Dissertation on

"Leveraging AI for Urban Spatial Planning"

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE21CS461A – Capstone Project Phase - 2

Submitted by:

Lakshya Singh	**PES2UG21CS252**
Luv Arora	**PES2UG21CS257**
Manasvi Varma	**PES2UG21CS305**
Manish I	**PES2UG21CS278**

Under the guidance of

Dr. Pooja Agarwal
Professor
PES University

June - Nov 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Leveraging AI for Urban Spatial Planning'

is a bonafide work carried out by

Lakshya Singh	**PES2UG21CS252**
Luv Arora	**PES2UG21CS257**
Manasvi Varma	**PES2UG21CS305**
Manish I	**PES2UG21CS278**

In partial fulfillment for the completion of seventh semester Capstone Project Phase - 2 (UE20CS461A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June 2024 – Nov. 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature	Signature	Signature
Dr. Pooja Agarwal	Dr. Sandesh B J	Dr. B K Keshavan
Professor	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ____________________________	____________________________
2. ____________________________	____________________________

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **“Leveraging AI for Urban Spatial Planning”** has been carried out by us under the guidance Dr. Pooja Agarwal , Professor , submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June – Nov. 2024. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2UG21CS252	**Lakshya Singh**	Lakshya Singh
PES2UG21CS257	**Luv Arora**	Luv
PES2UG21CS305	**Manasvi Varma**	Manasvi
PES2UG21CS278	**Manish I**	[illegible]

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Dr. Pooja Agarwal , Professor , Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE21CS461A Capstone Project Phase-2.
I am grateful to the Capstone Project Coordinators, Dr. Farida Begam, Professor and Prof. Vandana M. Ladwani, Associate Professor, Department of Computer Science and Engineering, PES University for organizing, managing, and helping with the entire process.
I take this opportunity to thank Dr. Sandesh B J, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor, PES University, Dr. Suryaprasad J, Vice Chancellor, PES University and Prof. Nagarjuna Sadineni, Pro Vice Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

This paper presents an innovative AI framework tailored for urban planning and development to address the challenges faced by urban planners in rapidly evolving city environments while adhering to stringent environmental guidelines. The framework aims to bridge critical gaps in current urban planning practices by focusing on overlooked support systems such as water and waste management, incorporating detailed plans for environmental sustainability and energy strategies, addressing socio-economic factors impacting community well-being, and promoting pedestrian-friendly transportation options over outdated car-centric approaches.

Key features of the framework include

- the generation of comprehensive city plans integrating multiple factors
- smart analysis facilitated by AI algorithms to compare and evaluate city plans
- clear visualization through layered visualizations for better understanding of city systems
- intuitive interface enabling designers to personalize city designs effortlessly

By leveraging AI technologies and addressing the identified gaps in urban planning, this framework offers a promising approach to enhance the efficiency, sustainability, and user-friendliness of urban development processes, ultimately contributing to the creation of more resilient and livable cities.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Incorporating technological advancements, Urban City Planning covers the strategic arrangement and management of urban spaces, including housing, commercial areas, and markets.

Urban city planning encompasses strategically organizing and managing urban spaces including housing, commercial areas, and markets, along with auxiliary systems such as electricity, road connectivity, water, and waste management. The main goal is to improve the quality of human life through sustainable, comprehensive, and integrated urban development strategies that aim to address environmental considerations.

Primary objective is the improvement of the quality of living for residents, achieving the set goals in sustainability, and considering environmental concerns by integrating comprehensive strategies in urban development.

Modern urban planning must improve the reintegrating of infrastructure such as water and waste management as well as more sustainable environmental and sustainable strategies. Socio-economic considerations, including crucial components such as culture and sports, must be taken seriously in ensuring a successful community. Priority should be given to sociable transportation over the old-fashioned vehicle-first approach to develop fair and sustainable urban environments.

CHAPTER 2

PROBLEM DEFINITION

Developing a dynamic and perplexing AI framework which can be efficient and user-friendly for urban planning and development. It should specifically address the challenges faced by urban planners when dealing with rapidly changing city environments that demand for strict adherence to environmental protection policies.

This project is aimed at creating an AI system for urban planning which is both efficient and easy to use. The goal is to overcome the complexities that come with it since these are the problems faced by city planners who have to adapt themselves quickly enough towards their cities' new look while still keeping up with tough standards regarding environmental conservation. This framework heavily borrows from technology-driven ideas hence; it provides an elaborate strategic organization and control of different parts of towns such as residential areas, business districts among others not forgetting about supporting facilities like power lines, roads networks, water supply systems plus waste disposal units too.

The main aim is to increase the standard of living for city dwellers and achieve sustainability goals, urban strategies must be systematically addressed by dealing with environmental concerns arising. One of the key driving factors behind this move is that there are gaps in modern planning systems which do not take into account important things like water & waste management. Additionally, energy usage as well as strategies for environmental sustainability have been poorly developed while socio-economic aspects affecting community wellbeing have always been looked at superficially. Outdated methods of planning usually give priority to car traffic against pedestrian friendly alternatives hence posing more problems in terms of getting around in cities and how enjoyable they can be.

CHAPTER 3

LITERATURE SURVEY

3.1 Spatial Planning of Urban Communities via Deep Reinforcement Learning

Year: 2023

The paper compares its Deep Reinforcement Learning (DRL)-based model against two rule-based heuristics for urban spatial planning: centralized and decentralized approaches. These conventional methodologies rely on predefined rules and guidelines to determine the layout of urban elements like roads, residential areas, and facilities, which often depend on extensive transportation systems.

The methodology utilizes a unique approach to urban planning by employing DRL. It introduces an urban contiguity graph to represent the spatial contiguity of urban elements, overcoming the challenge of urban blocks' diversity and irregularity. The DRL model comprises a value network and two policy networks, which navigate the vast action space of urban planning by predicting the quality of spatial plans and selecting optimal actions for land use and roads placement.

The DRL model demonstrates superior performance over rule-based heuristics in several key metrics across both synthetic and real-world scenarios. The model's effectiveness is evident in its ability to renovate communities according to the 15-minute city concept, optimizing the placement of various facilities to enhance accessibility and reduce dependency on vehicular transportation

Pros:

- **Flexibility and Efficiency:** The DRL model adapts to different urban planning needs and requirements without re-training, showcasing remarkable flexibility and efficiency in generating spatial plans.
- **Advanced Representation:** By utilizing a graph neural network (GNN)-based state encoder, the model effectively captures the spatial relationship between urban elements, facilitating more informed planning decisions.

Cons:

- **Complex Action Space:** The inherent complexity of urban planning presents challenges in navigating the vast number of possible actions and outcomes.

- **Advanced Representation:** With a GNN-based state encoder, the model effectively captures the spatial relationship between elements in the urban, thus ensuring better decision-making when making plans.

Cons:

- **Complexity of Action Space:** There is complexity of action space inherent in urban planning, which affects navigation due to numerous possible actions as well as their vast outcomes.
- **Dependence on Data Accuracy:** This mostly depends highly on the accuracy and completeness of input data, especially the conditions and configurations of the urban environment at the beginning.

Fixed planning order: The fixed planning order for land-use planning may reduce the complexity of the action space, but it may not provide any flexibility to increase the model sensitivity in the possibility of identifying alternative planning sequences with the potential to have a better optimum

3.2 Artificial Intelligence Models for Urban Planning

Year: 2023

This study focuses on employing Generative Adversarial Networks (GANs) alongside remote sensing data analysis to generate synthetic urban imagery. By integrating GANs with remote sensing techniques, researchers have achieved promising results in creating images that closely resemble real-world urban scenes. These synthetic images are beneficial for tasks like urban planning, infrastructure assessment, and environmental monitoring. The combination of GANs and remote sensing data analysis enhances image resolution and quality, enabling more accurate analysis of urban environments. Consequently, this approach improves various aspects of urban planning, including land use classification, urban growth prediction, infrastructure planning, and disaster management, showcasing its potential for revolutionizing urban analytics and planning methodologies.

The integration of Generative Adversarial Networks (GANs) with remote sensing data analysis has yielded significant benefits in urban planning and related fields. Notably, GAN-generated images exhibit improved image quality and resolution compared to conventional remote sensing methods, enhancing the accuracy of urban analysis tasks.

Pros:

- GANs offer a novel approach to augmenting limited datasets, enhancing spatial resolution, and generating data in challenging scenarios
- Integration of GANs with remote sensing data analysis improves the accuracy and richness of information available for urban planning and related tasks.
- GAN-generated images can support informed decision-making and planning strategies for sustainable urban development

Cons:

- Challenges remain in ensuring the spatial and contextual realism of GAN-generated urban imagery, particularly in aligning with real-world constraints and regulations.

3.3 Building-GAN: Graph-Conditioned Architectural Volumetric Design Generation

Year: 2021

Introduces a groundbreaking approach to 3D architectural design generation. It employs a voxel graph representation method to encode irregular voxel grids, effectively bridging input program graphs with output voxel graphs. This innovative technique is integrated into a Graph-Conditioned Generative Adversarial Network (GAN), which incorporates a Graph Neural Network (GNN) and a pointer-based cross-modal module. The result is an automated pipeline capable of generating valid volumetric designs with minimal user interaction, enhancing the creation of realistic architectural structures. Collaborating with professional architects, the authors curated a synthetic dataset of 120,000 designs, ensuring the model's training and evaluation on realistic building requirements. Through user studies and quantitative evaluations, including metrics like quality and diversity, the model's performance is thoroughly assessed. Additionally, professional architect involvement ensures the practicality and realism of the generated designs. The outcome is a sophisticated generative model that not only produces realistic 3D designs with clean facade surfaces but also empowers architects and designers to explore and visualize architectural designs more efficiently and creatively.

Pros:

- Outperformance of House-GAN: The model outperforms the House-GAN method in terms of both Fréchet Inception Distance (FID) and connectivity accuracy. The designs generated by the proposed model are preferred over those generated by House-GAN, indicating superior quality and realism.
- Identification of Common Flaws: The paper identifies common flaws in the generated volumetric designs, such as missing nodes, missing edges, and disconnected rooms. These flaws are attributed to limitations in the discriminator's consideration of program nodes' attention, indicating areas for future improvement and research.

Cons:

- Limited Generalization to Non-Cuboid Geometries: The paper concentrated on producing volumetric designs within valid design spaces mostly represented as irregular voxel grids. In this respect, it did not consider the effectiveness of the model in having or dealing with non-cuboid geometries or other complex architectural structures beyond simple rectangular buildings. Such a limitation could, therefore, be a hindrance to the further application of the model towards a wider array of architectural design considerations.
- Complexity and Computational Resources: Pipelines using graphs, GANs, and Pointer networks require both human and computational resources and are overly complex, reducing their wide adoption.

3.4 A Planning Support Tool for Street Network Design

Year: 2022

Incorporating planning intelligence into deep learning for street network design proposes a generation of context-aware, example-based, and user-guided street networks. This approach addresses a key limitation of deep learning: the challenge of melding complex urban knowledge with AI algorithms, which often operate as a "black box."

- **Combination of Gansand Planning Knowledge:** The tool utilizes Conditional Generative Adversarial Networks (cGANs) trained on fundamental context (surrounding street networks, topography, slope aspect) and planning guidance (major transport junctions, street network pattern types) to generate street networks that align with specific planning needs and contexts.
- **Context-Aware Generation:** Inspired by image completion tasks, the model recognizes various street patterns within a given context and adapts generated street networks to fit seamlessly with existing infrastructure.

- **Example-Based and User-Guided Design:** The system learns from real-world examples and allows users to input specific planning guidance, such as the location of key transport junctions and desired street patterns, facilitating the generation of customized street network proposals.

Pros:

- **Improved Realism and Diversity:** The incorporation of planning intelligence enables the generation of street networks that are realistic compared to traditional deep learning models and also offer a wider array of design alternatives, reflecting the diverse needs and contexts of urban planning.
- **Enhanced Planning Support System (PSS):** The tool represents a significant advancement in PSS, allowing for the exploration of multiple planning scenarios and the generation of comparative proposals, thereby aiding decision-making in urban design and planning.
- **Interactive Design Tool:** The tool facilitates a collaborative human-computer design process, making it accessible for both professionals and lay users to explore and evaluate different planning options.

Cons:

- **Expansion of Training Data Sets:** Incorporating data from a broader range of cities could enhance the model's ability to generate diverse and globally relevant street network patterns.
- **Integration of Additional Planning Elements:** Extending the model to include other urban elements like building layouts, land use, and urban amenities could lead to a more comprehensive planning support tool, offering insights into the multifaceted aspects of urban morphology and its impact on social, environmental, and aesthetic outcomes.

3.5 Automated Urban Planning for Reimagining City Configuration via Adversarial Learning

Year: 2023

The paper presents an innovative framework called LUCGAN (Land-Use Configuration Generative Adversarial Network) for automating urban planning processes. The framework leverages multiple data sources such as housing prices, point of interests data, and taxi GPS traces to learn representations of surrounding contexts. It consists of two main phases: surrounding contexts embedding and land-use configuration generation. In the embedding phase, explicit features of surrounding contexts are extracted and mapped to a spatial attributed graph for graph embedding. The land-use configuration generation phase involves using a generative adversarial network (GAN) to generate well-planned land-use configurations based on the learned embeddings. The framework is compared against baseline methods like DCGAN and WGAN for performance evaluation. The paper aims to streamline urban planning by automating the generation of recommended land-use configurations.

The paper on Automated Urban Planning for Reimagining City Configuration using the LUCGAN framework offers several advantages:

Pros:

- **Automated Urban Planning**: LUCGAN automates the process of generating land-use configurations based on surrounding contexts, reducing the burden on urban planning specialists.
- **Flexibility and Generalization**: The framework utilizes publicly available geographical spatial data such as POI, traffic, economic, and demographic data, making it flexible and easily generalizable
- **Customized Configurations**: LUCGAN can produce customized land-use configurations tailored to specific requirements, enhancing its adaptability

- **Effective and Robust**: Through extensive experiments, LUCGAN has been shown to be more effective and robust compared to other baseline models, indicating its reliability and performance
- **No Strict Data Collection Requirement**: Unlike some existing methods that require expert layout data for training AI models, LUCGAN does not have strict conditions for data collection, making it more accessible and user-friendly

Cons:

- **Data Availability:** While LUCGAN utilizes publicly available data sources, the quality and availability of such data may vary across different regions or cities, potentially impacting the accuracy and generalizability of the generated land-use configurations
- **Model Complexity:** The use of deep generative adversarial networks (GANs) and graph embeddings in the LUCGAN framework may introduce complexity in the implementation and interpretation of the model, requiring additional computational resources and expertise for optimization and fine-tuning
- **Generalization:** While LUCGAN aims to generate customized configurations, ensuring the generalizability of the model across diverse urban contexts and planning scenarios may require further validation and refinement.

