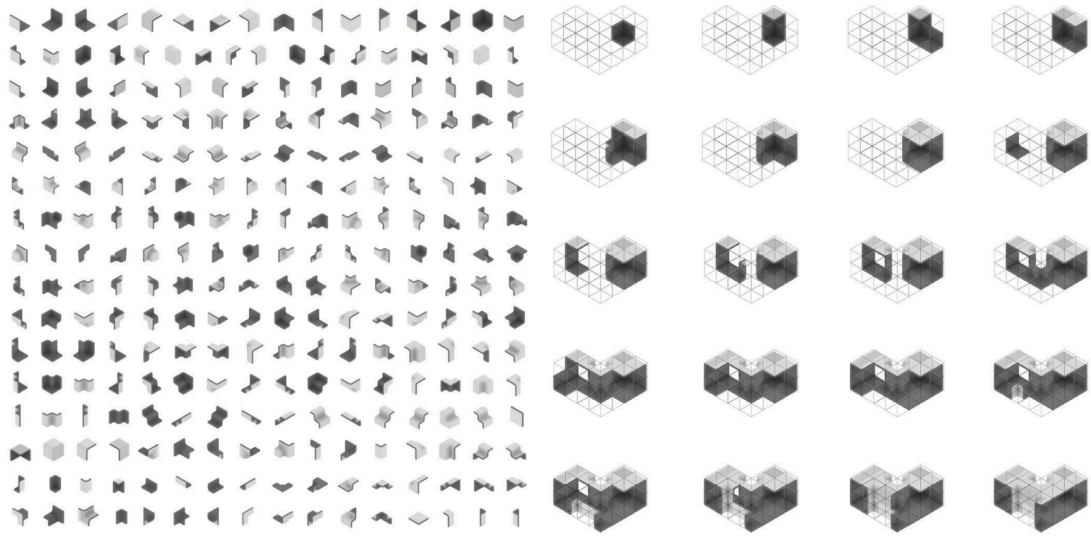


Figure 3.16) a) Spatial parts library (NoMAS 2019)

b) Spatial assembly sequence (NoMAS 2019).

Reproduced for academic and illustrative purposes from Hosmer et al. (2020), DSpatial Assembly with Self-Play Reinforcement Learning, ACADIA 2020: Distributed Proximities / Volume I: Technical Papers, pp. 384.

In this study, WFC and RL are used together in a complementary way. WFC acts as a local constraint solver, ensuring geometric validity by allowing only feasible part connections. RL operates on top of this filtered action space, guiding the agent's decisions based on performance objectives. This layered approach where WFC defines valid design moves and RL optimizes the sequence and strategy enables the generation of adaptive, buildable structures. This integrated framework is closely aligned with the approach developed in this thesis, where spatial constraints define available actions, and an RL agent learns to optimize modular housing configurations

through iterative design.

In summary, recent studies have explored the application of RL across a variety of design contexts and demonstrated its potential to address complex architectural problems. Although they differ in scope and methodology, together they establish a foundation that clarifies both the opportunities and the boundaries of this emerging approach. These works underline the need for design methods that go beyond static optimization and can adapt dynamically to spatial constraints.

Building on this foundation, this thesis positions RL not only as a computational tool but as a GD strategy. Its iterative, feedback-driven nature aligns closely with the exploratory processes of architectural design, offering a way to balance variation, constraint, and performance. The theoretical perspectives outlined in this chapter set the stage for the practical implementation that follows, where DRL is applied to modular housing massing.

The next chapter presents the proposed DRL-based design framework developed in this study, detailing how the method is structured, how the environment is defined, and how design-specific reward functions are constructed to guide the learning process.

CHAPTER 4

DESIGN OF HOUSING MASSING THROUGH MACHINE LEARNING

In this thesis, a simulation environment composed of a discrete voxel space is proposed, where a DRL agent can place a set of massing cells. The dynamics and constraints of this environment are designed explicitly to help the agent learn design policies that can gradually refine spatial arrangements through iterative feedback.

The housing massing is first abstracted using a CA framework. CA is selected for this initial step due to its discrete, grid-based structure, which is well-suited for modular housing systems and allows for direct spatial logic encoding. It provides a systematic way to discretize the design space into a grid structure where each cell can represent a building component. This abstraction enables the design problem to be transferred into a computable format that can be processed by the algorithm. In this context, CA functions as a problem-definition tool: it structures the design space into a set of possible cell states and defines spatial availability, neighborhood relationships, and adjacency rules, translating architectural constraints into a computable environment.

The ordering is deliberate: first, the problem is defined through CA to ensure that fundamental architectural logics are embedded into the environment. Next, the RL framework is integrated on top of this structured problem space, allowing the agent to interact with constraints that already reflect architectural reasoning. On top of this

structured space, the AI agent, trained using the PPO algorithm (Schulman et al., 2017), operates within a CA framework to autonomously generate and iteratively refine housing configurations. Within this setup, learning is guided by reward signals that reflect design-oriented objectives, such as maximizing the number of housing units within the buildable area, minimizing circulation space while ensuring accessibility, and encouraging diversity in housing unit mixtures. This setup allows the system to iteratively search for massing solutions that balance compactness, functionality, and variation. To clarify this choice further, it is useful to briefly revisit the basic logic of CA.

CA offers valuable insights into how complex behaviors can emerge from simple rules (Frazer, 1995). One to three or more-dimensioned regular arrays of cells define a cellular automaton. The simplest version of automata includes two states (0-1, true-false, black-white, alive-dead) for each cell, and for the more complex ones, there can be more states. Each cell holds a single, specific state, and the system undergoes cyclical transformations in which the states of individual cells are updated depending solely on the previous states of its surrounding cells (known as its "neighborhood").

A cellular automaton's state, consisting of a lattice of sites, evolves synchronously in discrete time steps according to identical rules and the values of variables at its "neighborhood" on the previous time step (Wolfram, 1983). In other words, the behavior of the cellular automaton is governed by transition rules that dictate each cell's state at any given time based on its prior state and the states of its neighboring cells (Frazer, 1995). A voxel cell can be defined with varying neighborhood configurations (two main neighborhood definitions are shown in Figure 1): in one

scenario, it may have four edge neighbors, while in another, it may include eight neighbors by accounting for diagonal connections. In three-dimensional space, a cubic cell may have six face neighbors or up to twenty-six if adjacent edge and vertex neighbors are included (Frazer, 1995).

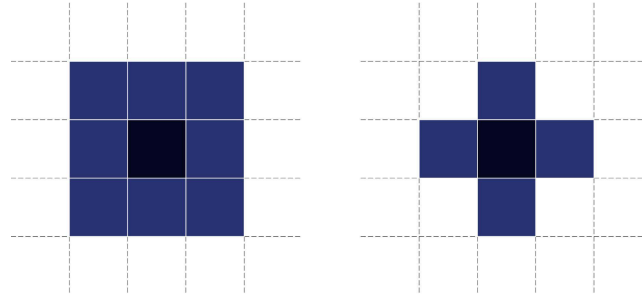


Figure 4.1. Left: Moore neighborhood. Right: Von Neuman neighborhood. Produced by author.

To proceed with the design process within the CA framework, a grid/lattice, neighborhood, cell states, time, and transition rules should be defined (Calisir & Cagdas, 2020). This framework enables complex design problems to be redefined as structured and well-defined problems, making them more suitable (machine-readable) for computational applications. This study uses a CA representation of the modular housing massing design problem. CA inherently involves a certain level of complexity and randomness, regardless of the simplicity of the rules (Wolfram, 2002, as cited in Calisir & Cagdas, 2020). In this study, a DRL agent is placed within a CA environment to handle the inherent complexity and learn sequential design policies of modular housing configurations guided by a set of design objectives.

4.1 Cell Design and Spatial Configuration

The housing structure, which simultaneously serves as the environment, is modeled as a $30\text{m} \times 30\text{m} \times 30\text{m}$ cubic building volume discretized into a three-dimensional

grid of voxel cells, each representing a volumetric unit of $3\text{m} \times 3\text{m} \times 3\text{m}$. Thus, the overall voxel lattice has dimensions of 10 floors, each consisting of a 10×10 grid, yielding a total of 1,000 voxels within the design space. In other words, the building is conceptualized as a three-dimensional array, where each index indicates the position of a voxel cell within the structure. This constitutes the mathematical and computational basis of the building in the project.

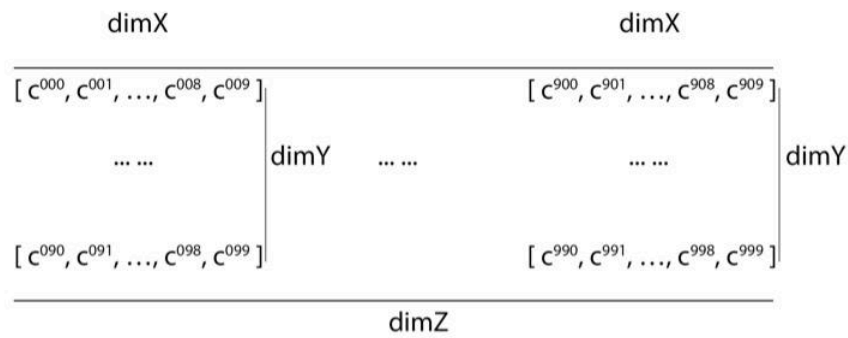


Figure 4.2. Representation of the building as a three-dimensional array. Produced by author.

Each voxel could be in one of the four following states:

- empty,
- built-up,
- hall, or
- building core (vertical circulation).

Built-up cells represent the housing units within the structure, hall cells provide horizontal access from the units to the core, and core cells provide vertical circulation.

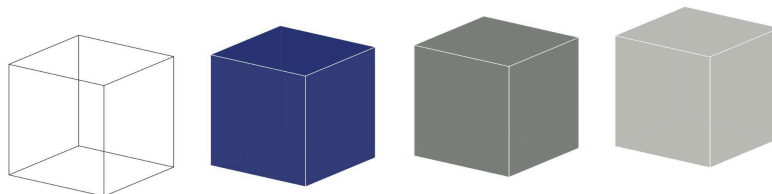


Figure 4.3. Cell types, from left to right: Empty Cell, Built-up Cell, Core Cell, and Hall Cell. Produced by author.

Initial state of the environment:

Initially, all cells are set to empty, ready for the agent's placement, with a set of cells forming a central vertical circulation core. The mid-four cells in the lattice for each floor are defined as the core cells, while others stay empty when the environment is initialized. While empty and core cells are allocated at the initial state of the building structure, the placement of built-up and hall cells is left to the RL agent to decide.

The initial state of the building:

$$\begin{array}{c} \text{dimX} \\ \hline \begin{array}{c} [0, 0, \dots, 0, 0] \\ \dots \dots \\ [0, 0, \dots, 0, 0] \end{array} \left| \begin{array}{c} \text{dimY} \quad \dots \quad \dots \end{array} \right. \begin{array}{c} \text{dimX} \\ \hline \begin{array}{c} [0, 0, \dots, 0, 0] \\ \dots \dots \\ [0, 0, \dots, 0, 0] \end{array} \left| \begin{array}{c} \text{dimY} \end{array} \right. \end{array} \\ \hline \text{dimZ} \end{array}$$

Cellular Automata Constraints:

In this framework, two layers of constraints are applied. At the environment level, adjacency-based rules derived from the CA ensure connectivity and accessibility, functioning as hard constraints that cannot be violated. At the reward level, additional design goals such as built-up and hall ratios, unit size limits, and compactness are incorporated as soft constraints. These are not enforced directly, but guide the agent's behavior through the reward structure. Together, this combination ensures that basic architectural feasibility is preserved while allowing the RL agent to explore diverse massing solutions within defined performance boundaries.

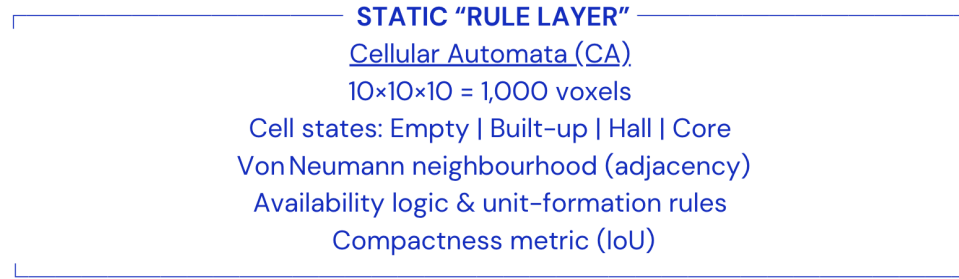


Figure 4.4. Static Rule Layer. Produced by author.

The CA framework prepares the ground for defining spatial rules and local interactions. A von Neumann neighborhood is employed, where each voxel's adjacent edge-connected neighbors in the same plane inform cell availability and placement decisions (Frazer, 1995). This local neighborhood definition ensures that design decisions remain computationally tractable while reflecting realistic adjacency-based spatial logic in architecture. This is essential to establish the decision environment for the RL agent.

To ensure that all housing units have access to the vertical core and to allow the agent to concentrate on cell placement decisions, the availability of cells within the building structure for built-up and hall cells is closely monitored. This availability status is checked according to the neighbor cells' condition:

For a cell:

1. If any adjacent neighbors are hall or core cells, the cell is available for either hall or built-up placement
2. If none of the adjacent neighbors are hall or core cells, but one or more neighbors are built-up cells, the cell's availability depends on the housing units to which the built-up neighbors belong:
 - If any housing unit has not yet reached its maximum limit for built-up cells, the cell is available for a built-up placement. However, it is not available for hall placement.

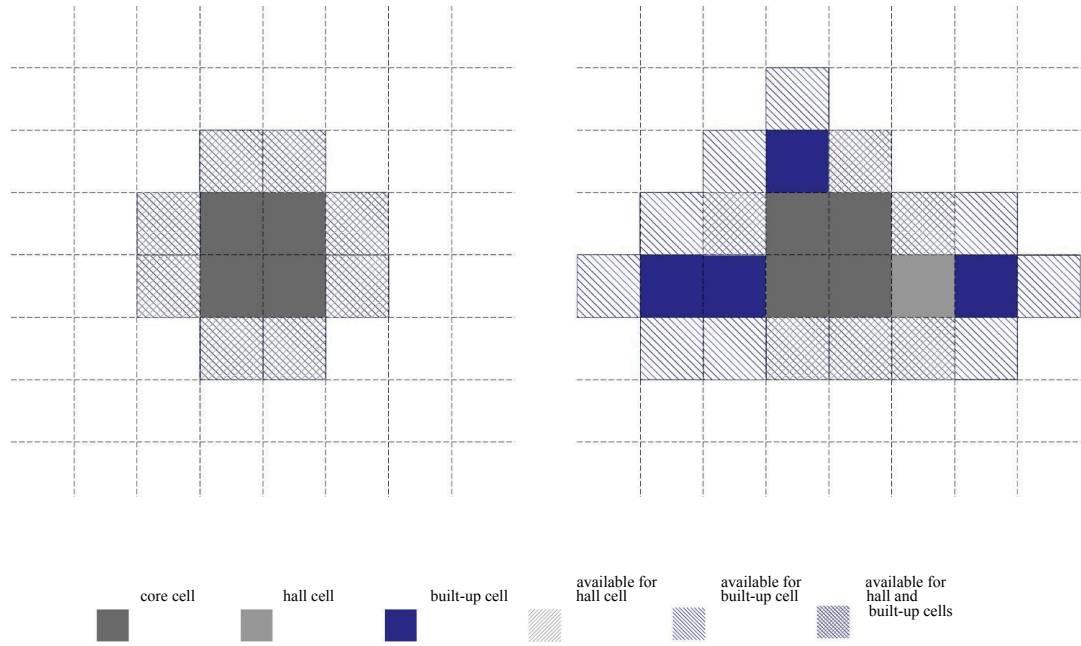


Figure 4.5. An instance of the rules for cell availability for placement. Produced by author.

These rules determine the availability of cells for built-up and hall placements. They help to ensure that all housing units remain accessible within the building structure via vertical and horizontal circulation.

In addition, the organization of housing units is governed by predefined rules and constraints based on neighboring conditions. By doing so, the model remains less complex, providing the agent with a clear and structured learning environment that minimizes confusion. If the definition of housing units were left entirely to the agent's decisions, the action space would become unstable. This is because the exact number of housing units is unknown at the initial state, making it impossible to define a fixed action space for decisions. This would lead to an unstable action space, which would significantly increase the complexity of the agent's learning process. Therefore, a rule-based approach ensures the agent operates within a well-defined and controlled environment, facilitating more effective learning.

When a built-up cell is placed:

- Its placement is based on the availability conditions. This means it is either placed due to being adjacent to a core or hall cell, or it is placed because it is adjacent to a housing unit that has not yet reached its maximum number of built-up cells.

After placement, the neighbouring cells of this built-up cell are checked:

- If the cell has no neighbouring housing unit that is below its maximum built-up cell limit, but it is adjacent to a core or hall cell, then this cell is defined as a new housing unit.
- If it is adjacent to only one housing unit that has not yet reached its maximum number of built-up cells, it directly becomes part of that housing unit.
- If it is adjacent to multiple housing units that have not yet reached their maximum built-up cell limits, the potential compactness rate of each unit is calculated as if the new cell were added to it. The cell is then assigned to the housing unit where its inclusion results in the highest compactness.

To quantify compactness, a calculation method devised by inspiration from Intersection-over-Union (IoU) Index.

Intersection over Union (IoU) which is also called the *Jaccard Index* (Jaccard, 1901), measures the similarity of two sets by dividing the size of their intersection by the size of their union: It is widely used in computer vision to judge how well a predicted bounding box or segmentation mask matches the ground-truth label (Rezatofighi et al., 2019).

Formally:

$$\text{IoU} = |\mathbf{A} \cap \mathbf{B}| / |\mathbf{A} \cup \mathbf{B}|$$

In this study, each housing unit is first projected onto the horizontal plane so that its occupied cells form a 2-D footprint. The footprint compared against predefined “ideal” shapes (e.g., 2×3 rectangle for six cells).

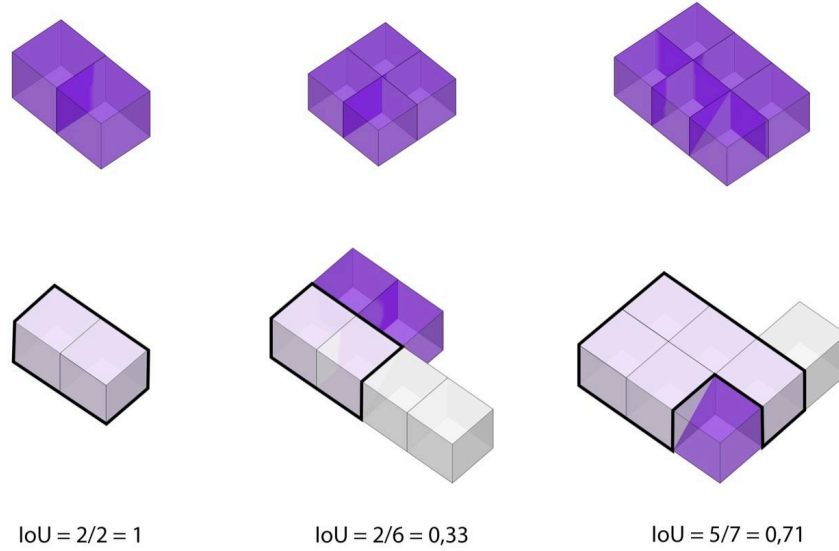


Figure 4.6. Instances of IoU calculations. Produced by author.

The IoU between the generated unit and each ideal reference provides a compactness score:

- $\text{IoU} = 1 \rightarrow$ perfect overlap, maximal compactness.
- $\text{IoU} < 1 \rightarrow$ increasing deviation, indicating elongated or irregular layouts.

4.2 Experiment Setting as Markov Decision Processes (MDPs)

RL is a computational approach to learning from interaction, rooted in the field of optimal control and dynamic programming (Sutton & Barto, 2018; Puterman, 1994).

In RL, an agent aims to discover an optimal policy by continuously interacting with its environment, using feedback signals to guide its decisions over time.

This interaction is formally modelled as a Markov Decision Process (MDP), which provides a mathematical framework for sequential decision-making under uncertainty (Puterman, 1994) . It is defined by a tuple (S, A, P, R, γ) . The Markov property states that the next state s' depends only on the present state s and action a , not on any earlier history (Puterman, 1994; Sutton & Barto, 2018). In this research, the computational design problem of optimizing modular housing mass configurations is formulated as a MDP to optimize modular housing mass configurations within a defined three-dimensional voxel-based simulation environment. The MDP formulation enables the integration of design objectives, spatial constraints, and sequential decision-making into a single optimization paradigm suitable for RL. An MDP consists of the following key components:

- **State (S):** Represents the current situation or configuration of the environment. It summarises all relevant information needed for decision-making at a given time.
- **Action (A):** Denotes the set of all possible moves or decisions the agent can make to influence the state of the environment.
- **Reward (R):** A scalar feedback signal received after taking an action, indicating the immediate benefit or cost associated with that action. It measures how desirable a particular state-action outcome is.
- **Policy (π):** Defines the agent's strategy. It is a mapping from states to actions, specifying the probability of selecting each action in a given state to maximise cumulative reward (Sutton & Barto, 2018).

