network

MDPI

Article

Integrating Reinforcement Learning and LLM with Self-Optimization Network System

Xing Xu [1], Jianbin Zhao [1], Yu Zhang [2] and Rongpeng Li [2,*]

[1] Information and Communication Branch of State Grid Hebei Electric Power Co., Ltd., Shijiazhuang 050051, China
[2] College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China
* Correspondence: lirongpeng@zju.edu.cn

Abstract

The rapid expansion of communication networks and increasingly complex service demands have presented significant challenges to the intelligent management of network resources. To address these challenges, we have proposed a network self-optimization framework integrating the predictive capabilities of the Large Language Model (LLM) with the decision-making capabilities of multi-agent Reinforcement Learning (RL). Specifically, historical network traffic data are converted into structured inputs to forecast future traffic patterns using a GPT-2-based prediction module. Concurrently, a Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm leverages real-time sensor data—including link delay and packet loss rates collected by embedded network sensors—to dynamically optimize bandwidth allocation. This sensor-driven mechanism enables the system to perform real-time optimization of bandwidth allocation, ensuring accurate monitoring and proactive resource scheduling. We evaluate our framework in a heterogeneous network simulated using Mininet under diverse traffic scenarios. Experimental results show that the proposed method significantly reduces network latency and packet loss, as well as improves robustness and resource utilization, highlighting the effectiveness of integrating sensor-driven RL optimization with predictive insights from LLMs.

Keywords: network self-optimization; adaptive routing; reinforcement learning; large language model

check for updates

Academic Editor: Christos Bouras

Received: 11 August 2025
Revised: 12 September 2025
Accepted: 15 September 2025
Published: 16 September 2025

Citation: Xu, X.; Zhao, J.; Zhang, Y.; Li, R. Integrating Reinforcement Learning and LLM with Self-Optimization Network System. *Network* **2025**, *5*, 39. https://doi.org/10.3390/network5030039

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the context of the rapid growth of data traffic and increasingly diversified service requirements, modern communication networks are facing unprecedented challenges. Efficiently utilizing limited network resources to achieve high-performance network optimization has become a core topic currently. As shown in Figure 1, with the rapid development of 5G, edge computing, and wireless network technologies, the complexity and dynamics of networks are increasing. However, based on fixed rules and static models, traditional methods, such as the improved genetic algorithm [1] and multiple access communication control algorithm [2], exhibit significant limitations in handling complex real-time network scenarios. Especially, their lack of adaptability and responsiveness makes them inadequate for dynamic and rapidly changing environments. Therefore, an intelligent optimization method that can adapt to the rapidly changing network environment has become an important issue. In this regard, integrating real-time sensor data into the optimization process becomes crucial. By embedding sensors within the network infrastructure to monitor

link conditions, delay, and packet loss in real time, the system can feed these dynamic observations into reinforcement learning agents. This sensor-driven approach empowers the optimization framework to respond swiftly to network fluctuations, supporting low-latency, high-efficiency, and adaptive resource management strategies.

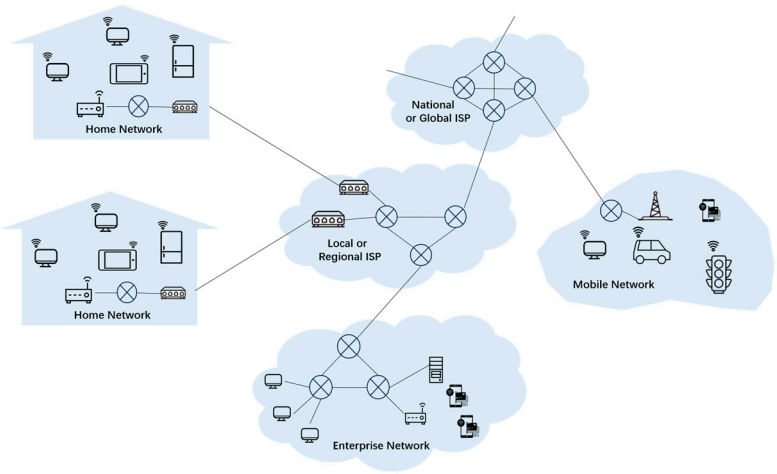

Figure 1. The structure of modern communication networks.

Notably, as a machine learning method with self-learning and autonomous decision-making ability, Reinforcement Learning (RL) [3,4] has been widely used in communication network optimization. Unlike traditional optimization methods, RL does not rely on predefined rules but continuously optimizes system performance through constant interaction with the environment and adjusting decisions based on real-time feedback. In the optimization of communication networks, RL can flexibly cope with complex problems (e.g., traffic management, bandwidth allocation, and path selection), especially when resources are limited and the environment changes dynamically [5]. This capability enables the autonomous optimization of resource allocation and enhancement of overall network performance, without relying on defined complex modeling. Meanwhile, Deep Learning (DL)-based methods have become popular in communication network optimization recently due to their powerful data processing and feature learning capabilities [6]. By combining deep neural networks with RL, Deep Reinforcement Learning (DRL) can handle more complex network state and decision-making problems, and it achieves remarkable results in bandwidth allocation, network traffic prediction, and load balancing [7,8].

Moreover, Large Language Models (LLMs) have shown great potential in processing time series data, prediction, and decision optimization in recent years [9,10]. LLM can effectively process a large amount of network data so as to provide more accurate network state prediction and optimization decisions for RL models, especially in the prediction of network traffic changes and communication patterns. In fact, LLMs are increasingly regarded by the research community as a key component of future communication systems.

On the other hand, although DRL has demonstrated strong adaptability in dynamic network environments, it primarily depends on current and short-term historical observations, which limit its ability to anticipate long-range traffic dynamics. By contrast, LLMs possess advanced sequence modeling capabilities that enable accurate forecasting of future traffic trends. Integrating these predictions with RL endows the agent with enhanced foresight [11], thereby shifting the paradigm from reactive adaptation to proactive resource scheduling and effectively mitigating traffic bursts and fluctuations that conventional DRL methods struggle to handle.

Therefore, we propose a self-optimization approach for communication networks by integrating RL with LLM, aiming to enhance the intelligence and self-adaptability of the

network, especially in bandwidth allocation and load balancing tasks. This integration enables analysis of intelligent network state and adaptive decision-making in dynamic environments, thereby improving network performance and service quality.

We summarize the major contribution of our work as follows:

- We propose a collaborative network optimization structure of LLM and RL, which integrates LLM's predictive abilities with RL's decision-making. LLMs forecast future traffic patterns using historical and contextual data, enabling proactive, informed optimization beyond relying solely on current network states.
- To manage network resources effectively, we model each communication link as an autonomous intelligent agent. These agents function in a cooperative multi-agent environment, where the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm facilitates continuous action control and policy learning.
- We validate the framework through extensive experiments on the Mininet simulation platform. Results demonstrate the superiority of LLM-enhanced RL decision-making over conventional RL baselines and further confirm our approach's practicality and scalability for future communication and real-world-like scenarios.

All source code has been released at https://github.com/zju-Collector/network-optimization (accessed on 14 September 2025).

2. Related Works

In this section, a literature review on the latest technologies for network system optimization is presented. This study covers bandwidth allocation and network slicing techniques based on deep reinforcement learning, as well as LLM-based traffic prediction techniques.