3.6 Benchmarking City Layouts

Year: 2022

The paper discusses a methodological approach for benchmarking city layouts and compares the accessibility of a real city (Coimbra, Portugal) with that of a Garden City model. The authors emphasize the importance of urban spatial layout and the need for quantitative analysis to understand the advantages of different urban design ideas. They use geographic information systems (GIS) to evaluate accessibility as a benchmarking indicator and apply this methodology to the case study of Coimbra. The results show that the Garden City layout provides better accessibility scores for urban facilities and jobs compared to the real city layout. The study also discusses the relevance of old city models like the Garden City in the context of modern urban planning and suggests future research directions for developing additional benchmarking indicators.

Pros:

- **Quantitative Analysis:** The approach provides a quantitative method for comparing the accessibility of different city layouts, allowing for objective evaluation and benchmarking.
- **GIS Capabilities:** By leveraging geographic information systems (GIS), the methodology enables the evaluation of urban layouts based on geographic characteristics, providing meaningful evidence for urban planning and development.
- **Relevance to Modern Urban Planning:** The research highlights the relevance of old city models, such as the Garden City, in the context of modern urban planning

Cons:

- **Practical Implementation**: The practical application of the methodology to existing cities may be limited by the static nature of real city layouts, which are unlikely to change based solely on benchmarking results.
- **Assumptions and Simplifications:** The methodology makes assumptions about the indicators used for benchmarking, such as accessibility, and may not fully capture the complexity of real-world urban environments.

3.7 Exploration of Campus Layout Based on Generative Adversarial Network

Year: 2020

The paper explores the automatic generation of campus layouts using a Generative Adversarial Network (GAN) approach, specifically the Pix2Pix model. The generator uses the U-net framework and the discriminator uses the Patch-GAN framework. The model is trained on a small dataset of campus layouts that adhere to specific guidelines, and it generates layouts based on input images of campus boundaries and surrounding roads.

By inputting pictures of the actual campus boundary and surrounding roads and outputting corresponding pictures of the actual campus layout , this model is trained to learn the rules of actual campus layouts and generate campus layout automatically.

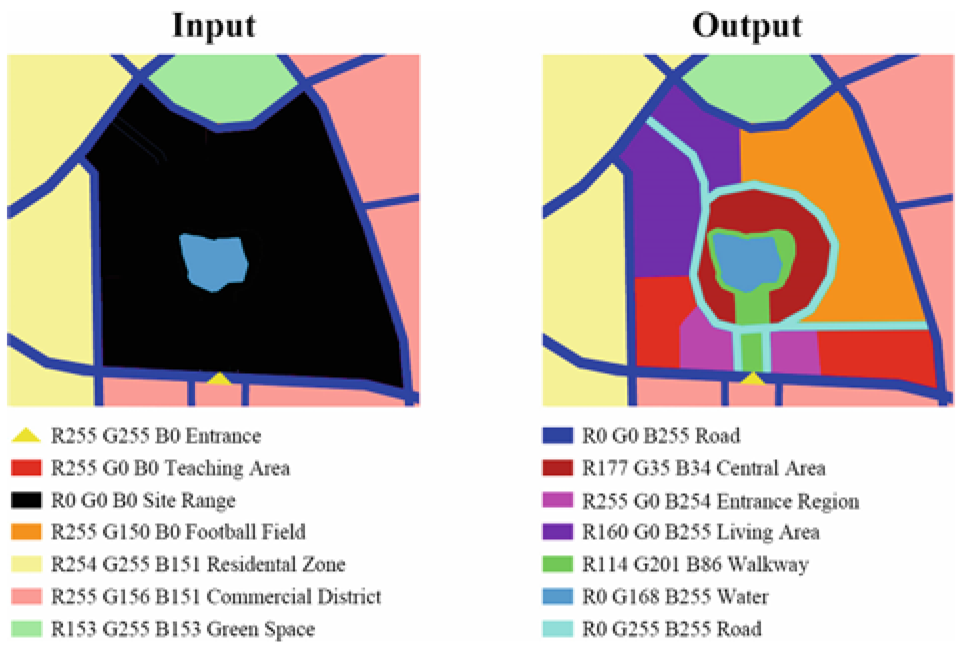

Input
Output
R255 G255 B0 Entrance
R255 G0 B0 Teaching Area
R0 G0 B0 Site Range
R255 G150 B0 Football Field
R254 G255 B151 Residental Zone
R255 G156 B151 Commercial District
R153 G255 B153 Green Space
R0 G0 B255 Road
R177 G35 B34 Central Area
R255 G0 B254 Entrance Region
R160 G0 B255 Living Area
R114 G201 B86 Walkway
R0 G168 B255 Water
R0 G255 B255 Road

Fig 3.1 Training data for step 1, top: input and output, bottom: labeling rule

The results show that the model successfully captures certain layout rules, such as building orientation and spacing, but may produce fuzzy areas in more complex central areas.

Pros:

- **Small Dataset Training:** The approach demonstrates the ability to train models effectively on relatively small datasets, which can be advantageous when limited data is available.
- **Two-Step Training:** The two-step training process allows for the generation of comprehensive campus layouts, addressing both functional zoning and internal architectural layout. The first step is to train a model that generates the main functional zoning of the campus based on the campus boundary ,surrounding roads and surrounding functional distribution. The second step is to train another model which is based on each functional zoning of the campus and generates the internal architectural layout of the Zone .In this way, a complete campus layout of the university is achieved

Cons:

- **2D Space Limitation:** The generated layouts are restricted to 2D space and do not consider topographical data, limiting the representation of complex terrain.
- **Fuzzy Nature of Output:** In complex central areas, the generated layouts may exhibit fuzzy areas due to weakly defined design relationships and characteristics, potentially impacting their practical usability.

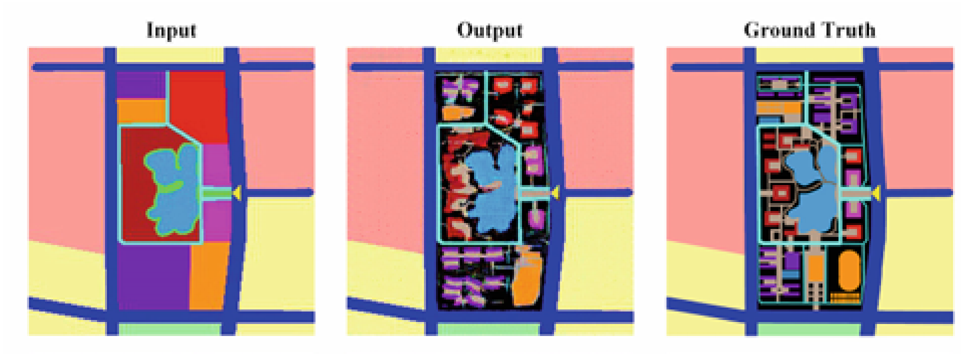


Fig 3.2 Selected results from the testing set

3.8 Road Planning for Slums via Deep Reinforcement Learning

Year: 2023

The paper introduces a deep reinforcement learning (DRL) approach for road planning in slums, leveraging a graph neural network (GNN) to model the topological structure of the slum layout. The proposed model outperforms random and greedy algorithms, as well as generative models, and achieves significant improvements in terms of accessibility, travel distance, and construction costs. The methodology includes the use of masked policy optimization and a two-stage process, where the first stage focuses on achieving universal connectivity, and the second stage adds more roads within budget constraints to decrease travel time.

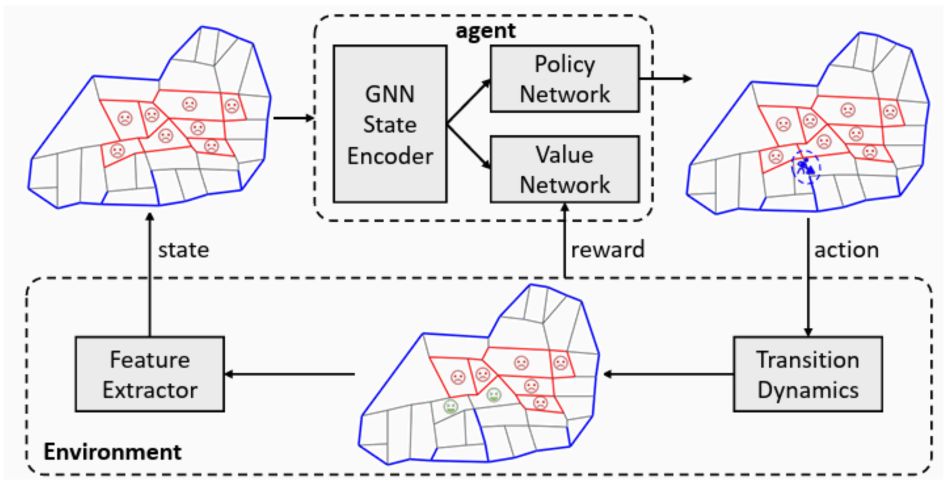


Fig 3.3 Schematic of Paper's Approach

Pros:

- **GNN Model:** The use of a GNN allows for the modeling of irregular polygon sites and various entities in slum layouts, providing a generic graph model capable of handling diverse geometric forms at different scales.
- **Multi-Objective Policy Optimization:** The methodology addresses multiple objectives simultaneously, ensuring universal connectivity while optimizing for travel time and construction costs.

Two-Stage Process: The two-stage process allows for a more comprehensive optimization strategy, first achieving universal connectivity and then refining the road network to decrease travel time within budget constraints.

Cons:

- **Insufficient Slum Data:** One limitation is the potential lack of sufficient real-world slum data to train the model effectively, which may impact the generalizability of the proposed approach to diverse slum scenarios.

Complex Topological Feature Conversion: Converting topological features such as roads, home sites, and other elements into graph entities like nodes, edges, and faces might pose challenges, indicating potential difficulties in accurately representing real-world features in the mod

CHAPTER- 4

Project Requirements and Specifications

4.1 High-Level Document

4.1.1 Design Considerations

1.1. Design Goals

Provide a comprehensive framework of AI tools to assist designers in generating city plans considering the following factors:

- Geology and current land usage
- Road networks and traffic management
- Allocation of urban spaces: housing, commercial, and market areas
- Population demographics and socio-economic data
- Water resources and drainage system
- Waste management strategies

The proposed framework aims to provide AI-powered tools to assist city planners in developing comprehensive and sustainable urban designs. It considers various factors, such as geology, land usage, transportation, resource management, and demographic data, to address common urban challenges from the outset.

The framework also includes advanced analytical and evaluation capabilities, allowing users to refine and compare city plans based on specific objectives. Additionally, it offers clear visualizations to enhance understanding of the complex urban systems, and an intuitive interface for designers to easily customize the city design according to their preferences.

This user-friendly system simplifies the design work, making it straightforward to adjust and tailor the design to meet the unique needs of urban planning projects.

2. High-Level System Design

High-Level Design View of the Urban Planning AI System:

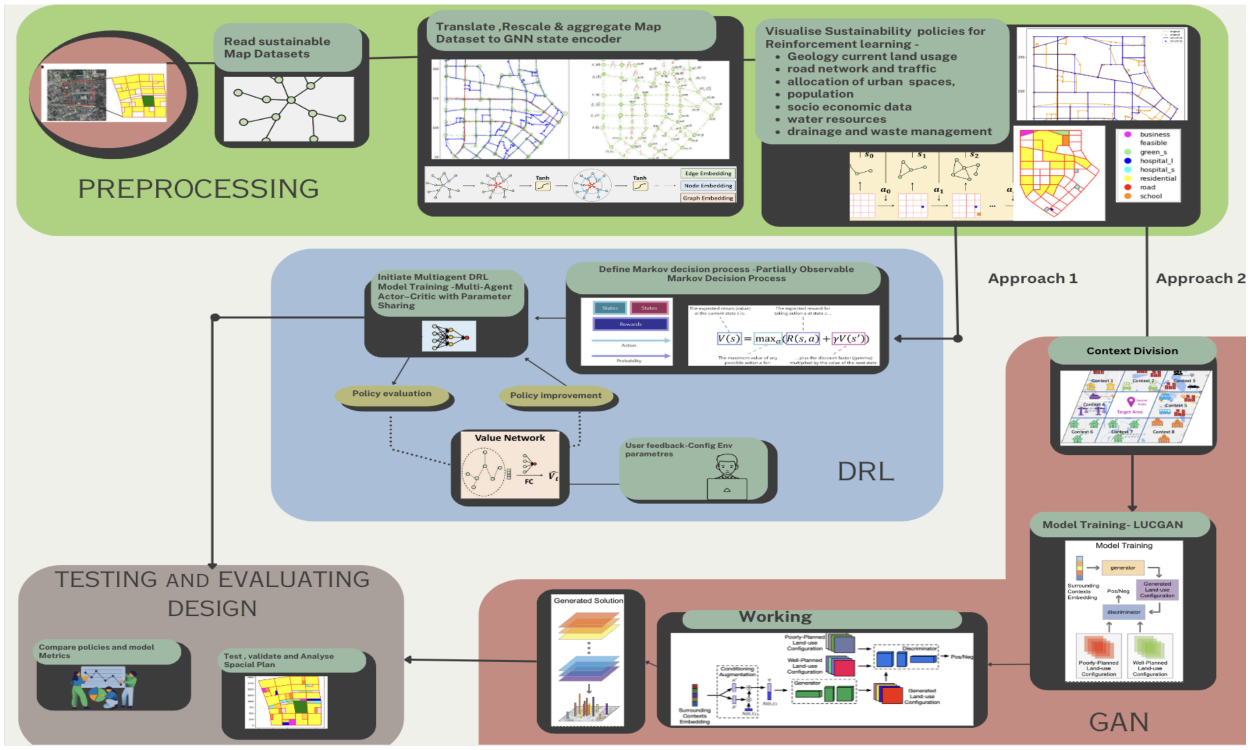

PREPROCESSING
Read sustainable Map Datasets
Translate ,Rescale & aggregate Map Dataset to GNN state encoder
Visualise Sustainability policies for Reinforcement learning -
Geology current land usage
road network and traffic
allocation of urban spaces,
population
socio economic data
water resources
drainage and waste management
Approach 1
Approach 2
Initiate Multiagent DRL Model Training -Multi-Agent Actor–Critic with Parameter Sharing
Define Markov decision process -Partially Observable Markov Decision Process
Policy evaluation
Policy improvement
Value Network
User feedback-Config Env parametres
DRL
Context Division
Model Training- LUCGAN
GAN
TESTING AND EVALUATING DESIGN
Compare policies and model Metrics
Test , validate and Analyse Spacial Plan
Generated Solution
Working

Fig 4.1: High-Level Design Diagram

Logical User Groups and Application Components

- Urban Initiatives: Stakeholders defining urban development goals and initiatives.
- Deep Reinforcement Learning (DRL) Model: Core AI component for generating and optimizing the urban plans.
- Data Processing Unit: Handles data aggregation, analysis, and feeds into the DRL model.