In addition to this four-tuple, the discount factor $0 \leq \gamma < 1$ is introduced to adjust the influence of future rewards on present decisions. The objective is to find a policy $\pi: S \rightarrow A$ that maximizes the expected total discounted reward (Puterman, 1994).

“The policy is the core of the agent’s behaviour, determining how it acts based on its current knowledge and experience.” (Sutton & Barto, 2018, p. 51)

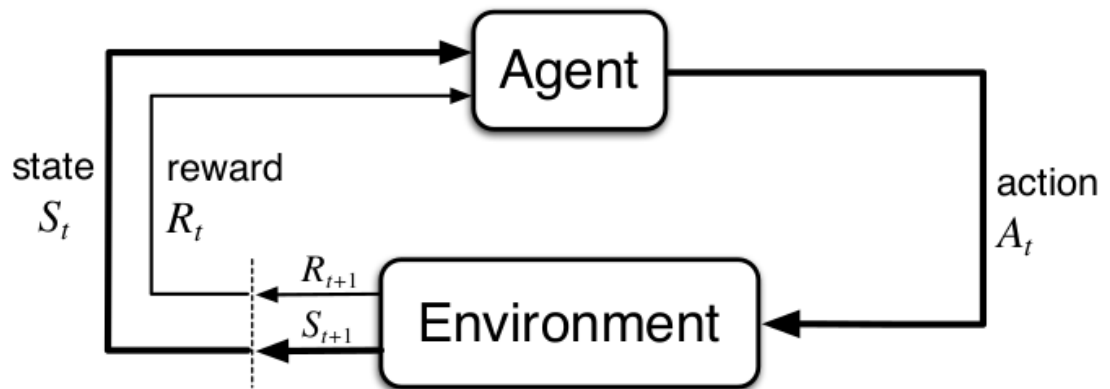


Figure 4.4. Reinforcement Learning: An Introduction 2nd Edition, Richard S. Sutton and Andrew G. Barto. Reproduced from Sutton & Barto (2018, p.54)

In summary, RL uses the MDP framework to formalise the process of learning optimal behaviour through trial-and-error interactions with the environment, guided by rewards that evaluate each action’s effectiveness (Puterman, 1994; Sutton & Barto, 2018). This framework reduces complex multi-objective design goals into a single numerical feedback signal, enabling the application of both classical optimisation and learning-based algorithms.

DYNAMIC “DECISION LAYER” (Deep RL)

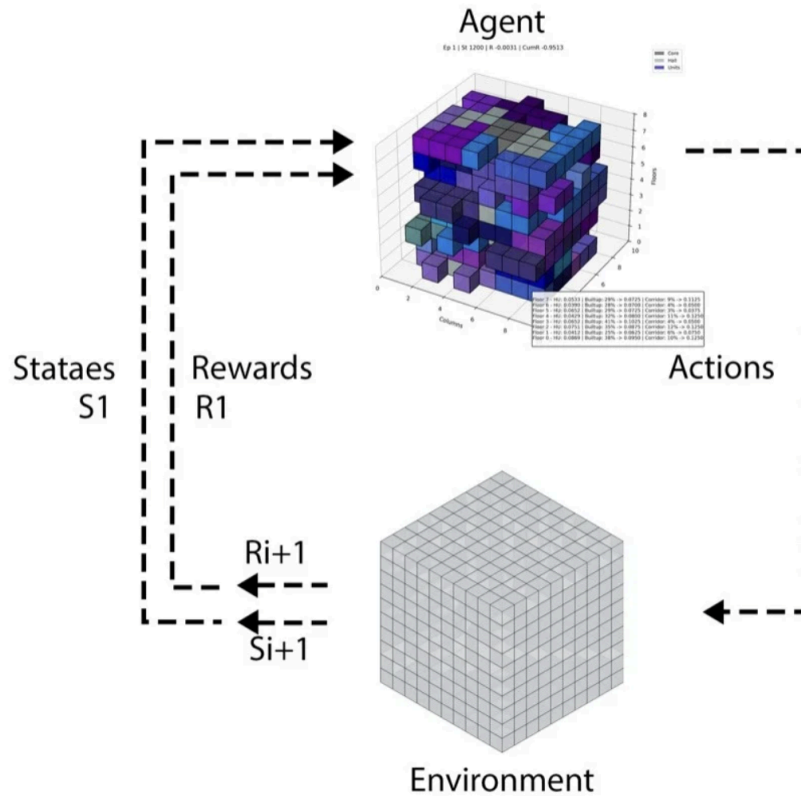


Figure 4.7. Dynamic, decision layer. Design problem within MDP framework. Produced by author.

By structuring the modular housing design problem within an MDP framework, it becomes possible to leverage RL algorithms such as PPO to autonomously generate and refine massing configurations through iterative, feedback-driven learning processes (Schulman et al., 2017).

4.3 Description and Results of Experimental Settings

This study includes two experimental settings developed for the same design task. Both use the same initial environment configuration and rule-based constraints. Most components of the RL environment remain the same. The observation space is identical. The reward structure also follows the same design goals, with some key updates to improve agent behaviour.

The second setting was introduced after analysing the outcomes of the first. In the first setting, the agent showed a strong tendency to take no action. This behaviour was caused by the way the reward signal was shaped, especially due to the use of original PBRS, and other implementation choices.

To address this, several updates were made in the second setting. These include changes in the reward calculation, the action space definition, and the way the environment responds to invalid moves. Together, these changes aimed to prevent passive behaviour and support more effective learning.

This section presents both experimental setups. It also explains the reasons behind the updates, and discusses their outcomes in terms of learning performance and design quality.

4.3.1 Experimental Setting I

This section introduces the first experimental setup used to evaluate the RL model. It includes the RL environment setup, the application of potential-based reward shaping (PBRS), and the results and discussion of the experiment.

4.3.1.1 RL Environment Setup for Experimental Setting I

There are four distinct components that define the RL environment. This subsection presents how each of these components is formulated specifically for the first experiment.

RL State (Observation) Space (S) Definition:

The state space in this experiment captures the complete configuration of the three-dimensional voxel environment at each timestep. It is represented as a flattened

one-dimensional array encoding the status of every voxel in the building.

Specifically:

- The voxel grid has dimensions $10 \times 10 \times 10$, resulting in 1,000 voxel cells in total.
- For each voxel, four features are recorded:
 1. The cell type encoding for each voxel (empty, built-up, hall, core),
 2. Built-up availability: indicating if the voxel is eligible for built-up placement,
 3. Hall availability: indicating if the voxel is eligible for hall placement,
 4. Unit ID: specifying the housing unit to which each voxel belongs.

Given the environment dimensions of $10 \times 10 \times 10$ voxels, and four features per voxel, the total observation vector has a dimension of 4,000.

The state space is defined in the environment as a continuous Box space with minimum and maximum bounds for each observation dimension:

```
self.observation_space = Box(
    low=self.min_obs,
    high=self.max_obs,
    shape=(self.obs_dim, ),
    dtype=np.float32
)
```

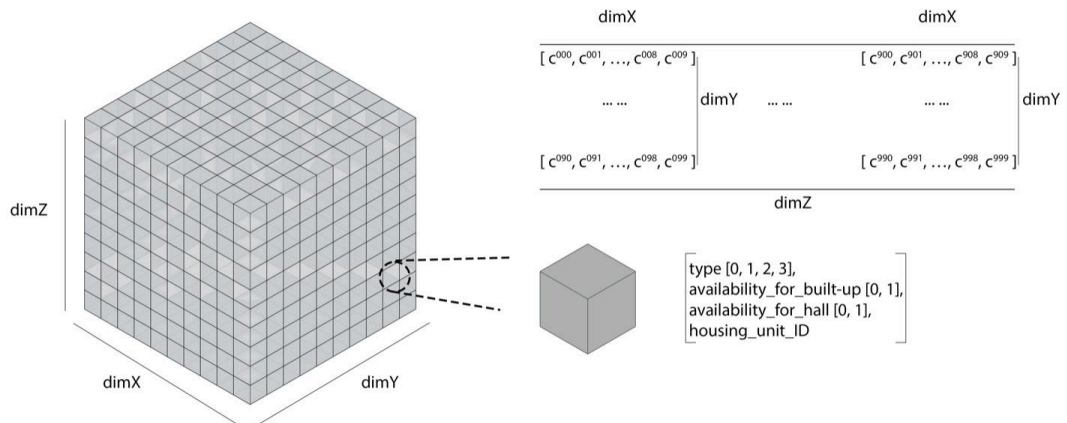


Figure 4.8. RL State (Observation) Space (S). Produced by author.

This detailed representation provides the agent with comprehensive spatial

awareness, enabling decisions that respect architectural constraints such as adjacency, circulation, and unit formation logic.

RL Action Space (A) Definition:

The action space, which is an important component of the RL environment, defines the set of placement decisions that the agent can take at each timestep, enabling it to interact with and modify the voxel-based design environment effectively. Also taking into account adjacency and accessibility constraints.

In the first experimental setting, a continuous Box action space was used, represented as a four-dimensional vector for each action:

1. Floor index (z): Normalised between [0,1] to cover the 10 building floors.
2. Row index (y): Normalised between [0,1] to cover the 10 rows per floor.
3. Column index(x): Normalised between [0,1] to cover the 10 columns per row.
4. Cell type selection scalar: Normalised between [0,1], interpreted as a binary choice between built-up or hall placement.

This action vector is defined in the environment as:

```
self.action_dim = 4
self.action_space = Box(
    low=self.min_action,
    high=self.max_action,
    shape=(self.action_dim, ),
    dtype=np.float32
)
```

These continuous outputs are subsequently discretised within the environment to map onto the voxel grid indices and cell type assignments. The first three components represent the coordinates within the building structure (floor, row, column numbers) where the action will take place. The last action component serves as a selection mechanism for the cell type (hall / built-up) to be allocated at each timestep.

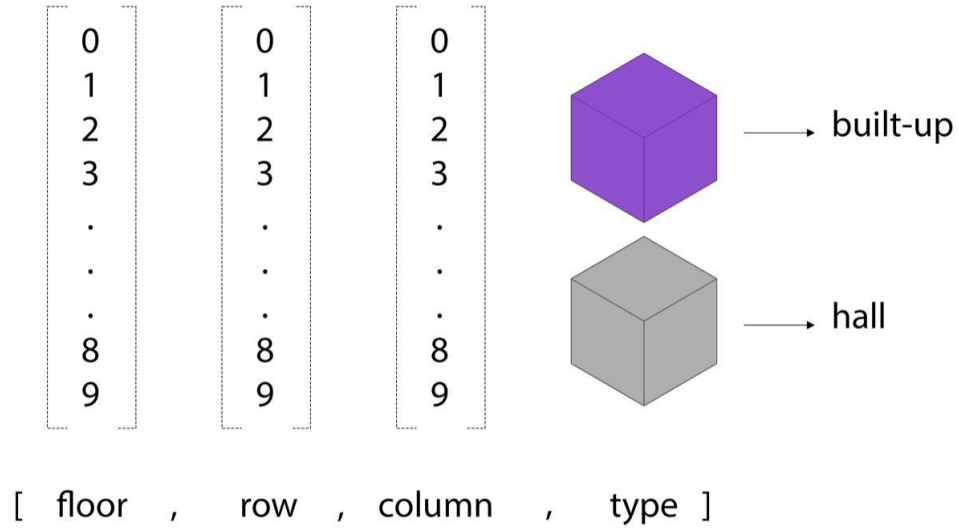


Figure 4.9. RL Action (A) Space (S). Produced by author.

RL Transition Dynamics (P) Definition:

Transition dynamics (P) define how the environment changes from one state to another after an action is taken. In this experimental setting:

- Executing an action updates the voxel environment state based on the placement decision logic.
- When the agent selects a voxel location and cell type:
 - If the placement satisfies availability rules, the voxel's state is updated accordingly (action performed) (e.g., from empty to built-up or hall).
 - If the placement is invalid, the state remains unchanged (action is not performed), and a time penalty is imposed, reducing the incentive for wasted actions.

When the agent selects a specific location and cell type for an action, the availability of the targeted cell is verified. If the cell is available for the chosen type, the action is performed. If the cell is not available, the action is not performed, and the agent incurs a time penalty, as a wasted step, within the step-limited RL environment

results in no reward. This availability logic ensures that every placement action contributes meaningfully to spatial connectivity, prevents isolated built-up placements, and maintains an accessible, coherent massing configuration. Finally, housing units are updated after each action if the new placement to be performed is a built-up type. They can vary in form and size within defined limits and are visually distinguished by different color coding.

RL Reward (R) Definition:

For the generation process the rules are transferred into the RL environment. The environment dynamics follow this set of general rules to direct the building configuration generation:

- **Connectivity Constraint:** All housing units in the building structure must be connected to the vertical core cells, either directly or through hall cells.
- **Housing Unit Size Constraint:** Housing units can vary in form and size, consisting of a minimum of two and a maximum of six built-up cells.

During the generation process, actions are only performed when the agent selects an available cell for the chosen type. Reward shaping is used to improve the learning performance of the agent by encouraging it to reach higher rewards with supplemental immediate rewards. It is important to apply them in a way that directs the learner to the intended goals in the environment. Otherwise, the learner adopts a policy that is optimal in reference to the reward function but suboptimal with respect to the original design task.

Rewards are defined to direct the agent's action as:

- **Built-up Cells Ratio Reward (R1) :** The agent is encouraged to maintain the floor area ratio within the prescribed limits for built-up cells.

- Hall Cells Ratio Reward (R2) :The agent is encouraged to maintain the floor area ratio within the prescribed limits for hall cells.
- Housing Units Size Reward (R3): The bigger housing units generated in between 2-6 cells in size, the greater positive reward the agent receives.
- Housing Units Compactness Reward (R4): The agent is encouraged to create housing units that are more compact in shape (with higher IOU rate).

Terminal Condition: If any housing unit consists of only a single built-up cell and cannot be extended further, it triggers a terminal state for the environment and a negative value will be added to the Reward.

Further details regarding the RL environment setup are provided in Appendix A.

4.3.1.2 Potential-Based Reward Shaping (PBRs)

In this study, the reward function was designed using Potential-Based Reward Shaping (PBRs), a widely recognised reward engineering technique to improve the learning efficiency of RL agents. PBRs works by augmenting the environmental reward with an additional shaping term at each timestep, calculated as the difference in a designer-defined potential function between consecutive states (Ng, Harada & Russell, 1999). The potential function, $\Phi(s)$, is a scalar measure of how “good” a particular state is according to the task’s objectives. In this project, $\Phi(s)$ represents the building’s spatial performance score calculated after each action, encompassing criteria such as the number of housing units, circulation efficiency, and compliance with design constraints.

Formally, PBRs is expressed as:

$$F(s_t, a_t, s_{t+1}) = \gamma \Phi(s_{t+1}) - \Phi(s_t),$$

where γ is the discount factor that reflects how much the agent values future rewards relative to immediate gains. The theoretical foundation of PBRS ensures that adding this shaping term to the reward function does not alter the optimal policy that the agent can learn, a property known as policy invariance (Ng et al., 1999). Instead, it accelerates the convergence towards this optimal policy by providing the agent with more frequent and informative feedback about the consequences of its actions, thereby reducing the reliance on sparse terminal rewards that may only become available at the end of an episode (Wiewiora, 2003).

In the context of this project, PBRS plays a critical role. The building mass configuration task involves a complex voxel-based environment where overall design performance depends not on a single isolated action but on a sequence of interdependent moves. If only final episodic rewards were used (e.g. a single reward signal after the full building configuration is completed) the learning process would become extremely slow and unstable. This is because the agent would struggle to identify which specific action led to improvement or deterioration in performance.

To address this, PBRS was integrated to provide dense, step-wise feedback by calculating the difference in building performance before and after each action.

4.3.1.3 Results and Discussion for Experimental Setting I

The results show that the RL environment has been created in a manner suitable for automatically learning a massing configuration generation policy according to the criteria set in this study. In the initial implementation of this approach for the experimental setup I, the shaping reward was formulated as:

$$R_t(\text{step}) = \gamma \cdot R_t - R_{t-1}$$

where R_t represents the building's performance score at timestep t . While this formulation aimed to reflect the discounted change in potential between states, it was observed to produce negative rewards even for beneficial actions due to the scaling down effect of the discount factor γ . As a result, once the training phase has been initiated, the agent's learning tendency has been directed towards 'no action' in the process. This is because of:

Actions are executed only when they are placed in available cells, otherwise they lead to a null reward. Also, as there are discounted changes between the states, which results in the agent receiving negative rewards, this leads to the generation of a negative cumulative reward constantly. When the agent receives a positive reward and is not able to take action after several attempts, the reward drops back to negative again. This situation has caused the agent to eventually avoid taking action, keeping its reward close to 0 or exactly 0. Figures 4.10 and 4.11 illustrate the learning progress.

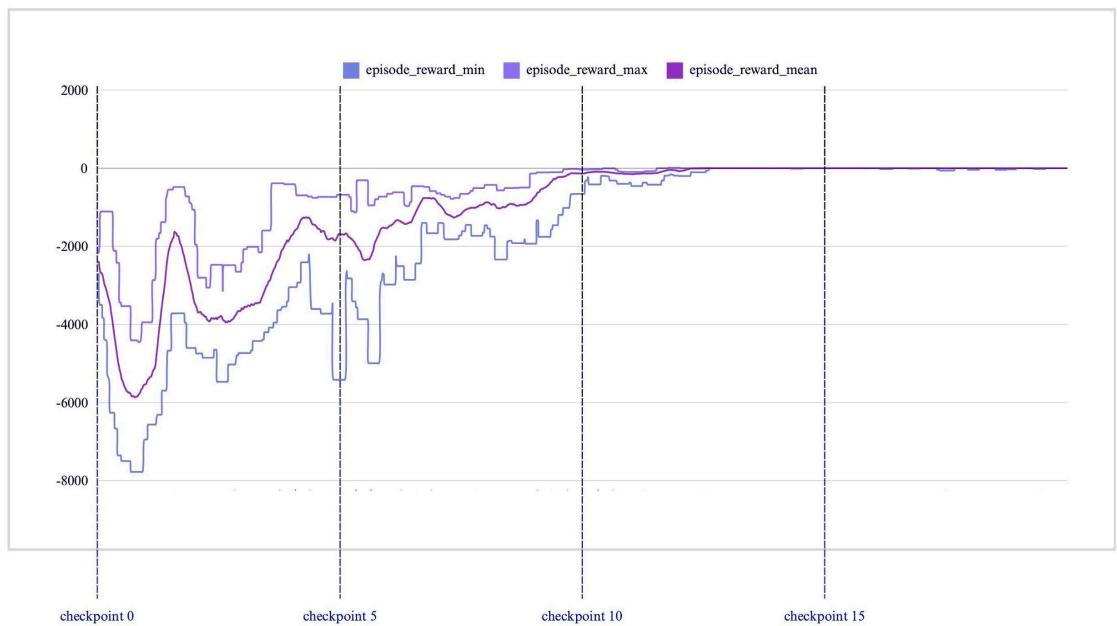


Figure 4.10. Reward progress graph showing minimum, maximum, and mean rewards during training, with every fifth checkpoint indicated. Visual outputs from these checkpoints are shown in Figure 4.11. Produced by author.

The graph displays the minimum, maximum, and mean rewards during training, while the visual outputs at every 5th checkpoint, as shown in the graph, enable a comparison between the two figures. In other words, it has learned to choose unavailable actions instead of available ones, to keep its score high. To overcome this problem, several potential solutions are explored. First of all, reward policy change is applied to solve this problem.

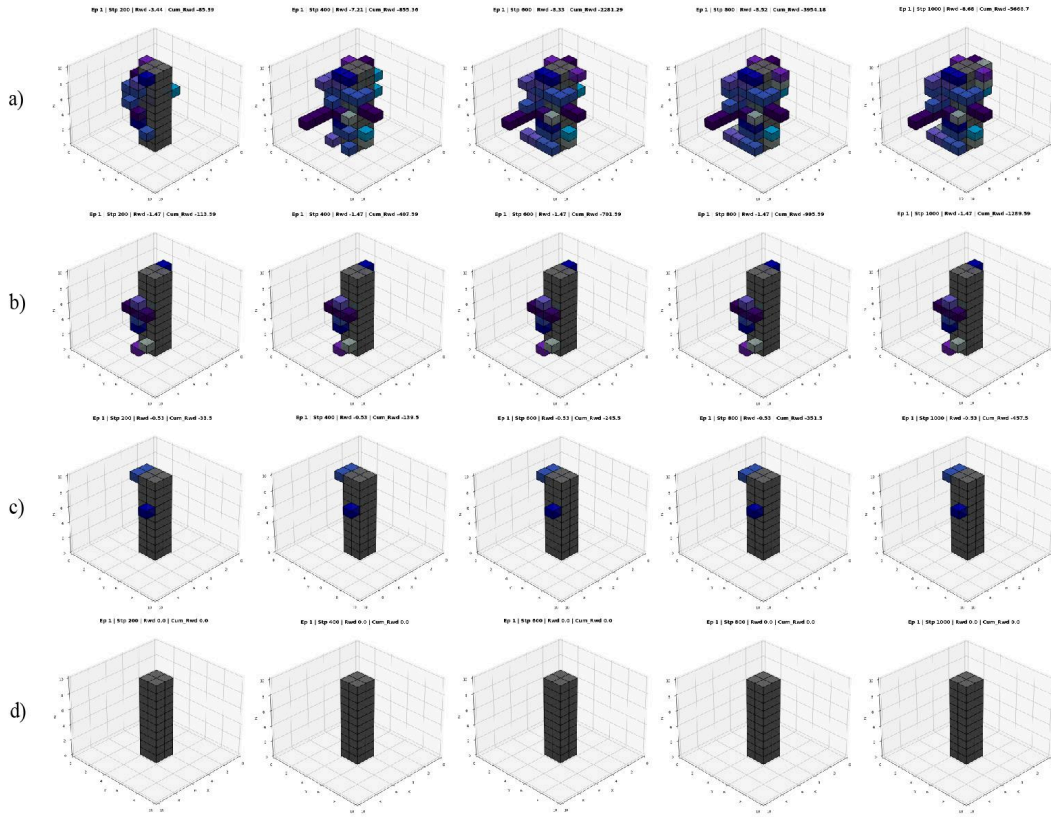


Figure 4.11. Outputs of the model from a) checkpoint 0, b) checkpoint 5, c) checkpoint 10, checkpoint 15. From left to right: Outputs from steps 200, 400, 600, 800, 1000. Produced by author.