2.1. Deep Reinforcement Learning for Bandwidth Allocation and Network Slicing Resource Management

The dynamic optimization of network resources can be formally modeled as a sequential decision-making problem. The foundational framework for such problems in reinforcement learning is the Markov Decision Process (MDP), a mathematical construct for modeling decision-making in environments where outcomes are partly random and partly under the control of a decision-making agent [12]. An MDP is characterized by a set of states, actions, transition probabilities, and a reward function, where the agent's objective is to learn an optimal policy that maximizes a cumulative reward signal. In complex systems, such as communication networks, where multiple entities interact and make decisions concurrently, this model is often extended to multi-agent frameworks, such as the Decentralized Partially Observable Markov Decision Process (Dec-POMDP), which addresses cooperative decision-making under conditions of uncertainty and incomplete information [13]. To address the high-dimensional state and action spaces inherent in these formulations, DRL has emerged as a particularly powerful paradigm, leveraging the power of deep neural networks for value function approximation and policy learning. Consequently, DRL has become an effective solution for bandwidth allocation and resource optimization in complex networks [14]. Do et al. [15] proposed a DRL-based dynamic bandwidth allocation scheme using deep Q-networks to improve resource utilization in wireless networks. Attiah et al. [16] applied DRL to cellular load balancing, achieving better system throughput and stability. Abu-Ein et al. [17] proposed a bandwidth allocation method based on RL, which addresses challenges such as service quality, fairness, security, and privacy. Furthermore, as a key architecture in 5G and future networks, network slicing addresses differentiated service requirements by isolating resources. To manage slice resources intelligently, Li et al. [18] introduced a DRL-based allocation method where

agents automatically learn resource adjustment strategies across virtual networks. This ensures fair and efficient bandwidth distribution within slices while maintaining high QoS. Liu et al. [19] proposed a constrained RL-based approach for network slicing, using the adaptive interior-point policy optimization and policy safety layer methods to deal with cumulative and instantaneous constraints. In addition, Wang et al. [20] developed a novel machine learning-based scheme for dynamic resource scheduling for network slicing, aiming to achieve automatic and efficient resource optimization and End-to-End (E2E) service reliability.

In summary, DRL enables real-time perception and adaptive bandwidth control in multi-user environments, substantially outperforming static strategies in dynamic network scenarios. It also demonstrates strong potential for orchestrating complex slice-level resource scheduling under heterogeneous and evolving traffic demands. However, the current research continues to face significant challenges in achieving global optimality, ensuring real-time performance, and maintaining scalability with respect to key issues such as resource allocation, network slicing, and load balancing within dynamic, heterogeneous, and multi-objective network environments. Therefore, the development of more efficient and intelligent solutions remains a critical necessity.

2.2. LLM-Based Traffic Prediction for Network Optimization

LLMs have recently been adopted for traffic prediction and network state modeling tasks, thanks to their strong temporal modeling capacity. Shokouhi et al. [21] proposed a multi-timespan traffic prediction method based on LLMs, while Yang et al. [22] leveraged retrieval-augmented generation to enhance forecasting accuracy. Guo et al. [23] proposed a traffic flow prediction model based on LLM to generate explainable traffic predictions. These models can anticipate network conditions and improve DRL decision-making by supplying an accurate traffic forecast. Nonetheless, the model still exhibits limitations in terms of generalization and adaptability when confronted with frequent and substantial traffic fluctuations in wireless networks, especially under extreme or previously unencountered conditions. To address the aforementioned challenges, our paper designs complex traffic patterns through data synthesis and the construction of a network architecture using Mininet.

While prior studies have extensively explored DRL for network resource optimization and preliminarily introduced LLMs into traffic modeling, systematic integration of LLMs and DRL for dynamic, adaptive optimization in communication networks remains underexplored. This work aims to address this gap by designing a co-optimization framework that combines predictive traffic insights from LLMs with adaptive bandwidth control from DRL, enhancing decision-making intelligence in complex network environments.

3. Materials and Methods

3.1. Overall Algorithm Structure

In this chapter, we will introduce our network self-optimization architecture integrating reinforcement learning and a large language model (LLM), which relies on real-time data from network sensors to enhance its performance, as shown in Figure 2.

Initially, we input historical traffic data into the LLM for training the network traffic prediction. Network traffic forecasting problems can often be modeled as time series prediction tasks, and the core goal of which is to predict future traffic value x_{t+1} based on historical traffic data series $\{x_1, x_2, \ldots, x_t\}$, where x_t is the actual flow value at time step t. In the network traffic prediction task, LLMs are trained in the following ways: As the initial step, appropriate preprocessing is performed on the historical traffic data to transform it into a hybrid format comprising textual and numerical representations.

This structured format enables the input data to be pre-tokenized, thereby effectively harnessing the predictive capabilities of the LLM. Subsequently, a trend decomposition module was incorporated into the LLM architecture, allowing the model to decompose the input sequence into trend and residual components, thereby improving its predictive performance. Finally, the transformer architecture [24] is used to encode the input data, and the dependencies between different time steps in the sequence were extracted through a multi-layer self-attention mechanism. Specifically, the loss function [25] of the model can be defined as follows:

$$L(\theta) = \frac{1}{N}\sum_t (\hat{x}_t - x_t)^2, \quad (1)$$

where $\hat{x}_t$ is the predicted flow value, and N is the batch size. In order to improve the accuracy of predictions, techniques such as multi-step forecasting, model integration, and regularization are employed. Ultimately, the well-trained LLM outputs the required traffic prediction values.

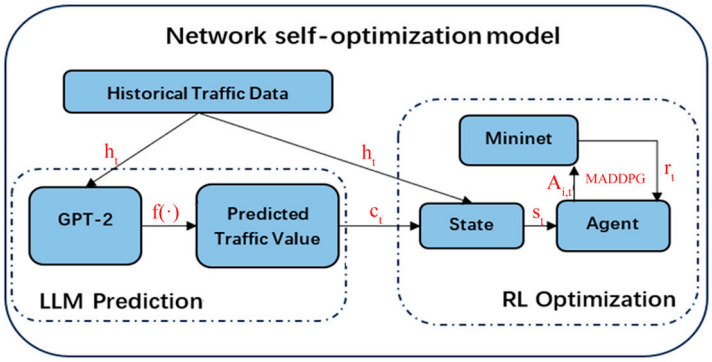

Figure 2. Proposed algorithm structure.

Subsequently, the predicted flow values and historical flow data are passed into the RL model as part of the state space. We optimize the system using the simulation network of Mininet [26] as the environment, and conduct interaction based on the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [27] algorithm. Sensors embedded within the network infrastructure collect critical metrics such as link states, delay, and packet loss rate, which serve as inputs for the MADDPG algorithm, enabling the system to dynamically optimize bandwidth allocation and adapt to changing network conditions. The agents are continuously trained and updated through ongoing interaction with the environment. The goal of RL is to learn the optimal traffic allocation strategy, so that the intelligent agent can reasonably allocate the input traffic across various links, and thereby maximize the expected value of its cumulative reward [28] in a given initial state:

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma_t r_t \mid s_0 = s, \pi\right], \quad (2)$$

where $V_\pi(s)$ is the value of the state s under the strategy π, and $\gamma \in [0, 1]$ is the discount factor, which indicates the degree of attenuation of future rewards. The goal of the agent is to find the π^* that maximizes the expected value of the accumulated reward, so as to complete the task of optimizing the overall network performance.

3.2. LLM-Based Network Traffic Prediction

We select the pre-trained GPT-2 [29] model as the backbone for traffic prediction, aiming to investigate the potential of large-scale pre-trained models in network traffic analysis. While its predictive performance is largely comparable to lighter alternatives such as LSTMs [30] or Temporal Convolutional Networks (TCNs) [31], GPT-2's large-scale

pre-training provides richer knowledge, greater flexibility in handling heterogeneous data, and stronger tolerance to variations in input format and scale [29]. These properties make GPT-2 a promising option for generating predictive insights that can guide reinforcement learning–based decision-making in complex environments.

In particular, LLMs are used to directly generate future traffic values rather than natural language text in this study; therefore, conversational input formats—often redundant and inefficient—are rendered unnecessary. The historical traffic values in a continuous time window are formatted as a space-separated numeric sequence (text-like form) as the model input, and the output is the traffic forecast value after that time period. The entire modeling process can be formalized as a function mapping as follows:

$$\hat{x}_{t+1} = f(x_{t-n}, \ldots, x_t), \tag{3}$$

where $f(\cdot)$ represents the mapping function learned by the LLM.