Data Components and Storage

- Geospatial Data Storage: Repository for geospatial data including land use, infrastructure, and environmental conditions.
- Traffic Management System: Contains data and algorithms for analyzing and proposing traffic flow optimizations.

Interfacing Systems

- GIS Tools: External geospatial information systems providing detailed mapping and land use data.

- Public Data Sources: Various open data sources offering demographic, economic, and environmental datasets.
- Urban Planning Tools: Professional software used for detailed urban design and planning tasks.

Collaboration and Interaction

- The Data Processing Unit serves as the central hub, interacting with both internal components like the DRL model and external systems like GIS tools. It processes and synthesizes data to inform the AI-driven planning process, which iteratively refines urban layouts through model training cycles.

Technical Specifications

- Architecture of Database to manage and retrieve urban planning data.

System Elements from Different Perspectives

- The system's architecture encompasses the processing unit, external systems, and data flow. The physical deployment spans servers, data storage, and network infrastructure.
- The codebase is structured around the main components, such as the DRL model, data processing, and storage systems. Additionally, the system incorporates security features to safeguard data integrity, confidentiality, and system resilience against unauthorized access and breaches.

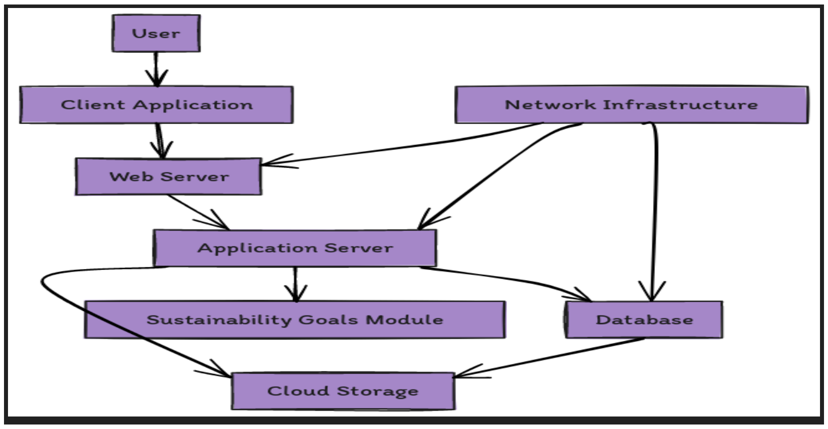


Fig 4.2 Logical dataflow, servers, storage, network infrastructure, database architecture

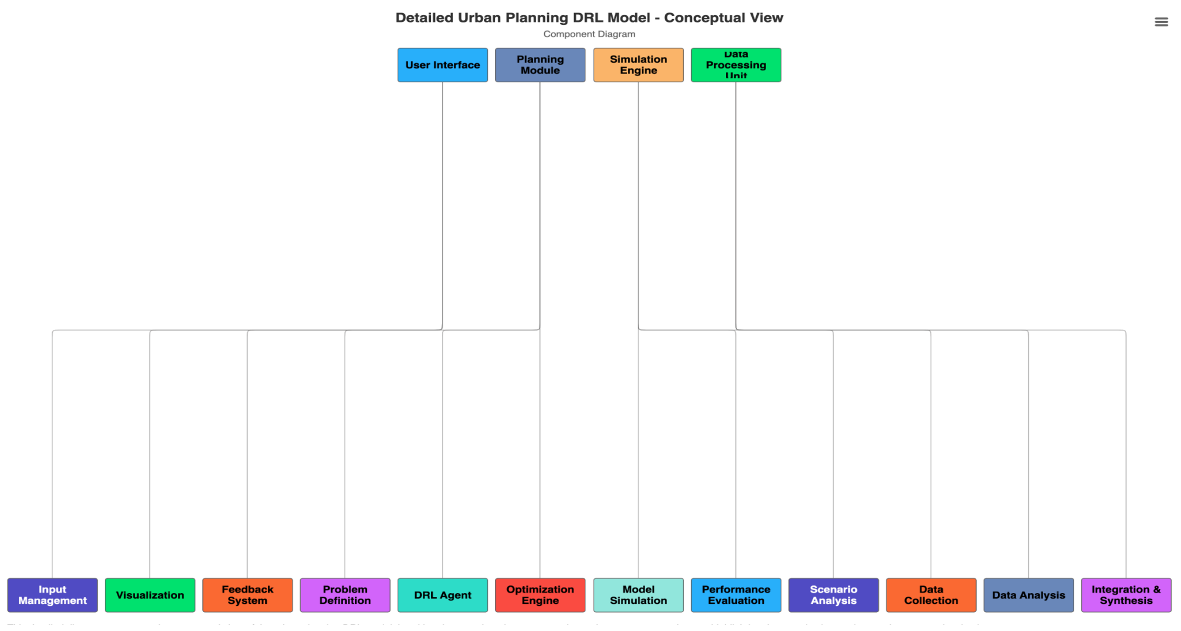


Fig 4.3 Conceptual/Logical

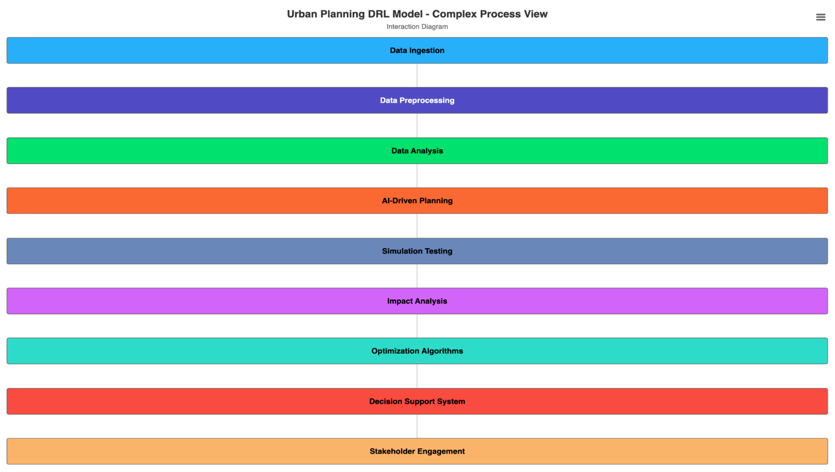


Fig 4.4 Process/Runtime

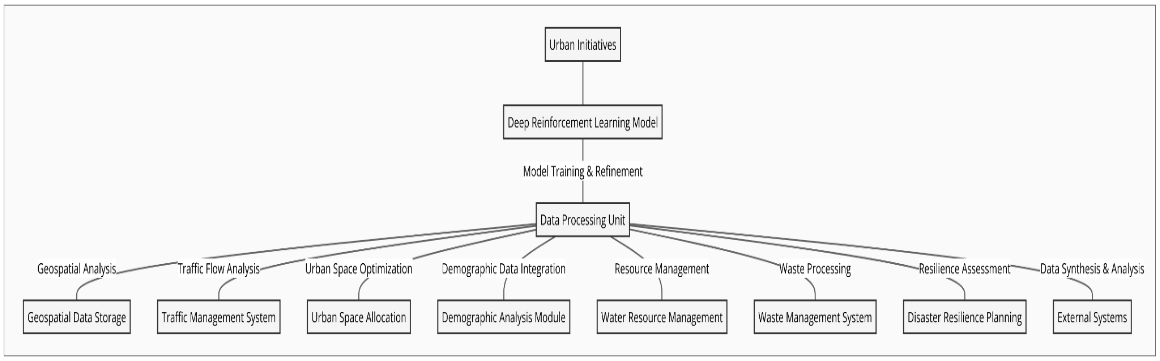


Fig 4.5 MODULES

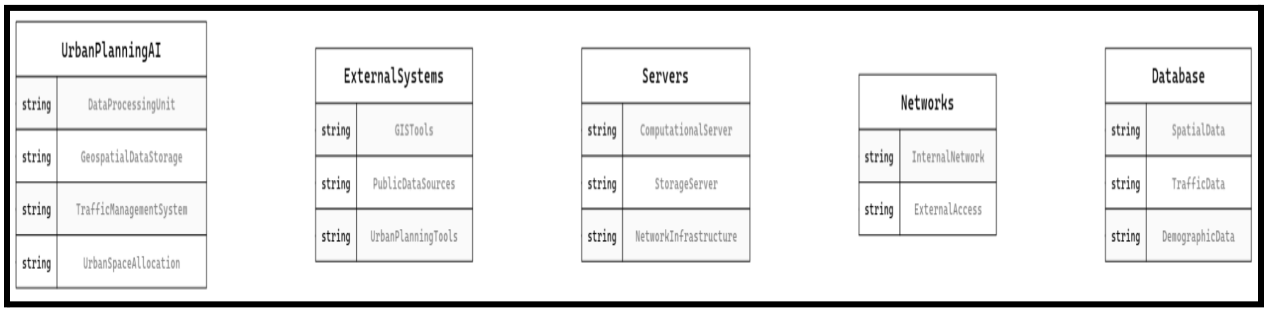


Fig 4.6 System Elements

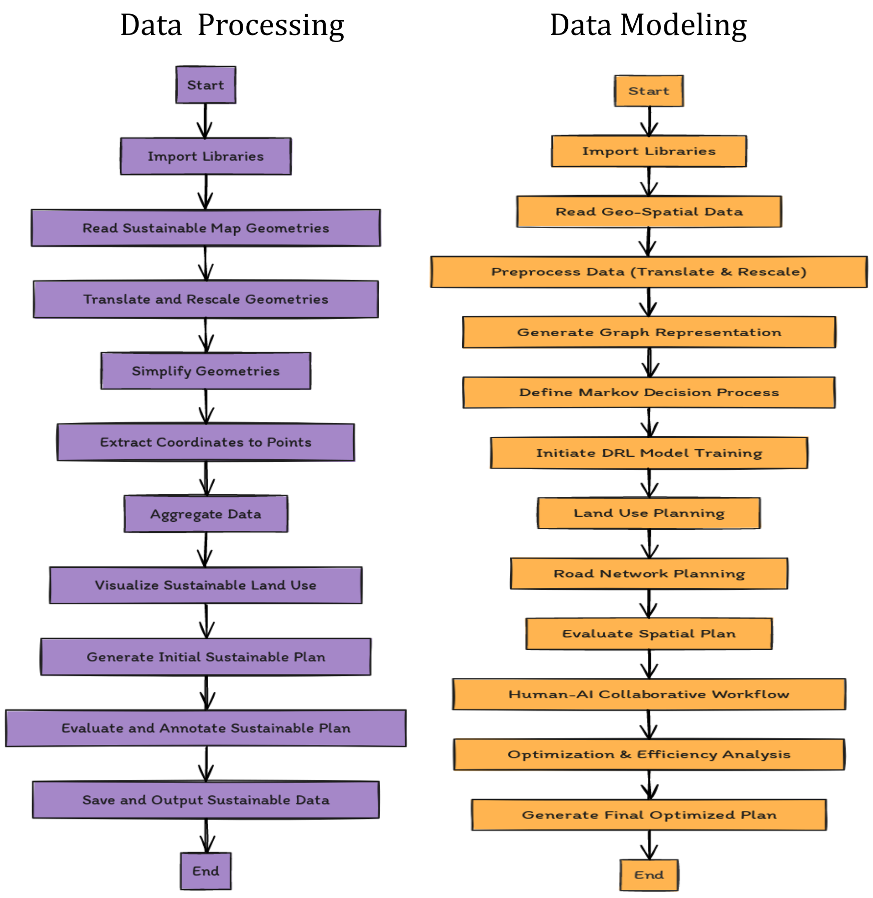

Data Processing
Start
Import Libraries
Read Sustainable Map Geometries
Translate and Rescale Geometries
Simplify Geometries
Extract Coordinates to Points
Aggregate Data
Visualize Sustainable Land Use
Generate Initial Sustainable Plan
Evaluate and Annotate Sustainable Plan
Save and Output Sustainable Data
End
Data Modeling
Start
Import Libraries
Read Geo-Spatial Data
Preprocess Data (Translate & Rescale)
Generate Graph Representation
Define Markov Decision Process
Initiate DRL Model Training
Land Use Planning
Road Network Planning
Evaluate Spatial Plan
Human-AI Collaborative Workflow
Optimization & Efficiency Analysis
Generate Final Optimized Plan
End

Fig 4.7 Data Flow

4.1.8 Design Description

The modules and classes of the Urban Planning System is as follows:

- UrbanPlanningAISystem: Central class managing the process of creating the urban planning, communicating with other modules.
- GeospatialDataStorage: Oversees space information, communicates with the Urban Planning AI System for data handling, and feeds the Data Processing Unit.
- DeepReinforcementLearningModel: Located at the centre of the planning logic, it takes input from the Urban Planning AI System and communicates with the Traffic Management System for traffic data.
- DataProcessingUnit: Processes data and gets input from the Urban Planning AI System as well as Geospatial Data Storage.
- TrafficManagementSystem: Implements traffic simulation and analysis that is connected to the Deep Reinforcement Learning Model for traffic data and analysis.
- UrbanSpaceAllocation: Coordinates the management of the areas in the cities as the living and business districts as managed by the Urban Planning AI System for spatial management.
- The following interfaces depict how the different modules of the system are integrated to support the urban planning process in detail.

4.1.9 Design Details

The high-level design of the "Leveraging AI for Sustainable Urban Planning" project uses emerging AI frameworks such as GANs and DRL for improving the design of urban plans. It puts great emphasis on the areas of interface compatibility, efficiency, protection, dependability, and easy serviceability due to its modular structure and practice of gradual enhancement. The proposed approach should also be easily compatible with the current applications in urban planning, as well as be optimized in terms of resource consumption.

4.1.9.1 Novelty

- Applying GCNs, and RL for generating, analyzing, and optimizing the layouts of urban plans, the introduction of a new form of spatial planning, based on the principles of flexibility and sustainability.
- Spatial graphs for the fine-grained analysis of urban spaces, a novel approach to making use of spatial graphs as tools for representing spatial data.

4.1.9.2 Innovativeness

- Sophisticated AI tools for smart analysis of the city's environment and for making better decisions about the city's development than traditional urban planning methods allow.
- Most layered visualizations are a rich and complete representation of the systems present in an urban setting, making it possible to gain increased understanding of the relationships in cities.