4.3.2 Experimental Setting II

This section introduces the second experimental setup used to evaluate the RL model. It includes the RL environment setup, the application of potential-based reward shaping (PBRS), and the results and discussion of the experiment.

4.3.2.1 RL Environment Setup for Experimental Setting II

There are four distinct components that define the RL environment. This subsection presents how each of these components is formulated specifically for the second experiment.

RL State (Observation) Space (S) Definition:

The state space definition in the second experimental setting remains identical to Experimental Setting I.

RL Action Space (A) Definition:

In Experimental Setting I, the action space was defined as a four-dimensional continuous Box space $[-1,1]^4$ with normalised outputs that required rescaling and discretisation into integer floor, row, column indices and a binary cell type. This extra conversion layer introduced rounding artefacts that can destabilise gradient updates when the underlying decision problem is categorical (Henderson et al., 2018).

In Experimental Setting II the same four decisions were encoded as a MultiDiscrete space $\langle \text{floor} \in \{0:9\}, \text{row} \in \{0:9\}, \text{col} \in \{0:9\}, \text{type} \in \{0,1\} \rangle$.

Gymnasium’s documentation defines MultiDiscrete as the Cartesian product of multiple independent *Discrete* spaces and notes that it is most useful when each action dimension is an independent categorical choice (Gymnasium Docs, *Spaces*).

This switch eliminates the need for the rounding step. Network outputs now map straight to discrete actions, without additional conversion layers. This direct mapping lowers gradient variance and stabilises policy updates, improving learning reliability (Sutton & Barto, 2018). Overall, switching to MultiDiscrete better reflects the discrete nature of voxel placement decisions in architectural massing tasks and

simplifies the agent-environment interaction logic.

RL Transition Dynamics (P) Definition:

In Experimental Setting II, the transition dynamics follow the same fundamental placement logic as in Setting I. When the agent selects a voxel location and cell type:

- If the placement satisfies the availability rules, the voxel state is updated accordingly (e.g. from empty to built-up or hall).
- If the placement is invalid, the voxel state remains unchanged, and the agent incurs a penalty.

Unlike Setting I, where invalid actions only resulted in a wasted step with no reward, Setting II introduces a constant negative penalty for invalid actions. This penalty was set to approximately 25% of the average per-step reward calculated from preliminary training runs. It was implemented as an alternative to an action mask since it was easier to implement. While action masking is often recommended due to its scalability and theoretical validity in policy gradient algorithms (Huang & Ontañón, 2022). By applying a fixed penalty to invalid actions, the agent is discouraged from selecting unavailable placements without requiring modifications to the action space itself.

The overall transition dynamics therefore remain as:

1. Valid action $\rightarrow$ state updated;
2. Invalid action $\rightarrow$ state unchanged + constant penalty applied.

RL Reward (R) Definition:

In Experimental Setting II, the reward structure maintains the same core constraints and reward definitions logic as in Setting I, with several key modifications to better align agent behaviour with the design objectives of this study.

The rewards guiding agent actions in this setting are defined as follows:

- **Built-up Cells Ratio Reward (R1):** The agent is encouraged to maintain the built-up cell floor area ratio within the range of 50-80% for each floor. Achieving this ratio across all floors yields a total of +1 point for entire building.
- **Hall Cells Ratio Reward (R2):** Similarly, the agent is incentivised to maintain the hall cell floor area ratio within the range of 10-20% for each floor. Achieving this ratio across all floors yields a total of +1 point for entire building.
- **Housing Units Compactness Reward (R4):** The agent is rewarded for creating more compact housing units, measured by the Intersection over Union (IoU) score of each unit. If all housing units in the building achieve perfect compactness, the agent gains a total of +1 point for entire building.

Note: The Housing Units Size Reward (R3) implemented in Setting I was removed in this setting. Promoting larger unit sizes was found to contradict the study’s objective of maximising variation among housing units. Thus, to prioritise diverse spatial configurations over unit expansion, R3 was excluded.

Terminal Condition:

The termination condition was also modified. In Setting I, episodes ended if any housing unit consisted of only a single built-up cell that could not be extended further. However, agents began exploiting this condition to terminate episodes strategically by creating such units deliberately, thereby avoiding penalties from invalid actions. To counteract this behaviour, the terminal condition was revised: an episode now ends only when no available cells remain for placement, ensuring that the agent fully explores possible configurations before termination.

Further details regarding the RL environment setup are provided in Appendix A.

4.3.2.2 Potential-Based Reward Shaping (PBRs): Special Case of $\gamma=1$

Since the structure of the problem causes many invalid actions, the discounted reward policy is considered to be changed to:

$$R_t(step) = R_t - R_{t-1}$$

to use the simpler potential difference form without applying γ within the shaping term. This adjustment aligns with the recommendation by Wiewiora (2003), who demonstrated that using $\gamma=1$ within the shaping difference preserves policy invariance while avoiding the unintended consequence of double discounting. In practice, this revised formulation enabled the agent to receive clearer feedback about how each action influenced spatial performance, ultimately stabilising learning and facilitating the development of effective placement policies within the voxel environment.

4.3.2.3 Results and Discussion for Experimental Setting II

To avoid the issues observed in the previous experimental setting and to ensure that the agent’s policy was functioning effectively, initial tests in Experimental Setting II were conducted on a simplified environment configuration. Specifically, a single-floor building model was used during the early training runs. This simplification allowed for a more controlled evaluation of agent behaviour by reducing the spatial complexity and isolating fundamental learning dynamics from multi-floor coordination challenges. Starting with a single-floor environment made it easier to observe whether the agent could correctly learn placement rules, maintain the required built-up and hall ratios, and generate valid housing unit configurations without encountering early termination or invalid action loops. Once the agent demonstrated stable learning performance in this simpler setup, the environment was

planned to be gradually scaled up to include multiple floors and additional design constraints. This incremental complexity strategy was adopted to ensure that the policy generalised well and that improvements at each stage were both measurable and interpretable within the architectural mass configuration task.

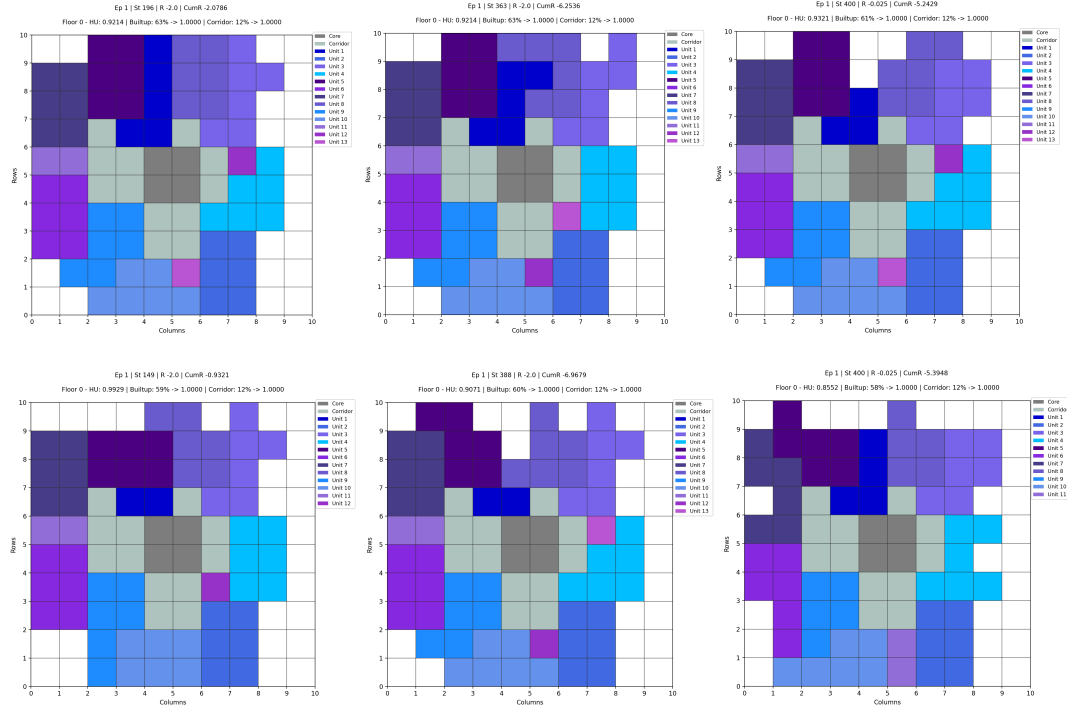


Figure 4.12. Six different outputs generated by the trained model: single-floor building with a fixed core location. Produced by author.

The results from these preliminary runs indicated that while the agent successfully learned to comply with placement rules and maintain targeted built-up and hall ratios, the generated outputs displayed very limited variation across episodes. Six sampled layouts showed highly similar configurations, with housing units clustering around the core in nearly identical patterns (Figure 4.12). This consistency suggests that the agent rapidly converged to a specific optimal solution given the fixed spatial constraint.

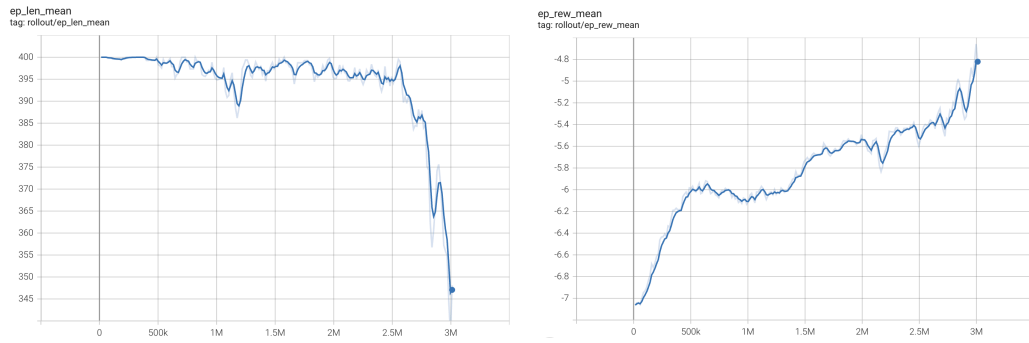


Figure 4.13. Training graphs: left - mean episode length, right - mean episode reward during training for the fixed core location. Produced by author.

Training metrics further supported this interpretation: the mean episode length remained near the maximum for most of training but decreased towards the end, suggesting the agent was learning to reach termination conditions more efficiently rather than exploring diverse configurations. Additionally, the mean episode reward improved steadily, indicating policy optimization progress (Figure 4.13). This consistent yet repetitive behaviour suggested that although the fundamental policy logic was functional, the agent was converging to a single local optimum without adequate exploration. To address this and encourage spatial diversity, subsequent training runs were conducted with randomised core locations in each episode. This modification aimed to force the agent to adapt its strategy to varying core placements, promoting generalisable policy learning that could produce alternative massing solutions within the design constraints of the modular housing environment.

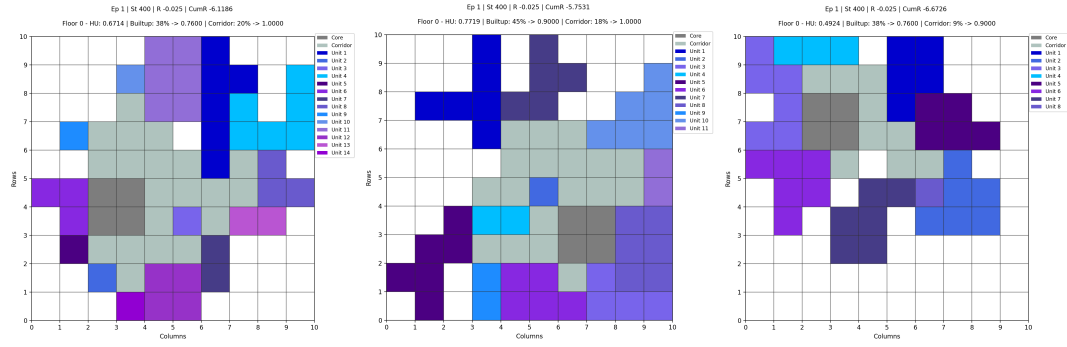


Figure 4.14. Three example training cases illustrating the randomly placed core cell conditions that the agent encountered during learning. Produced by author.

When the core location was randomised at the beginning of each episode, the mass configurations became noticeably more diverse (Figure 4.14). Unlike the fixed-core setting where units consistently clustered in similar patterns, here the housing units extended in multiple directions around different core positions. This indicates that the agent was able to adjust its strategy to varying circulation cells configurations instead of memorising a single placement routine.

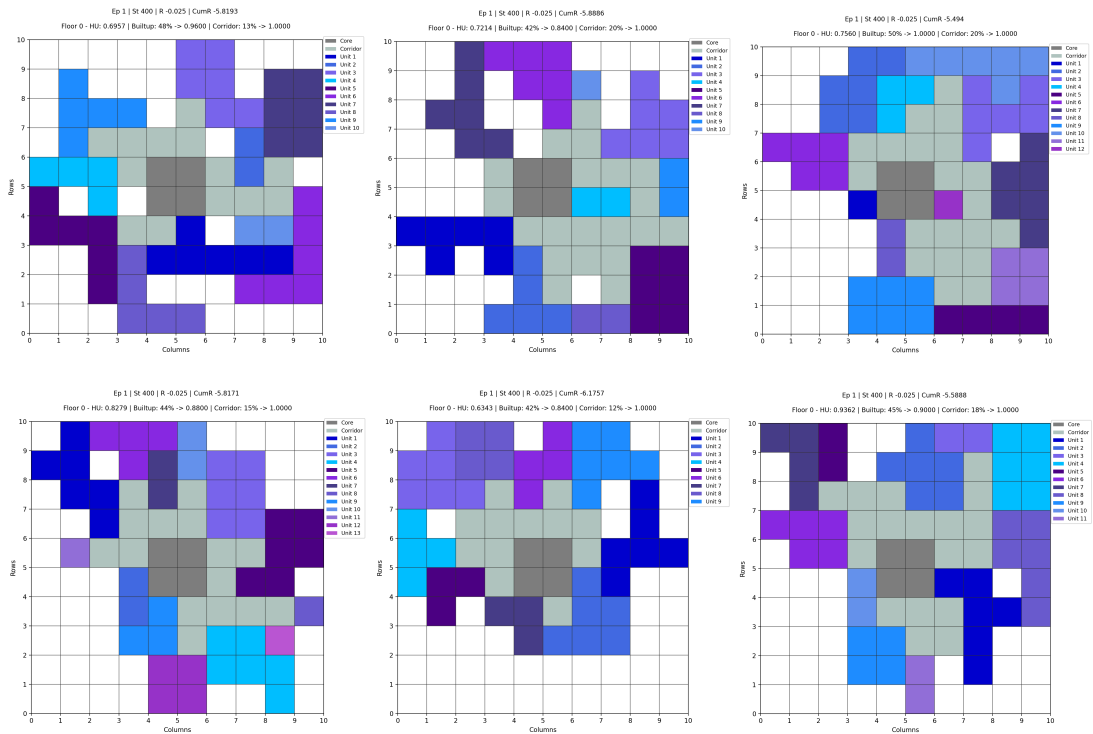


Figure 4.15. Six different outputs generated by the model trained with random core placement, evaluated here in a fixed core scenario (central four cells) for direct comparison with the fixed core model results. Produced by author.

However, this improvement in output diversity came with several trade-offs. As shown in Figure 4.16 (training graphs), the random core setup (red) exhibited a slower and less stable increase in mean episode rewards compared to the fixed core setup (blue). By the end of three million steps, the random core agent had not yet reached the performance levels achieved in the fixed core runs.

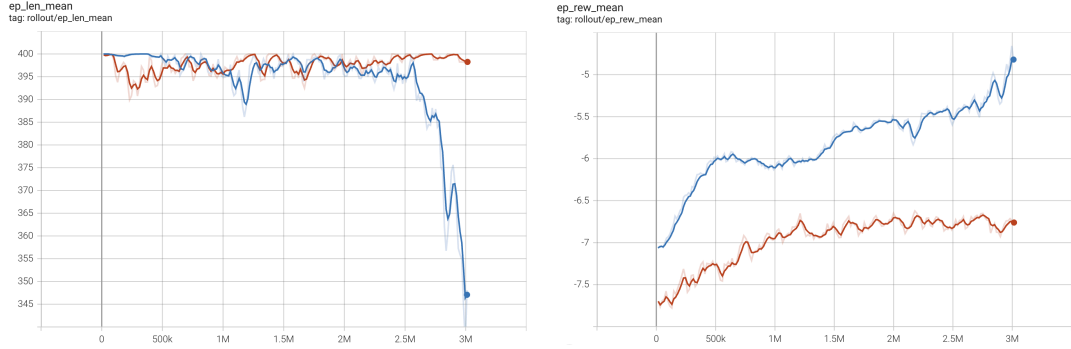


Figure 4.16. Training graphs: left - mean episode length, right - mean episode reward during training, comparing fixed core (blue) and random core (red) location. Produced by author.

The mean episode length graph shows that while the fixed core agent’s episodes began to terminate earlier as it learned efficient policies, the random core agent maintained near-maximum episode lengths, suggesting ongoing exploration without converging to an optimal stopping condition. Overall, while randomising the core location effectively enhanced spatial output diversity, the random core agent failed to reach performance objectives defined by R1 (built-up ratio) and R4 (compactness) within the same training horizon. To overcome this, future training stages may extend the number of training steps.

In the second experimental setting, the agent was trained within a fixed-core scenario for an 8-story building, where each episode consisted of 1200 steps and training was conducted for a total of 1 million timesteps (See Appendix B for RL configuration, best-performing hyperparameters). A trained model was sampled every 10,000 steps for evaluation. The training curve depicting the mean episode reward (Figure 4.17)

and 3D visualizations of episode outputs every 10k steps (Figure 4.18) were used to analyze the progression of learning and spatial design behaviors over time.

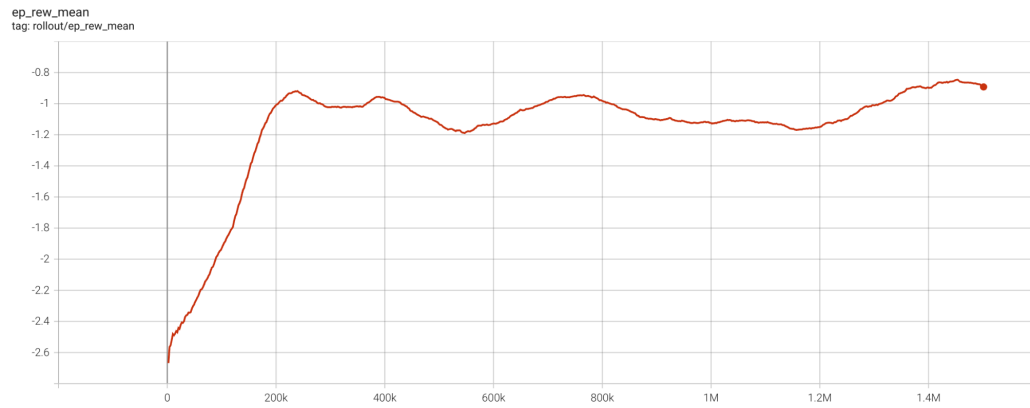


Figure 4.17. Training graph: Mean Episode Reward over 1,5M Episodes, eight-floor building with a fixed core location trained for 1200 steps. Produced by author.