Network traffic data is usually highly time-dependent, and large data volumes are affected by many complex factors, such as network latency and bandwidth fluctuations. Moreover, there are sometimes a lot of random fluctuations and noise in network traffic, such as sudden traffic fluctuations or interference in the network. Therefore, in order to improve the generalization ability and robustness of the model in complex network environments, we introduce a variety of enhancements to the original network traffic data, as shown in Table 1. By adding Gaussian white noise to the original data, we simulated certain random minor fluctuations in the actual network traffic. By employing methods such as traffic merging, splitting, and zeroing, we were able to simulate abnormal events, including DDoS attacks and sudden traffic drops, within the network. These typical data augmentation strategies expand the diversity of training samples and significantly improve the adaptability of the model to a changing network environment. In addition, all raw data is normalized before enhancement to ensure input scale consistency. The prediction results are reverted by reverse normalization in the output phase.

Table 1. Sample data augmentation method.

Name	Principle	Effects	Parameter Value
Time offset	Shift the entire sequence of traffic forward or backward by several time steps	Improve the time adaptability of the model	{−5, −3, −1, +1, +3, +5} step
Noise disturbances	Add Gaussian noise to the raw data	Improve model robustness and prevent overfitting	σ =0.1
Extreme event synthesis	Artificially synthesize abnormal traffic events, such as Distributed Denial of Service attacks	Enhance the ability of the model to identify and respond to burst traffic	Training: one event per 1700 data points; Testing: one event at a random position

In order to improve the learning efficiency and performance of the model, we also adopt the "stepwise window expansion" strategy in this study. In the early stage of training, in order to reduce the training overhead, we choose to use a small time window to let the model learn simple local patterns through rapid training. As the model gradually learns more features and trends, we increase the size of the window gradually to capture longer-term dependencies, which also learns information on different time scales and improves the accuracy of prediction.

3.3. Implementation of Self-Optimization Strategy Based on Reinforcement Learning

Based on the aforementioned traffic prediction and simulation platform, we further construct an RL-based intelligent agent model. The following section will elaborate on the

design of the state space, action space, and reward function, as well as the deployment mechanism of the multi-agent architecture.

In this paper, the network optimization problem is modeled as a multi-agent RL task, and each link in the network is treated as an independent agent. In order to achieve efficient scheduling and resource allocation for dynamic complex networks, we adopt a method framework based on MADDPG [27].

In terms of implementation, MADDPG builds an Actor Network and a Critic Network for each agent. The Actor network generates actions based on the current link state (i.e., the bandwidth allocation ratio), while the Critic network evaluates the overall reward performance of the entire system under that action in a centralized structure. Through the architecture of centralized evaluation and distributed execution, agents can not only optimize policies based on the local state but also consider the behavior and feedback of other links during the training process, so as to achieve system-level collaborative optimization [32].

The key to MADDPG lies in its centralized Critic network, which evaluates the overall performance of the multi-agent system, while each agent maintains its own Actor network for policy optimization. This structure allows agents to update their policies based on global feedback, converging toward a system-wide optimal strategy rather than local optima. By mitigating inconsistencies from independent decision-making, the centralized Critic enables more efficient resource scheduling and optimization in complex network topologies.

In order to realize the efficient decision-making of agents in the network environment, we reasonably design the state space$S$, action space$A$ and reward function R under the framework of MADDPG. That is, a Markov process based on tuples $\langle S,\ A,\ R\rangle$ makes the agent learn effective resource scheduling strategies in a dynamically changing network.

In terms of the construction of S, the state perceived by the agent at time t is defined as follows:

$$s_t = [d_t, l_t, c_t, h_t], \tag{4}$$

where d_t represents the current delay of the link; l_t represents the packet loss rate of the link; c_t represents the future incoming traffic as predicted by the LLM module; and h_t represents historical traffic to introduce time series features. These metrics can comprehensively reflect the health status and load level of the current link, which is convenient for the agent to evaluate and perceive the network status, and finally make decisions based on status feedback. In addition, historical traffic information can help capture the time series dependence of the network state and improve the forward-looking and stable strategy.

The definition of action space A corresponds to the bandwidth allocation ratio of the link. Each agent selects the corresponding action to adjust the link bandwidth allocation ratio based on the current perceived state so as to implement dynamic resource scheduling. Specifically, each agent's $A_{i,t}$ consists of two actions, a_{i1} and a_{i2}, which correspond to the link for the distribution of two input traffic on the link. In particular, in order to make the action physically meaningful and not too small, it is subject to certain restrictions, namely:

$$\begin{cases} a_{i1}, a_{i2} \in A \in (0.001, 1) \\ \sum_{i=1}^{5} a_{i1} = \sum_{i=1}^{5} a_{i2} = 1 \end{cases}. \tag{5}$$

The continuous action design enables the RL model to control the link resources in a fine-grained manner so as to optimize the overall latency and packet loss performance while ensuring network stability.

In order to effectively guide agents to learn the expectation strategy, we design a reward function R that is closely related to network performance. The reward function aims

to minimize the link delay and packet loss rate, which represent the most critical and widely accepted QoS requirements in such simulation scenarios [33], thereby promoting the convergence of agents' strategies toward the global optimum. A linear combination is adopted as it represents a widely used and general reward design strategy in reinforcement learning, ensuring stability, interpretability, and flexibility. It can be expressed mathematically as follows:

$$r_t = R(d_t, l_t) = -(l_t + \alpha \cdot d_t), \tag{6}$$

where α represents the discount factor, allowing adjustment of the relative importance of the two terms under different service requirements. α serves to balance packet loss and delay in the reward function: A larger α places more emphasis on delay reduction, whereas a smaller α increases tolerance to delay but may lead to higher packet loss. Similarly, the adopted learning rate achieves a compromise between fast convergence and stability, avoiding oscillations or divergence while maintaining sufficient training efficiency.

According to the reward function, during the training process, when the bandwidth allocation strategy adopted by the agent helps to improve the network performance, the agent will receive a positive reward. On the contrary, a negative incentive is given, which guides the strategy to converge to the global optimal solution.

4. Results

In this section, we carry out partial and overall simulation verification of the network self-optimization model based on real-time data from network sensors and fully analyze the obtained data results to verify the performance of the network self-optimization system.

4.1. Network Traffic Forecasting

In this paper, we select GPT-2 [29] as an LLM-based traffic prediction model to predict the time series based on historical network traffic series. The training dataset used in this study is an open-source historical Internet traffic dataset that contains traffic measurements (in bits) with corresponding timestamps and other related information. (All privacy and compliance issues have been adequately addressed at the source level, and the dataset can be accessed online: https://github.com/xiaohuiduan/network-traffic-dataset (accessed on 13 September 2025)). The data was collected by a private ISP operating in 11 European cities and records traffic on a transatlantic link at five-minute intervals. The measurement period spans from 06:57 on 7 June to 11:17 on 31 July 2005. In total, the dataset comprises 8000 entries, covering various typical network states (e.g., stable transmission, bursty traffic), with noticeable correlations observed between traffic volume and time. In terms of data preprocessing, we adopt standard normalization to ensure training stability. The dataset is further divided into training, validation, and test sets using an 8:1:1 ratio. Moreover, the time series is segmented into sliding windows of size initially five steps, enlarged by one step every 1500 data points until reaching a maximum of 10 steps to construct the model inputs. We use constructed datasets to train the model and evaluate it through simulation experiments. The performance of the model is mainly evaluated by indicators such as Mean Absolute Error (MAE) and Mean Square Error (MSE), which can measure the generalization ability in different types of traffic scenarios. Table 2 shows the configuration of the core training parameters for GPT-2. We set the maximum input length to 128 to enable the model to capture flow patterns across the broadest temporal scale when the time window reaches its maximum extent. The initial learning rate was set to 1×10^{-5}; however, due to slow convergence under this configuration, the learning rate was subsequently increased to 1×10^{-4}, which still allowed for effective convergence without compromising model performance. Moreover, a weight decay value of 0.01 was applied to mitigate the risk of overfitting. Table 3 presents the configuration of the core architectural and opti-

mization parameters of MADDPG. Both the actor and critic networks are designed with two hidden layers, each containing 64 units. To ensure stable learning, the target network update rate is set to $\tau = 0.001$, enabling smooth soft updates. Exploration is facilitated through a Gaussian noise process with parameters $\mu = 0$ and $\sigma = 0.1$, which introduces temporally correlated perturbations in continuous action spaces. In addition, the replay memory is configured with a capacity of 1×10^{-6} and a minibatch size of 1024 to provide a diverse sampling of past experiences. Finally, the role of the α weighting factor in the reward function is explicitly defined in this table, with its sensitivity further examined in the experimental analysis. Through repeated trials, we found that the chosen value of the α weighting factor and the selected learning rate consistently yielded the most stable training performance.