4.1.9.3 Interoperability

- This system is built to conform to standard data formats and protocols so that it can easily interface with other data sources, analytical tools and other urban planning software. It does this by avoiding the need to rip and replace current connections and instead aims to optimize data exchange and organizational coordination without disrupting integration.

4.1.9.4 Performance

- Deep Reinforcement Learning (DRL) for Road Network Planning:

    - Improve the efficiency of the proposed deep reinforcement learning (DRL) technique that plans road networks for slums.
    - Develop graph mathematical models to model the slums' topological structure and choose the best roads using graph neural networks (GNN).
    - Use of multi-objective policy optimization techniques in order to achieve the goals of universal connectivity and reduced travelling time while at the same time minimizing construction costs.

- Multi-Agent Actor-Critic (MA2C) Algorithm:

    - OThe specific aims are as follows: Improve the novel Multi-Agent Actor–Critic (MA2C) algorithm specifically designed for multi-autonomous vehicle (AV) lane-changing in mixed-traffic environments to exhibit high performance.
    - Optimise resources by applying the necessary data structures and algorithms for reasonable handling of the multi-agent interactions of dynamic world.

Effectiveness for high performance, more capable of handling large datasets and running complex spatial analyses than causing the user experience delays.
Optimised data handling and processing designs for reducing response time and increasing transaction rate.
High performance is realised through the selection of appropriate data processing pathways and algorithms that can handle large volumes of urban data. The concept is designed to offer efficient and fast responses to queries from the users and to update urban models in real-time making the planning process more useful and efficient.

4.1.9.5 Reliability

Fault-Tolerant System Design:

- **Redundancy:** Employ redundancy at the hardware, software and network sub-systems in order to continue with operation when there are failures. For instance, the use of multiple servers in a cluster system can help to guarantee an easy switchover in the event of hardware problems.
- **Load balancing:** Use load balancing methods in order to spread the incoming traffic among several servers or instances to avoid overload of a single server or instance and guaranteeing the work efficiency.
- **Disaster recovery:** Create and enforce a sound disaster recovery plan, along with backups and replication techniques, failover processes, and other features that will help reduce the time required to recover from natural disasters and other critical system failures.

Regular Updates and Maintenance:

- **Patch management**: Establish a proactive patch management process to regularly update and patch software components, operating systems, and dependencies to address known vulnerabilities and security flaws. It also assists in the reduction of attack vectors from exploitation from the various malicious players out there.
- **Monitoring and alerting:** The following should be implemented constantly while in use in order to ensure the health, performance and security of the system are well checked: Use notifications if you have to be notified once there are signs of low performance, or security issues.
- **Proactive maintenance:** Carry out system checkups and reviews, maintain system health checks, diagnosis exercises and performance optimization activities in the context of the system, that should be able to determine any other latent conditions or effects, before they transform themselves into major disasters. This entails reading of system logs, collected performance statistics and received feedback and identifying issues that are likely to develop into problems as well as enhancing system performance.

Continuous Improvement:

- **Agile methodologies:** Use Scrum or Kanban as a useful working software development framework for planning, developing, testing and implementing system improvements and updates. It is possible to adjust and develop the solution in parallel with the changing needs, the new technologies, and users' expectations.
- **Feedback loop:** Create a mechanism with which users, stakeholders and system administrators can put forward their thoughts, observations, experiences and suggestions on how to enhance on the reliability of the system and its performance. Incorporate this feedback into the development roadmap to prioritize and implement enhancements that align with user needs and expectations.

4.1.9.6 Maintainability

Modular Design:

- Embracing an SOA approach where system is broken down into components, which are, relatively independent of each other and developed to perform a singular task. This makes its update, maintenance and scale more manageable since changes to one module does not affect the others.

- Apply design patterns like, – micro services, or service oriented architecture where some function logically can be packaged as a service and invoked later, in a way it can be easily maintained and scaled when required as changes do not affect the entire structure of the application.

Clear Documentation and Coding Standards:

Keep good documentation that will encompass architecture, design choices, API and the structure of the codebase. This documentation should be kept at a more convenient place and should be updated periodically with the changes and new additions to the system.

4.1.9.7 Portability

Cross-Platform Compatibility:

To ensure cross-platform compatibility, develop the system using technologies and frameworks that can work across multiple operating systems and environments. This allows the system to be deployed and used seamlessly on different platforms without significant modifications. Utilize programming languages like Java, Python, or JavaScript, along with platform-independent libraries and dependencies to minimize platform-specific dependencies and facilitate portability.

4.1.9.8 Legacy to Modernization

Plan in Phases:

- **Assessment Phase:** thoroughly examine the existing systems, workflows, and requirements. Identify the areas that need modernization and improvement.
- **Prioritization**:Prioritize the components or functionalities for modernization based on factors such as business impact, technical complexity, and user needs.
- **Define Phases:** Divide the modernization process into manageable phases, each with specific objectives, deliverables, and timelines.

Start with Low-Risk Changes:

Begin the modernization process with low-risk changes. Starting with a proof of concept or pilot project to demonstrate the feasibility and benefits of AI-driven processes within the existing environment. Gradually integrate AI-powered analytics or decision support tools into the existing systems. Gather feedback from users and stakeholders during each phase of implementation to identify areas for improvement and refinement.

4.1.9.9 Resource utilization

The system resources are optimized to operate efficiently, even on limited infrastructure. Resource monitoring tools are integrated to provide insights into system performance and resource usage, enabling proactive management and optimization to avoid issues and ensure long-term viability.

4.2 Low-Level Design Document

4.2.1 Overview

This Low Level Design document for sustainable city planners addresses the use of additive learning to improve sustainability in cities by providing detailed information about architecture and design. This covers key activities such as traffic improvement, land allocation, waste management, life support and environmental impact reduction. Each model explains its specific role in the overall process to ensure that the design is consistent with the broad goals of urban development. It provides detailed modeling and data flow to facilitate implementation and future development.

The main goal of this model is to create an intelligent platform for sustainable urban planning using additive learning models to make decision-making changes in areas such as housing construction, waste management and high-speed connectivity. Our design uses a design model that focuses on key areas such as modeling and real-time decision making with the help of adaptation strategies to ensure scalability, efficiency, and environment.

4.2.2 Purpose

The purpose of this low-level design document is to provide detailed information to developers, urban planners, researchers, and other stakeholders interested in design, testing, and improved urban planning. It acts as a bridge between the high-level architecture and the detailed methods by specifying the behavior of each relevant class, module, and system. This information ensures that the development process is reliable, integrated, adaptable to future development, and aligned with the sustainability goals of the project.

- Determine the appropriate design and context behind it.
- provides detailed information about the processes, their roles, and interactions.
- demonstrates how to load security and city symbols into the RL model.

4.2.3 Scope

The scope of this low-level design information extends to the definition of key structures that empower urban planning, such as the promotion of learning-based optimization, planning and simulation tools, management systems, and environmental information. It describes internal and external processes, including interactions with urban data repositories, geospatial tools, and third-party simulation libraries. The documentation also includes diagrams, flowcharts, and pseudocode for the module, providing a clear understanding of the libraries, frameworks, and languages involved in its development. The dependencies, limitations, and assumptions surrounding the design and implementation of the model are also detailed.

4.2.4 Master Class Diagram

The main drawing of this project represents the basic architecture of urban planning and shows how various components interact to perform the required tasks. The main components of the system include Environment, which simulates urban environments, and PlanClient, which manages and updates land use and network configurations. Other basic classes such as CityEnv, InfeasibleActionError, and ObservationExtractor manage the city analog logic, error handling, and data recovery accordingly.

The system is used for actions such as land use, road construction, energy calculation, and allows landuse to be adjusted according to required needs.This class diagram shows the structure of the relationship between the structures, clarifying the flow of information in the system and how each part affects the planning of the city.

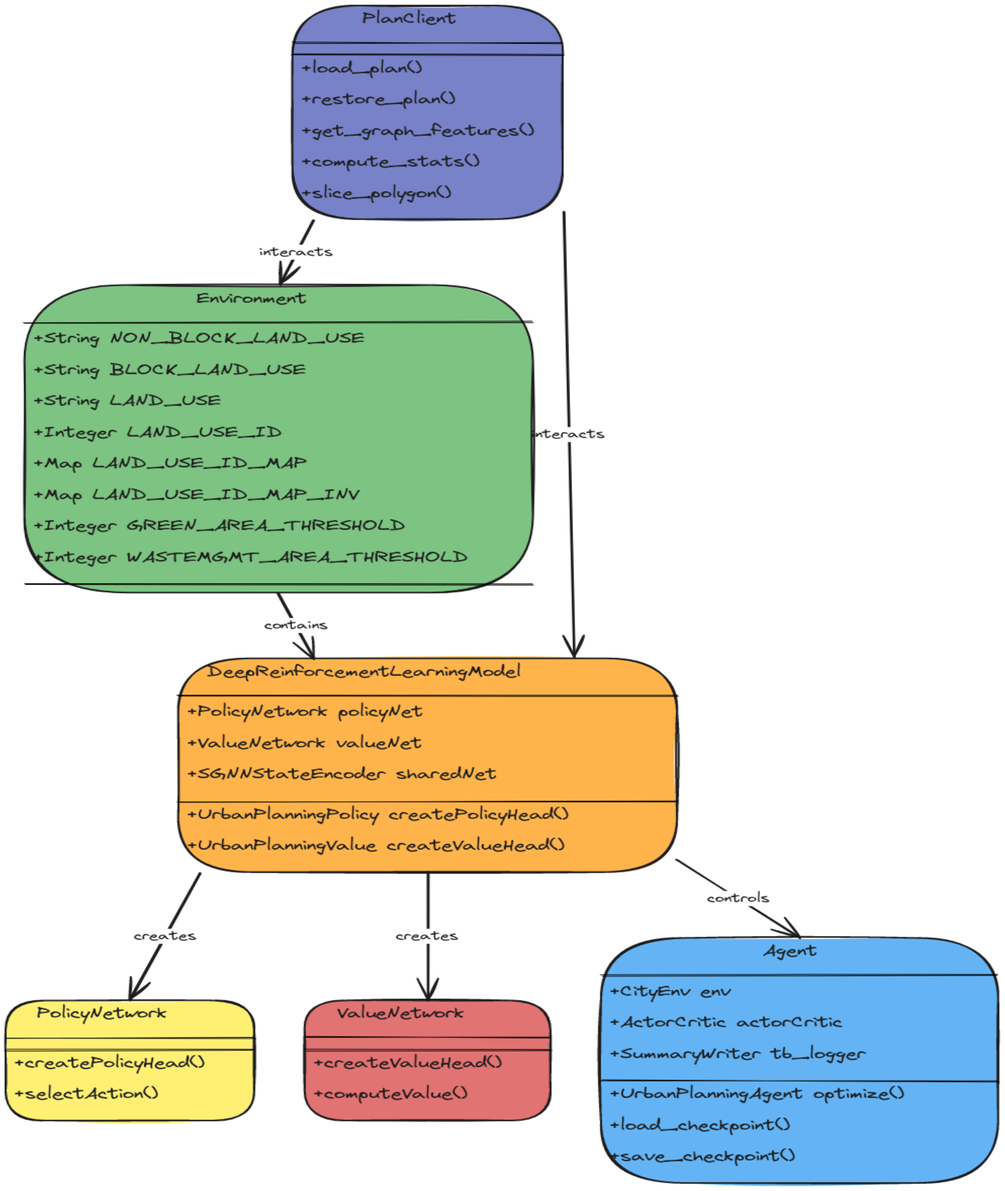
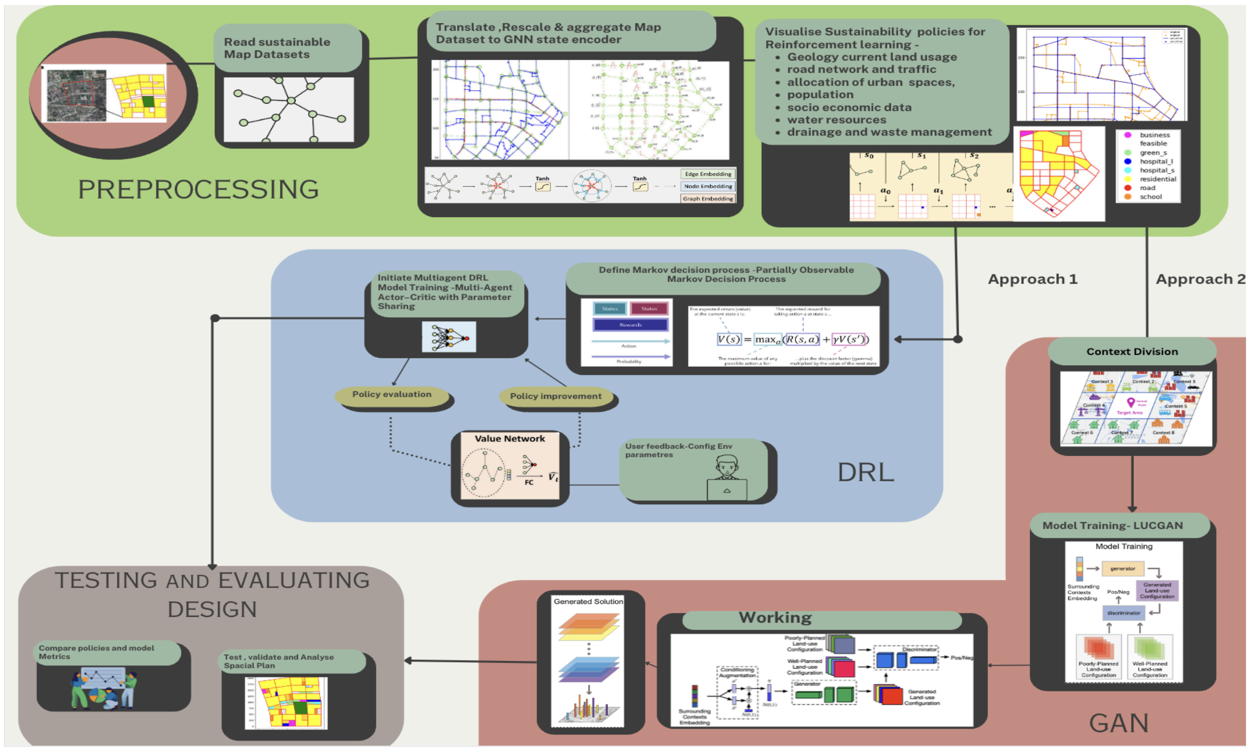


Fig 4.8 Master Class Diagram

CHAPTER - 5

SYSTEM DESIGN



Read sustainable Map Datasets
PREPROCESSING
Translate ,Rescale & aggregate Map Dataset to GNN state encoder
Tanh
Edge Embedding
Node Embedding
Graph Embedding
Visualise Sustainability policies for Reinforcement learning -
Geology current land usage
road network and traffic
allocation of urban spaces,
population
socio economic data
water resources
drainage and waste management
business
feasible
green_s
hospital_l
hospital_s
residential
road
school
Approach 1
Approach 2
Initiate Multiagent DRL Model Training -Multi-Agent Actor–Critic with Parameter Sharing
Define Markov decision process -Partially Observable Markov Decision Process
$V(s) = \max_a(R(s,a) + \gamma V(s'))$
Policy evaluation
Policy improvement
Value Network
User feedback-Config Env parametres
DRL
Context Division
Model Training- LUCGAN
Model Training
generator
discriminator
GAN
Generated Solution
Working
TESTING AND EVALUATING DESIGN
Compare policies and model Metrics
Test , validate and Analyse Spacial Plan

Fig 5.1 Overview

5.2 Module - Neural Networks

5.2.1 Description

Reinforcement learning neural networks are the core component of the presented framework, as they enable the Agent to make rational decisions during the design of the layout of an urban environment. They work through and build the relations between the elements of urbanism and help the Agent come up with the best optimal solution. The system incorporates three specialized neural networks:

- Graph Neural Network (GNN):

The GNN is developed as a way to handle and learn from graph-based representations of urban features such as land use polygons, road networks and drainage networks. It retains spatial interconnection and dependency of these elements and infers a state representation of the city at each time step. The subsequent networks use this state for decision-making and evaluation.