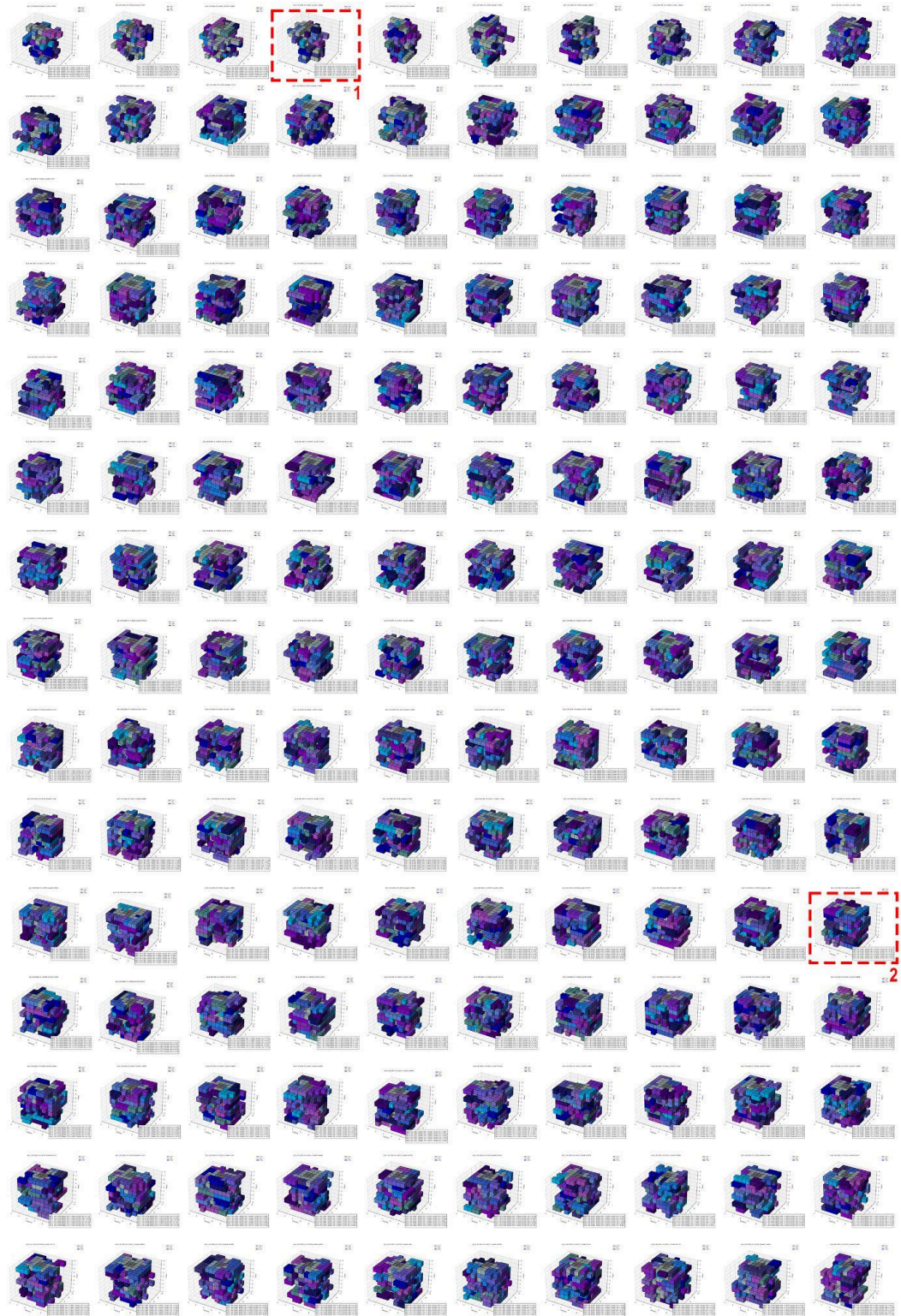


Figure 4.18. Outputs generated every 10k episodes by the trained model in order to demonstrate the learning process of the model: eight-floor building with a fixed core location trained for 1200 steps. Produced by author.

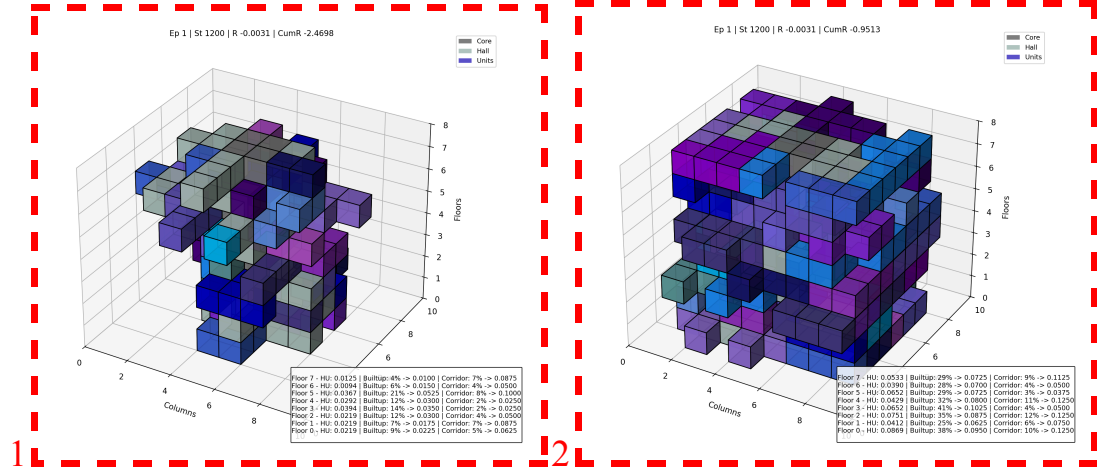


Figure 4.19. 3D Isometric Visualization of Building Output at (left) Episode 40,000 | (right) Episode 1,100,000. Produced by author.

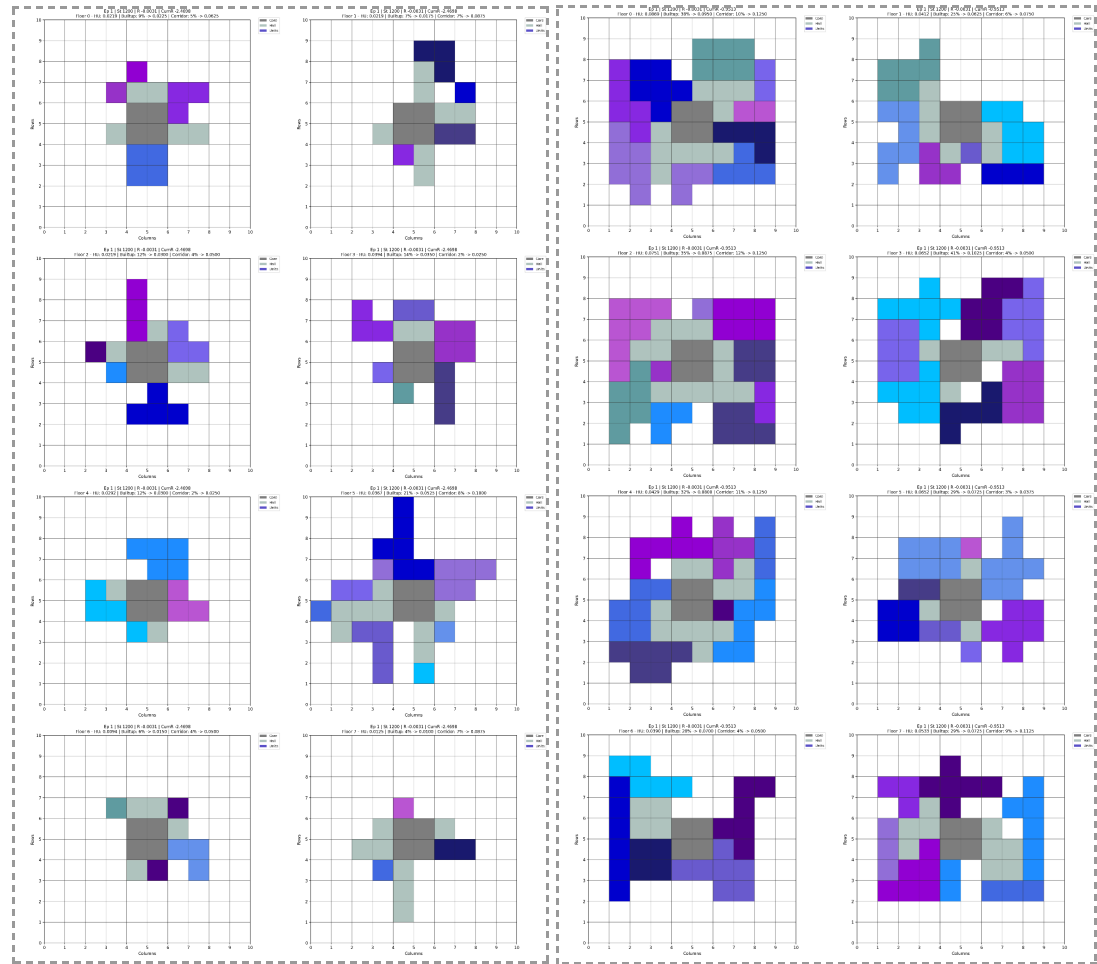


Figure 4.20. Floor-by-Floor Plan Views Episode at (left) Step 40,000 | (right) Step 1,100,000. Produced by author.

A clear upward trend in the mean episode reward is observed during the first 1.5 million steps (Figure 4.17). This early phase represents a learning window where the agent gradually improves its performance in selecting feasible actions within the fixed episode length. Initially, the agent struggles to populate the design space meaningfully; however, by step $\sim 240k$, the model demonstrates significant improvements in spatial expansion, spatial coherence, and reward optimization. This is supported by both the increase in mean episode reward and the qualitative improvements seen in the 3D visualizations of building outputs.

After this initial phase, a performance plateau and slight decline follow, indicating that the agent reaches a local optimum in policy space and fails to make further significant advances. This stagnation may result from increased combinatorial complexity in 3D spatial decision-making, especially under multi-objective reward constraints and limited training budget.

A detailed comparison between two selected episodes, one from an early phase (Episode at Step 40k) and one from the more optimized phase (Episode at Step 1.1M) reveals tangible improvements in architectural layout and performance metrics.

Early-stage output (Step 40k): The design is sparse, fragmented, and exhibits low coverage.

Floor 7	- HU: 0.0125	Builtup: 4%	-> 0.0100	Hall: 7%	-> 0.0875
Floor 6	- HU: 0.0094	Builtup: 6%	-> 0.0150	Hall: 4%	-> 0.0500
Floor 5	- HU: 0.0367	Builtup: 21%	-> 0.0525	Hall: 8%	-> 0.1000
Floor 4	- HU: 0.0292	Builtup: 12%	-> 0.0300	Hall: 2%	-> 0.0250
Floor 3	- HU: 0.0394	Builtup: 14%	-> 0.0350	Hall: 2%	-> 0.0250
Floor 2	- HU: 0.0219	Builtup: 12%	-> 0.0300	Hall: 4%	-> 0.0500
Floor 1	- HU: 0.0219	Builtup: 7%	-> 0.0175	Hall: 7%	-> 0.0875
Floor 0	- HU: 0.0219	Builtup: 9%	-> 0.0225	Hall: 5%	-> 0.0625

Built-up cell ratios are relatively low. The total cumulative reward is also notably low (CumR = -2.4698), indicating underperformance across all defined reward

metrics. This is especially evident in the hall distribution, where isolated clusters fail to achieve meaningful circulation continuity (Figure 4.20 (left)).

Later-stage output (Step 1.1M): The agent produces denser, more organized configurations with higher built-up and hall ratios, with noticeable hall connectivity.

Floor 7	- HU: 0.0533	Builtup: 29%	-> 0.0725	Hall: 9%	-> 0.1125
Floor 6	- HU: 0.0390	Builtup: 28%	-> 0.0700	Hall: 4%	-> 0.0500
Floor 5	- HU: 0.0652	Builtup: 29%	-> 0.0725	Hall: 3%	-> 0.0375
Floor 4	- HU: 0.0429	Builtup: 32%	-> 0.0800	Hall: 11%	-> 0.1250
Floor 3	- HU: 0.0652	Builtup: 41%	-> 0.1025	Hall: 4%	-> 0.0500
Floor 2	- HU: 0.0751	Builtup: 35%	-> 0.0875	Hall: 12%	-> 0.1250
Floor 1	- HU: 0.0412	Builtup: 25%	-> 0.0625	Hall: 6%	-> 0.0750
Floor 0	- HU: 0.0869	Builtup: 38%	-> 0.0950	Hall: 10%	-> 0.1250

The total reward improves significantly to CumR = -0.9513, reflecting learning progress, particularly in Built-up Cells Ratio (R1) and Hall Cells Ratio (R2). The floor plans exhibit improved zoning and internal layout logic, suggesting the agent has acquired an understanding of local adjacency and connectivity patterns to (Figure 4.20 (right)).

Despite the agent's improved ability to place more units over time, many of them remain elongated or fragmented due to vertical misalignment or scattered growth, resulting in suboptimal IoU values. This is clearly seen in the 3D isometric visualizations where unit volumes fail to form simple geometric clusters and often wrap around hall or core spaces inefficiently.

This limited performance in R4 underscores a key shortcoming of the current policy: while it learns to maximize occupancy (R1) and circulation logic (R2), it does not consistently prioritize spatial cohesion at the unit level. This could be attributed to the challenge of reconciling compactness objectives with other competing rewards under fixed episode length and constrained design space. Further architectural logic may be necessary to guide the model toward generating higher-quality unit layouts

with better compactness.

The performance observed in this multi-story training scenario does not match the level of success achieved in the fixed-core single-story training. This discrepancy is likely due to the significantly increased complexity of the design problem when extended to an eight-story configuration. While the single-floor setting enabled clearer learning and reward optimization, the multi-story environment introduced additional spatial and vertical coordination challenges, which appear to have limited the agent's learning capacity.

For this reason, the fixed-core training strategy, originally applied to enforce standardization in the single-story scenario, was reintroduced in the multi-story setting, in order to reduce design variability and stabilize the learning process. Given the inherently higher difficulty of the problem, constraining the core location was considered essential. Outputs generated under the alternative random-core setup are included in the Appendix C for comparison purposes.

The results of Experimental Setting II indicate that the agent achieved more stable learning compared to Setting I, largely due to the use of the PBRS special case with $\gamma = 1$. From a performance perspective, the trained agent demonstrated an improved ability to produce massing outputs that were consistent within the given constraints.

From an architectural perspective, however, the results highlight both opportunities and limitations. On one hand, the generated massings ensured accessibility and compactness within the defined voxel space, and a certain degree of variation was observed in unit mixtures. On the other hand, it did not fully learn to utilize hall cells in the most efficient way, and the simplification of constraints limited the

architectural richness of the results. This limitation reduces architectural richness and suggests the need for more advanced integration of diverse design criteria into the reward system.

These observations suggest that while the framework can provide meaningful outputs, it should not be considered as a final or exhaustive architectural solution. Instead, it demonstrates how RL can be positioned as a baseline exploration for architectural massing problems. The simplifications made in this study, such as treating the building as a mass configuration, rather than establishing inter-floor connections, underline the methodological focus. Although the voxel grid was set up as an eight-floor structure, vertical relationships beyond circulation were not explicitly modeled. Future studies could extend the framework by incorporating visual and physical inter-floor connections, as well as additional architectural constraints such as daylight, structural performance, and environmental criteria.

In summary, Experimental Setting II demonstrated more stable learning behavior and produced coherent spatial arrangements compared to the first setting. However, its architectural limitations indicate the need for further development.

CHAPTER 5

CONCLUSION AND FUTURE IMPLICATIONS

This thesis introduces a framework that integrates CA and DRL to explore the optimization of modular housing massing. The study demonstrates how the problem of spatial configuration can be reformulated into a machine-readable environment, enabling an RL agent to autonomously generate and refine housing layouts through iterative feedback. Although the resulting configurations do not aim to present a definitive or perfect architectural solution, they offer an initial step in examining how RL can operate within architectural problem spaces as a feedback-driven generative process.

From an architectural perspective, the outputs highlight both potentials and limitations. On one hand, the system was able to generate massing arrangements within architectural constraints, ensuring accessibility to all housing units and introducing variation in housing unit mixtures under defined rules. On the other hand, when assessed architecturally, the system did not fully learn to utilize hall cells in the most efficient way, and the simplification of constraints points to clear limitations that restrict architectural richness and quality. These shortcomings underline the need for a more advanced integration of architectural criteria into computational frameworks. In this sense, the study does not claim to resolve the design problem exhaustively, but rather to illustrate what emerging AI tools can begin to offer for architecture, and how they might be meaningfully incorporated into

design processes.

Key contributions of this thesis can be summarized as:

- This study proposes a new GD framework using DRL to optimize the spatial mass configuration of modular housing. Unlike rule-based or conventional generative methods, the agent learns through reward feedback, making the system more adaptable and capable of handling conflicting design objectives.
- A unique aspect of the framework is the use of CA to impose design constraints and define spatial availability. This hybrid approach reduces the agent's action space while preserving spatial logic, contributing a structured yet learnable environment design in architectural RL research.
- The study introduces a multi-faceted reward system tailored for architectural design goals, including built-up and hall area optimization, circulation efficiency, and housing unit compactness. These metrics bridge the gap between quantitative RL rewards and qualitative architectural performance indicators.
- This study reveals a critical policy failure in RL: the agent's tendency toward repeated "no-action" behaviors when trained with a poorly tuned shaping term in Potential-Based Reward Shaping (PBRS). In the initial experiment, the discount factor γ applied within the shaping reward caused beneficial actions to be undervalued, resulting in an unintended incentive where the agent deliberately avoided actions to preserve a non-negative reward. This unintended behavior highlighted a structural risk in applying discounted PBRS in constrained design environments with frequent invalid actions (sparse rewards during training). To address this, the thesis introduces a

special case of PBRS with $\gamma = 1$, based on the theoretical work of Wiewiora (2003), preserving policy invariance while avoiding double-discounting. This adjustment successfully mitigated the “no-action” failure mode, stabilized learning, and significantly improved agent performance across architectural design metrics. As a result, the PBRS tuning process and its observed effects provide a valuable **methodological case study** in the application of reward shaping within spatial design problems, offering transferable insights for future research in constrained, rule-based RL environments.

Despite the demonstrated potential of the proposed framework, several limitations were encountered throughout the study:

- The framework requires **significant computational resources**, particularly as the spatial complexity of the environment increases. As the number of design constraints, floors, and voxel cells grows, the training time and memory consumption increase substantially. This poses a scalability challenge for extending the framework to real-world design problems.
- The agent’s learning performance is highly sensitive to the design of the reward function. Fine-tuning the reward structure requires **extensive experimentation**, as even minor imbalances can lead to suboptimal or undesired behavior (See Appendix D).
- The availability of cells, as defined by the CA model, poses challenges by causing a high number of unperformed actions. When the agent selects an unavailable cell, it leads to multiple attempts before a valid action is taken. This leads to repeated failed attempts before a valid placement can be made, producing a sequence of negative or zero rewards that **dilute learning signals**

and slow policy convergence. To mitigate the learning inefficiency caused by these invalid actions, a constant penalty was introduced in this study.

While this approach was easy to implement, it does not fully prevent the agent from exploring non-viable actions. As a result, learning progresses more slowly, particularly in complex multi-story configurations where the number of available actions is significantly lower than the total voxel count.

- The current implementation uses a Multi-Layer Perceptron (MLP) policy architecture, which limits the agent's ability to generalize across different building sizes and topologies. Since MLPs do not encode spatial relationships explicitly, the building dimensions were kept fixed throughout the experiments to maintain architectural consistency.

This study serves as a foundation for future applications. Building on the findings and limitations discussed in this thesis, several directions for future research are proposed to enhance the scalability, efficiency, and design relevance of the presented DRL-based framework:

- Since DRL introduced adaptability and the capacity to search through design alternatives beyond fully predefined rules. RL, as a simulation-based process, could be further strengthened by the integration of architectural simulation tools (e.g., daylight, energy, or environmental performance feedback). Such an integration could enable multi-objective optimization that aligns more closely with architectural design goals.
- Vertical connections and inter-floor relationships, which were beyond the scope of this study, can be included to investigate multi-level architectural performance. Treating the building not only as a massing configuration but as

a spatially and visually interconnected system would significantly enhance its architectural relevance.

- The high frequency of invalid actions observed in the current implementation hampers learning progress and slows convergence. Future studies could integrate dynamic **action masking** to prevent the agent from selecting unavailable cells, thereby increasing reward signal clarity and training efficiency.
- In the current framework, spatial connectivity (e.g., access to the vertical core) is enforced through hard-coded CA constraints. An alternative direction would be to embed **accessibility as a soft objective** within the reward function, allowing the agent to learn diverse spatial strategies that meet circulation goals while optimizing other performance metrics. This may introduce additional learning complexity but would offer greater design flexibility.
- The use of a MLP in this study constrains the agent to operate within fixed-size voxel grids. Future work could explore more expressive architectures such as **Graph Neural Networks (GNNs) or attention-based encoders**, which can better capture spatial relationships and support generalization across buildings of different shapes and sizes.
- While the current reward structure includes basic spatial metrics (e.g., built-up ratio, compactness), future research can enhance the practical use of this method by considering **additional performance factors** not addressed in this study. Specifically, environmental performance factors could be incorporated to optimize spatial arrangements.

In conclusion, this thesis should be considered as a baseline exploration: it does not deliver a finalized design solution, but rather opens a pathway for how RL can be positioned in architectural research. The findings highlight the potential of RL to contribute to design by offering new ways of navigating complexity, exploring variation, and integrating feedback systems. Ultimately, the value of this study lies not in the exhaustiveness of its outputs, but in showing how emerging computational methods may reshape the questions architects can ask and the processes they can adopt in the future.