Table 2. Training parameter settings.

Item	Value
Maximum input length	128
Learning rate	1×10^{-4}
Batch size	8
Weight decay	0.01
Optimizer	Adam
Loss function	MAE, MSE

Table 3. Hyperparameter settings for MADDPG.

Component	Configuration
Actor network	[64, 64] with ReLU and tanh
Critic network	[64, 64] with ReLU and tanh
Target update parameter	0.001
Noise process	$\mathcal{N}(0,\ 0.2^2)$
Replay memory	1×10^{-6}
Discount factor (α)	0.95
Optimizer	Adam
learning rate	1×10^{-3} (actor), 1×10^{-3} (critic)

After the simulation, we obtain the following images, which show the training process and evaluation results in the network traffic prediction task using LLM. Through the analysis of graphs, we were able to examine the trends in learning rate adjustment, training loss, and evaluation loss, and how these factors impacted the model's performance.

The left figure in Figure 3 shows the training loss of the GPT-2 model. It can be observed that the model loss is high at the beginning, and then decreases rapidly and gradually converges, indicating that the model is gradually learning and fitting the training data.

The local fluctuations in the curve reflect the temporary instability of the model in response to some traffic mutation patterns, suggesting that further tuning of model hyperparameters is warranted to enhance generalization performance.

The right figure in Figure 3 shows that the evaluation loss on the validation set fluctuates more dramatically than the training loss and shows a slow upward trend overall. This phenomenon suggests a tendency of the model to overfit, meaning that while it achieves high performance on the training data, its generalization capability deteriorates when exposed to unseen or diverse traffic patterns.

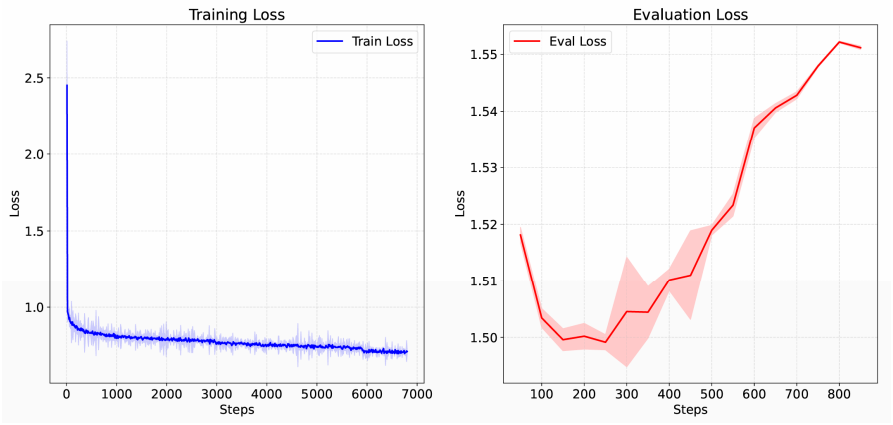

Figure 3. Loss during the training and evaluation process of the GPT-2 traffic data prediction model.

In order to alleviate the above problems, we introduce two improvement strategies: one is to adopt an early stop mechanism [34] to prevent overfitting caused by an excessively long training process, which terminates training if the validation loss does not improve for 10 consecutive epochs; the other is to replace the original simplified noise disturbance with Gaussian white noise, so as to simulate the network traffic fluctuation more realistically. Figure 4 shows the training loss and the evaluation loss after optimization. It can be observed that the application of these strategies does not significantly impact the training loss, which continues to converge, demonstrating the model's ability to effectively learn the optimal solution. Meanwhile, the evaluation loss shows an overall downward trend. When the loss increased, the early stopping strategy was triggered promptly, effectively avoiding overfitting.

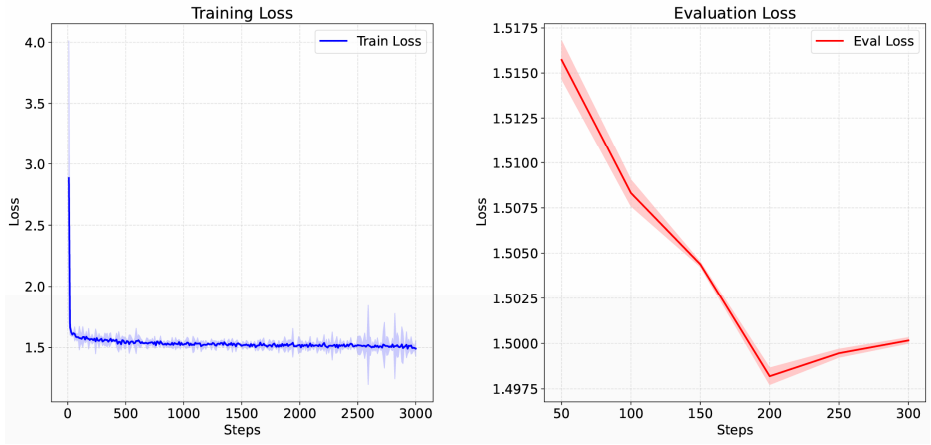

Figure 4. Model training loss and evaluation loss images of the GPT-2 prediction model based on early stop Gaussian white noise.

Moreover, we conducted comparative experiments with the LLM-based traffic prediction method against traditional prediction models on the normalized score. As shown in Figure 5, the GPT-2 model outperforms classical methods such as random forest (RF in the figure) [35] and linear regression (LR in the figure) in terms of prediction accuracy and responsiveness to violent fluctuations in traffic, which verifies its generalization ability in complex network environments, although it performs slightly weaker than specialized time-series models such as LSTM and TCN under full-input settings. In addition, to further assess robustness, we compared GPT-2 with specialized time-series models under perturbed input conditions, where 30% of the input sequence was randomly masked to simulate disturbances. As shown in Figure 6, GPT-2 achieves superior performance compared

to LSTM and TCN in this setting. This demonstrates that GPT-2 is more resilient to noisy or incomplete observations. Moreover, given their flexibility in processing heterogeneous data types and structures, LLMs present greater potential for practical deployment and broader applicability in future networking scenarios.

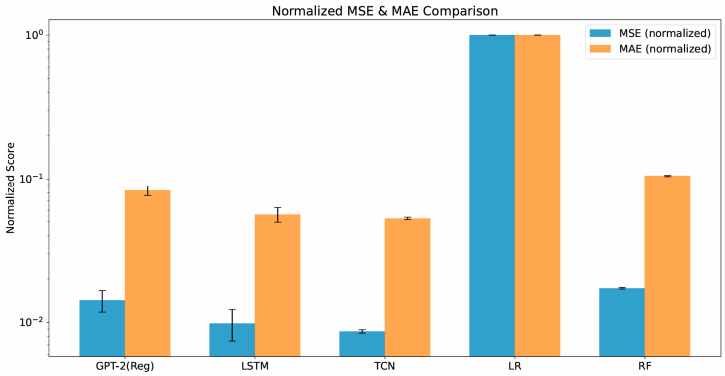

Figure 5. Comparison of traffic data prediction performance between the GPT-2 model and other models.