- Policy Network:

The Policy Network receives the current state from the GNN and produces the best action to be taken by the organization. This could be change of use of the land, alterations in the geometric design of roads or alteration of the physical facilities. The Policy Network adapts the ability to choose actions that bring the urban design closer to an optimal state while attending to short-term goals.

5.2.2 Class 1: Urban Planning Policy

Class Description 1

This class defines the policy network for planning decisions for land use and road planning actions, based on a shared neural network state.

Data members 1

Data Type	Data Name	Access Modifiers	Initial Value	Description
dict	cfg	Public	N/A	Configuration dictionary containing network parameters.
urban planning agent	agent	Public	N/A	Agent responsible for executing actions.
nn.Module	shared_net	Public	N/A	Shared neural network to extract feature

				embeddings for the policy.
nn.Sequential	policy_land_use_head	Public	N/A	Policy head for land use decisions.
nn.Sequential	policy_road_head	Public	N/A	Policy head for road planning decisions.

Method - create_policy_head

Purpose: To design a multiple level policy head for the land use or road planning decision..
Input: input_size, hidden_size, name
Output: policy_head (nn.Sequential)
Parameters:

- input_size (int): Size of the input layer.
- hidden_size (list[int]): Sizes of the hidden layers.
- name (str): The head (land use or road) name.

Pseudo-code:

- Initialize a sequential model.
- Adding linear and Tanh layers based on hidden_size.
- If the last layer has size 1, we flatten the output.
- Return the policy_head.

Method - select_action

Purpose: Select an action based on the current policy distribution.

Input: x, mean_action=False

Output: action (tensor)

Parameters:

- x (tensor): Input state for action selection is defined as the knowledge the agent has concerning the complete environment it is operating in.
- mean_action (bool): Whether to pick the mean action.
- Exceptions: N/A

Pseudo-code:

- Please initiate a call forward to obtain these policy distributions.

- If available, then sample from land use and road distributions.

5.2.3. Class 2: SGNNStateEncoder

Class Description

A state encoder for urban planning based on a Graph Neural Network (GNN) that takes node and edge characteristics from the city graph and produces values important to policy.

Data Members

Data Type	Data Name	Access Modifiers	Initial Value	Description
dict	cfg	Public	N/A	Configuration dictionary containing network parameters.
urban planning agent	agent	Public	N/A	Agent responsible for executing actions.
nn.Module	edge_fc_layers	Public	N/A	Fully connected layers for processing edges in the GNN.
nn.	attention_layer	Public	N/A	Policy head for land use decisions.
nn.MultiheadAttention	attention_layer	Public	N/A	Multi-head attention layer for node attention.

Method - scatter_to_nodes

Purpose: Accumulate the embeddings of aggregation to the nodes.

Input: h_edges, edge_index, edge_mask, max_num_nodes

Output: h_nodes (tensor)

Parameters:

- h_edges (tensor): Edge embeddings.
- edge_index (tensor): A vector indicating the positions of each node of the neighbors.
- edge_mask (tensor): Mask for valid edges.
- max_num_nodes (int): Maximum number of nodes.

Exceptions: N/A

Pseudo-code:

- To accumulate edges onto nodes, make use of the scatter_count.
- Return the updated node embeddings.

Method - self_attention

Purpose: When in the current node, apply attention to the rest of the node embeddings of the remaining graph nodes.

Input: h_current_node, h_nodes, node_mask

Output: h_current_node_attended (tensor)

Parameters:

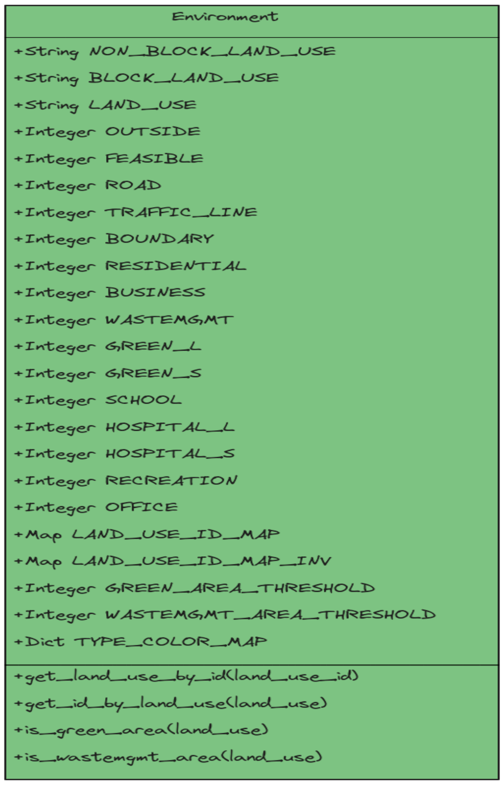
- h_current_node (tensor): Embedding of the current node.
- h_nodes (tensor): Embeddings of all nodes.
- node_mask (tensor): Mask indicating valid nodes.

Exceptions: N/A

Pseudo-code:

- Calculate attention scores from the transformed query, key and values.
- Pay attention to the current node embedding.
- Return the updated current node embedding.

5.2.3 Class 3 Environment



Description

The Environment class is designed to model a simple city-simulation in which some of the land uses can be blocked while other cannot. It handles various types of the land use, green areas, public services, and serves as an input to place the land use areas and calculates the rewards.

5.2.4 Data Members

Data Type	Data Name	Access Modifiers	Initial Value	Description
list	NON_BLOCK_LAND_USE	Public	N/A	List of non-blockable land uses (e.g., outside, feasible, road).
list	BLOCK_LAND_USE	Public	N/A	List of blockable land uses (e.g., residential, business, school).
list	LAND_USE	Public	N/A	List of all land uses (both blockable and non-blockable).
int	NUM_TYPES	Public	N/A	Total number of land use types.
dict	LAND_USE_ID_MAP	Public	N/A	Maps the land use types to their corresponding ID numbers.
float	GREEN_AREA_THRESHOLD	Public	N/A	Threshold area for green spaces.
float	WASTEMGMT_AREA_THRESHOLD	Public	N/A	Threshold area for waste management services.

PlanClient	_plc	Private	N/A	Interface to the land use plan client that manages land use placement and configuration.
Config	cfg	Private	N/A	Configuration object containing settings and parameters for the environment.

Method -place_land_use

Purpose: Put a certain land use in the environment at the given place and do.

Input:

- land_use (dict): A dictionary with attributes regarding the land use type, position of the type, and the dimensions of the type.
- action (int): The step to be taken, which is equivalent to putting a land use block at a particular point.

Output: None, but also changes the internal state of the environment.

Parameters:

- land_use (dict): A dictionary containing information about type; x and y coordinates; and dimensions.
- action (int): The action that prescribes the location that the land use will be implemented.

Pseudo-code:

- Bring the block to exercise the action of placing land use.
- Further, it must be validated whether or not the block is usable for placement.
- If possible, put the land use and then update the environment.
- If the land use is encompassed in the block, then the block should be filled in.
- It is also important to update the statistics for the placement of land use.

- Fine those areas that are close to residential zones and provide incentives for those that are closer to green zones.
- Return the total amount of reward for the placement of waste management.

5.2.5 Class 4 Agents

The Sustainable Urban Planner uses a multi-agent system in which each agent is in charge of important decisions concerning urban planning. Some of these agents employ RL models to adjust their decisions on real-time data produced by the environment, in order to improve a number of urban sustainability indicators including land-use distribution, traffic patterns, and waste disposal. The following explains the different agents and their functions in the system with the help of the following figure.

Urban Planning Agent

The Urban Planning Agent is the central decision maker in the system and its key function is to enforce land use and road planning policies. It discusses with other classes like the Urban Planning Policy and SGNNStateEncoder to understand the current status of the city and suggest the best course of action for the urban city.

- *Policy-Based Decision Making:* The Urban Planning Agent employs the Urban Planning Policy class that encompasses a dual purpose neural network for both land use and road planning. This agent chooses actions according to the policies derived from the distribution functions estimated by the neural part of the system taking into account the current state of the city and distribution of resources.

- *Action Selection:* The agent makes decisions through the select_action method where actions such as placing of land use or road construction are determined from policy heads which are learnt (policy_land_use_head and policy_road_head). These actions are intended for addressing the infrastructure requirements of a city while addressing the sustainable practices of the city including availability of green areas and traffic patterns.

- *Multi-Layer Policy Networks:* The action-selection mechanism of the Urban Planning Agent is based on the multi-layer policy networks. When using the create_policy_head method, it defines different policy heads for the land use and road planning so that each action type will have its own policies and its parameters. The policy networks help the agent to assess a number of possible scenarios and choose the most suitable one.

5.2.6 SGNNStateEncoder Agent

The SGNNStateEncoder (Graph Neural Network-based agent) takes the city's graph, where nodes correspond to areas (land use) and edges are roads or connections between areas. This agent is very important in the process of translating spatial data into meaningful embeddings that can be used by a policy network.

- *Graph-Based Representation:* This agent acts in a graph with nodes that are city blocks and edges that connect city blocks or represent other spatial interactions. That is, it accepts both node and edge features to give a holistic representation of the urban setting. The agent is able to learn how different areas integrate with each other through the methods like scatter_to_nodes that collect edge embeddings into node level representations.

- *Self-Attention Mechanism:* The self_attention method enables this agent to perform multi-head attention on the current node thereby improving on its awareness of the node in the overall city graph. This allows the agent to concentrate on areas that would require its decision, for example, areas with many vehicles or few parcels of land, which is important for policy network input.

5.2.7 Environmental Agent

The Environment Agent is engaged in direct communication with the simulation environment to make sure that foreseen actions do not contradict the principles of sustainability and existent environmental limitations. This agent is tasked with the responsibility of appraising and implementing decisions on land use and infrastructure, and the computation of the pay offs of the different courses of action.

- *Reward Calculation:* The Environmental Agent computes several incentives that go back to the RL models. For example, the get_greenness_reward method allows the system to be rewarded for increasing green coverage in and around residential areas in order to balance the need for expansion of urban infrastructure with environmental conservation. Likewise, the get_wastemgmt_reward method provides a reward for the best placement of waste management facilities to areas close to residences and green areas.

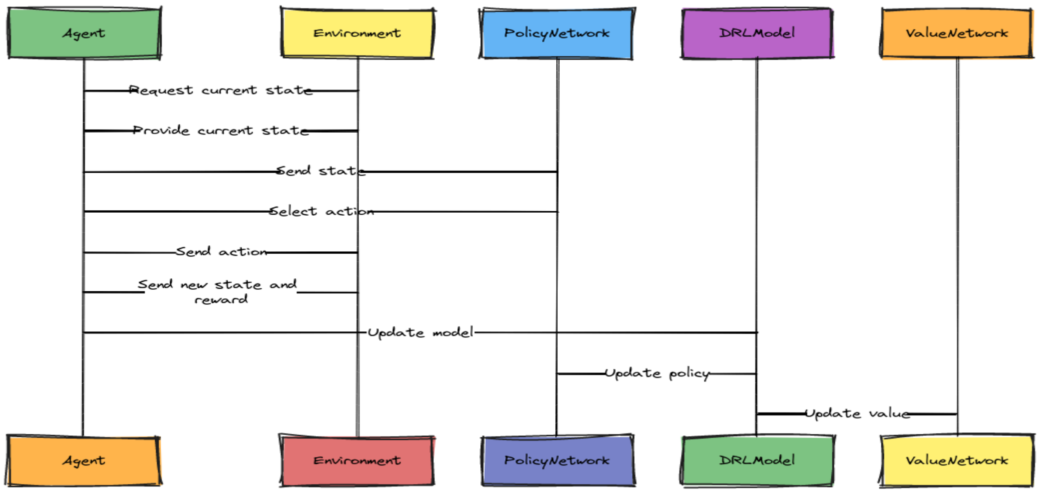
- *Land Use Placement :* Using the place_land_use method, the Environment Agent assigns particular types of land use such as residential, commercial or green areas according to the decisions of the Urban Planning Agent. It confirms the

possibility of placing land use and refreshes the environment. This agent safeguards green areas and makes sure that decisions on the use of land improve the condition of the total urban environment.

- *Infrastructure Development:* The agent is also in charge of infrastructure choices including building roads (build_road). It assesses the constructibility of roads in certain locations and alters the geography of the city. This means that transport systems are extended in the most cost-effective and least environmentally-sensitive manner.

- *Policy Learning Loop:* The agents make their own decisions in which they act, acquire their rewards and update their own policies in a cycle. For example, if an agent decides to locate a waste management facility near a residential area, the system will punish the decision. This negative feedback helps the agent to make more sustainable decision in the next time step of the process.

5.3 Sequence diagram



CHAPTER - 6

PROPOSED METHODOLOGY

6.1 Agents

The Agents is the key decision maker in this study. This agent employs a reinforcement learning paradigm, specifically designed to optimize urban planning strategies through the following processes:

▪ **State Representation:** The agent gets the copy of the current state of the city in the form of a number of parameters such as traffic density, available resource, land use, and environment, etc. These states are important for the agent to determine the effects of its actions within the urban environment.

