CHAPTER 05: DESIGN

5.1. Chapter Overview

In this design chapter, we unfurl the blueprint of our software ecosystem, spotlighting the elegant prowess of the 3-tier architecture. The selection of design paradigm for the "Network Optimizer" is discussed with reasons and related design diagrams have been used to achieve the expected design goals. Finally the algorithm design and UI wireframes are presented.

5.2. Design goals

Design Goal	Description
Correctness	The primary goal is to create an optimizer model that predicts future traffic accurately in a way the loss between the predicted and the actual value is minimized. Inaccurate predictions about traffic can misguide resource optimization decisions..
Performance	Ability of the optimizer model to carry out the task efficiently and effectively is crucial. For the proactive network optimization, a model delivering results with the least possible latency would affect in achieving the ultimate goal of the prediction task.
Adaptability	The system should be designed in such a way, that it can adapt to changes in the network data with the dynamically fluctuating network conditions. By focusing on adaptability, the network resource optimizer can effectively navigate the complexities of dynamic network environments, ensuring optimal resource utilization and performance in the face of changing conditions.
Scalability	In production environment, the model should be capable of handling increased network data volumes and variety of network data

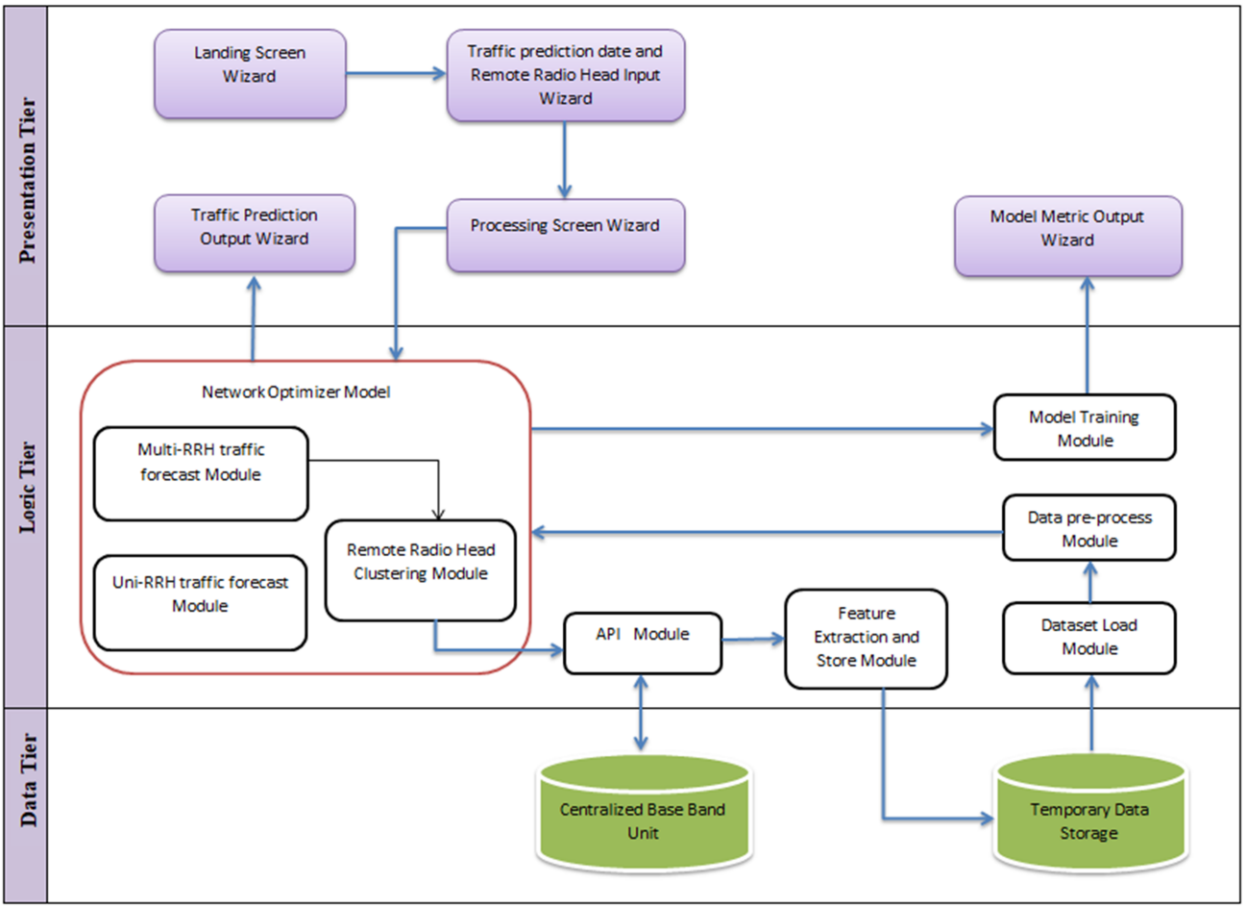
Resource Efficiency	In the midst of a complex network architecture, designing an optimizer model considering the factors such as memory usage of the model, computation speed, power consumption etc. is important.

Design goals of the proposed system

5.3 System Architecture Design

5.3.1 System Architecture Diagram

An architecture diagram serves as the visual backbone of a system's design, offering a comprehensive overview of its structure, components, and interrelationships. The deliberate layering of presentation, business logic, and data storage is not a mere structural hierarchy; it is an orchestration of efficiency, encapsulation, and modularity.



Three Tiered Architecture (self-composed)

5.3.2 Discussion of System Architecture Tiers

5.3.2.1 Presentation Layer

This is the outermost layer that interacts directly with users. It focuses on presenting information to users in a meaningful way and capturing user input.