REFERENCES

- Adeyemi, A. B., Ohakawa, T. C., Okwandu, A. C., Iwuanyanwu, O., & Gil-Ozoudeh, I. (2024). Integrating modular and prefabricated construction techniques in affordable housing: Architectural design considerations and benefits. *Comprehensive Research and Reviews in Science and Technology*, 2(1), 10-19. <https://doi.org/10.57219/errst.2024.2.1.0030> (errjournals.com, researchgate.net)
- Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623-2631. <https://doi.org/10.1145/3292500.3330701>
- Akizuki, Y., Bernhard, M., Kakooee, R., Kladeftira, M., & Dillenburger, B. (2020). Generative modeling with design constraints: Reinforcement learning for object generation. In D. Holzer, W. Nakapan, A. Globa, & I. Koh (Eds.), *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th CAADRIA Conference - Volume 1* (pp. 445-454). Chulalongkorn University. <https://doi.org/10.52842/conf.caadria.2020.1.445>
- Almashaqbeh, M., & El-Rayes, K. (2021). Optimizing the modularization of floor plans in modular construction projects. *Journal of Building Engineering*, 39, 102316. <https://doi.org/10.1016/j.jobbe.2021.102316>
- de Almeida, A., Taborda, B., Santos, F., Kwiecinski, K., & Eloy, S. (2016). A genetic algorithm application for automatic layout design of modular residential homes. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2774–2778). IEEE. <https://doi.org/10.1109/SMC.2016.7844659>
- Aman, J., Tabassum, N., Hopfenblatt, J., Kim, J. B., & Haque, M. O. (2021). Optimizing container housing units for informal settlements: A parametric simulation & visualization workflow for architectural resilience. In A. Globa,

- J. van Ameijde, A. Fingrut, N. Kim, & T. T. S. Lo (Eds.), *PROJECTIONS – Proceedings of the 26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2021), Volume 1* (pp. 51–60). The Chinese University of Hong Kong.
<https://doi.org/10.52842/conf.caadria.2021.1.051>
- American Institute of Architects. (2019). *Design for modular construction: An introduction for architects*. Retrieved from,
https://content.aia.org/sites/default/files/2019-03/Materials_Practice_Guide_Modular_Construction.pdf
- Araghi, S. K., & Stouffs, R. (2015). Exploring cellular automata for high density residential building form generation. *Automation in Construction*, 49(Part A), 152-162. <https://doi.org/10.1016/j.autcon.2014.10.007>
- Ashour, S., & gogo, S. (2024). Boosting the Design Process Using a Proposed Methodology Based on Computational Design. *ERJ. Engineering Research Journal*. <https://doi.org/10.21608/erjm.2024.244713.1301>
- Baghdadi, A., Heristchian, M., & Kloft, H. (2020). Design of prefabricated wall-floor building systems using meta-heuristic optimization algorithms. *Automation in Construction*, 114, 103156.
<https://doi.org/10.1016/j.autcon.2020.103156>
- Banihashemi, S., Tabadkani, A., & Hosseini, M. R. (2018). *Integration of parametric design into modular coordination: A construction waste reduction workflow*. *Automation in Construction*, 88, 1–12.
<https://doi.org/10.1016/j.autcon.2017.12.026>
- Bertram, N., Fuchs, S., Mischke, J., Palter, R., Strube, G., & Woetzel, J. (2019, June). *Modular construction: From projects to products*. McKinsey & Company.
<https://www.mckinsey.com/~media/mckinsey/business%20functions/operation/s/our%20insights/modular%20construction%20from%20projects%20to%20products%20new/modular-construction-from-projects-to-products-full-report-new.pdf>
- Bonafede, A., & Erioli, A. (2022). Versus Habitat – Multi-agent spatial negotiation

for topology-aware, large-scale architectural assemblages. In *eCAADe 2022: Co-creating the Future – Inclusion in and through Design. Proceedings of the 40th eCAADe Conference – Volume 2* (pp. 113–122).

<https://doi.org/10.52842/conf.ecaade.2022.2.113>

Calisir Adem, P., Cagdas, G. (2020). Computational Design Thinking through Cellular Automata: Reflections from Design Studios. *Journal of Design Studio*, 2(2), 71-83.

Cao, Y., Sheng, Q. Z., McAuley, J., & Yao, L. (2023). *Reinforcement learning for generative AI: A survey*. arXiv. <https://doi.org/10.48550/arXiv.2308.14328>

Chaillou, S. (2019). AI + Architecture | Towards a New Approach. Harvard GSD Thesis, 2019, retrieved from <http://stanislaschaillou.com/articles.html>

Chaillou, S. (2022). Artificial Intelligence and Architecture: From Research to Practice. Chaillou, S. (2022). Birkhäuser. Retrieved from, <https://www.degruyter.com/document/doi/10.1515/9783035624045/html>

Chang, K.-H., Cheng, C.-Y., Luo, J., Murata, S., Nourbakhsh, M., & Tsuji, Y. (2021). *Building-GAN: Graph-conditioned architectural volumetric design generation* (arXiv preprint arXiv:2104.13316). arXiv. <https://doi.org/10.48550/arXiv.2104.13316>

ChatGPT (OpenAI, 2024) was used to (1) was used to correct errors in spelling and grammar and (2) to enhance the clarity and flow of the text.

Chilton, J. (2000). *Space grid structures*. Architectural Press. Retrieved July 24, 2025, from https://www.academia.edu/32969545/CHILTON_2000_Space_grid_structures

Choi, J. O., Chen, X. B., & Kim, T. W. (2019). Opportunities and challenges of modular methods in dense urban environment. *International Journal of Construction Management*, 19(2), 93-105. <https://doi.org/10.1080/15623599.2017.1382093>

Dörfler, V. (2022). Artificial Intelligence. The SAGE Encyclopedia of Theory in Science, Technology, Engineering, and Mathematics, DOI:

- Djukanovic, M., Alegre, A., & Teixeira Bastos, F. (2025). Prefabricated solutions for housing: Modular architecture and flexible living spaces. *Buildings*, 15(6), 862. <https://doi.org/10.3390/buildings15060862>
- El Messeidy, R. (2018). Adapting shipping containers as temporary shelters in terms of recycling, sustainability and green architecture: Reuse as accommodation in Egypt. *Engineering Research Journal*, 160, 173-191. <https://doi.org/10.21608/erj.2018.139572>
- El-Zanfaly, D. (2009). Design by algorithms: A generative design system for modular housing arrangement. *Proceedings of the 2009 Sigradi Conference*, 65. Retrieved from https://papers.cumincad.org/data/works/att/sigradi2009_835.content.pdf
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338. <https://doi.org/10.1007/s11263-009-0275-4>
- Ferdous, W., Bai, Y., Ngo, T. D., Manalo, A., & Mendis, P. (2019). New advancements, challenges and opportunities of multi-storey modular buildings - A state-of-the-art review. *Engineering Structures*, 183, 883-893. <https://doi.org/10.1016/j.engstruct.2019.01.061>
- Frazer, J. (1995). An evolutionary architecture. *Architectural Association publications*, Themes VII.
- Ghannad, P., & Lee, Y.-C. (2022). Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN). *Automation in Construction*, 139, 104234. <https://doi.org/10.1016/j.autcon.2022.104234>
- Gibb, A., & Isack, F. (2003). Re-engineering through pre-assembly: Client expectations and drivers. *Building Research & Information*, 31(2), 146-160. <https://doi.org/10.1080/09613210302000>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. *MIT Press*.

- Goodier, C. I., & Gibb, A. G. F. (2007). Future opportunities for offsite in the UK. *Construction Management and Economics*, 25(6), 585–595.
<https://doi.org/10.1080/01446190601071821>
- Goulding, J. S., Pour Rahimian, F., Arif, M., & Sharp, M. D. (2014). New offsite production and business models in construction: priorities for the future research agenda. *Architectural Engineering and Design Management*, 11(3), 163-184. <https://doi.org/10.1080/17452007.2014.891501>
- Graham Foundation. (2015). *2015 grants to individuals*. Retrieved from http://www.grahamfoundation.org/system/press/pdf/40/original/PressRelease_2015GrahamFoundation_GrantstoIndividuals_20150527_LINKED_2.pdf
- Grammarly Inc. (2024) Grammarly: AI Writing and Grammar Checker App (Version 14.1198.0) [Large Language Model] was used to correct errors in spelling and grammar.
- Günay, G. (2025). *Development of standards for modular construction: Learning from the freight container* (Doctoral dissertation, Middle East Technical University). Middle East Technical University Institutional Repository.
- Hu, R., Huang, Z., Tang, Y., Van Kaick, O., Zhang, H., & Huang, H. (2020). *Graph2Plan: Learning floorplan generation from layout graphs*. *ACM Transactions on Graphics*, 39(4), Article 118.
- Huang, S., & Ontañón, S. (2022). *A closer look at invalid action masking in policy gradient algorithms* (arXiv:2006.14171). arXiv.
<https://doi.org/10.48550/arXiv.2006.14171>
- Hosmer, T., Tigas, P., Reeves, D. & He, Z. (2020). Spatial Assembly with Self-Play Reinforcement Learning. In 40th Annual Conference of the Association of Computer Aided Design in Architecture in Online and Global: Distributed Proximities, ACADIA 2020 (pp. 382-393). The Association of Computer-Aided Design in Architecture (ACADIA).
<https://doi.org/10.52842/conf.acadia.2020.1.382>
- Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des

Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37, 547-579.

<https://www.biodiversitylibrary.org/page/42443655>

Kakooee, R., & Dillenburger, B. (2025). *Illuminating spaces: Deep reinforcement learning and laser-wall partitioning for architectural layout generation* (arXiv preprint arXiv:2502.04407). arXiv. <https://doi.org/10.48550/arXiv.2502.04407>

Kakooee, R., Dillenburger, B. (2024). Reimagining space layout design through deep reinforcement learning. *Journal of Computational Design and Engineering*, 11(3), 43-55. <https://doi.org/10.1093/jcde/qwae025>

Kakooee, R., & Dillenburger, B. (2022). Design process is a reinforcement learning problem (modified: 2024, November 3). Deep RL Workshop 2022. Retrieved from: <https://openreview.net/forum?id=INWF02POfp>

Kamali, M., & Hewage, K. (2016). Life cycle performance of modular buildings: A critical review. *Renewable and Sustainable Energy Reviews*, 62, 1171-1183. <https://doi.org/10.1016/j.rser.2016.05.031>

Kihlstedt, F. T. (1984). The Crystal Palace. *Scientific American*, 251(4), 132-143. <http://www.jstor.org/stable/24969462>

Jaillon, L., & Poon, C. S. (2010). Design issues of using prefabrication in Hong Kong building construction. *Construction Management and Economics*, 28(10), 1025–1042. <https://doi.org/10.1080/01446193.2010.498481>

Laovisutthichai, V., Lu, W., & Xue, F. (2020). Modular construction: Design considerations and opportunities. In *Proceedings of the 25th International Symposium on Advancement of Construction Management and Real Estate (CRIOCM 2020)* (Outstanding Paper Award; in press). Springer. <https://frankxue.com/pdf/laos20modular.pdf>

Lawson, M., Ogden, R., & Goodier, C. (2014). *Design in modular construction* (1st ed.). CRC Press. [\[https://doi.org/10.1201/b16607\]](https://doi.org/10.1201/b16607)(<https://doi.org/10.1201/b16607>)

Lee, C.-J., & Lim, S.-H. (2015, December 25). An Analysis on Architectural

Characteristics of Domestic Modular Housing and Building Material Standardization Effect through MC Design. Journal of the Korean housing association. The Korean Housing Association.

<https://doi.org/10.6107/jkha.2015.26.6.103>

Lee, G., & Widrig, D. (2020). *Making a Case for Modularity*. 3, 74-103.

<https://doi.org/10.31182/CUBIC.2020.3.025>

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M. I., & Stoica, I. (2018). RLlib: Abstractions for distributed reinforcement learning. *Proceedings of the 35th International Conference on Machine Learning (ICML)*. <https://arxiv.org/abs/1712.09381>

Lin, X., Su, P., Lu, W., & Guo, H. (2025). *ModulePacking: A top-down generative design approach for modular key plans*. Journal of Computing in Civil Engineering, 39(1), 04023065.

Liu, S., Li, Z., Teng, Y., & Dai, L. (2022). A dynamic simulation study on the sustainability of prefabricated buildings. *Sustainable Cities and Society*, 77, 103551. <https://doi.org/10.1016/j.scs.2021.103551>

Liu, J., Qiu, Z., Wang, L. L., Liu, P., Cheng, G. H., & Chen, Y. (2024). Intelligent floor plan design of modular high-rise residential building based on graph-constrained generative adversarial networks. *Automation in Construction*. <https://doi.org/10.1016/j.autcon.2023.105264>

Li, Y., Hu, Y., Tan, T., Yu, B., Li, J., & Fingrut, A. (2023). AR-Assisted Assembly in Self-Build Construction with Discrete Components. Paper presented at 30th EG-ICE International Workshop on Intelligent Computing in Engineering, London, United Kingdom.

Modular Building Institute. (2019). *2019 Permanent Modular Construction Report*.

Modular Building Institute. Retrieved from

<https://www.modular.org/documents/public/images/2019-PMC-Report-reduced.pdf>

Mekawy, M., & Petzold, F. (2017). Exhaustive exploration of modular design options to inform decision making. In A. Fioravanti, S. Corsi, S. Elahmar, S. Gargaro,

- G. Loffreda, G. Novembri, & A. Trento (Eds.), *ShoCK! - Sharing Computational Knowledge! Proceedings of the 35th eCAADe Conference – Volume 2* (pp. 107–114). Sapienza University of Rome.
<https://doi.org/10.52842/conf.ecaade.2017.2.107>
- Mukkavaara, J., & Sandberg, M. (2020). Architectural Design Exploration Using Generative Design : Framework Development and Case Study of a Residential Block. *Buildings*, 10(11), 201. <https://doi.org/10.3390/BUILDINGS10110201>
- Muradov, A. N., & Grabovyy, K. P. (2024). *Ecological aspects of development of modular real-estate construction industry in modern conditions*. E3S Web of Conferences, 535, 01022. <https://doi.org/10.1051/e3sconf/202453501022>
- Nam, S., Yoon, J., Kim, K., & Choi, B. (2019). Optimization of Prefabricated Components in Housing Modular Construction. *Sustainability*, 12(24), 10269. <https://doi.org/10.3390/su122410269>
- Nauata, N., Chang, KH., Cheng, CY., Mori, G., Furukawa, Y. (2020). House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, JM. (eds) Computer Vision - ECCV 2020. ECCV 2020. Lecture Notes in Computer Science(), vol 12346. Springer, Cham.
https://doi.org/10.1007/978-3-030-58452-8_10
- Ng, A. Y., Harada, D., & Russell, S. J. (1999). *Policy invariance under reward transformations: Theory and application to reward shaping*. ICML.
- O'Neill, D., & Organ, S. (2016). A literature review of the evolution of British prefabricated low-rise housing. *Structural Survey*, 34(2), 191-214.
<https://doi.org/10.1108/SS-08-2015-0037>
- OpenAI. (2024). ChatGPT Model: GPT-4 | Temp: 0.7 [Large Language Model].
<https://openai.com/chatgpt/>
- Pasetti Monizza, G., Bendetti, C., & Matt, D. T. (2018). Parametric and generative design techniques in mass-production environments as effective enablers of Industry 4.0 approaches in the building industry. *Automation in Construction*,

92, 270–285. <https://doi.org/10.1016/j.autcon.2018.02.027>

- Peña, M.L.C., Carballal, A., Rodríguez-Fernández, N., Santos, I., & Romero, J. (2021). *Artificial intelligence applied to conceptual design: A review of its use in architecture*. *Automation in Construction*, 124, 103550.
- Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley.
- Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 1-8.
<https://github.com/DLR-RM/stable-baselines3>
- Razkenari, M., Fenner, A., Shojaei, A., Hakim, H., & Kibert, C. (2020). Perceptions of offsite construction in the United States: An investigation of current practices. *Journal of Building Engineering*, 29, 101138.
<https://doi.org/10.1016/j.jobe.2019.101138>
- Safdie Architects. (n.d.). *Habitat 67*. Retrieved July 25, 2025, from <https://www.safdiearchitects.com/projects/habitat-67>
- Sajid, Z. W., Ullah, F., Qayyum, S., & Masood, R. (2024). Climate change mitigation through modular construction. *Smart Cities*, 7(1), 566-596.
<https://doi.org/10.3390/smartcities7010023>
- Samadi, P., Sedghimehr, S., Shevidi, S., & Andaji Garmaroodi, A. (2025). A generative framework for designing linear-modular housing integrated with a real-time user interface. *Nexus Network Journal*, 27(2), 355–371.
<https://doi.org/10.1007/s00004-024-00802-y>
- Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*,
<https://doi.org/10.48550/arXiv.1707.06347>
- Shahi, S., Wozniczka, P., Rausch, C., Trudeau, I., & Haas, C. (2021). A computational methodology for generating modular design options for building extensions. *Automation in Construction*, 127, 103700.

<https://doi.org/10.1016/j.autcon.2021.103700>

Sharafi, P., Samali, B., Ronagh, H., & Ghodrat, M. (2017). Automated spatial design of multi-story modular buildings using a unified matrix method. *Automation in Construction*, 82, 31-42. <https://doi.org/10.1016/j.autcon.2017.06.025>

Song, H., Ghaboussi, J., & Kwon, T.-H. (2016). Architectural design of apartment buildings using the Implicit Redundant Representation Genetic Algorithm. *Automation in Construction*, 72(Part 2), 166–173. <https://doi.org/10.1016/j.autcon.2016.09.001>

Su, P., Lu, W., Chen, J., & Hong, S. (2023). *Floor plan graph learning for generative design of residential buildings: A discrete denoising diffusion model*. Building Research & Information (online first).

Sutton, R. and Barto, A. (1998). Reinforcement Learning: An Introduction. MIT Press.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction (2nd ed.)*. The MIT Press. Retrieved from <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>

Terzidis, K. (2006). *Algorithmic Architecture (1st ed.)*. Routledge. <https://doi.org/10.4324/9780080461298>

Torrado, R. R., Ramos, J., Sánchez-González, A., Hesse, C., Besse, F., & Vinyals, O. (2023). Gymnasium: A standard API for reinforcement learning environments. <https://github.com/Farama-Foundation/Gymnasium>

United Nations. (2015). *Transforming our world: The 2030 Agenda for Sustainable Development* (A/RES/70/1). <https://sdgs.un.org/2030agenda>

Wang, D., & Snooks, R. (2021). Intuitive behavior - The operation of reinforcement learning in generative design processes. In A. Globa, J. van Ameijde, A. Fingrut, N. Kim, & T. T. S. Lo (Eds.), *PROJECTIONS - Proceedings of the 26th International Conference on Computer-Aided Architectural Design*

- Research in Asia (CAADRIA)*, Volume 1 (pp. 101-110). The Chinese University of Hong Kong. <https://doi.org/10.52842/conf.caadria.2021.1.101>
- Wang, D., & Snooks, R. (2021). Artificial intuitions of generative design: An approach based on reinforcement learning. In P. F. Yuan, J. Yao, C. Yan, X. Wang, & N. Leach (Eds.), *Proceedings of the 2020 DigitalFUTURES* (pp. 255-266). Springer. https://doi.org/10.1007/978-981-33-4400-6_18
- Weber, R.E., Müller, C., & Reinhart, C. (2022). *Automated floorplan generation in architectural design: A review of methods and applications*. *Automation in Construction*, 140, 104385.
- Wei, Y., Choi, H., & Lei, Z. (2022). A generative design approach for modular construction in congested urban areas. *Smart and Sustainable Built Environment*, 11(4), 1163–1181. <https://doi.org/10.1108/SASBE-04-2021-0068>
- Wiewiora, E. (2003). Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19, 205-208.
- Wolfram, S. (1983). Statistical Mechanics of Cellular Automata. *Reviews of Modern Physics*, 55, 601-644
- Young, B.E., Seidu, R.D., Thayaparan, M. & Appiah-Kubi, J. (2020). Modular Construction Innovation in the UK: The Case of Residential Buildings. The 10th Annual Industrial Engineering and Operations Management (IEOM) Conference. Dubai 10 - 12 Mar 2020 IEOM Society, <https://openresearch.lsbu.ac.uk/item/8q1x6> .
- Zhang, W., Lee, M. W., Jaillon, L., & Poon, C. S. (2018). The hindrance to using prefabrication in Hong Kong's building industry. *Journal of Cleaner Production*, 204, 70–81. <https://doi.org/10.1016/j.jclepro.2018.08.190>
- Zhang, J., Liu, N., & Wang, S. (2021). Generative design and performance optimization of residential buildings based on parametric algorithm. *Energy and Buildings*, 244, 111033. <https://doi.org/10.1016/j.enbuild.2021.111033>
- Zhao, H., Savov, A., Zhang, H. & Dillenburger, B. (2023). A Framework for the Design and Evaluation of Architectural Tilesets. In *41st Conference on*

Education and Research in Computer Aided Architectural Design in Europe: Volume 2, eCAADe 2023 (pp. 491-500). The Association of Education and research in Computer Aided Architectural Design in Europe (eCAADe), <https://doi.org/10.52842/conf.ecaade.2023.2.491>.