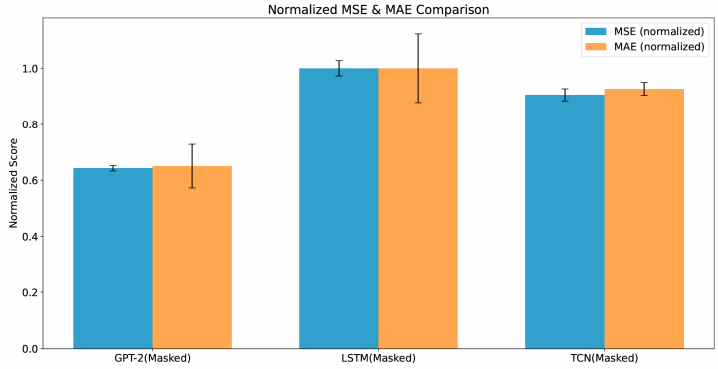

Figure 6. Comparison of traffic data prediction performance under perturbed input data.

4.2. Network Self-Optimization Model Designing

Based on the traffic prediction model and network optimization model, in order to verify the system performance, we construct a heterogeneous network topology using Mininet [26] and simulate the real network scenario. In addition, in order to evaluate the performance of the network self-optimization system under controllable and reproducible conditions, a representative simulated network environment was constructed based on the Mininet platform.

It is worth noting that the proposed model can be flexibly deployed across cloud and edge environments, which allows the LLM to operate on resource-rich servers. From a complexity standpoint, the GPT-2 (6-layer) predictor has $T_{LLM}(L, d) = O\left(6\left(L^2 d + L d^2\right)\right)$ [29] with activation memory $O\left(6\left(L^2 + Ld\right)\right)$, favoring cloud execution. In contrast, the MADDPG actor per agent is $O\left(|S| \cdot 64 + 64^2 + 64 \cdot |A|\right)$, yielding an overall $O(N)$ edge cost for N agents. Accordingly, we assume LLM inference in the cloud and lightweight RL on the edge to minimize edge-side computation while preserving low-latency decision-making.

Consequently, issues such as computational resources and deployment overhead are not the primary focus of this study. Instead, we concentrate on network optimization metrics. Correspondingly, network topology has far-reaching significance for model training and evaluation. The change in traffic is not only affected by the physical distance between hosts, but also closely related to network parameters such as link bandwidth and latency. In this paper, we design a five-path transport topology consisting of two pairs

of hosts and multiple switches. The topology consists of five independent paths between hosts, each with different bandwidth and latency settings, simulating the characteristics of heterogeneous links commonly found in real-world networks. These include the main path with high bandwidth and low latency, the path with low bandwidth and high latency, and the path with medium bandwidth and latency. Table 4 lists the configurations of each path. Based on these paths, we set up the Mininet network topology, as shown in Figure 7. In the diagram, h_i stands for the number of the host and s_i stands for the number of the switch, and the network topology makes good use of the multipath design described above. To simplify the system without affecting the experiment, we set the flow transmission from h1 and h2 to s1, as well as the transmission from s7 to h3 and from s8 to h4, to be zero-delay and zero-loss transmissions. For the five switches from s2 to s6, the links connected to their respective sides are set as the links numbered 1 to 5 in Table 4. This multipath design introduces the heterogeneity and uncertainty of link performance, which helps simulate practical challenges in the network (e.g., link bottlenecks, congestion migration, delay-sensitive path switching). Also, it provides a more complex decision-making environment for path selection and resource scheduling of subsequent self-optimization strategies.

Table 4. Network link settings.

Number	Bandwidth	Latency (ms)	Description
1	20	5	High bandwidth and low latency
2	10	10	Medium bandwidth and latency
3	5	15	Low bandwidth and high latency
4	20	15	High bandwidth and high latency
5	5	5	Low bandwidth and high latency

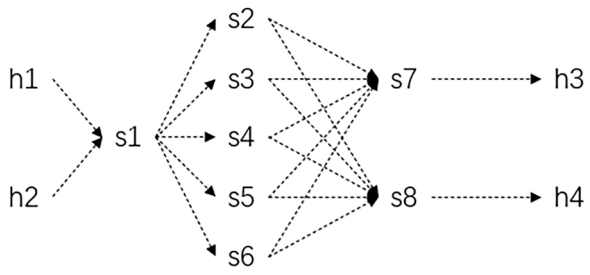

Figure 7. Mininet network topology.

During the simulation, we use the low-level interface provided by Mininet to dynamically configure and control the link parameters.

The above parameters will be passed to the subsequent self-optimization system as the environment state, and the agent can sense the current network quality and make corresponding control decisions.

4.3. Network Self-Optimization Model Testing

In order to verify the practical effect of the RL-based network optimization method, we built a simulated network environment on the Mininet platform. The experimental topology consists of two host nodes and five links with different characteristics to simulate different network congestion and bandwidth conditions. By interacting with the reinforcement learning module with the simulation network, we can observe the learning process and optimization effect of the agent in the dynamic network environment.

Considering the uncertainty and complexity of the real-world network environment, we used a variety of scenarios for training, including network latency fluctuations, link congestion, and burst traffic. By simulating these complex scenarios, agents

can better cope with various challenges in real networks and improve the robustness of optimization strategies.

For the RL model, we set the batch size to 32 and the maximum length of the replay buffer to 1000, which contributes to enhancing the stability of the training process. To highlight the importance of long-term network performance, we set the discount rate to 0.95. Furthermore, in order to balance training speed and effectiveness, the learning rates of the actor network and critic network were adjusted from 2×10^{-3} to 5×10^{-4} and 1×10^{-3}, respectively, to ensure optimal model performance.

The training process of the RL model is shown in Figure 8. It can be seen that in the early stage of training, the reward value fluctuates greatly because the strategy has not yet converged. However, with the progress of learning, the agent gradually masters the bandwidth allocation strategy, and the reward value increases and tends to stabilize as a whole. This shows that the RL model can effectively extract network state features and gradually approximate the global optimal strategy.

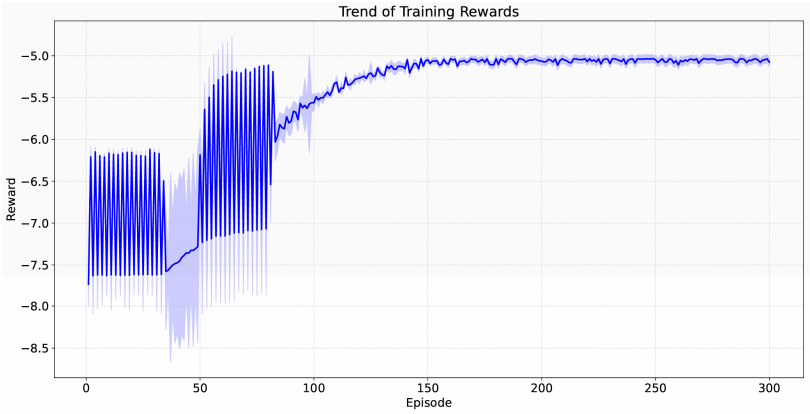

Figure 8. Training loss of the reinforcement learning system under the constant network traffic mode input.

Figure 9 shows that, in the initial stage of training, the intelligent agent conducts thorough exploration and tries various traffic distribution strategies. As a result, the delay and packet loss rate are relatively high and fluctuate greatly. As the intelligent agent gradually learns the reasonable traffic distribution strategy, the curve gradually decreases and approaches stability. Thus, it can be seen that the intelligent agent can conduct exploration and learning in a relatively short period of time and allocate network bandwidth reasonably, thereby gradually reducing and converging the network delay and packet loss rate and achieving better network optimization performance.