- Landing Screen Wizard
- Traffic prediction date and Remote Radio Head Input Wizard
- Processing Screen Wizard
- Traffic Prediction Output Wizard

5.3.2.2 Business Logic Layer

The Business Logic Layer emerges as the layer that business rules, processes, and system functionality converge. Processing user input, system login execution and system activity orchestration can be summarized as the main functionality of this layer

- Multi-RRH traffic forecast Module
- Uni-RRH traffic forecast Module
- Remote Radio Head Clustering Module
- API Module
- Feature Extraction and Store Module
- Dataset Load Module
- Data pre-process Module
- Model Training Module

5.3.2.3 Data Storage Layer

The Data Storage Layer stands as the architectural backbone and is responsible for managing and storing data. It serves as the repository where information is stored, retrieved, and updated.

- Centralized Base Band Unit
- Temporary Data Storage

5.4 System Design

5.4.1 Design Paradigm

In the vast landscape of software development, design paradigms act as compass points, providing direction amid the complexities of designing scalable, maintainable, and adaptable solutions. They provide a conceptual framework consisting of set of principles, patterns, and best practices that guide developers to make design decisions, organize their code, and create architectures that align with specific goals and requirements.

5.4.1.1 Selection of design paradigm

The choice of design paradigm will be determined by factors such as problem domain, specific project requirements, long term goals of the project etc.

The two most commonly used design paradigms are OOAD (Object Oriented Analysis and Design) and SSADM (Structured Systems Analysis and Design Method). Out of the above two SSADM was chosen as the choice of design paradigm mainly for below reasons.

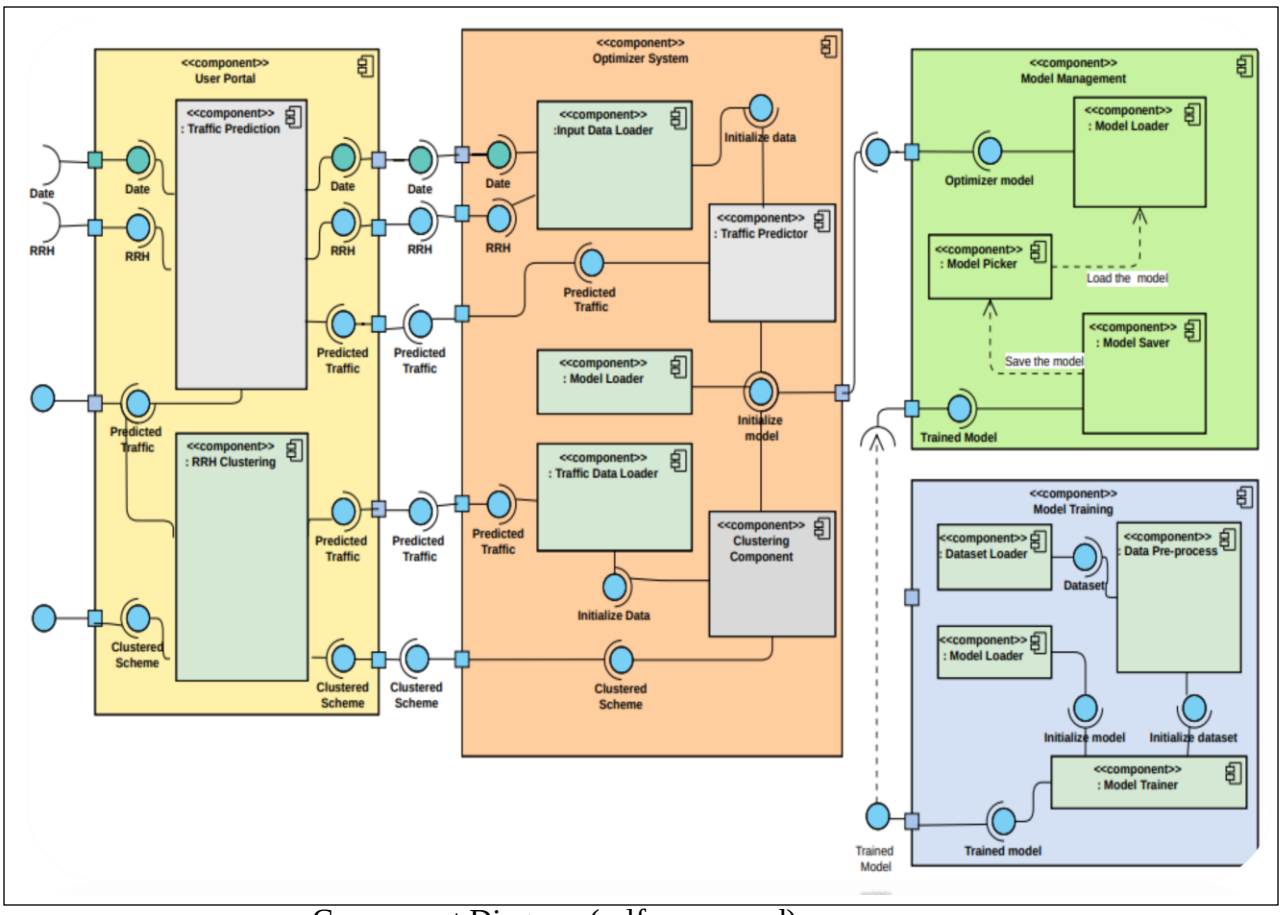
- The research component is mainly inclined towards Data Science which doesn't gain a noticeable benefit by using Object Oriented approaches.
- Ease of implementation of an MVP (Minimum Viable Product) for demonstrating the prototype
- SSADM tends to be more detailed and focused on specific processes and data flows within the system.

5.4.2 Design Diagrams

Design diagrams serve as blueprints for the architecture and functionality of software systems, offering a visual roadmap that guides developers through the complexities of implementation. They serve as communication tools fostering collaboration among stakeholders.

5.4.2.1 Component Diagram

Below diagram provides a high-level view of the "Network Resource Optimizer" system architecture, depicting how different components interact with each other to achieve the systems functionality.



