Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/64/20241370

Integration of computer networks and artificial neural networks for an AI-based network operator

Binbin Wu[1,*],Jingyu Xu[2],Yifan Zhang[3], Bo Liu[4],Yulu Gong[5],Jiaxin Huang [6]

[1]Heating Ventilation and Air Conditioning Engineering, Tsinghua University, Beijing China
[2]Computer Information Technology, Northern Arizona University,1900 S Knoles Dr, Flagstaff, AZ, USA
[3]Executive Master of Business Administration, Amazon Connect Technology Services (Beijing), Co., Ltd. Xi'an, Shaanxi, China
[4]Software Engineering, Zhejiang University, HangZhou China
[5]Computer & Information Technology, Northern Arizona University, Flagstaff, AZ, USA
[6]Information Studies, Trine University, Phoenix USA

*Corresponding author: wubinbin.1@gmail.com

Abstract. This paper proposes an integrated approach combining computer networks and artificial neural networks to construct an intelligent network operator, functioning as an AI model. State information from computer networks is transformed into embedded vectors, enabling the operator to efficiently recognize different pieces of information and accurately output appropriate operations for the computer network at each step. The operator has undergone comprehensive testing, achieving a 100% accuracy rate, thus eliminating operational risks. Additionally, a simple computer network simulator is created and encapsulated into training and testing environment components, enabling automation of the data collection, training, and testing processes. This abstract outline the core contributions of the paper while highlighting the innovative methodology employed in the development and validation of the AI-based network operator.

Keywords: AI-based Network Operator, Integration of Computer Networks and Artificial Neural Networks, Embedded Vector Representation, Automated Training and Testing Environment.

1. Introduction

The current enterprise network architecture faces numerous challenges during the process of digital transformation. As a critical and complex component of IT infrastructure, computer network systems are undergoing a paradigm shift in architecture and design principles from traditional hardware-centric network operations to business and application-oriented network operations. Challenges such as insufficient agility in network architecture, inadequate automation and intelligence in network operations, and difficulties in network fault localization are among the primary obstacles. These challenges further result in issues such as failure to meet [1]Service Level Agreement (SLA)

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/64/20241370

requirements for business continuity and high network operation costs. To address these challenges, this paper introduces the design of an intelligent operator capable of analyzing network information, identifying network faults, and providing appropriate actions to resolve them. This intelligent operator represents a novel approach towards enhancing network resilience, automating network operations, and meeting the evolving demands of digital transformation in enterprise networks.

2. Related work

2.1. System architecture

In the example of a minicomputer network in this article, we will show how to make an intelligent operator. This computer network is created randomly through simulators, and in doing so increases the diversity that allows our intelligent operators to make appropriate decisions and reactions in a variety of situations. Our intelligent operators will manage and maintain this network by monitoring network traffic, analyzing packets, detecting network anomalies, and more. It will perform various tasks according to pre-set policies and algorithms, such as: 1) Network monitoring and fault detection: Intelligent operators will regularly scan network traffic, monitor equipment operating status, and detect any abnormal conditions; 2) Traffic management: It can dynamically adjust the network according to the traffic load and optimize the data transmission path to ensure the efficient operation of the network; 3) Security protection: Intelligent operators will monitor network security vulnerabilities and attacks in real time and take necessary measures to protect network security; 4) Automated tasks: It can perform a variety of automated tasks to reduce the workload of administrators.; 5) Performance optimization: Intelligent operators will provide optimization suggestions and adjustment plans based on network usage and performance indicators to improve the overall performance and stability of the network[2].

2.2. Intelligent operator

To achieve the goal of automating data collection and operator training, we need to integrate computer networks and neural networks together[4]. To achieve this, we can take the following steps to integrate computer networks and neural networks: First, we need to deploy data collectors in the network to capture network traffic data; Next, we design a neural network model suitable for network traffic management tasks. This model can be a deep reinforcement learning model, such as [5-7]Deep Q networks (DQN) or policy gradient methods, for learning optimal policies for network traffic management; Then, reinforcement learning algorithms, such as the experiential playback mechanism of deep Q networks, are then used to train neural network models from data collected in real time. During the training process, the neural network will constantly interact with the network, learn and adjust according to the actual network state and feedback signals, in order to improve its network traffic management ability; After the training is completed, the trained neural network model is deployed to the actual network environment. The intelligent operator will monitor network traffic in real time and make operational decisions based on the output of the neural network model. At the same time, on-the-ground data can be continuously collected for further optimization and adaptation of the neural network model to the changing network environment and needs[8]. Through this integrated approach, we achieve a close combination of computer networks and neural networks, enabling intelligent operators to collect and train data from the network in real time, thereby continuously improving their network management and optimization capabilities.

3. Methodology

This paper employs four key modules to implement the designed architecture: a computer network simulator, a computer network environment component, a trainer, and a tester. The simulator creates a simulation environment, diversifying data by randomly generating computer networks for training intelligent operators. The environment component acts as a bridge, collecting real-time data for processing by the operator. [9-10] The trainer module utilizes collected data and a pre-designed neural network model to train the operator via reinforcement learning algorithms, continuously refining its

DOI: 10.54254/2755-2721/64/20241370

strategy. Lastly, the tester module evaluates the operator's performance in real or simulated network environments, assessing its efficacy across various scenarios. Together, these modules form a comprehensive system for intelligent network management and optimization, crucial for building efficient network systems.

3.1. Computer network simulator

The simulator has the ability to randomly generate simple computer networks, which allows users to easily create networks with different topologies, thus increasing the flexibility and diversity of the system (Figure 1). In addition, the simulator also provides several other functions, such as real-time monitoring of network performance, simulating network failures and attacks, and evaluating network security. Through these functions, users can fully understand and evaluate the operation of the simulated network, so as to better understand the working principle and characteristics of the computer network.