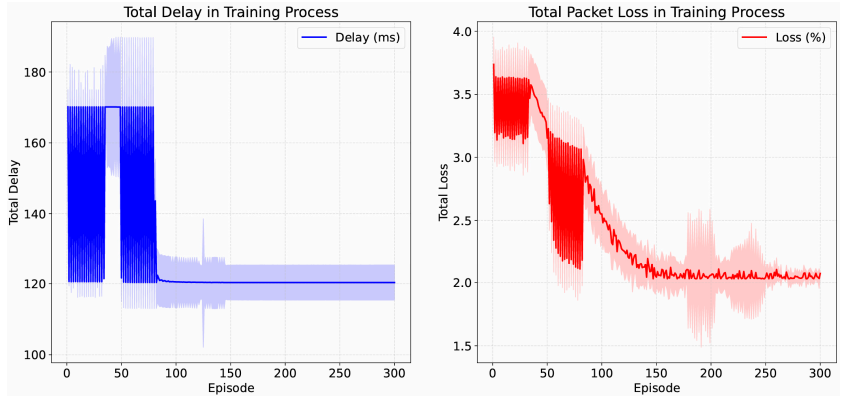

Figure 9. The total delay and packet loss during the training process of the constant network traffic mode input.

We also test the integration of the LLM-based traffic prediction module with the RL optimization system, as illustrated in Figure 10. The network optimization delay packet loss curve is based on the RL model of network traffic changes over time. At the beginning of the training process, the intelligent agent could not effectively utilize the predicted traffic data sent by LLM. It merely relied on random exploration to allocate the traffic. The allocation strategies explored by the agent varied in quality, with many exhibiting suboptimal performances during the early stages of training. Consequently, the network latency and packet loss fluctuate greatly, and the overall values are quite large. After a period of training, the intelligent agent effectively acquired optimal network traffic allocation strategies and learned to leverage traffic predictions provided by the LLM (Large Language Model) to optimize future network states. As a result, both the delay and packet loss curves gradually converged toward lower, more stable levels. These findings demonstrate that the system maintains robust optimization capabilities even under highly dynamic and fluctuating traffic conditions.

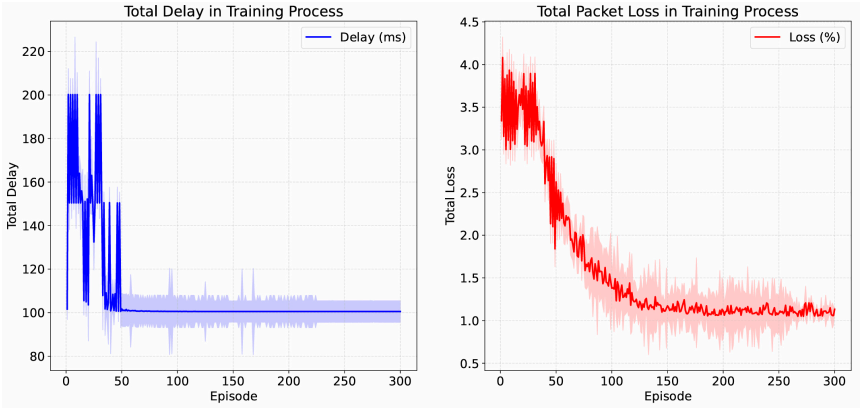

Figure 10. The total delay packet loss curve for the integrated system during the training process of the fluctuating network traffic mode input.

Moreover, we compare the MADDPG with priority allocation strategy (Uniform), the Deep Deterministic Policy Gradient (DDPG) algorithm [36], the Independent Q-Learning (IQL) algorithm [37], the Q-value Mixing (QMIX) algorithm [38], and the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm [39]. Figure 11 presents the reward curves obtained during the training process of each algorithm. The analysis indicates that the MADDPG algorithm converges at approximately 100 epochs, whereas the DDPG algorithm requires nearly 250 epochs to reach convergence. Further investigation of the MAPPO and QMIX algorithms reveals that they necessitate a minimum of 500 epochs to achieve basic convergence. In addition, ablation experiments comparing standard MADDPG with the LLM-enhanced version show that incorporating traffic prediction yields higher rewards and greater robustness, validating the proposed approach and its shift from reactive adaptation to proactive optimization. Consequently, MADDPG demonstrates superior performance compared to the other algorithms in terms of convergence behavior and training stability.

Figure 12 compares the optimization performance of the integrated system and traditional methods for fluctuating network traffic input. It can be seen that the traditional method lacks the ability to adapt to various traffic conditions, and the overall network performance is poor; it cannot be self-optimized. For the DDPG algorithm, the multi-input multi-output nature of network traffic combined with the complexity of network topology makes it challenging for a single agent to effectively learn optimal optimization patterns, often leading to subpar optimization performance. For the IQL algorithm, it only divides the task of multiple agents into a single agent. As it does not account for the direct co-

operation and competition relationship between each agent representing each link in the complex network environment, it cannot effectively handle the interaction between agents and has poor optimization performance. The QMIX algorithm has a relatively high training efficiency; nevertheless, its assumption of the monotonicity of the global value function makes it less suitable for the traffic allocation task, resulting in poorer performance. For the MAPPO algorithm, although it has certain collaborative capabilities and simplifies the training process, it lacks the modeling of the strategies of other intelligent agents. Therefore, its optimization effect is not as good as that of the MADDPG algorithm. This strongly proves that the network self-optimization system based on MADDPG strategy with LLM and RL has effective optimization performance and high robustness to various complex network traffic conditions.

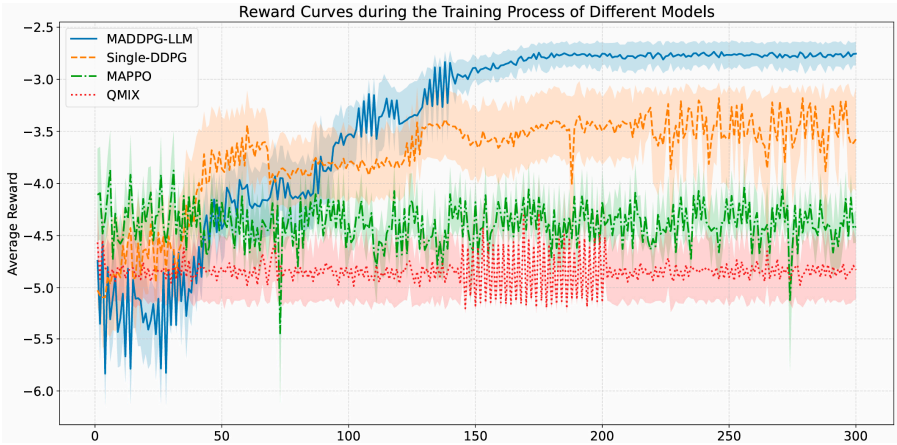

Figure 11. Comparison of the convergence behavior of the integrated system and traditional methods for fluctuating network traffic input.

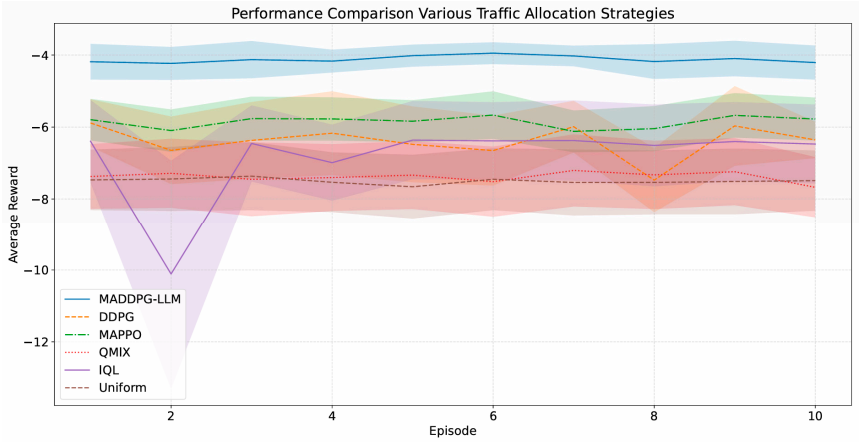

Figure 12. Comparison of the optimization performance of the integrated system and traditional methods for fluctuating network traffic input.