Component Diagram (self-composed)

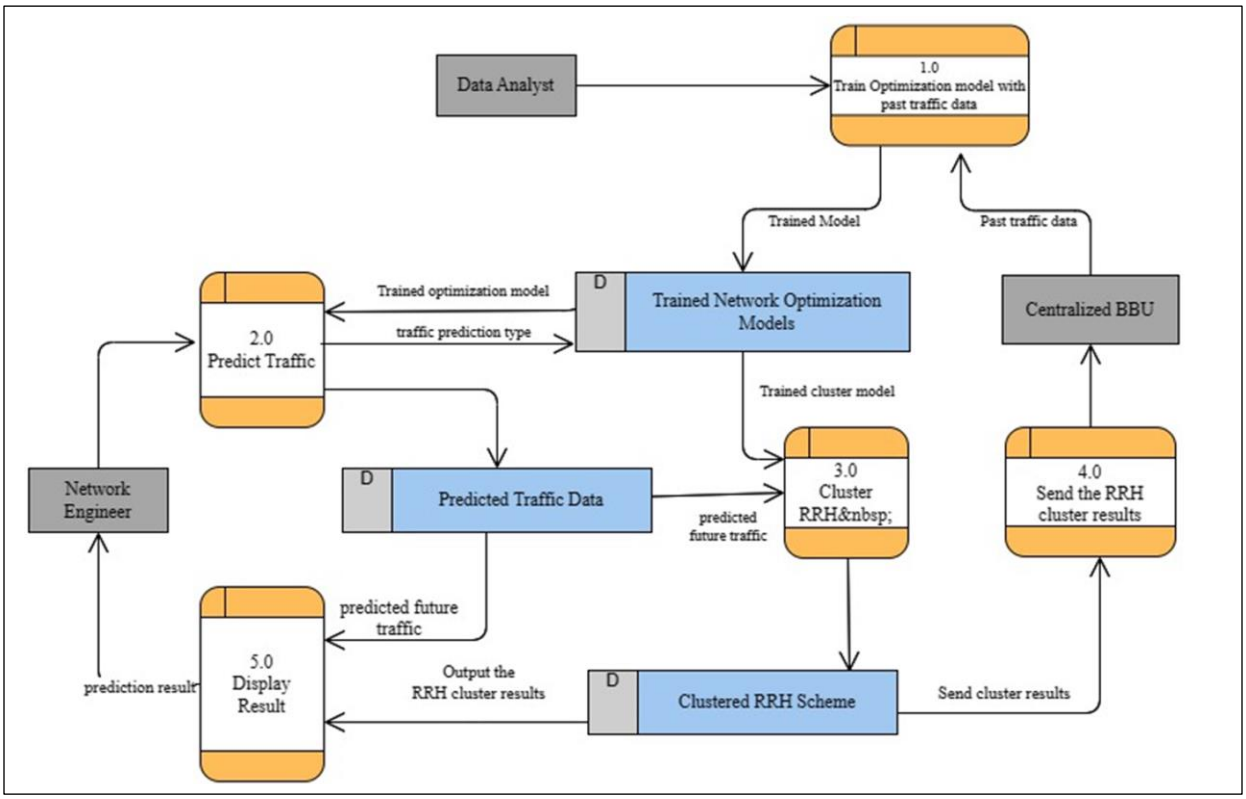
5.4.2.2 Data Flow Diagram

Below diagram visualizes the Level1 DFD of the proposed "Network Resource Optimizer" system which elaborates the level 0 DFD that was described during the SRS chapter 04. This illustration delves deeper into the internal structure and processing logic of the system by decomposing the system into sub-processes or modules.

Level 1 DFD

This consists of 3 main processes mentioned below.

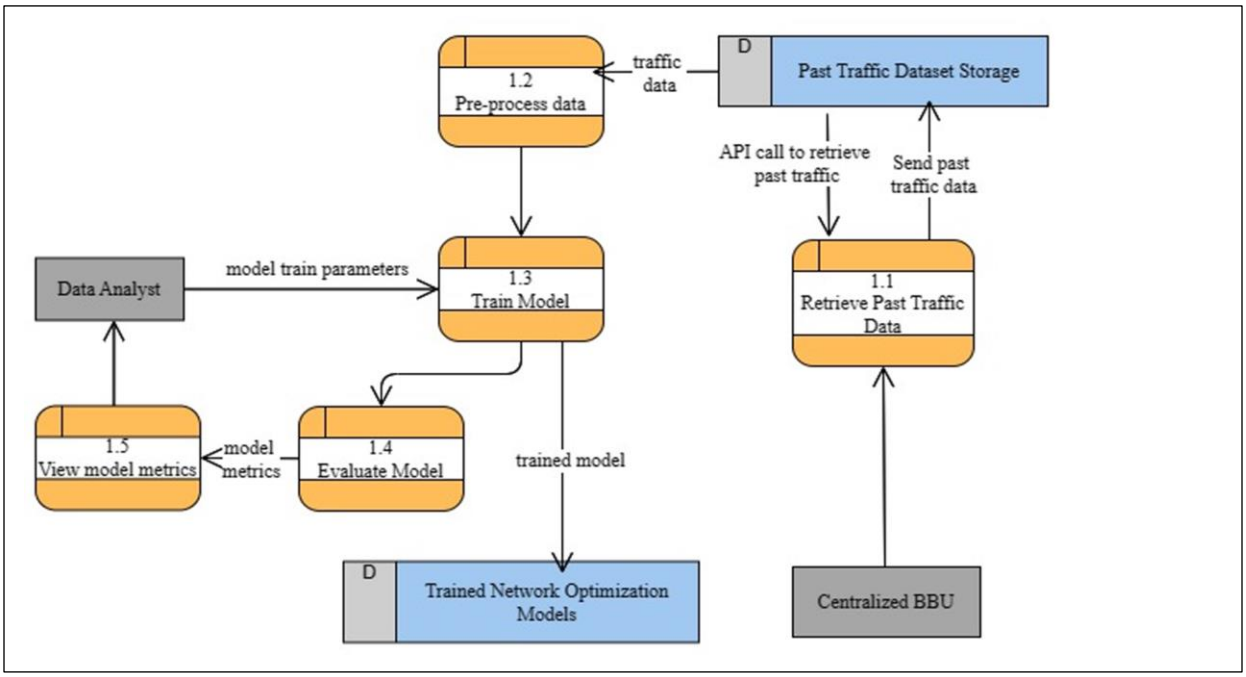
1. Train Optimization models with past traffic data
2. Predict Traffic
3. Cluster RRH