▪ **Action Selection:** The agent uses the policy function, to determine an appropriate action in a given state. Examples of measures would include decisions on investment in infrastructure, resource mobilization or policy adjustments. The policy is obtained from a mix of learned actions and ad hoc exploration schemes so that the agent can learn better solutions over time.

· **Reward Mechanism:** The agent receives feedback in the form of rewards, which are determined with reference to the accomplishment of certain actions towards the achievement of certain urban planning objectives such as traffic flow density, pollution levels or resource exploitation. This feedback loop is used so that the agent repeats the decision-making policies through a learning process.

· **Learning Algorithm:** The agent implements a state-of-the-art reinforcement learning algorithm (e.g., Q-learning, Deep Q-Networks) to modify the action plan according to the results achieved. The agent gains better planning based on experience replay and target networks that helps it to estimate the consequences of actions in similar states.

6.2 Environment

The urban environment is modeled through several interconnected components that simulate real-world urban dynamics:

city.py: The module which encapsulates the physical and socio-economic characteristics of the city. It defines elements such as:

- **Geographical Layout**: The spatial arrangement of districts, neighborhoods, and key infrastructure (e.g., roads, parks, public transport).

- **Dynamic Variables:** Changes in population density, land use, and environmental factors that evolve over time based on the agent's decisions and external influences (e.g., economic shifts, demographic changes).

- **Inter-agent Interactions:** Mechanisms for simulating how multiple agents or factors within the city interact, providing a realistic portrayal of urban systems.

city_config.py: This configuration module allows for the customization of simulation parameters, including:

- **Initial Conditions:** Set up the starting state of the city, including population size, resource availability, and existing infrastructure.

- **Simulation Parameters:** Define the duration of the simulation, frequency of decision-making intervals, and other operational parameters to control the complexity and length of the simulations.

plan_client.py: This module facilitates communication between the urban planning agent and the city environment. It acts as an intermediary, managing the flow of information through:

- **Observation Relay:** Transmitting the state of the city to the agent for evaluation.

- **Action Execution:** Implementing the agent's decisions within the city model and updating the environment based on the outcomes of these actions.

observation_extractor.py: This component plays a critical role in data collection and preprocessing. It extracts relevant features from the city environment, such as:

- **Traffic Data:** Real-time information about traffic congestion, vehicle flow, and public transport utilization.

- **Resource Distribution:** Data on the availability and usage of essential resources (e.g., water, energy) across different city regions.

6.2 Environment

The urban environment is modeled through several interconnected components that simulate real-world urban dynamics:

city.py: The module which carries the physical and socio-economic attributes of the city. It defines elements such as:

- Geographical Layout: The geographical distribution of districts, neighborhoods, main facilities (roads, parks, public transport).
- Dynamic Variables: Population density alteration, change of land use type, the alteration of environmental parameters that vary over time as the result of the decision-making process of the agent and other factors such as the economic or demographic shifts.
- Inter-agent Interactions: The approaches for modeling how many agents or forces inside the city might communicate and behave, and offering a genuine image of urban systems.

city_config.py: This configuration module includes the possibility of setting the following simulation parameters:

- Initial Conditions: Define the initial conditions of the city which include initial population size in the city and initial resource base and initial infrastructure.
- Simulation Parameters: Explain the number of periods in a simulation process and the frequency of decision making intervals and other operational parameters in order to manage the simulation and its length.

plan_client.py: This module assists in communication of the urban planning agent with the surrounding city environment. It acts as an intermediary, managing the flow of information through:

- Observation Relay: Informing the state of the city to the agent for assessment.
- Action Execution: Appling the actions defined by the agent to the city model and changing the environment according to the results of performed steps.

observation_extractor.py: This component is very important in the process of data collection and data cleaning. It extracts relevant features from the city environment, such as:

- Traffic Data: Information on the state of traffic congestion, traffic intensity, and the use of public transportation in real-time.
- Resource Distribution: Information concerning the accessibility and consumption of the basic necessities (water, energy) in various zones of the city.

6.3 Models

The ***baseline*** module is used as a point of reference and the agent's learned policies are compared against the baseline performance. It emulates basic heuristic strategies that approximate conventional city planning processes in order to generate a baseline against which the enhancements achieved by the agent's reinforcement learning are compared.

Integral to the agent, and at the core of the reinforcement learning algorithm is the ***model module*** It includes:

Central to the agent's functionality, the ***model*** module encompasses the architecture of the reinforcement learning algorithm. It includes:

- **Training Procedures:** Enactment of training schedules that enable the specific learning activities of the agent in its interaction with the environment.

- **Policy and Value Function Updates:** Methods for changing the agent's policy and value estimates in light of newly acquired experience, so that learning takes place in a way that is simultaneously economical and beneficial.

  The policy module is responsible for the architecture of the agent's policy, that is, for defining what action the agent will take given the encoded state information.

The ***policy*** module defines the structure of the agent's policy, determining how it selects actions based on the encoded state information. Key features include:

- **Deterministic vs. Stochastic Policies:** Several possibilities to enable deterministic (always choose the best action) or stochastic (randomly select action based on probability values) kind of behaviour during learning phase.

- **Policy Optimization Techniques**: Implementation of advanced methods (e.g., Proximal Policy Optimization) to improve the stability and reliability of the agent's action selections over time.

This ***state encoder model*** improves the learning process effectiveness by reducing the dimensionality of the observations. This

- **Feature Extraction:** The process of analyzing the environment and selecting which of its aspects will be most suitable to the current situation of the agent.

· **Dimensionality Reduction Techniques:** Using processes for reducing dimensionality of the state representation and yet maintain important characteristics; such as Principal Component Analysis (PCA) or an autoencoder.

The ***VALUE*** component calculates the value function that evaluates the expected future returns of different state-action pairs. Its functionalities include:

· **Temporal Difference Learning:** Utilizing TD learning methods to update the value estimates based on new information from the environment, ensuring that the agent's predictions are continually refined.

· **Value Function Approximation**: Implementing neural networks or other function approximators to generalize value estimates across similar states, improving the agent's performance in complex urban scenarios.

6.4 Reward Functions

These are the reward functions in this reinforcement learning framework as they are a direct representation of the objectives/use-case of our project. Each function assesses distinct facets of the urban layout, including the efficiency of the road network, accessibility to public services, and the availability of green spaces.For instance:

- Road Network Reward: This function evaluates the compactness of the roads, and the presence of dead-end roads, short roads and long roads is penalized. This is because a good road network structure that has high connectivity and density is of essence to the mobility of an urban center.

- Life Circle Reward: It evaluates the proximity of key public facilities to residential places, with focus on how services can be within the reachable region. This reward helps to foster urban designs that benefit the public.This function assesses the density of green areas in relation to living quarters and encourages solutions that will improve environmental quality and the health of the inhabitants.

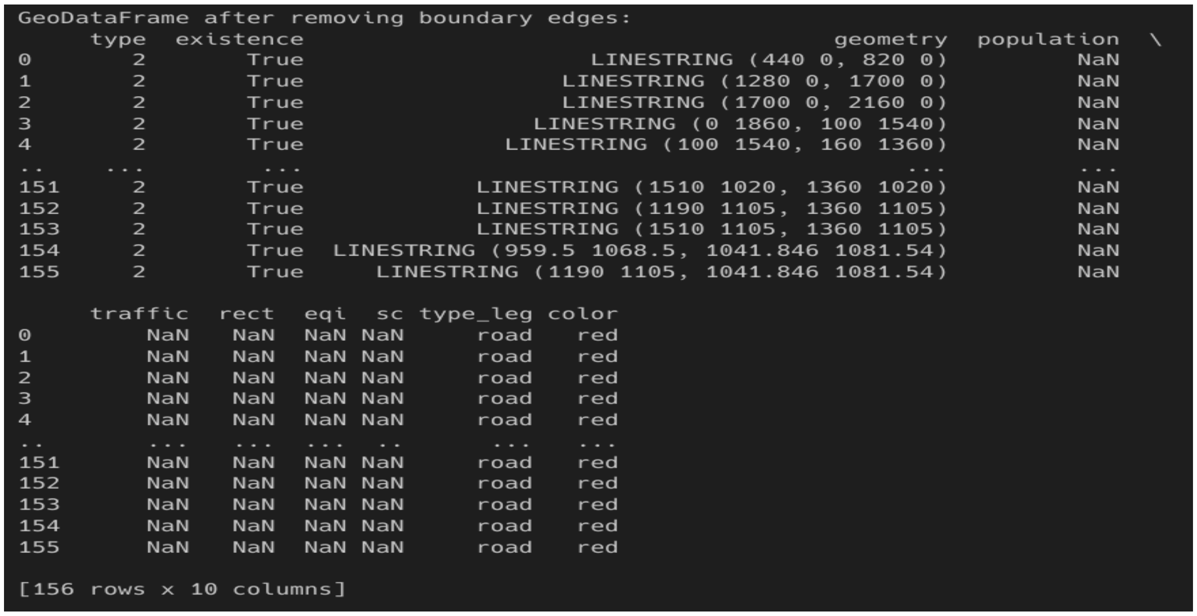
- Greenness Reward: This function evaluates the proportion of green spaces in relation to residential areas, promoting designs that enhance environmental quality and community health

CHAPTER- 7
IMPLEMENTATION AND PSEUDOCODE

7.1 Data Collection and preprocessing

Our project is focused on developing an urban planning model that prioritizes sustainability and at the same time is feasible. The previous urban planning models don't take sustainability factors into picture. To achieve this we have gathered numerous datasets like electricity lines, sewage treatments and ground water index from the OpenCity portal, a valuable resource with urban data. These datasets provide us with crucial information essential for urban planning initiatives. We have also used detailed satellite imagery from Java OpenStreetMap(JOSM), with a comprehensive view on the layout of our target area.

The image collected by the JOSM is then converted into GeoDataFrame(GDF) using python libraries like pandas, geopandas, matplotlib, etc. The GDFs store attribute data like type, existence, etc. and geometric data like lines, polygons, linestrings, etc. enabling spatial operations. The GDF is then converted to a pickle file that is passed to the model.

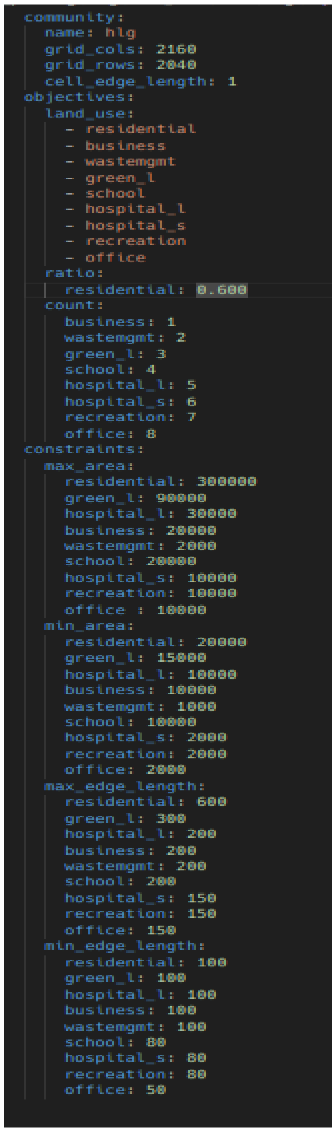


```
GeoDataFrame after removing boundary edges:
     type  existence                                     geometry  population  \
0       2       True                       LINESTRING (440 0, 820 0)         NaN
1       2       True                     LINESTRING (1280 0, 1700 0)         NaN
2       2       True                     LINESTRING (1700 0, 2160 0)         NaN
3       2       True                   LINESTRING (0 1860, 100 1540)         NaN
4       2       True                 LINESTRING (100 1540, 160 1360)         NaN
..    ...        ...                                             ...         ...
151     2       True               LINESTRING (1510 1020, 1360 1020)         NaN
152     2       True               LINESTRING (1190 1105, 1360 1105)         NaN
153     2       True               LINESTRING (1510 1105, 1360 1105)         NaN
154     2       True  LINESTRING (959.5 1068.5, 1041.846 1081.54)         NaN
155     2       True     LINESTRING (1190 1105, 1041.846 1081.54)         NaN

     traffic  rect  eqi  sc type_leg color
0        NaN   NaN  NaN NaN     road   red
1        NaN   NaN  NaN NaN     road   red
2        NaN   NaN  NaN NaN     road   red
3        NaN   NaN  NaN NaN     road   red
4        NaN   NaN  NaN NaN     road   red
..       ...   ...  ...  ..      ...   ...
151      NaN   NaN  NaN NaN     road   red
152      NaN   NaN  NaN NaN     road   red
153      NaN   NaN  NaN NaN     road   red
154      NaN   NaN  NaN NaN     road   red
155      NaN   NaN  NaN NaN     road   red

[156 rows x 10 columns]
```

7.1 Geodataframe

7.2 User configuration files

The configuration files are used to define the key parameters of the land layout. It outlines the grid dimensions, land-use and non land-use blocks, etc. User can set these constraints and parameters according to his needs. These configurations will guide the reinforcement learning model in optimizing land-use allocation within the grid.



```yaml
community:
  name: hlg
  grid_cols: 2160
  grid_rows: 2040
  cell_edge_length: 1
objectives:
  land_use:
    - residential
    - business
    - wastemgmt
    - green_l
    - school
    - hospital_l
    - hospital_s
    - recreation
    - office
  ratio:
    residential: 0.600
  count:
    business: 1
    wastemgmt: 2
    green_l: 3
    school: 4
    hospital_l: 5
    hospital_s: 6
    recreation: 7
    office: 8
constraints:
  max_area:
    residential: 300000
    green_l: 90000
    hospital_l: 30000
    business: 20000
    wastemgmt: 2000
    school: 20000
    hospital_s: 10000
    recreation: 10000
    office : 10000
  min_area:
    residential: 20000
    green_l: 15000
    hospital_l: 10000
    business: 10000
    wastemgmt: 1000
    school: 10000
    hospital_s: 2000
    recreation: 2000
    office: 2000
  max_edge_length:
    residential: 600
    green_l: 300
    hospital_l: 200
    business: 200
    wastemgmt: 200
    school: 200
    hospital_s: 150
    recreation: 150
    office: 150
  min_edge_length:
    residential: 100
    green_l: 100
    hospital_l: 100
    business: 100
    wastemgmt: 100
    school: 80
    hospital_s: 80
    recreation: 80
    office: 50
```