In conclusion, we conducted multiple rounds of testing and analyzed the average values of latency, packet loss rate, and response time, as illustrated in Figure 13. While the proposed algorithm exhibits a longer response time compared to certain other algorithms, the differences remain within an acceptable range, enabling effective real-time network optimization. Furthermore, the algorithm demonstrates significantly better performance in terms of both latency and packet loss rate. Therefore, by leveraging real-time sensor data, the system ensures accurate monitoring and proactive resource scheduling, ultimately reducing latency and improving overall network efficiency.

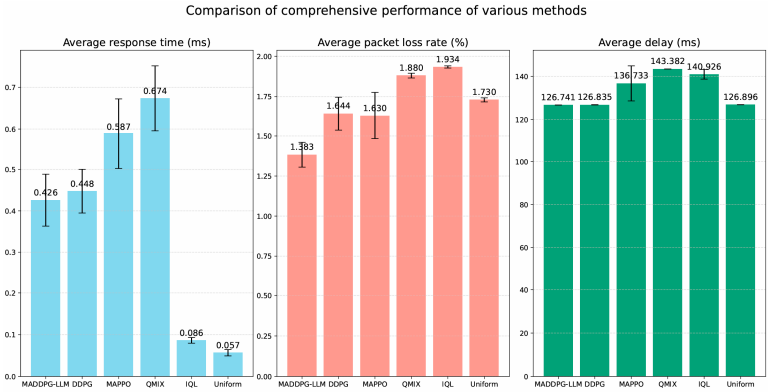

Figure 13. Comparison of the comprehensive performance of the integrated system and traditional methods across multiple evaluation iterations.

To further evaluate the robustness and scalability of the proposed framework, we extended the experiments to four distinct network topologies, namely three-path, seven-path, branched, and ring structures. For each topology, we present both the network structure and the corresponding performance comparison results against baseline strategies (DDPG, IQL, QMIX, MAPPO, and Uniform). As shown in Figure 14, our LLM-enhanced MADDPG consistently achieves superior performance across all topologies, confirming its robustness under longer paths, shifting bottlenecks, and non-stationary loads.

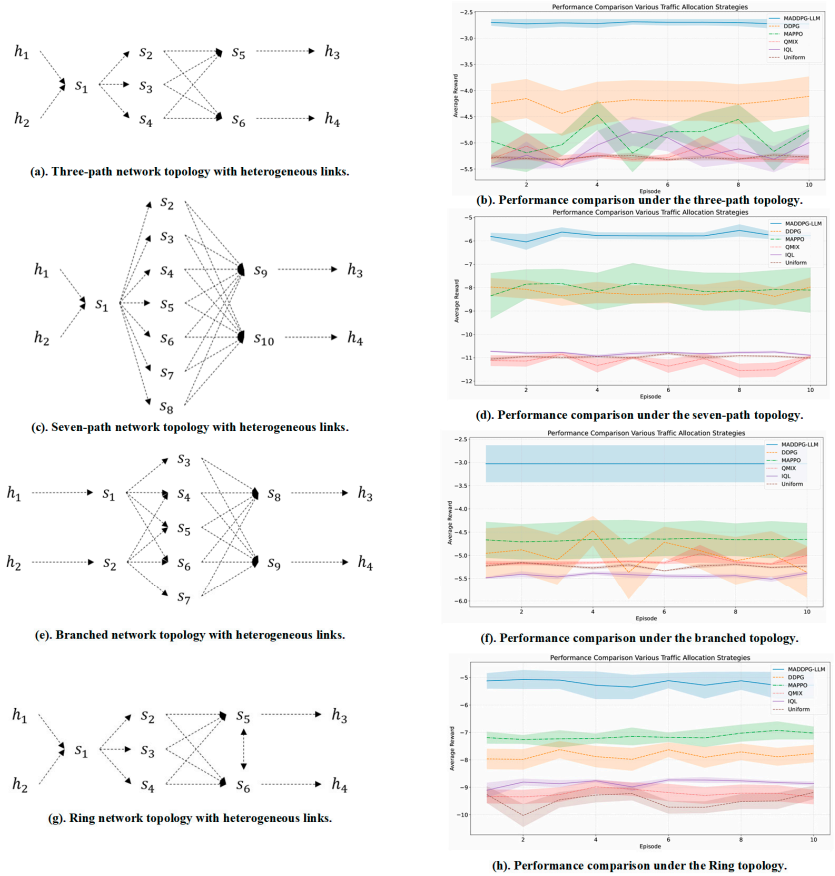

Figure 14. Network topology structures (**left**) and performance comparison results (**right**) under five topologies: (**a**,**b**) three-path, (**c**,**d**) seven-path, (**e**,**f**) branched, and (**g**,**h**) ring. The five-path topology was already presented as the baseline in Figures 6–12.

Moreover, since each communication link is modeled as an independent agent, scalability can be achieved by simply increasing the number of agents when extending to larger

or more complex topologies, without requiring modifications to the core framework. To quantify the computational overhead, Table 5 reports the inference time across the five topologies on our experimental platform (Intel i7-11800H CPU and NVIDIA RTX 3060 GPU). The results demonstrate that inference remains at the millisecond level, confirming the practicality of our approach for real-time deployment.

Table 5. Inference time (ms) across different network topologies on Intel i7-11800H CPU and NVIDIA RTX 3060 GPU.

Topology	Three-Path	Five-Path	Seven-Path	Branched	Ring
Inference time (ms)	0.358	0.586	1.002	0.436	0.753

Table 6 summarizes the key performance metrics of our method. The upper block presents latency, loss, and reward under different topologies, together with the relative improvement of reward over baselines. The middle block reports the inference time of our method, and the bottom block shows prediction error (mean $\pm$ std) compared with baseline predictors.

Table 6. Performance summary across topologies and prediction error.

Topology	Latency (ms)	Loss	Reward	Rel. Improv. (Reward)
Three-path	133.84	1.72	−2.64	+42.77%
Five-path	159.36	2.17	−5.17	+8.19%
Seven-path	82.87	1.69	−4.69	+51.89%
Ring	133.81	2.21	−2.93	+36.33%
Branched	129.65	2.19	−5.74	+22.16%
Inference time of our method (ms)				
Three-path	0.358			
Five-path	0.586			
Seven-path	1.002			
Ring	0.753			
Branched	0.436			
	Prediction error (perturbed)			
Model	MSE (normalized)		MAE (normalized)	
LSTM	1.000 ± 0.027		1.000 ± 0.122	
TCN	0.906 ± 0.022		0.927 ± 0.023	
GPT-2	0.643 ± 0.010		0.651 ± 0.078	

5. Conclusions

In this paper, we design a network self-optimization system integrating RL and LLM to enhance communication network performance in dynamic environments, where the GPT-2 model is employed for traffic prediction to improve future network condition forecasting, and MADDPG is used for bandwidth scheduling and adaptive resource allocation in heterogeneous networks. The proposed network self-optimization system relies on real-time data from network sensors to enhance its performance. The simulation environment built with Mininet provides an effective platform for evaluating system performance under diverse dynamic traffic scenarios. Experimental results demonstrate that the proposed fusion model achieves efficient resource allocation with reduced delay and packet loss, showcasing strong adaptability and robustness—especially with traffic prediction integration—while highlighting the synergy between RL and LLMs to lay a practical foun-

dation for future predictive and intelligent network optimization systems. This work has demonstrated the effectiveness of multi-agent RL based on LLM-based traffic prediction in network self-optimization, thereby providing a feasible paradigm for future applications of LLMs in communication systems. In the future, efforts will be made to explore more lightweight LLMs to enhance deployment feasibility. While the current design leverages cloud deployment to mitigate computational cost and training overhead, real-time constraints and limited edge resources remain important challenges. Addressing these issues through model compression and lightweight architectures will be a key step toward practical deployment in real-world networks. We will focus on enhancing training efficiency and deployment feasibility through model compression, meta-learning, and edge integration, alongside exploring end-to-end optimization to further improve system performance in real-world conditions.