Data Flow Diagram-Level 1(self-composed)

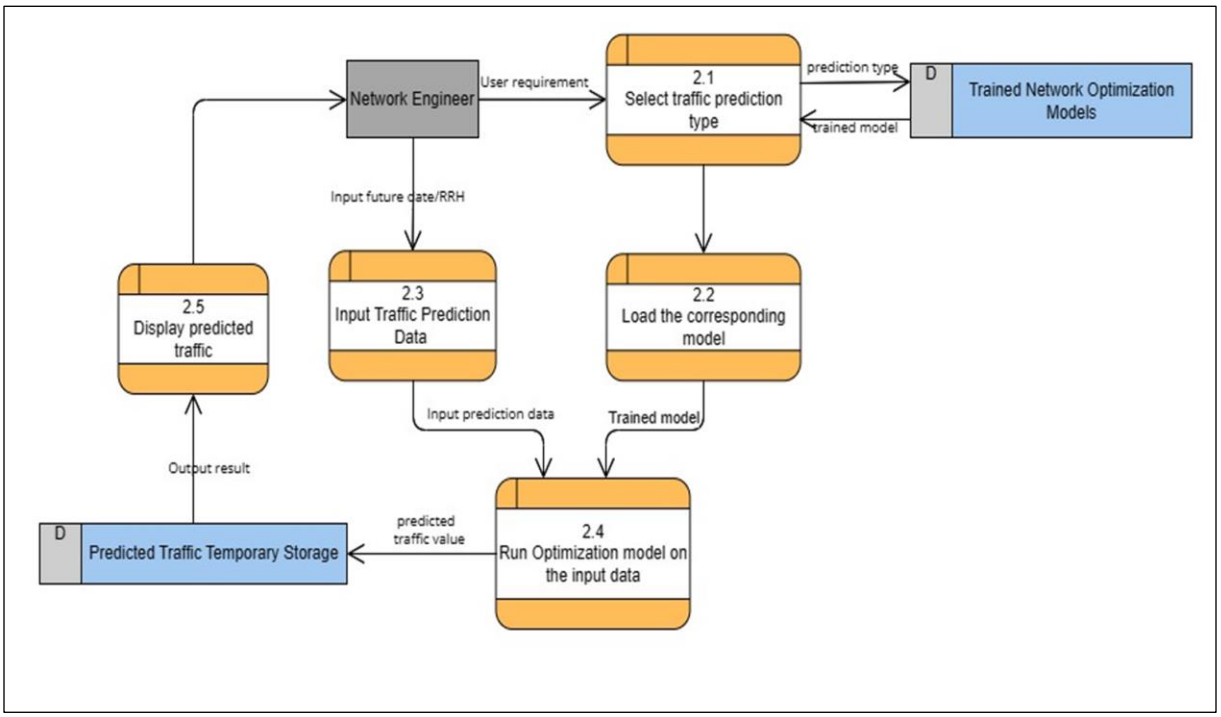
Level 2 DFD

More detailed breakdown of the main process in DFD 1 diagram above are illustrated in Level 2 DFD diagrams.



Optimization Model Trainer - Data Flow Diagram-Level 2(self-composed)

Above Level 2 DFD diagram depicts the training process of the hybrid model elaborated in detail using past traffic data of RRHs retrieved from the centralized BBU.



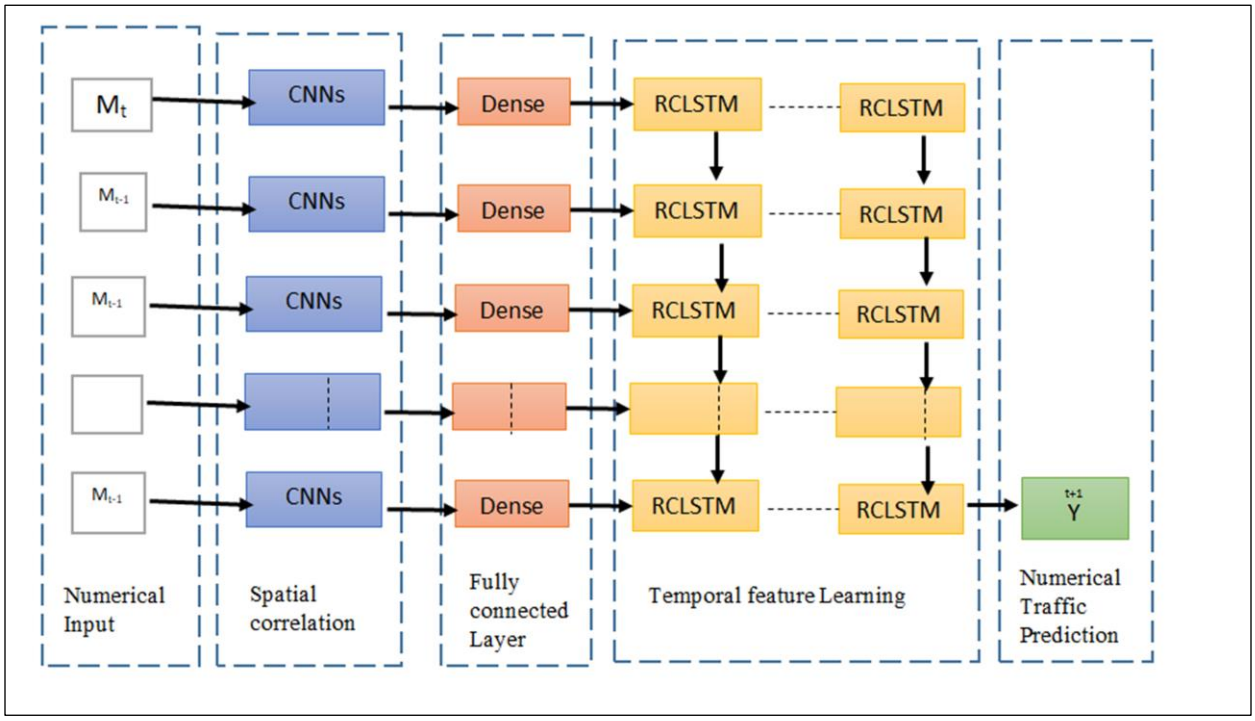
Traffic Predictor - Data Flow Diagram-Level 2(self-composed)

Above Level 02 DFD diagram depicts the core module of the system which is the traffic predictor process using the trained ML model.