Zhu, A., Dai, T., Xu, G., Pauwels, P., de Vries, B., & Fang, M. (2023). Deep Reinforcement Learning for RealTime Assembly Planning in Robot-Based Prefabricated Construction. *IEEE Transactions on Automation Science and Engineering*, 20(3), 1515-1526. Article 10021301. <https://doi.org/10.1109/TASE.2023.3236805>

APPENDICES

APPENDIX A

RL ENVIRONMENT SETUP

This appendix provides further details on the RL environment setup, implementation, and training workflow developed in this study. It aims to ensure the reproducibility of the experiments and support future research. The following sections outline the development process across different libraries, training stages, and tuning procedures.

The model was trained within a customized Gymnasium environment. Gymnasium, developed by OpenAI, is a lightweight Python library that provides a standard interface for RL environments (Torrado et al., 2023). In this study, it is used to implement a custom 3D voxel-based environment, allowing the RL agent to interact with the spatial design space through structured observations and actions. The agent was trained using the PPO algorithm (Schulman et al., 2017) with the MLP network. Two RL libraries were used during the implementation phase: Ray RLlib in the first experiment setting and Stable-Baselines3 (SB3) in the second.

In experiment setting 1, the RL model was implemented using Ray RLlib, which is built on PyTorch and supports distributed training and flexible algorithm structures (Liang et al., 2018). RLlib provides useful features for scaling and experimentation. However, its configuration model is complex and requires additional effort to adapt to the specific needs of this study. Managing custom environment definitions, reward modifications, and training parameters involved navigating multiple layers of abstraction. Due to this complexity, and relatively small scale of the environment

defined in this study, a simpler framework was sought for the second stage.

In experiment setting 2, the implementation was migrated to SB3. Also based on PyTorch, SB3 offers a lighter and more user-friendly structure with a well documented clean API (Raffin et al., 2021). Its compatibility with Gymnasium made integration straightforward, and it provided built-in support for TensorBoard and Optuna (Akiba et al., 2019), which helped streamline training and tuning. For these reasons, SB3 was preferred for the later and final phases of the project implementation.

The training workflow consisted of four main stages:

1. Environment setup and Initial Training with Default Parameters

The PPO agent was first trained using SB3’s default hyperparameters. This phase helped to test and debug the custom Gymnasium environment, as well as the reward logic. Running initial experiments with default values allowed for validating the basic behavior of the agent and refining the spatial rules within the environment through fast iterations before more computationally intensive processes.

2. Reward Shaping and Environment Refinement

Once the basic environment was functioning as expected, the reward function was shaped in alignment with design goals. The reward signal was structured using a PBRS method (Ng et al., 1999) to provide continuous feedback and guide the agent more effectively.

3. Hyperparameter Tuning with Optuna

After finalizing the environment and reward logic, the hyperparameters were optimized using Optuna, an open-source hyperparameter search framework.

The tuning process aimed to maximize the average episodic reward by testing various combinations of model and training parameters. Each trial consisted of a short training run and was evaluated based on the agent’s performance. The tuning process was conducted approximately 48 hours on a MacBook Pro with Apple M1 chip (8-core CPU, 16GB memory) and represented the most computationally intensive phase of the entire process.

To support the evaluation of different hyperparameter configurations during the tuning process, key training metrics were logged and visualized using TensorBoard. These graphs collectively illustrate the behavior of multiple Optuna trials over the course of training. Metrics such as average episode length, episodic reward, policy and value losses, entropy, KL divergence, explained variance, and learning rate were tracked to assess learning stability, convergence, and exploration quality across candidate configurations. The visualizations helped identify unstable trials, overfitting behavior, and early saturation, guiding the selection of the most promising hyperparameter set used in the final model.

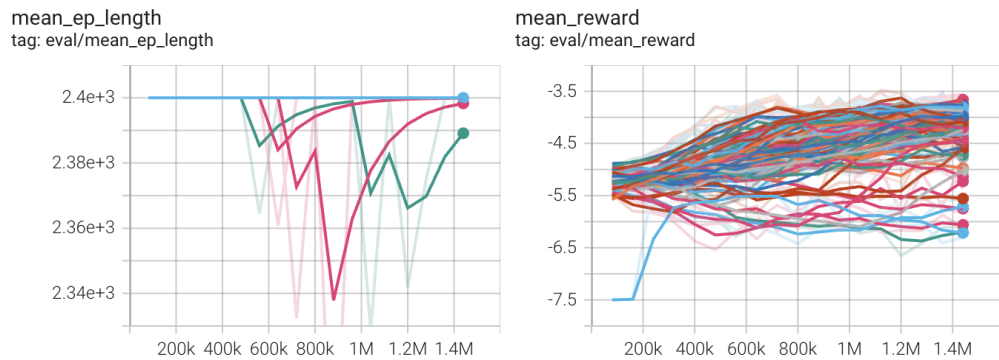


Figure A.1. Training graphs: left - mean episode length, right - mean episode reward during hyperparameter tuning. Produced by author.



Figure A.2. Training diagnostics for all Optuna trials, shown from left to right, top to bottom: approximate KL divergence, clip fraction, clip range, entropy loss, explained variance, learning rate, total loss, policy gradient loss, and value loss. These graphs provide insights into convergence behavior, stability, and the learning dynamics across sampled configurations. Produced by author.

APPENDIX B

HYPERPARAMETER CONFIGURATION

The final model was trained using the best-performing hyperparameters configuration found through the Optuna search process. These values are summarized in Table below.

Category	Hyperparameter	Value	Description
Optimization	learning_rate	0.00019	Step size used in gradient updates
	batch_size	64	Number of samples used in each training batch
	n_epochs	7	Number of passes over each batch during training
Training Dynamics	n_steps	2048	Number of steps to run in each rollout
	gamma	0.932	Discount factor for future rewards
	gae_lambda	0.930	Controls bias vs. variance in advantage estimation
Loss Function Weights	clip_range	0.378	Range for clipping the policy objective (for stability)
	vf_coef	0.451	Weight of the value function loss in the total loss
	ent_coef	5.94e-04	Entropy bonus to encourage exploration
Model Architecture	activation_fn	tanh	Activation function used in the MLP policy network
	net_arch	64, 64	Hidden layer sizes in the policy/value networks

Table A.1. Tuned hyperparameters used in the final model, grouped by function. Values were selected via Optuna and rounded for clarity. Produced by author.

4. Final Training and Evaluation

Once the best-performing configuration was identified, the final model was trained over a longer duration to ensure convergence and policy stability. During this phase, training progress was continuously monitored using TensorBoard, which provided real-time insights into reward progression, loss trends, and policy behavior. This final model was used for evaluation and result analysis presented in the main chapters of the thesis.

The full implementation, custom environment, and training scripts are available at:

<https://github.com/BetulKelesEkici/kaizen>

APPENDIX C

COMPARATIVE TRAINING DATA

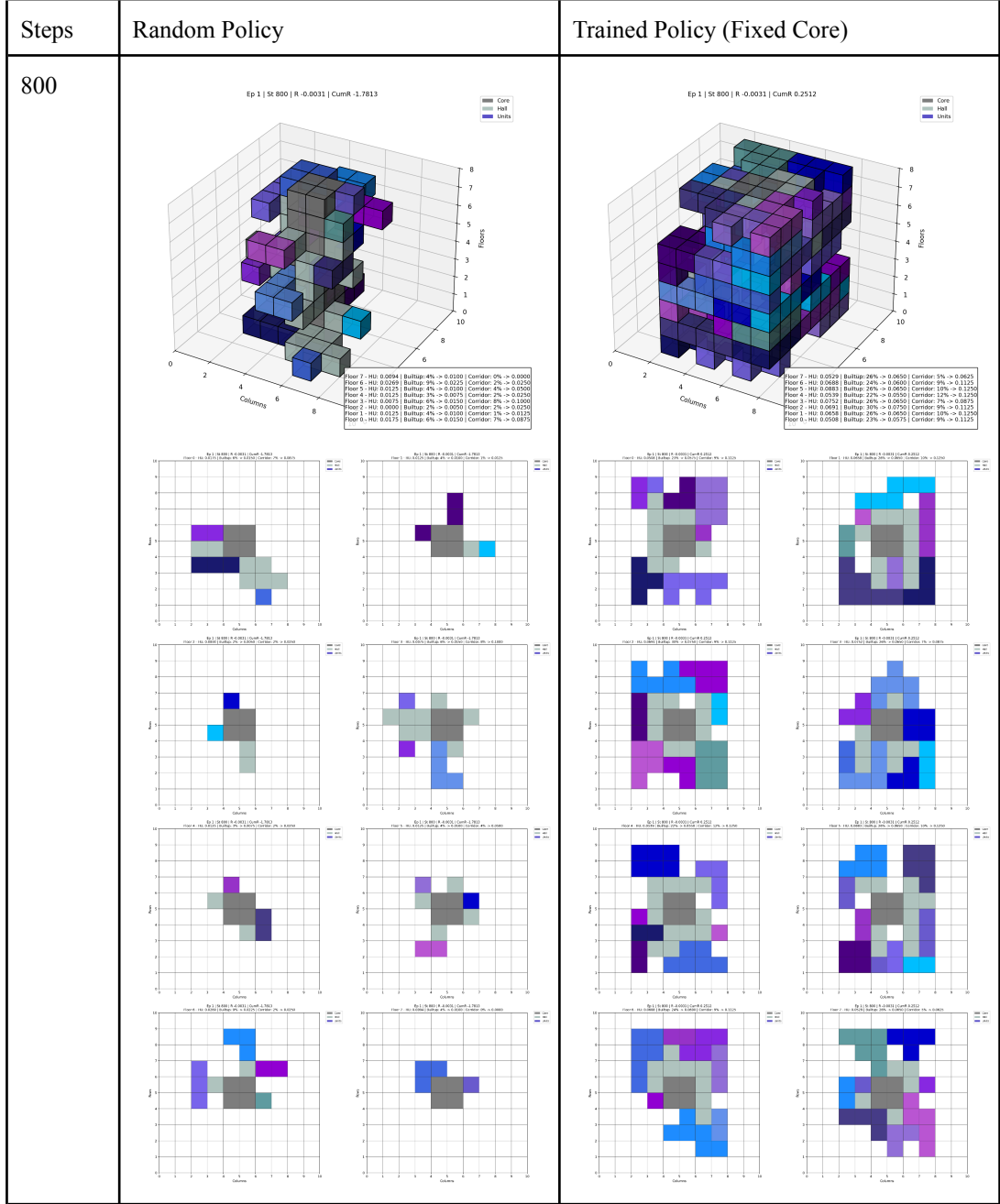


Table A.2. Comparison of design outputs generated by an untrained agent (random actions) and a trained agent with a fixed core location at 800 steps. The top row displays the 3D voxel configuration, while the subsequent rows show 2D floor slices. The trained agent exhibits more organized and compact spatial arrangements, reflecting learning progress at this stage. Produced by author.

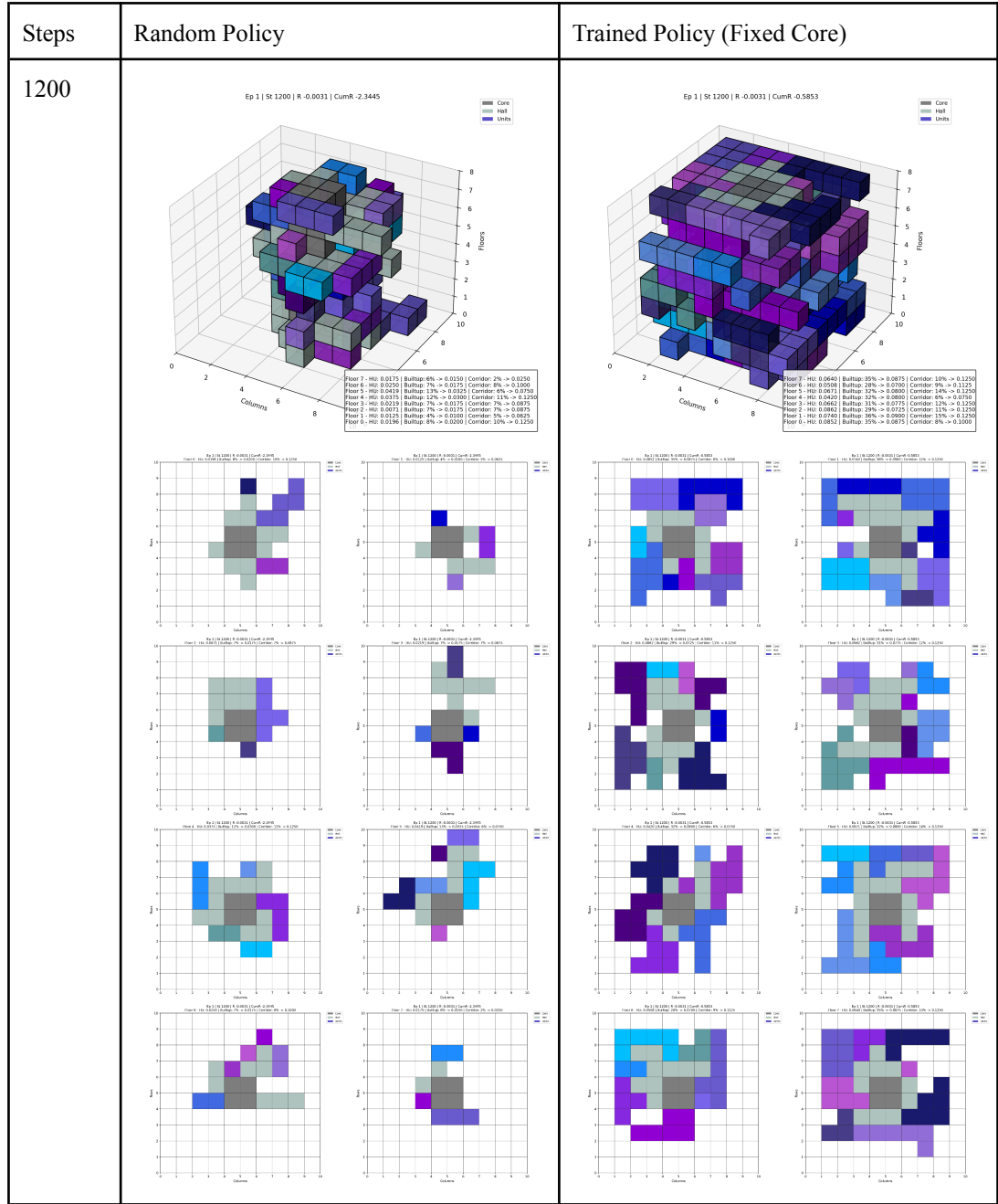


Table A.3. Comparison of untrained and trained model outputs at 1200 steps. Format and interpretation follow Table A.2. Produced by author.

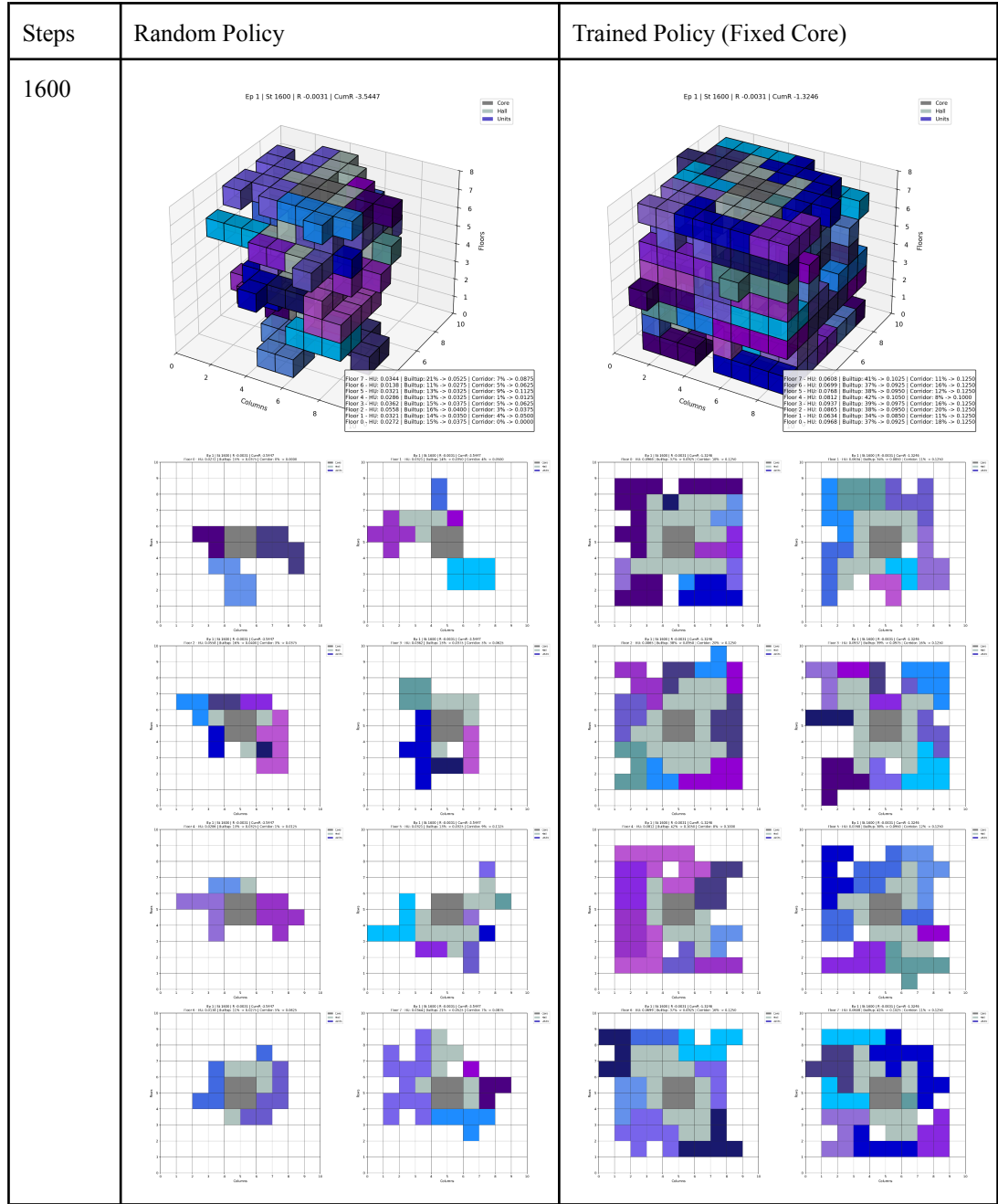


Table A.4. Comparison of untrained and trained model outputs at 1600 steps. Format and interpretation follow Table A.2. Produced by author.

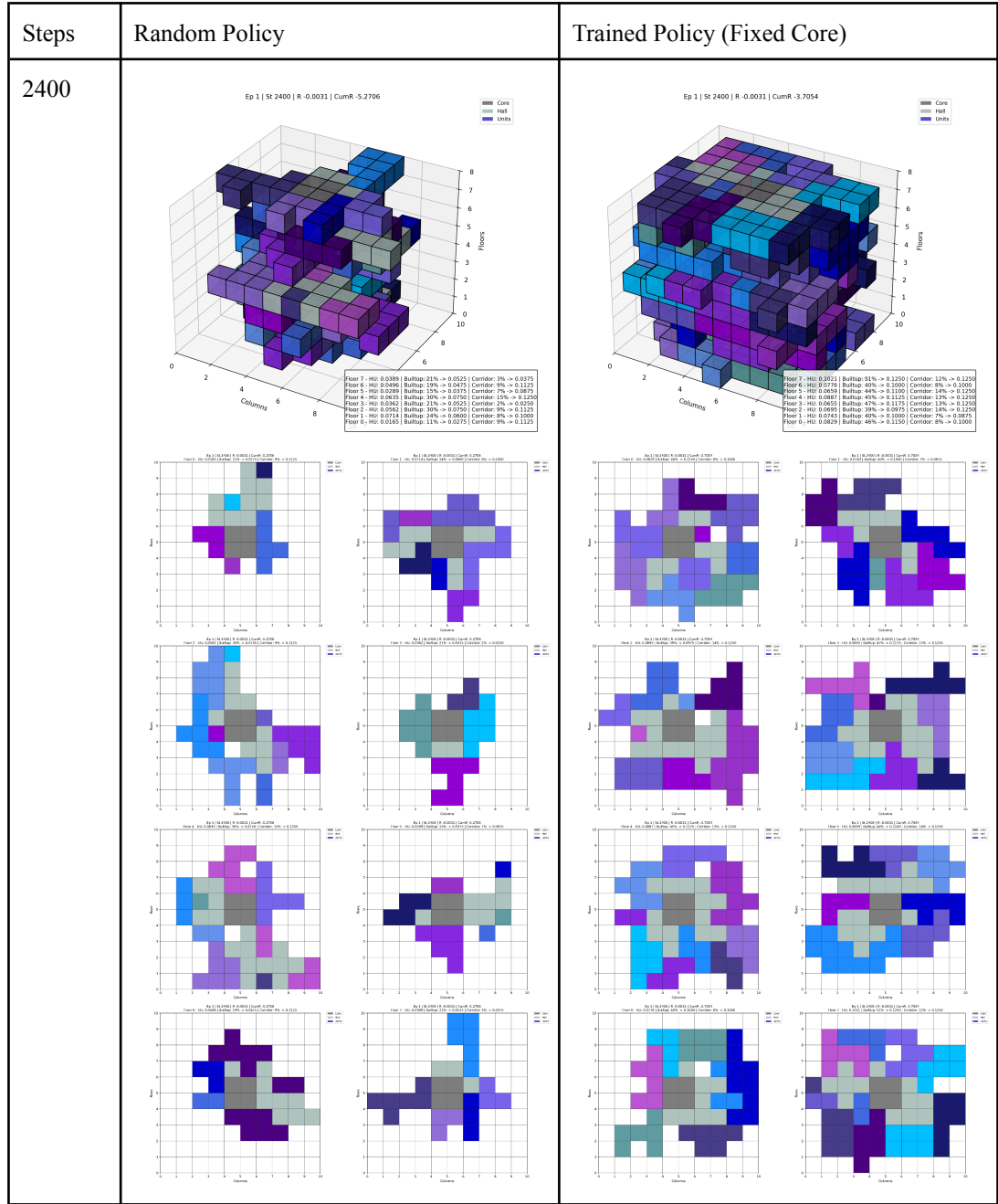


Table A.5. Comparison of untrained and trained model outputs at 2400 steps. Format and interpretation follow Table A.2. Produced by author.