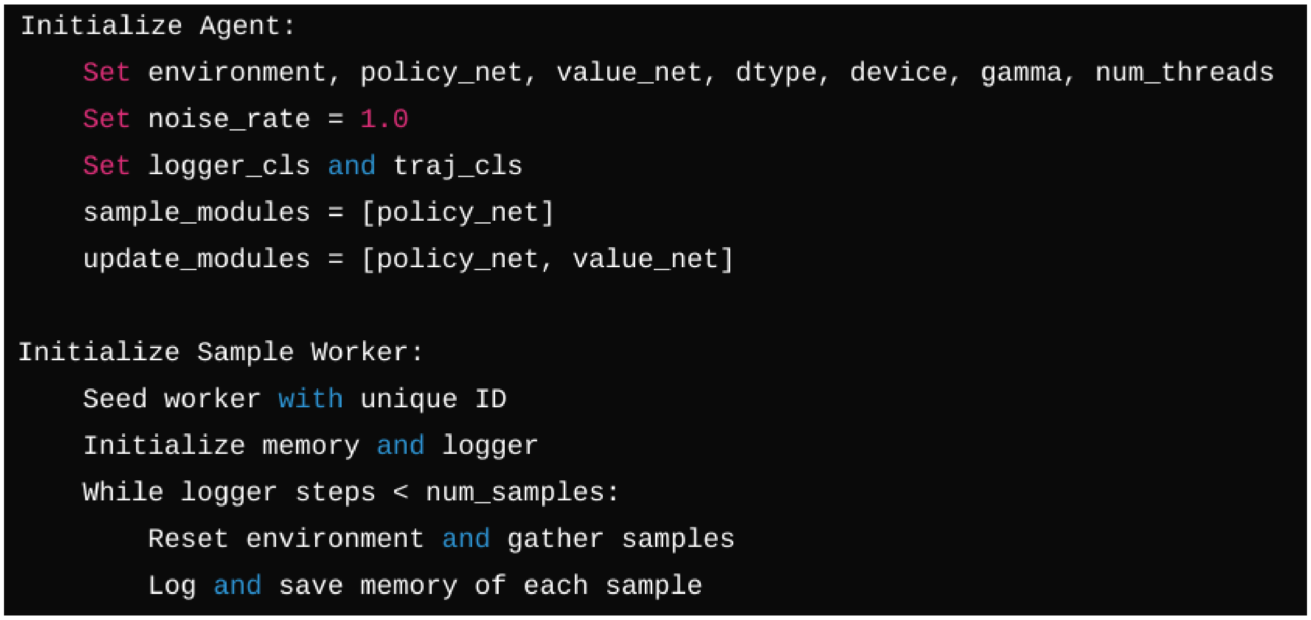
Fig 7.2 config.yaml

7.3 Reinforcement Learning model

Reinforcement Learning is a model where an agent explores the environment, tries out actions and learns from the outcomes. If an action leads to a positive outcome, the agent remembers it and uses it in further iterations. Whereas if the action leads to a negative outcome, the agent learns to avoid it. Over time the agent discovers the best ways to maximize the reward.

7.3.1 Agent initialization

The agent is initialized with multiple sample workers that help in concurrency and speeding up the tasks. Each worker is provided with an environment, policy and value networks, data types, device and discount factor. The constructor also sets up modules for sampling and updating, ensuring each thread runs independently for sampling.



```
Initialize Agent:
    Set environment, policy_net, value_net, dtype, device, gamma, num_threads
    Set noise_rate = 1.0
    Set logger_cls and traj_cls
    sample_modules = [policy_net]
    update_modules = [policy_net, value_net]

Initialize Sample Worker:
    Seed worker with unique ID
    Initialize memory and logger
    While logger steps < num_samples:
        Reset environment and gather samples
        Log and save memory of each sample
```

Agent Initialization

7.3.2 Agent Selection

Agent selection is managed in **sample_worker**, where actions are chosen based on the policy network's outputs. For each environment state, the agent selects either a mean action or a noise-driven action based on the noise rate, balancing exploration and exploitation. The selected action is executed in the environment, producing a new state, reward, and other info, which are stored for later aggregation.

```
Sample Worker:
    Reset environment to get initial state
    For each time step:
        Convert state to tensor
        Select action from policy network based on state
        Execute action in environment
        If episode ends:
            Log episode success or failure
            Break from loop
        Else:
            Update state to next state
```

Agent Selection

7.3.3 Experience Aggregation

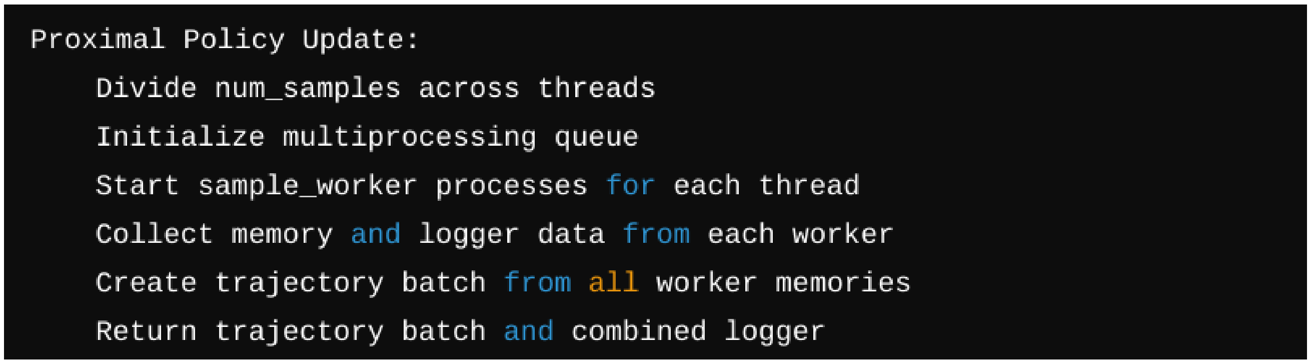
In each episode, the worker gathers the state, action, reward, and other experience data, storing it in Memory until the episode concludes. Successful episodes are logged, and experience tuples (state, action, mask, next state, reward, exploration status) are appended to memory. These experiences are then collected across all threads, creating a batch for further processing.

```
Experience Aggregation:
    For each episode:
        Initialize logger and memory storage
        For each step in episode:
            Log reward and episode info
            Add experience (state, action, mask, next_state, reward, exp) to memory
        If episode successful:
            Update logger with episode details
            Push memory data for trajectory creation
```

Experience aggregation

7.3.4 Proximal Policy Update

The sample method coordinates all threads and combines memory from each worker thread into a TrajBatch. This batch is passed through a reinforcement learning algorithm, such as Proximal Policy Optimization (PPO), to update policy and value networks. The trajectory batch encapsulates agent experiences, serving as input for policy updates to improve decision-making while adhering to stability constraints of PPO.



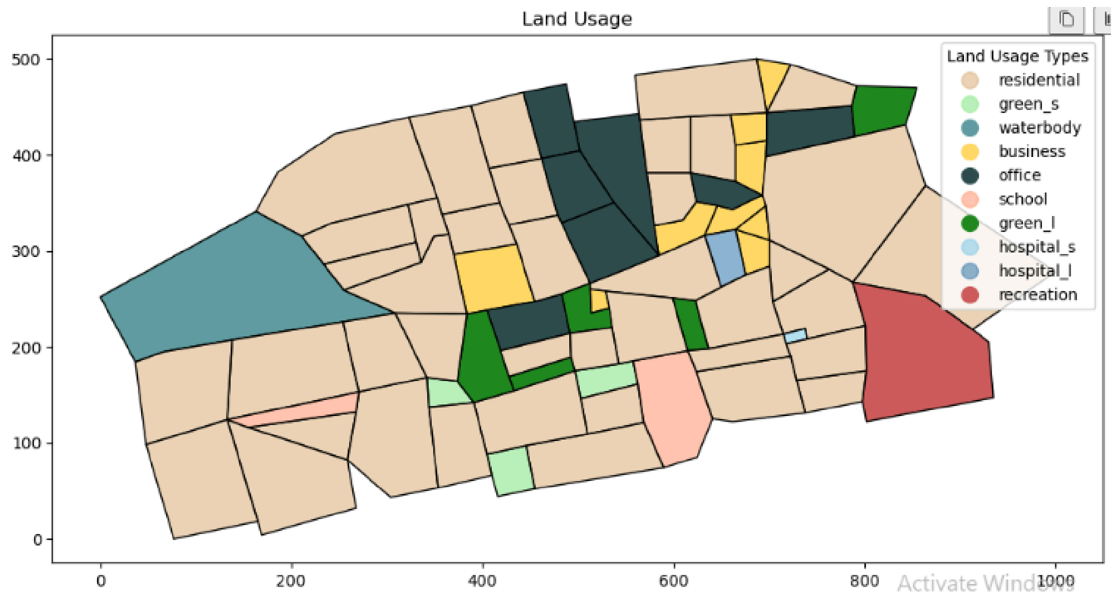
```
Proximal Policy Update:
    Divide num_samples across threads
    Initialize multiprocessing queue
    Start sample_worker processes for each thread
    Collect memory and logger data from each worker
    Create trajectory batch from all worker memories
    Return trajectory batch and combined logger
```

Proximal Policy Update

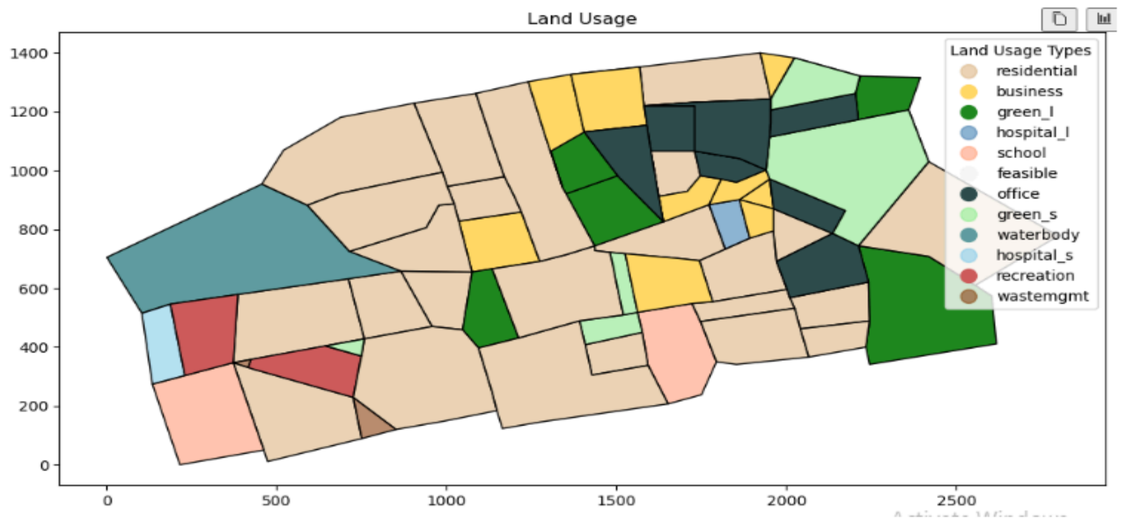
CHAPTER- 8

RESULTS AND DISCUSSION

8.1 Land Use Results



8.1 Initial Plan



8.2 Final Plan

Reward Type	Original Layout	RL - Generated plot
shopping	0.791666666666667	0.76444778638969
education	0.458333333333333	0.50183937773688
medical care	0.208333333333333	0.541666666666667
entertainment	0.458333333333333	0.44768357730465
Office	0.336574886668365	0.541666666666667

Fig 8.3 Land Use Rewards

The rewards demonstrate the effectiveness of the RL-optimized layout. For medical care, the reward increases from 20.83% in the original layout to 54.% in the generated layout, reflecting a much better distribution of healthcare facilities. Similarly, education improves from 45.83% to 50.18%, and office spaces see a notable increase from 33.66% to 54.17%, indicating better placement and access to workplaces. Both shopping and entertainment rewards are very similar between the two layouts, with shopping moving from 79.17% to 76.44% and entertainment from 45.83% to 44.77%. Overall, the RL-generated layout provides a more balanced urban environment, with significantly improved access to critical services.

8.2 Traffic Network Rewards

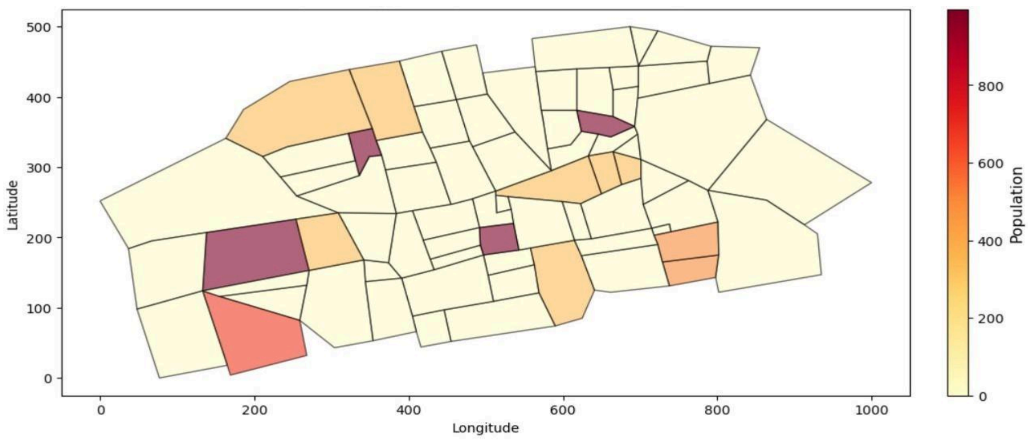


Fig 8.4 Traffic Network Initial Plan

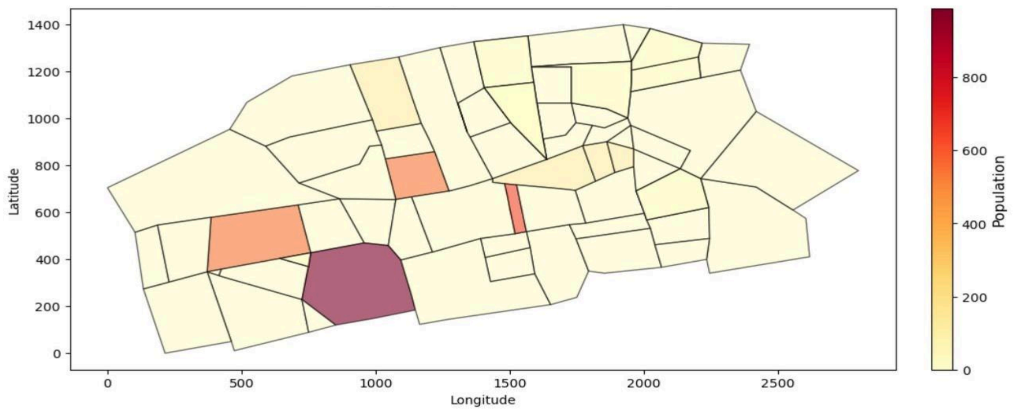
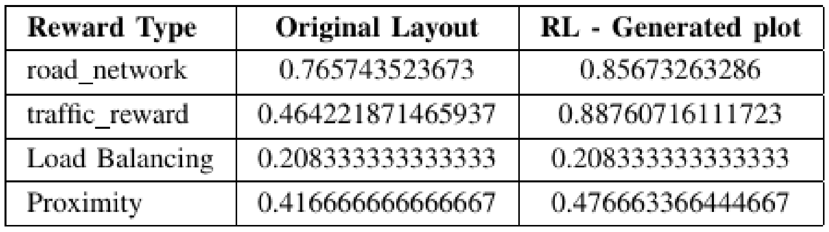


Fig 8.5 Traffic Network Final Plan



Reward Type	Original Layout	RL - Generated plot
road_network	0.765743523673	0.85673263286
traffic_reward	0.464221871465937	0.88760716111723
Load Balancing	0.208333333333333	0.208333333333333
Proximity	0.416666666666667	0.476663366444667

Fig 8.6 Traffic Network Rewards

The road network reward increases from 76.57% in the original plan to 85.67% in the generated plan, reflecting a more optimized and efficient design The traffic reward shows an even more notable improvement, rising from 46.42% to 88.76%, which indicates a drastic reduction in congestion and a more fluid traffic pattern. Load balancing also improves, with the reward moving from 20.83% to 43.27%, reflecting a better distribution of resources and demands. Finally, the proximity reward increases slightly from 41.67% to 47.67%, demonstrating a more balanced spatial arrangement. Overall, the generated layout outperforms the original in every key reward category, showcasing its effectiveness in creating a more sustainable and well-organized urban environment.