5.4.3 Model Architecture Design

Design of the algorithm is the significant component of the system. The process of algorithm design involves crafting correct as well as optimized solutions in terms of time, space and other resources. A hybrid model capable of capturing both temporal dependency and spatial correlation among base stations to predict traffic with a higher accuracy and a lowered computational complexity being the main novelty of the research, a combination of the CNN architecture and RCLSTM (Random Connectivity Long Short Term Memory) architecture have been used.

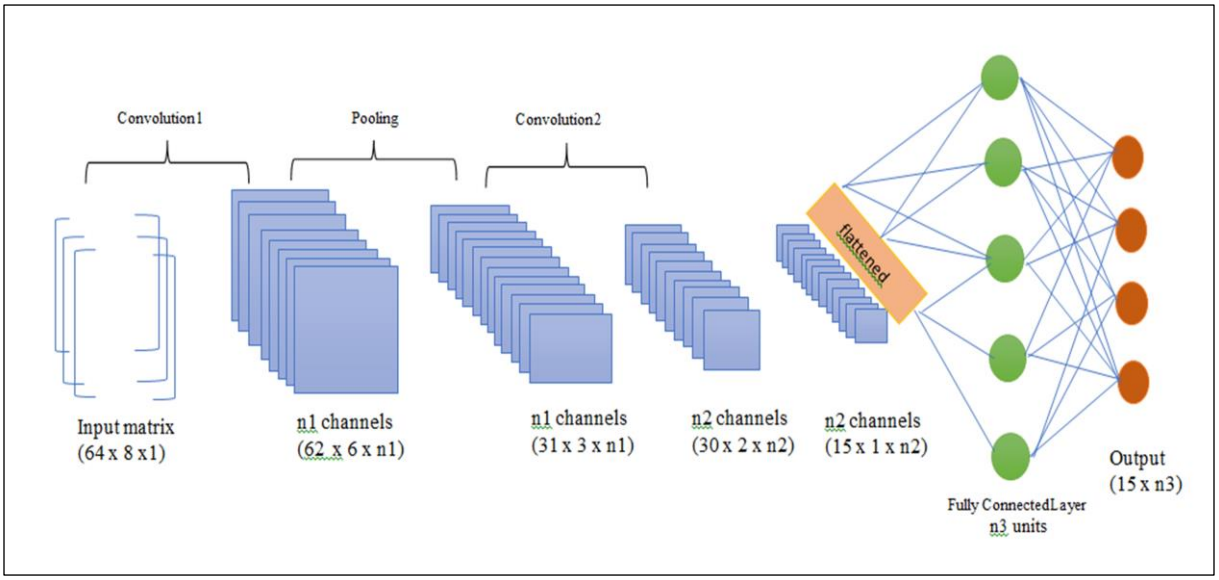
5.4.3.1 Framework of the proposed algorithm



Proposed hybrid algorithm framework (self-composed)

The framework is composed of a combination of two algorithms intermediate with a dense layer. First numerical traffic input is passed to the CNN model which is powerful at capturing spatial features in the input traffic data and then passed to RCLSTM model via a Dense layer. The purpose of a dense layer in a neural network is to provide full connectivity between its input and output. Each model architecture is described in detail as below.

5.4.3.2 CNN architecture



CNN model architecture

Step01: Converting traffic data to traffic matrices as images

- First network traffic of different base stations is transformed into interactive traffic matrices, each of whose element represents traffic value at each base station
- Secondly, the values of matrices are normalized and scaled to (0,1)
- Finally matrices are served as images

Step 02: Convolution and Pooling

CNN network was formed by stacking together two convolution layers with each layer having 32 convolution filters, all of which can extract useful spatial features of the input traffic data and produce set of feature maps. Filters denoted by $\theta = (\theta_1, \ldots\ldots, \theta_k)$, with the given input vector $X \in R^n$, the i-th entry of the output y resulting from the transformation of x by filter θ