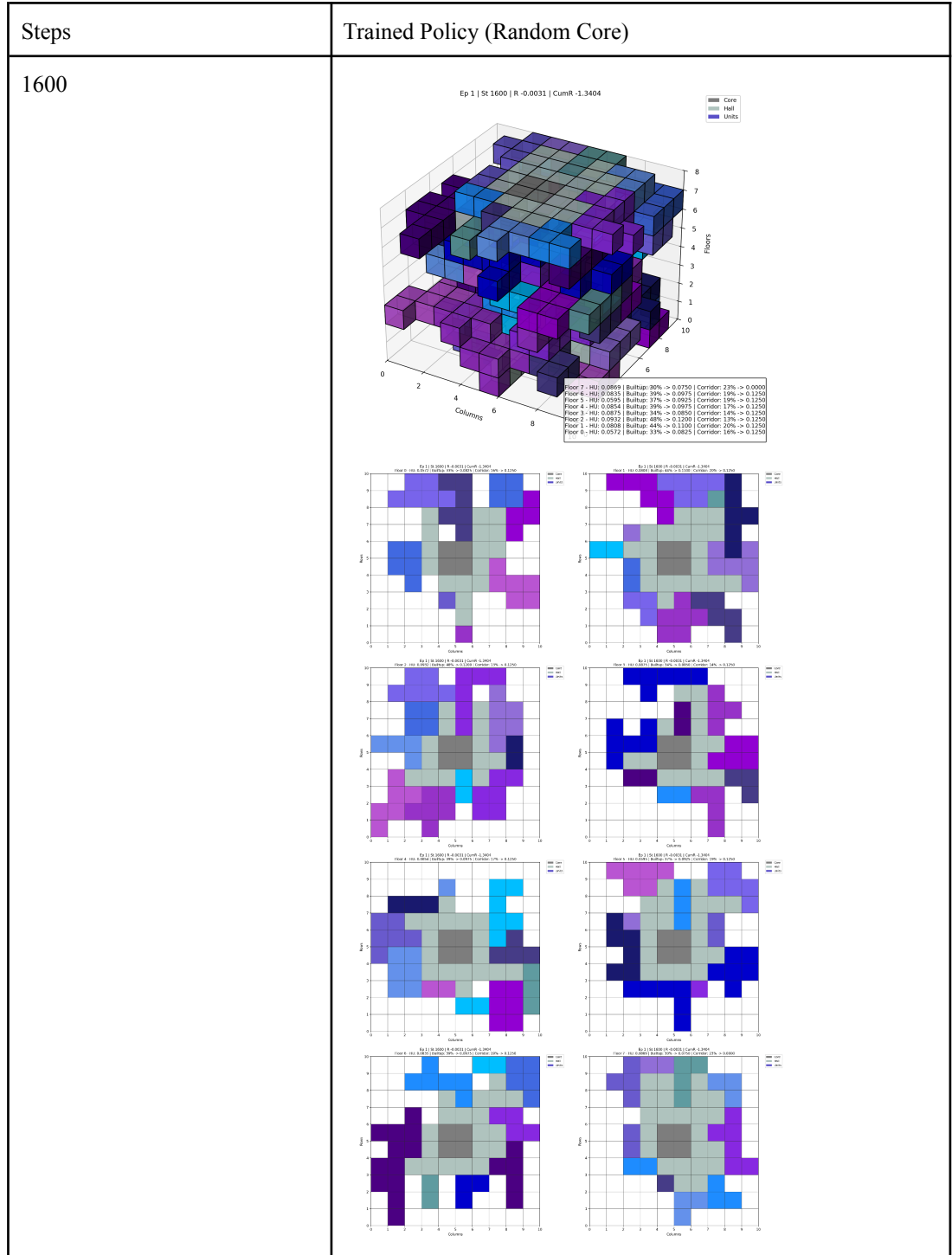


Table A.6. Design output generated by a trained agent using randomly placed core locations at 1600 steps. The top row shows the 3D voxel configuration, while the subsequent rows present corresponding 2D floor slices. Spatial patterns reflect the early learning behavior under variable core constraints. Produced by author.

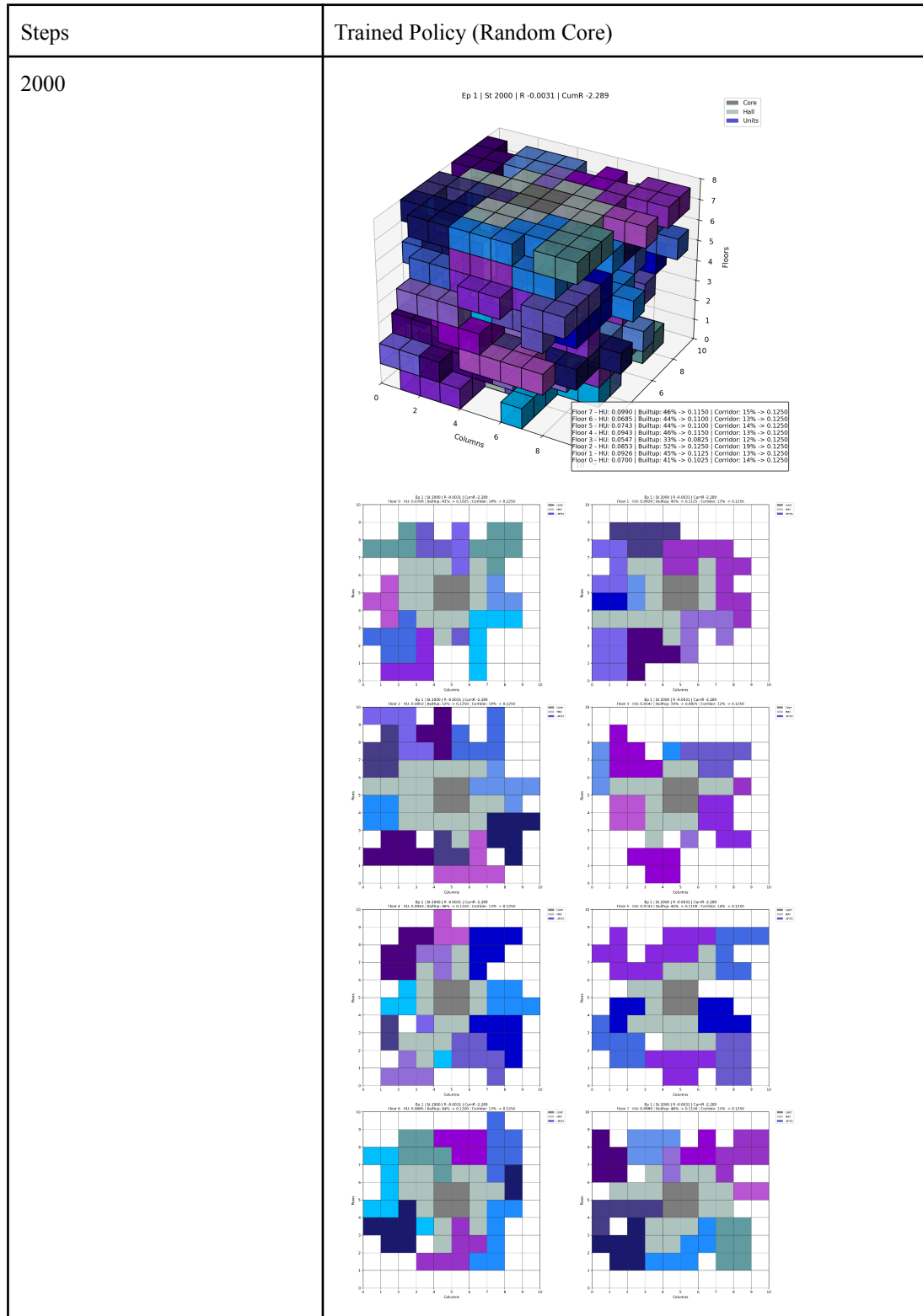


Table A.7. Design output at 2000 steps with randomly placed core locations. Format and interpretation follow Table A.6. Produced by author.

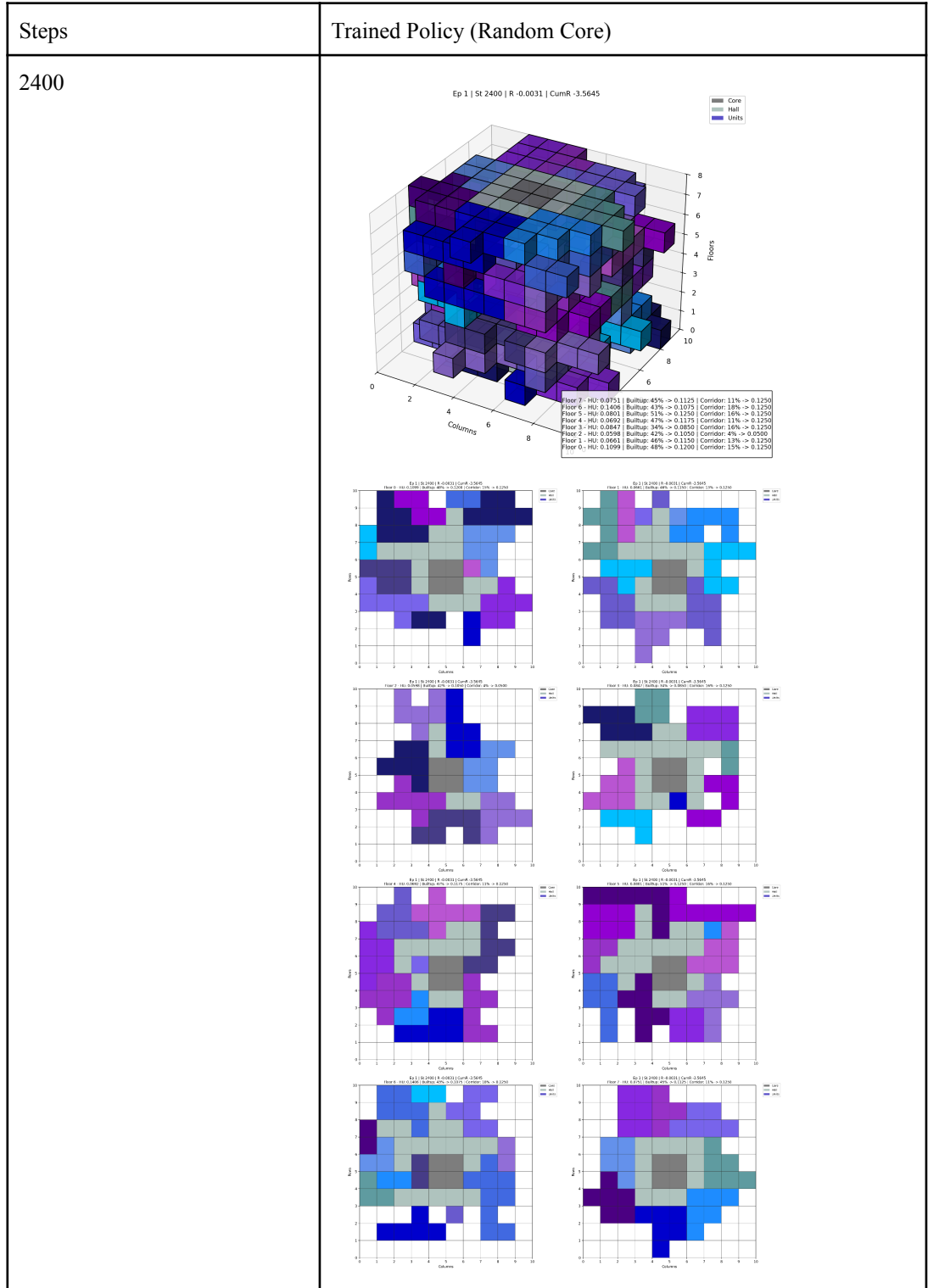


Table A.8. Design output at 2400 steps with randomly placed core locations. Format and interpretation follow Table A.6. Produced by author.

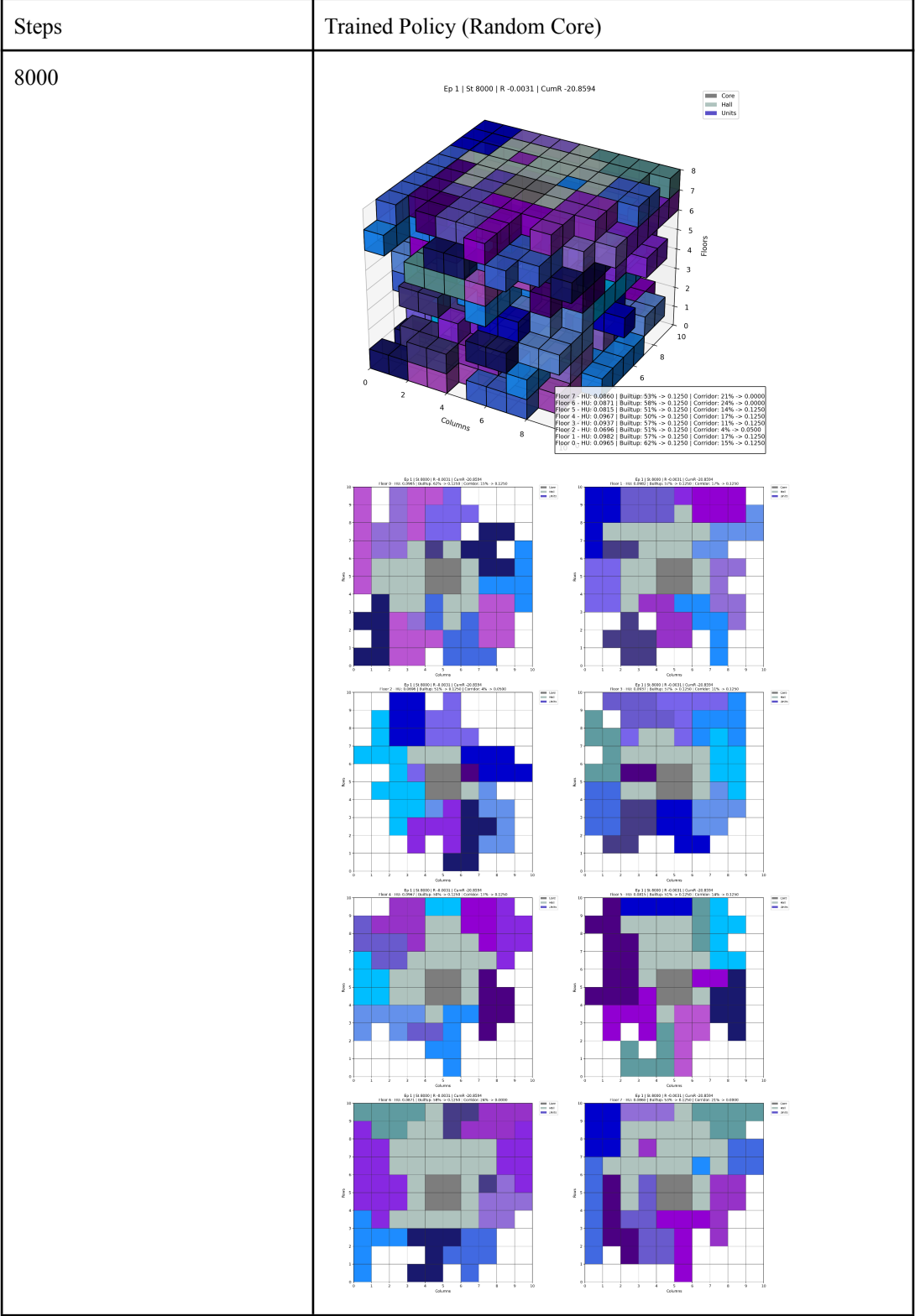


Table A.9. Design output at 8000 steps with randomly placed core locations. Format and interpretation follow Table A.6. Produced by author.

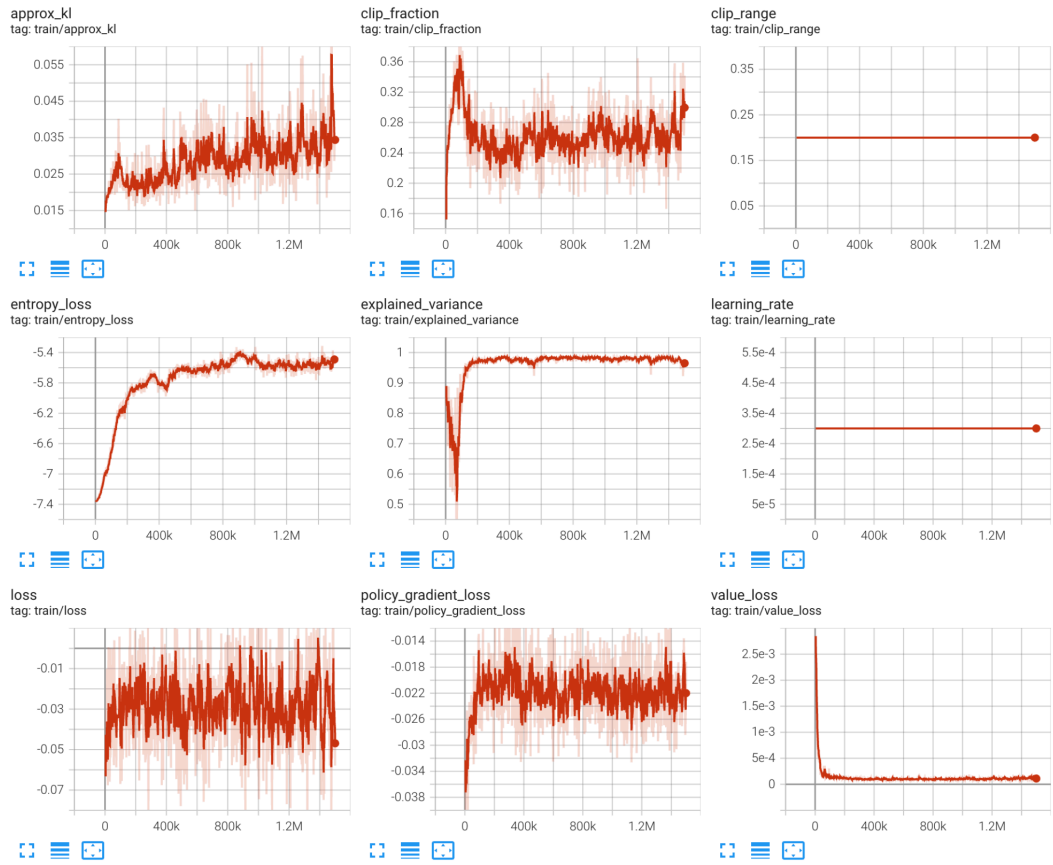


Figure A.3. Additional training metrics for the PPO agent, extending the analysis presented in Figure 4.13 and Section 4.3.2.2. The graphs show the progression of key indicators such as KL divergence, entropy loss, explained variance, value loss, and policy gradients throughout training. These metrics offer further insight into the convergence, stability, and learning dynamics of the model. Produced by author.

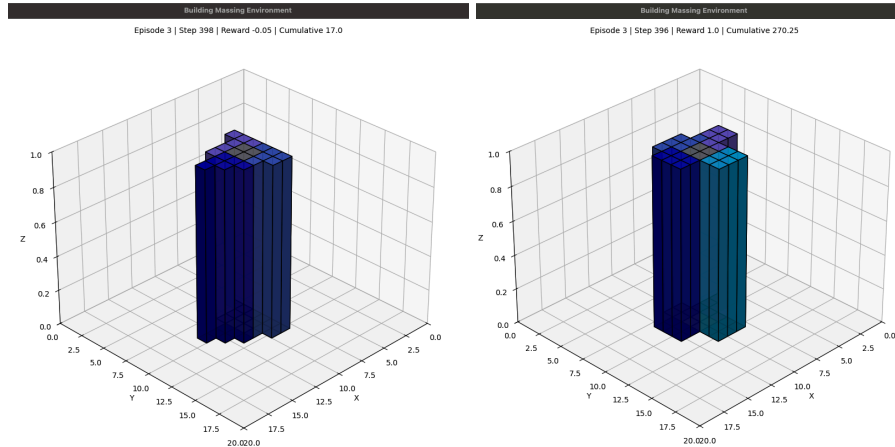
APPENDIX D

REWARD SHAPING EXAMPLE

This training setup was developed and tested separately from Experimental Settings I and II. It was initially used for exploring reward shaping strategies. However, it was later excluded from the main experiments as the reward formulation did not align well with the full delta approach adopted in the potential-based reward structure. Nevertheless, this appendix is included as it demonstrates the sensitivity of the training process to reward shaping design and highlights the need for extensive experimental exploration.