Author Contributions: Conceptualization, X.X.; Methodology, J.Z. and Y.Z.; Validation, J.Z.; Formal analysis, R.L.; Investigation, Y.Z.; Resources, Y.Z.; Writing—original draft, X.X.; Writing—review & editing, X.X. and R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Science and Technology Project of State Grid Hebei Information Telecommunication Branch (contact number kj2024-017).

Data Availability Statement: The original contributions presented in this study are included in the article material. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Xing Xu and Jianbin Zhao are employed by the company Information and Communication Branch of State Grid Hebei Electric Power Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Dandy, G.; Simpson, A.; Murphy, L. An improved genetic algorithm for pipe network optimization. *Water Resour. Res.* **1996**, *32*, 449–458. [CrossRef]
2. Ephremides, A.; Verdú, S. Control and optimization methods in communication network problems. *IEEE Trans. Autom. Control* **1989**, *34*, 930–942. [CrossRef]
3. Mammeri, Z. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access* **2019**, *7*, 55916–55950. [CrossRef]
4. Pan, Z.; Yang, J. Deep reinforcement learning-based optimization method for D2D communication energy efficiency in heterogeneous cellular networks. *IEEE Access* **2024**, *12*, 140439–140455. [CrossRef]
5. O'Shea, T.; Hoydis, J. An introduction to deep learning for the physical layer. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 563–575. [CrossRef]
6. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3133–3174. [CrossRef]
7. Chen, Y.; Guo, Y. Network link weight optimization based on antisymmetric deep graph networks and reinforcement learning. In Proceedings of the 2024 Sixth International Conference on Next Generation Data-Driven Networks (NGDN), Shenyang, China, 26–28 April 2024.
8. Gómez-delaHiz, J.; Galán-Jiménez, J. Improving the traffic engineering of SDN networks by using local multi-agent deep reinforcement learning. In Proceedings of the NOMS 2024 IEEE Network Operations and Management Symposium, Seoul, Republic of Korea, 6–10 May 2024.
9. Wu, D.; Wang, X.; Qiao, Y.; Lu, J.; Zhang, M.; Wang, K. NetLLM: Adapting large language models for networking. In Proceedings of the ACM SIGCOMM 2024 Conference, Sydney, Australia, 4–8 August 2024.
10. Liu, B.; Liu, X.; Gao, S.; Cheng, X.; Yang, L. LLM4CP: Adapting large language models for channel prediction. *J. Commun. Inf. Netw.* **2024**, *9*, 113–125. [CrossRef]
11. Nascimento, N.; Alencar, P.; Cowan, D. Self-adaptive large language model (LLM)-based multiagent systems. In Proceedings of the 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion, Toronto, ON, Canada, 25–29 September 2023.

12. Mondal, A.; Mishra, D.; Prasad, G.; Hossain, A. Joint Optimization Framework for Minimization of Device Energy Consumption in Transmission Rate Constrained UAV-Assisted IoT Network. *IEEE Internet Things J.* **2022**, *9*, 9591–9607. [CrossRef]
13. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPs*; Springer: Berlin, Germany, 2016; Volume 1.
14. Li, Z.; Wang, X.; Pan, L.; Zhu, L.; Wang, Z.; Feng, J.; Deng, C.; Huang, L. Network topology optimization via deep reinforcement learning. *IEEE Trans. Commun.* **2022**, *71*, 2847–2859. [CrossRef]
15. Do, Q.V.; Koo, I. Dynamic bandwidth allocation scheme for wireless networks with energy harvesting using actor-critic deep reinforcement learning. In Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication, Okinawa, Japan, 10–16 August 2019.
16. Attiah, K.; Ammar, M.; Alnuweiri, H.; Shaban, K. Load balancing in cellular networks: A reinforcement learning approach. In Proceedings of the 2020 IEEE 17th Annual Consumer Communications & Networking Conference, Las Vegas, NV, USA, 10–13 January 2020.
17. Abu-Ein, A.; Abuain, W.; Alhafnawi, M.; Al-Hazaimeh, O. Security enhanced dynamic bandwidth allocation-based reinforcement learning. *WSEAS Trans. Inf. Sci. Appl.* **2024**, *22*, 21–27. [CrossRef]
18. Li, R.; Zhao, Z.; Sun, Q. Deep reinforcement learning for resource management in network slicing. *IEEE Access* **2018**, *6*, 74429–74441. [CrossRef]
19. Liu, Y.; Ding, J.; Liu, X. A constrained reinforcement learning based approach for network slicing. In Proceedings of the 2020 IEEE 28th International Conference on Network Protocols, Madrid, Spain, 13–16 October 2020.
20. Wang, H.; Wu, Y.; Min, G.; Xu, J.; Tang, P. Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach. *Inf. Sci.* **2019**, *498*, 106–116. [CrossRef]
21. Shokouhi, M.H.; Wong, V.W.S. Large language models for wireless cellular traffic prediction: A multi-timespan approach. In Proceedings of the GLOBECOM 2024 IEEE Global Communications Conference, Cape Town, South Africa, 8–12 December 2024.
22. Yang, S.; Wang, D.; Zheng, H.; Jin, R. TimeRAG: Boosting LLM time series forecasting via retrieval-augmented generation. In Proceedings of the ICASSP 2025 IEEE International Conference on Acoustics, Speech and Signal Processing, Hyderabad, India, 6–11 April 2025.
23. Guo, X.; Zhang, Q.; Jiang, J.; Peng, M.; Zhu, M.; Yang, H. Towards explainable traffic flow prediction with large language models. *Commun. Transp. Res.* **2024**, *4*, 100150. [CrossRef]
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.
25. Raiaan, M.A.K.; Mukit, M.S.H.; Fatema, K.; Nahid, A.A.; Islam, M.; Uddin, M.S.; Podder, P.; Alsaikhan, F.; Alqahtani, A. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access* **2024**, *12*, 26839–26874. [CrossRef]
26. de Oliveira, R.L.S.; Schweitzer, C.M.; Shinoda, A.A.; Prete, L.R. Using Mininet for emulation and prototyping software-defined networks. In Proceedings of the 2014 IEEE Colombian Conference on Communications and Computing, Bogota, Colombia, 4–6 June 2014.
27. Li, T.; Zhu, K.; Luong, N.C.; Niyato, D.; Wu, Q.; Zhang, Y.; Chen, B. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 1240–1279. [CrossRef]
28. Sutton, R.S.; Barto, A.G. Reinforcement learning: An introduction. *IEEE Trans. Neural Netw.* **1998**, *9*, 1054. [CrossRef]
29. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
30. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
31. Lea, C.; Vidal, R.; Reiter, A.; Hager, G.D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; Springer: Cham, Switzerland, 2016; pp. 47–54.
32. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Red Hook, NY, USA, 4–9 December 2017.
33. Li, B.; Chen, L.; Yang, Z.; Xiang, H. preDQN-Based TAS Traffic Scheduling in Intelligence Endogenous Networks. *IEEE Syst. J.* **2024**, *18*, 997–1008. [CrossRef]
34. Teerapittayanon, S.; McDanel, B.; Kung, H.T. Branchynet: Fast inference via early exiting from deep neural networks. In Proceedings of the 2016 23rd International Conference on Pattern Recognition, Cancun, Mexico, 4–8 December 2016.
35. Louppe, G. Understanding Random Forests: From Theory to Practice. Ph.D. Thesis, Université de Liège, Liège, Belgium, 2014.
36. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2019**, arXiv:1509.02971. [PubMed]
37. Kostrikov, I.; Nair, A.; Levine, S. Offline reinforcement learning with implicit Q-learning. *arXiv* **2021**, arXiv:2110.06169. [CrossRef]

38. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 7234–7284.
39. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, J. The surprising effectiveness of PPO in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.