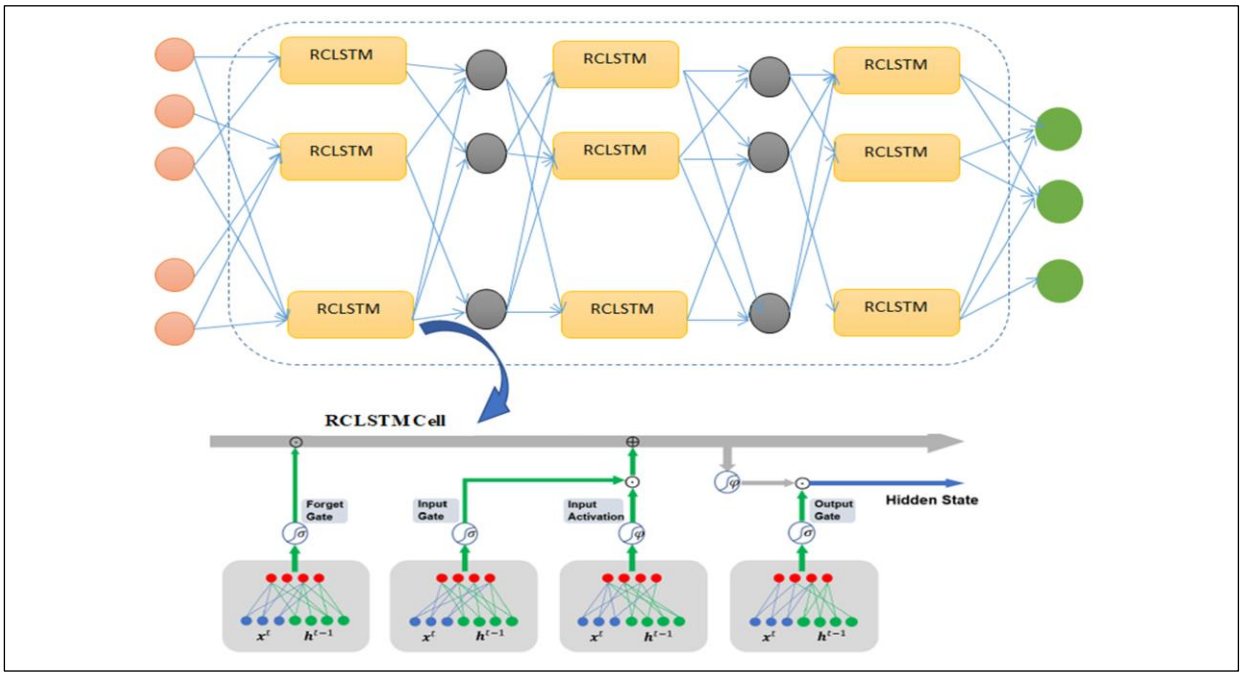
$$y_i = f\left(\sum_{k=1}^{w}(\theta_k \; X_{(i+k)})\right)$$

Pooling layers are used to down sample and aggregate the data in the neighborhood region, so as to reduce the scale and extract the most salient features.

Step 03: Fully Connected Layer

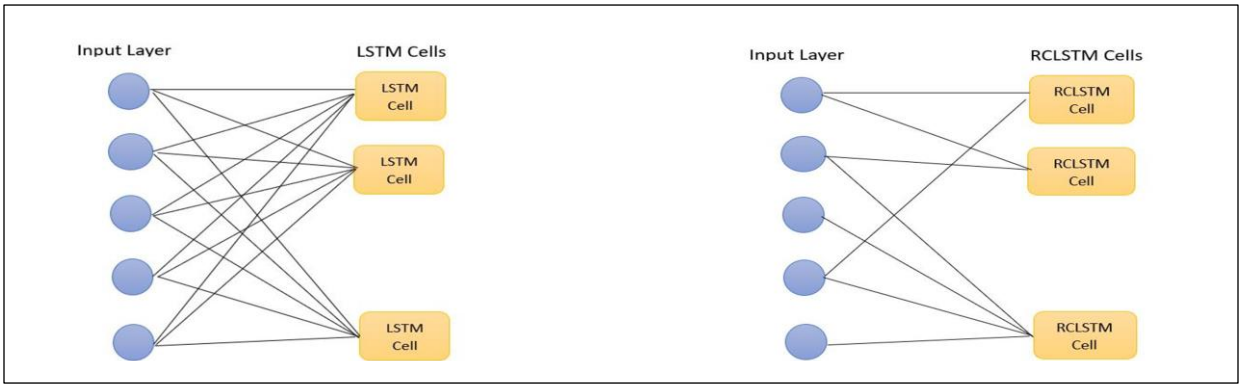
Final feature map is then flattened into a high dimensional vector that can be fed to the next algorithm. Each neuron in the fully connected layer consider information from all neurons in the previous layer, enabling it to capture intricate patterns and relationships in the data.

5.4.3.3 RCLSTM architecture



RCLSTM model architecture

Above diagram illustrates the basic RCLSTM memory block arrangements along with its inputs, outputs and the information flow. This maintains random connectivity in the formation of neural connections in LSTM memory block whereas the conventional LSTM exhibits full neural connections in each memory block.



Graph representation of LSTM Vs RCLSTM memory block

Connectivity in conventional LSTM memory block can be represented as $G(V, P_g[j \to k])$ where V denotes the set of neurons $V = \{v_{glk} | g \in \{i, o, f, z\}, l \in \{1,2\}, 1 \leq k \leq n_1\}$ where

v_{glk} => k^{th} neuron at layer l in g network

g => one of three gates or input activation

n_1 => number of neurons at layer l

$P_g[j \to k]$ represents the probability that neural connection occurs between neuron v_{g1j} and v_{g2k}. Using above definitions, neural connections within LSTM are randomly inserted among the set of neurons independently. The strategy to establish neural connections in RCLSTM is as below.

$$e_g [j \to k] \text{ exists where } P_g[j \to k] \geq T$$

Here $e_g [j \to k]$ denotes the neural connection from node j to node k, in the network of g and T is the threshold which indicates the sparsity of neural connectivity in RCLSTM.