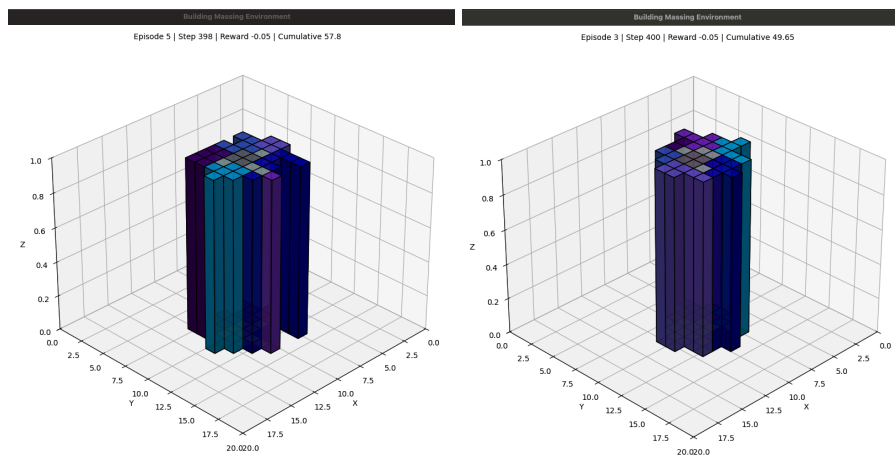
Number of floors:1 | Number of rows: 20 | Number of columns: 20 | Max steps in an episode: 400

PPO_211	
Current action rewards:	Performance-related rewards:
• Placement of built-up $+=1$ • Placement of hall $+=2$ • Invalid action $= 0.05$ • Terminal condition reward $= -100$ (alone built_up, no expand potential)	• Housing units (made of 2-6 cells) rewards: $+= \text{Number of cells for housing units} * 1$ • Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): $+= \text{Number of cells for housing units} * 1$



Since the results show that there is an expansion problem due to a lack of hall cell placement, a terminal condition is added to the problem. If there is no available place left for placement for built-up or hall cells (no available action left), the episode will be terminated as well.

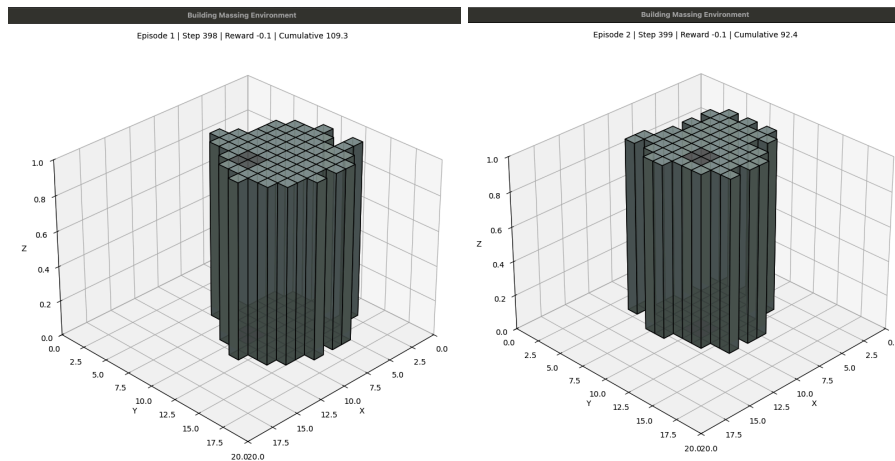
PPO_212	
Current action rewards:	Performance-related rewards:
- Placement of built-up +=1 - Placement of hall +=2 - Invalid action -= 0.05 - Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	- Housing units (made of 2-6 cells) rewards: += <i>Number of cells for housing units</i> * 1 - Extra rewards for perfect shape housing units (2 cells - 2*1, square shape 4 cells - 2*2, rectangular shape 6 cells - 2*3): += <i>Number of cells for housing units</i> * 1



This improvement promoted the agent for the placement of a few hall cells to get rid

of the terminal condition. However, the expansion is not enough to fill the space. So, punishment is increased for invalid actions for the next training.

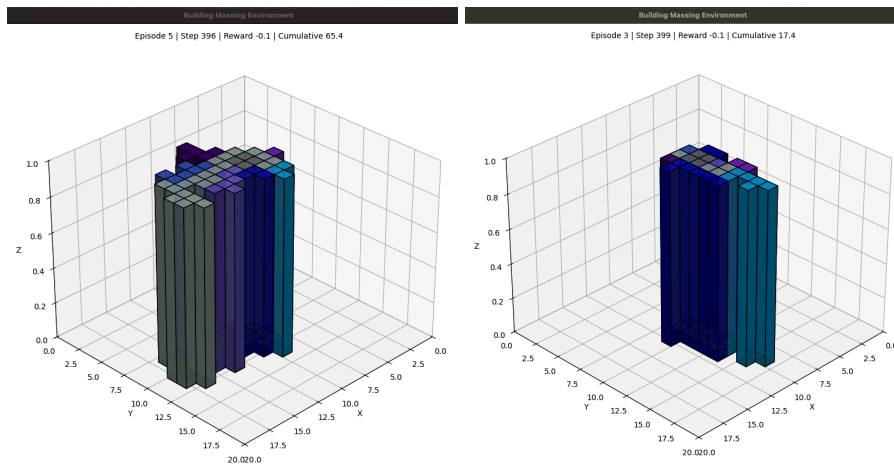
PPO_213	
Current action rewards:	Performance-related rewards:
• Placement of built-up $\text{+=}1$ • Placement of hall $\text{+=}2$ • Invalid action $\text{=-}0.1$ • Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	• Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing units} * 1$ • Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): $\text{+= Number of cells for housing units} * 1$



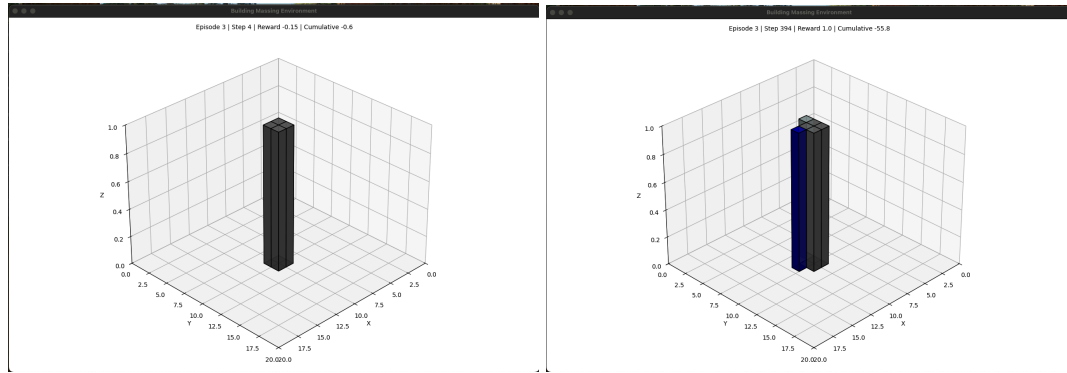
The configuration indicates that increasing the punishment for invalid actions encourages the agent to place hall cells. This may be due to the fact that neighboring cells of hall cells are available for both hall and built-up cell placement. This situation could provide an advantage for expanding the space, leading to fewer invalid actions. Additionally, the reward for placing hall cells is higher than for placing built-up cells. If at least two built-up cells are adjacent, the reward is equal to or greater than the reward for placing hall cells. This might result in a lack of motivation to create housing units. By increasing the additional reward for placing housing units, the agent is tried to be encouraged to focus on placing built-up cells

over the longer term.

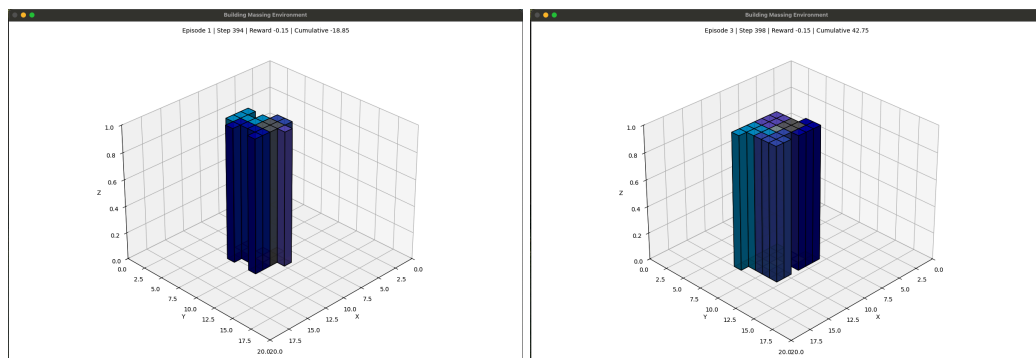
PPO_214	
Current action rewards:	Performance-related rewards:
• Placement of built-up $\text{+=}1$ • Placement of hall $\text{+=}2$ • Invalid action $\text{=-}0.1$ • Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	• Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing units} * 1.5$ • Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): $\text{+= Number of cells for housing units} * 1$



PPO_217	
Current action rewards:	Performance-related rewards:
• Placement of built-up $\text{+=}1$ • Placement of hall $\text{+=}2$ • Invalid action $\text{=-}0.15$ • Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	• Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing units} * 1.5$ • Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): $\text{+= Number of cells for housing units} * 1$

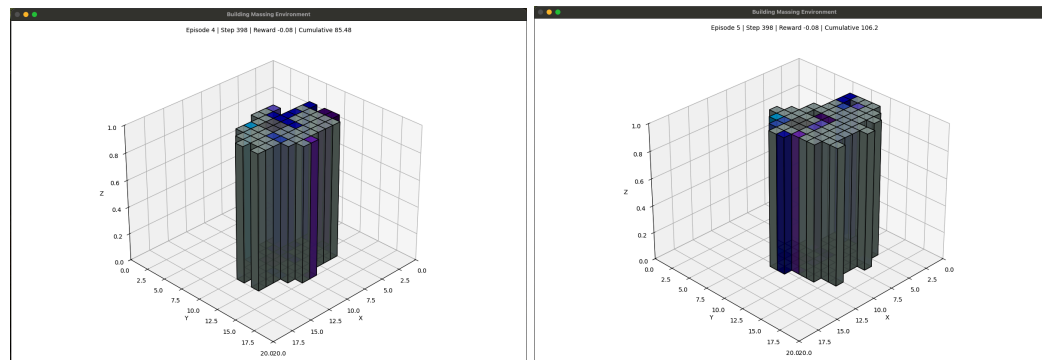


PPO_218	
Current action rewards:	Performance-related rewards:
- Placement of built-up $\text{+=}1$ - Placement of hall $\text{+=}2$ - Invalid action $\text{=-}0.15$ - Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	- Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing units} * 1.5$ - Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): $2\text{cells} \text{+= Number of cells for housing units} * 1$ $4 \text{ cells} \text{+= Number of cells for housing units} * 3$ $6 \text{ cells} \text{+= Number of cells for housing units} * 7$

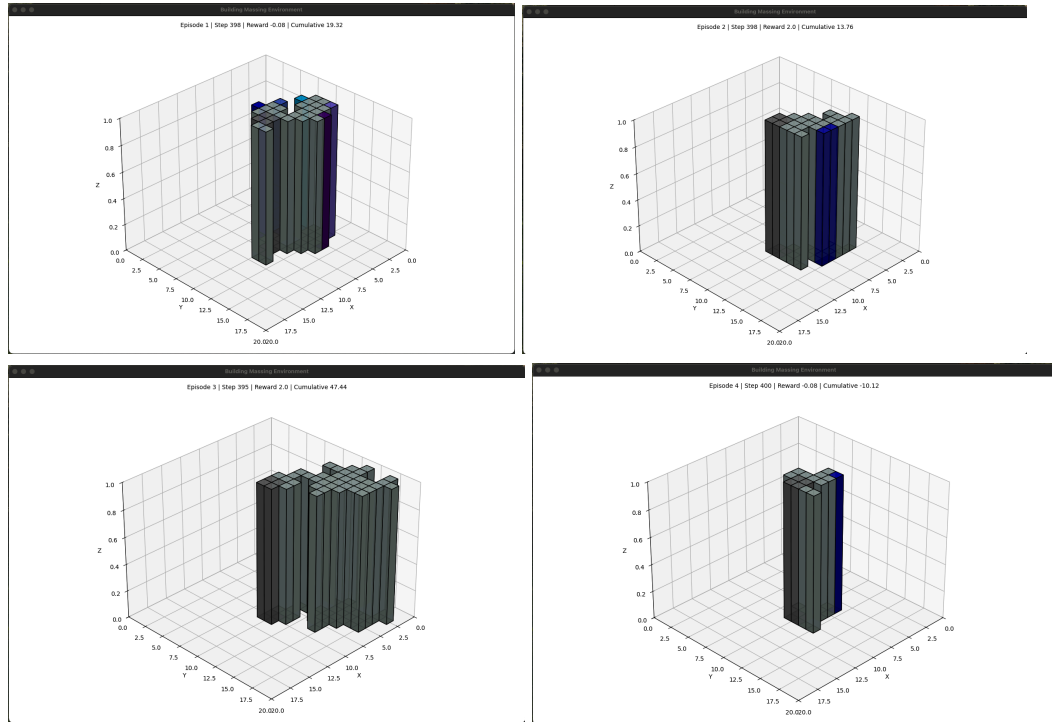


PPO_219	
Current action rewards:	Performance-related rewards:
- Placement of built-up $\text{+=}1$ - Placement of hall $\text{+=}2$ Invalid action $\text{=-}0.08$ - Terminal condition reward =	Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing}$

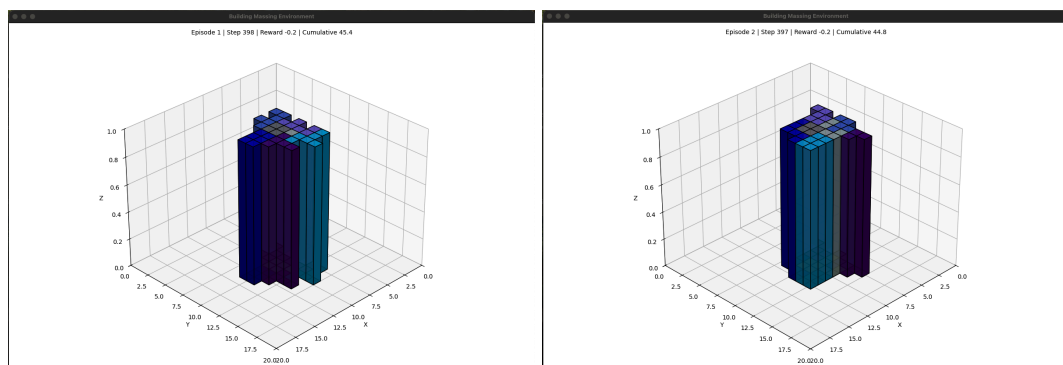
-100 (alone built_up, no expand potential) -100 (no available place left)	<i>units * 1.5</i> Extra rewards for perfect shape housing units (2 cells - 2*1, square shape 4 cells - 2*2, rectangular shape 6 cells - 2*3): 2cells += <i>Number of cells for housing units * 1</i> 4 cells += <i>Number of cells for housing units * 3</i> 6 cells += <i>Number of cells for housing units * 7</i>
---	---



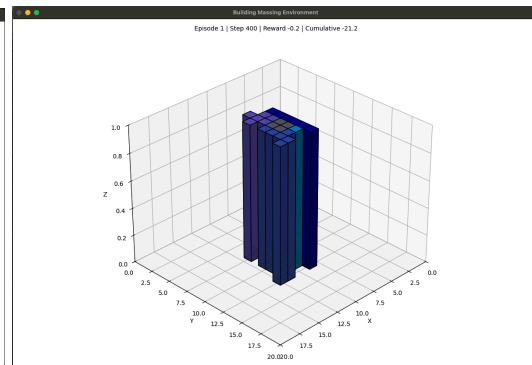
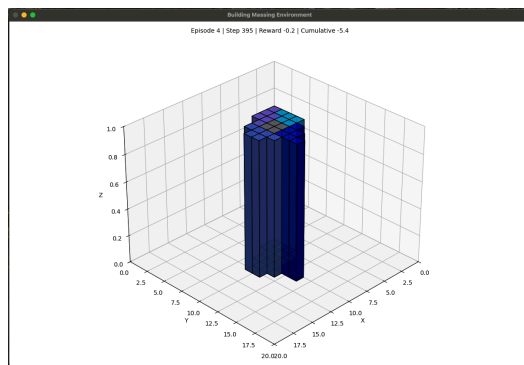
PPO_220	
Current action rewards:	Performance-related rewards:
- Placement of built-up +=1 - Placement of hall +=2 - Invalid action -= 0.08 - Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	- Housing units (made of 2-6 cells) rewards: += <i>Number of cells for housing units * 1.8</i> - Extra rewards for perfect shape housing units (2 cells - 2*1, square shape 4 cells - 2*2, rectangular shape 6 cells - 2*3): 2cells += <i>Number of cells for housing units * 1.2</i> 4 cells += <i>Number of cells for housing units * 3</i> 6 cells += <i>Number of cells for housing units * 5</i>



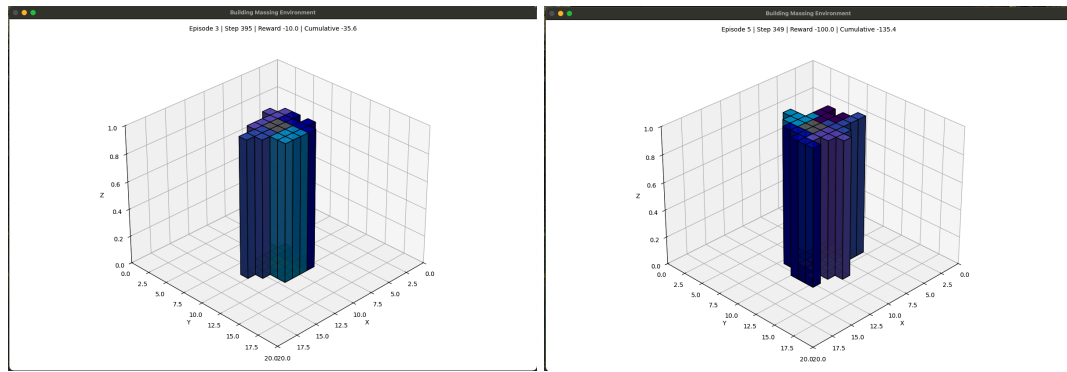
PPO_221	
Current action rewards:	Performance-related rewards:
- Placement of built-up ± 1 - Placement of hall ± 2 - Invalid action -0.2 - Terminal condition reward = -40 (alone built_up, no expand potential) -40 (no available place left)	- Housing units (made of 2-6 cells) rewards: $\pm \text{Number of cells for housing units} * 2$ - Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): $2\text{cells} \pm \text{Number of cells for housing units} * 1$ $4\text{ cells} \pm \text{Number of cells for housing units} * 3$ $6\text{ cells} \pm \text{Number of cells for housing units} * 5$



PPO_222	
Current action rewards:	Performance-related rewards:
• Placement of built-up $\text{+=}1$ • Placement of hall $\text{+=}2$ • Invalid action $\text{=-}0.2$ • Terminal condition reward = -40 (alone built_up, no expand potential) -40 (no available place left)	• Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing units} * 2$ • Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): 2cells $\text{+= Number of cells for housing units} * 1$ 4 cells $\text{+= Number of cells for housing units} * 3$ 6 cells $\text{+= Number of cells for housing units} * 5$



PPO_223	
Current action rewards:	Performance-related rewards:
• Placement of built-up $\text{+=}1$ • Placement of hall $\text{+=}2$ • Invalid action $\text{=-}0.3$ • Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	• Housing units (made of 2-6 cells) rewards: $\text{+= Number of cells for housing units} * 1$ • Extra rewards for perfect shape housing units (2 cells - $2*1$, square shape 4 cells - $2*2$, rectangular shape 6 cells - $2*3$): 2cells $\text{+= Number of cells for housing units} * 1$ 4 cells $\text{+= Number of cells for housing units} * 3$ 6 cells $\text{+= Number of cells for housing units} * 5$



PPO_224	
Current action rewards:	Performance-related rewards:
• Placement of built-up ± 1 • Placement of hall ± 2 • Invalid action -0.3 • Terminal condition reward = -100 (alone built_up, no expand potential) -100 (no available place left)	• Housing units (made of 2-6 cells) rewards: $\pm$ Number of cells for housing units $\times 1$ • Extra rewards for perfect shape housing units (2 cells - 2×1, square shape 4 cells - 2×2, rectangular shape 6 cells - 2×3): $2\text{cells} \pm$ Number of cells for housing units $\times 0.2$ $4\text{ cells} \pm$ Number of cells for housing units $\times 0.5$ $6\text{ cells} \pm$ Number of cells for housing units $\times 1$