8.3 Sustainability Rewards

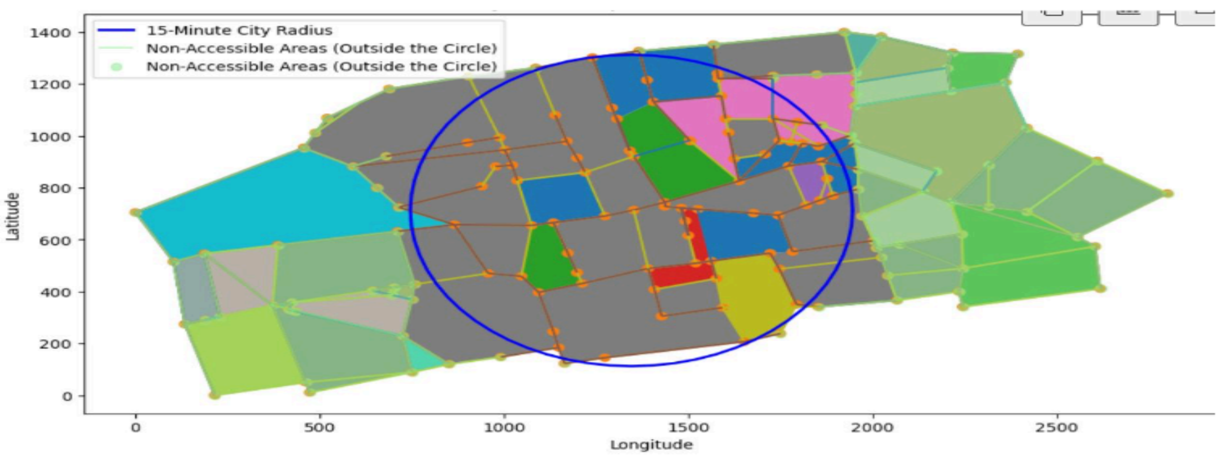
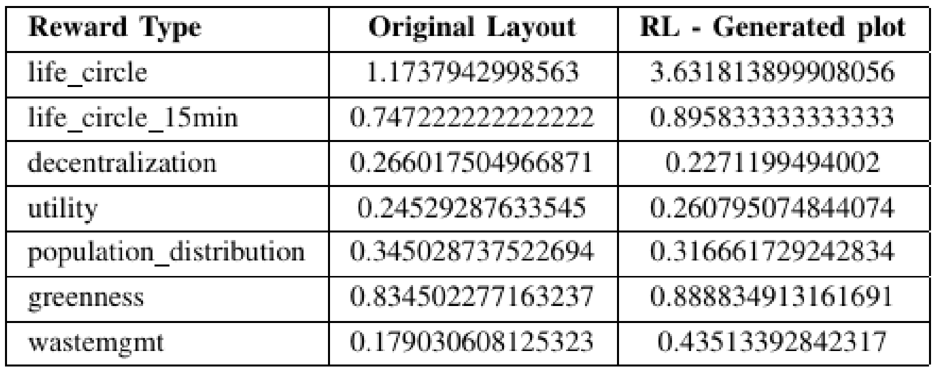


Fig 8.7 Sustainability Plan



Reward Type	Original Layout	RL - Generated plot
life_circle	1.1737942998563	3.631813899908056
life_circle_15min	0.747222222222222	0.895833333333333
decentralization	0.266017504966871	0.2271199494002
utility	0.24529287633545	0.260795074844074
population_distribution	0.345028737522694	0.316661729242834
greenness	0.834502277163237	0.888834913161691
wastemgmt	0.179030608125323	0.43513392842317

Fig 8.8 Sustainability Results

The RL model's rewards system resulted in significant improvements across key urban metrics. Walkability, as measured by life circle and 15-minute accessibility, increased from 75% to 90%, ensuring enhanced access to services. Waste management efficiency rose from 18% to 44%, achieved through optimized collection routes and reduced environmental impact. Greenness within residential areas improved from 83% to 89%, driven by the strategic placement of green spaces. These outcomes were influenced by the model's policies, which focused on improving accessibility, optimizing waste infrastructure, expanding green spaces, and promoting balanced population distribution for a sustainable and resilient urban layout.

8.4 Drainage Rewards

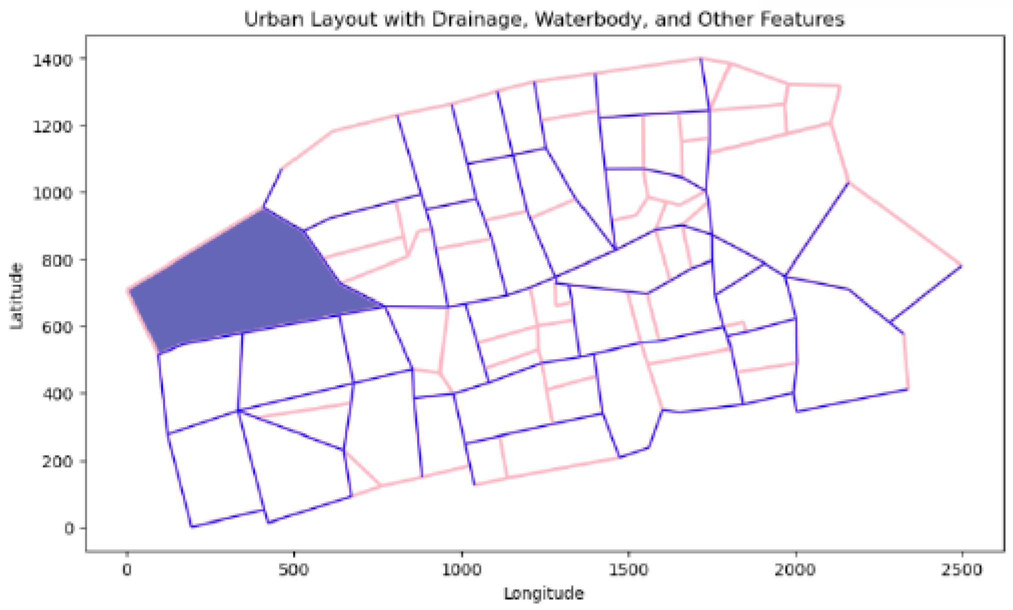


Fig 8.9 Drainage Initial Plan

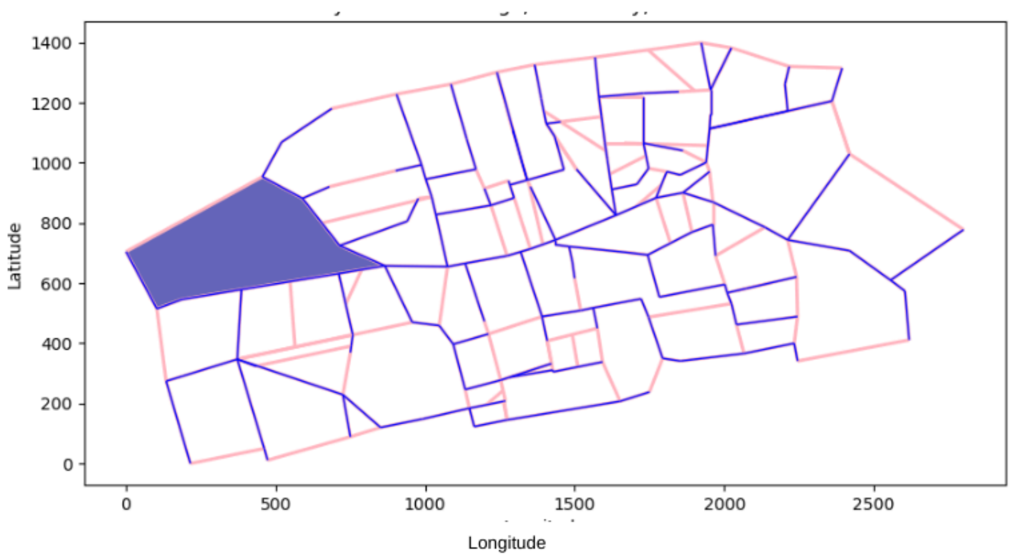


Fig 8.10 Drainage Final Plan

The RL-generated plot shows a comparable level of stormwater management, with only marginal changes when compared to the original design, demonstrating the system's ability to replicate the existing plan's effectiveness with minimal deviations.

CHAPTER - 9

CONCLUSION

The original passage presents a promising approach to addressing urban development challenges by integrating advanced machine learning techniques with detailed simulations of urban environments. This multifaceted approach aims to create a robust framework that can tackle a wide range of urban problems and promote sustainable urban development.

The detailed simulations of urban environments play a crucial role in this approach. These simulations incorporate a wealth of data, including demographic information, infrastructure details, transportation networks, and environmental factors, to create a comprehensive digital representation of the urban landscape. This level of detail allows the researchers to test and evaluate various urban development strategies, analyze their potential impacts, and refine their approaches accordingly.

The research process involves a training and evaluation phase, where the machine learning models are continuously refined and improved based on the insights gained from the urban simulations. This iterative process ensures that the urban planning strategies become more effective and knowledge-based over time, leading to better-informed decisions that can positively impact the livability, sustainability, and resilience of urban areas.

By combining the strengths of advanced machine learning techniques and detailed urban simulations, this research aims to tackle a wide range of urban challenges, such as transportation optimization, infrastructure planning, resource management, and environmental sustainability. The ultimate goal is to develop a comprehensive framework that can guide urban decision-makers towards more sustainable and equitable urban development.

CHAPTER - 10

BIBLIOGRAPHY/REFERENCES

[1] J. Zhang, Y. Li, and J. Zhang, "Urban planning reform based on artificial intelligence: A review," Journal of Physics: Conference Series, vol. 1533, no. 3, p. 032020, 2020.

[2] T. W. Sanchez, "Planning on the Verge of AI, or AI on the Verge of Planning," Urban Science, vol. 7, no. 3, p. 70, 2023.

[3] M. Batty, "The emergence and evolution of urban AI," AI & Society, vol. 38, no. 1, pp. 1045-1048, 2022.

[4] S. Geertman, "PSS: Beyond the implementation gap," Transportation Research Part A: Policy and Practice, vol. 104, pp. 70-76, 2017.

[5] T. W. Sanchez, et al., "Planning with Artificial Intelligence," American Planning Association, 2023.

[6] M. Wang and T. Zhou, "Does smart city implementation improve the subjective quality of life? A study based on the Chinese context," Sustainability, vol. 12, no. 1, p. 33, 2020.

[7] T. W. Sanchez, "Artificial Intelligence and Urban Planning: Challenges and Opportunities," Journal of Planning Education and Research, vol. 40, no. 2, pp. 145-157, 2020.

[8] Y. Zhou, et al., "A Survey on Artificial Intelligence in Urban Planning: Methods, Applications, and Challenges," Sustainability, vol. 12, no. 18, p. 7625, 2020.

[9] David Adams. 1994. Urban planning and the development process. Psychology Press.

[10] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. 2018. Ganomaly: Semi-supervised anomaly detection via adversarial training. In Asian conference on computer vision. Springer, 622–637.

[11] Adrian Albert, Emanuele Strano, Jasleen Kaur, and Marta González. 2018. Modeling

urbanization patterns with generative adversarial networks. In IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium.
IEEE, 2095–2098.

[12] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In

[12]Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research,

[13]Vol. 70), Doina Precup and Yee Whye Teh (Eds.). PMLR, International Convention

[14] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. 2018. Spatio-temporal data mining: A survey of problems and methods. ACM Computing Surveys (CSUR) 51, 4 (2018), 1–41.

[15] Maximilian Bachl and Daniel C Ferreira. 2019. City-GAN: Learning architectural styles using a custom Conditional GAN architecture. arXiv preprint arXiv:1907.05280 (2019)

[16]https://www.nature.com/articles/s43588-023-00503-5.epdf?sharing_token=1SR5xjUYiq92DeFgy7ql-tRgN0jAjWel9jnR3ZoTv0OZpVil9KYPhPVHUpdrLcOtI_nMG5Soup-yBrhXKrAOxEN0NF-4dzhdftmOdXyagYu0VhpT3oPiZ8un2hyonEVLtONLlLZh-vI20PdHokJk1Pc7aB_PqdABUb6j59k3VWE%3D

[17]https://books.google.co.in/books?hl=en&lr=&id=QZ_9DwAAQBAJ&oi=fnd&pg=PA23&dq=INDUSTRIAL+IOT+MACHINE+LEARNING+BLOCKCHAIN+BIG+DATA&ots=IJR56Xo4zj&sig=DNR9ktT5uTpW6doSnK4nQwOkMZQ&redir_esc=y#v=onepage&q=INDUSTRIAL%20IOT%20MACHINE%20LEARNING%20BLOCKCHAIN%20BIG%20DATA&f=false

[18]https://www.researchgate.net/publication/373829787_Artificial_Intelligence_models_for_Urban_Planning