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f)$$

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i)$$

$$C_t^{\wedge} = \tanh(W_c[h_{t-1}, X_t] + b_c)$$

$$C_t = f_t \times C_{t-1} + i_t \times C_t^{\wedge}$$

$$O_t = \sigma(W_o[h_{t-1}, X_t] + b_o)$$

$$h_t = O_t * \tanh(C_t)$$

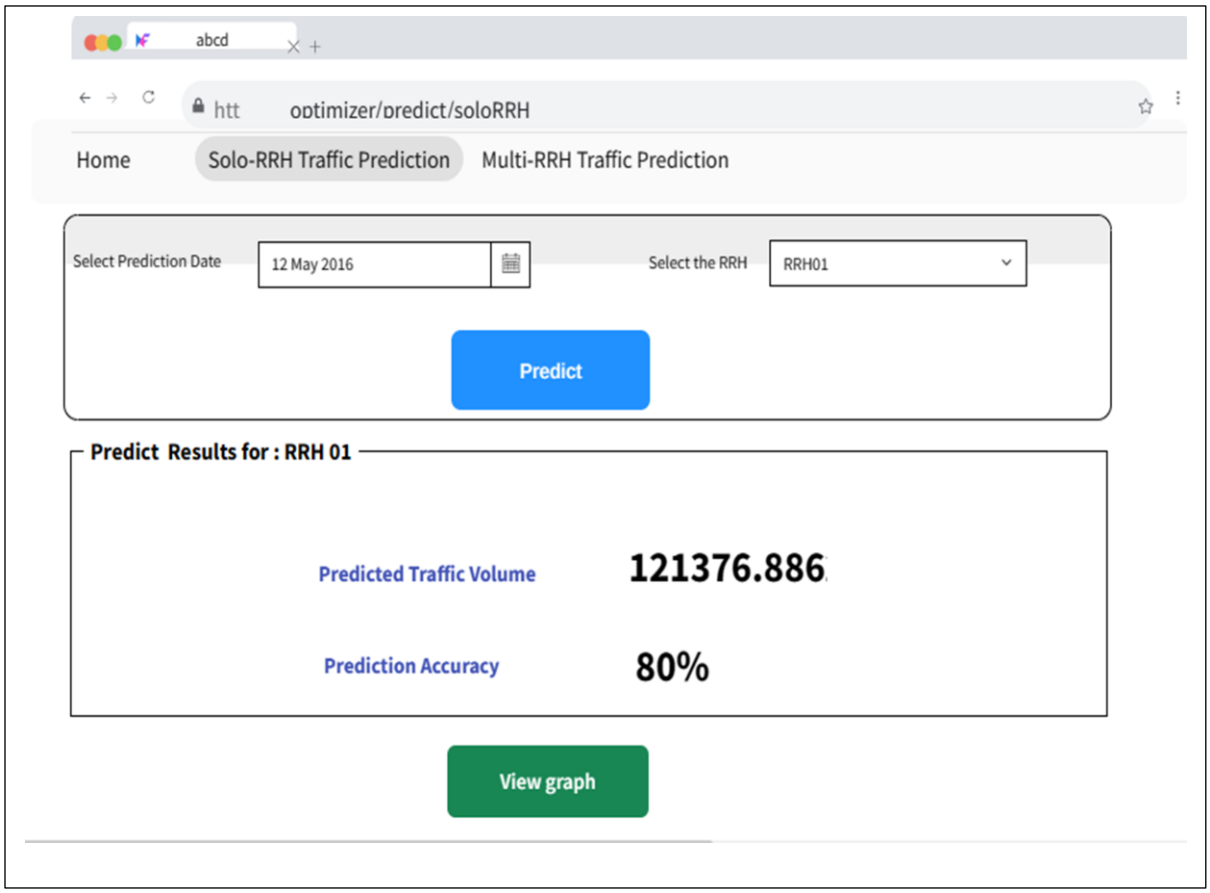
Suppose the input vector at time t is x^t, the hidden state vector at time t is h^t, the memory cell at time t is c^t. The i, o and f denote the input gate, output gate, and forget gate respectively and z denotes the input activation. $\sigma(.)$ usually takes the sigmoid function and $\varphi(.)$ is usually the hyperbolic tangent function.

5.4.4 UI Design Diagrams

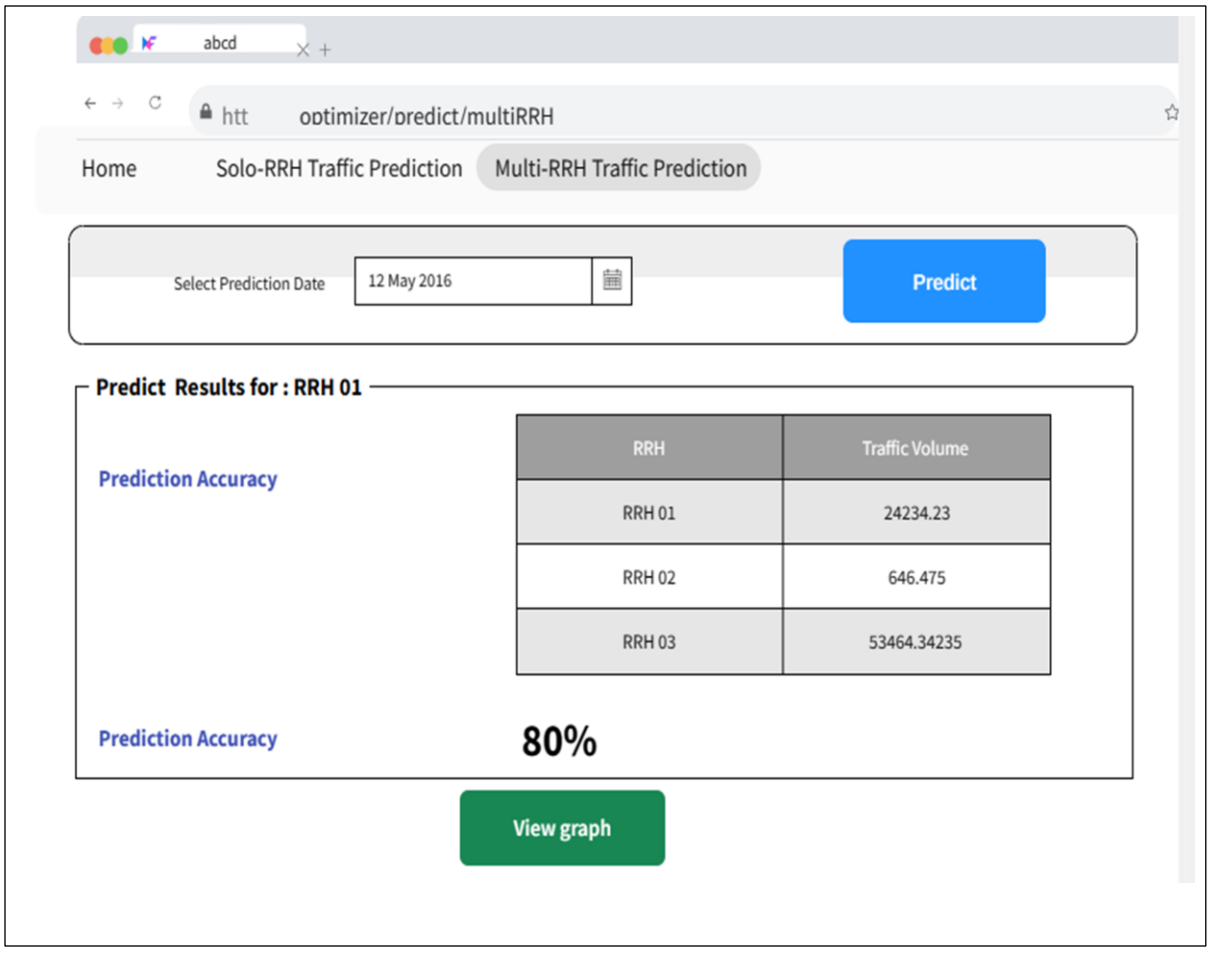
Below illustrations depict the planned UI designs of the two core features of the "Network Resource Optimizer" system stated below.

1. SoloRRH traffic prediction where traffic of the selected base station on the selected date is predicted
2. MultiRRH traffic prediction where traffic of all base stations on the selected date is predicted

Mock UI designs of the other screens have been placed in the Appendix.



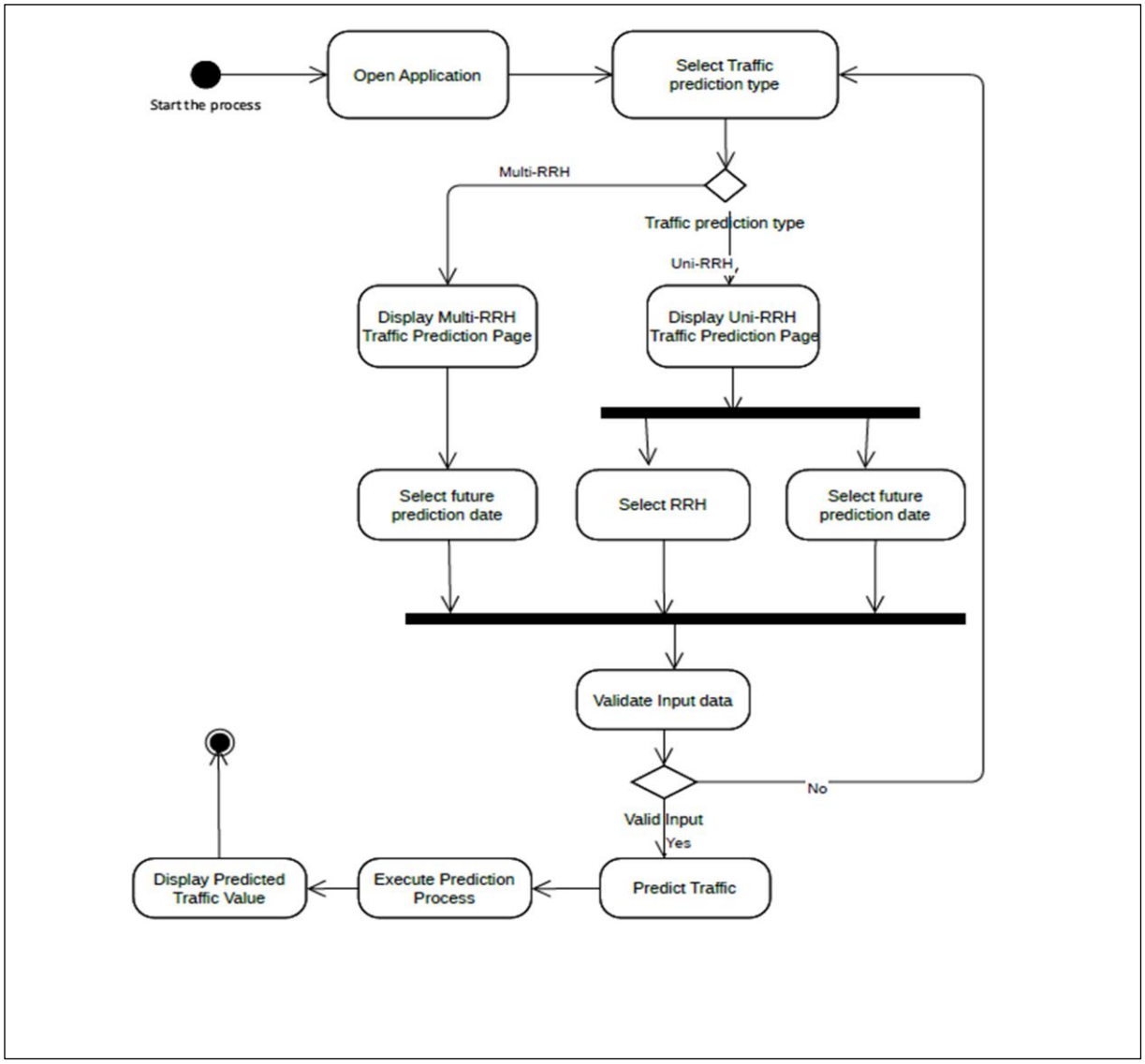
Single RRH Traffic Prediction Wizard



Multiple RRH Traffic Prediction Wizard

5.4.5 System Process Flow Diagram

The process flow diagram illustrates the flow of data, sequence of activities and decision points within the "Network Resource Optimizer". This mainly offers a visual narrative to the use of structured programming in the journey from system input to output by reaching decisions in the system logic tier.



System Activity Diagram (self-composed)

5.5 Chapter Summary

This chapter has commenced with the design goals of the proposed "Network Resource Optimizer" system with their proper justifications. Then the 3 tier system architecture diagram and other required diagrams were presented, in order to provide an overview of the system context and other interactions with external entities. Lastly, the initial UI designs of the proposed system have been presented.