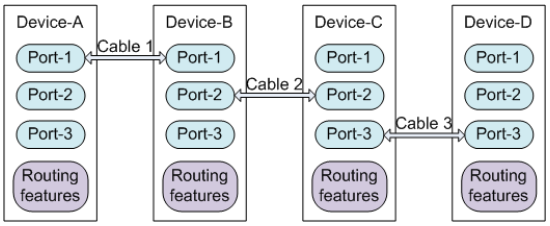


Figure 1. Computer network diagram

The computer network generated by the emulator shown in Figure 1 has the following characteristics:

a. The number of devices ranges from 4 to 10; **b.** Devices are connected through the tree topology; **c.** IP Subnet Select a subnet from a pool containing 32 subnets. For brevity, the subnets are named IP-Subnet-1 through IP-Subnet-32. The mask of a subnet is a 24-bit 255.255.255.0; **d.** Select an IP address from a pool of 63 IP addresses corresponding to each subnet. For brevity, IP addresses are named from ip-address -1 to ip-address -63. The full name of an IP Address is "ip-address -x in IP- subnet-y" (x and y are numbers). In this article, only "ip-address -x" is used to indicate the name of the IP Address, which is more concise and still valid because the IP Subnet to which the IP address belongs is fixed and immutable; **e.** Routing Protocol Select RIP version 2, EIGRP (AS 1), and OSPF (Process ID 1). The automatic summary function of RIP and EIGRP is disabled. The area of all OSPF subnets is 0; **f.** In this example network, the following six types of faults are randomly inserted:

Table 1. Device Status and Possible Reasons

Status	Possible Reasons
Port Closed	1. Port misconfiguration or not properly enabled.
	2. Port not specified in device configuration.
	3. Hardware fault or connectivity issues.
Incorrect IP Address	1. Static IP address misconfigured.
	2. Dynamic IP address allocation failure or misconfiguration.
	3. Subnet mask misconfigured.
Incorrect IP Subnet	1. Subnet mask misconfigured.
	2. IP address not matching the network of the device.
	3. Subnet mask not matching the network of the device.
Missing IP Subnet	1. Device not configured with an IP address.
	2. Device not joined to any network.
	3. Non-existent network or connectivity issues.
Automatic Summary Routing	1. Automatic summary function misconfigured.
	2. Router not enabled with automatic summary function.
	3. Erroneous summary routes in the static routing table.
Routing Protocol Version	1. Routing protocol misconfiguration.
	2. Protocol version mismatch between router and neighboring devices.
	3. Router not enabled with any routing protocol.

DOI: 10.54254/2755-2721/64/20241370

The above features are very important for any type of computer network, and more functions can be added in future work, such as: security functions, quality of service functions, traffic engineering functions, etc.

3.2. Computer network environment components

The environment component inherits all the functionality of the simulator and adds two additional features: an encoder that converts the observation information text into numbers (the observation information encoder) and an encoder that converts the observation information into embedded vectors (the observation information embedder). The observation information embedder embeds numeric information into an embedded vector consisting of floating-point numbers (vectors). This article uses numbers between 0 and 1 because the relationships between network information used in the environment components are relatively simple. The principle is to bring the embedding vectors to which the same kind of numeric information belongs close to each other and distance them from the embedding vectors of other classes, so that adjacent numbers will have some common feature or meaning. For example, IP addresses are classified as category 1, and during training, operators will be able to learn this feature. At each step, the environment component uses a portion of the computer network information (the corresponding information obtained from the initial network information dictionary and the current network information dictionary) as observation information, such as link state information for a port, or IP address information for a port. It loops through all network information until all problems are fixed or the maximum number of allowed steps is reached. The environment component also breaks down the task of generating complex instructions into substeps. For example, in order to configure the correct IP address, there are 3 sub-steps:

a. The operator calculates the observation information and outputs an instruction indicating that there is a problem with the port.

b. The operator calculates the observation information and outputs a command indicating the correct command.

c. The operator calculates the observation information and outputs an instruction, representing the parameters of the instruction.

The environment component then combines these three subinstructions into a complete network device instruction for further processing.

3.3. Practical application

In practice, the simulator randomly generates a simple computer network by first determining the number of devices (between 4 and 10) and assigning each device a name (e.g., Device-a). Then, it selects a root device and connects other devices to it one by one until all devices are connected. A routing protocol (RIP, EIGRP, or OSPF) is randomly chosen, and IP addresses and subnets are assigned to each device. Initial network information is generated, and faults are randomly injected into the network to create a dictionary of current network information. Additionally, the simulator includes a simplified network estimator component, which assigns rewards based on the correctness of the operator's actions: 1 for correct actions and -1 for incorrect ones.

4. Critical Loss Algorithm

4.1. Model training

The intelligent operator (neural network model) is randomly initialized at the beginning, at which point it can only randomly generate instructions. During training, the operator gradually learns the characteristics of the input data (observation information) and begins to generate more and more correct instructions. The process is described in detail as:

a. The environment component reads a portion of the initial network information dictionary and the current network information dictionary as text observations, encodes and emphases the observations

DOI: 10.54254/2755-2721/64/20241370

into an embedding vector (current observations), and then feeds it to the operator for analysis; b. The operator calculates the embedded vector and outputs an instruction (current instruction); c. The environment component decodes the instructions into network device instructions, applies them to the computer network, then evaluates new network information and issues rewards (current rewards. Correct instructions use the number 1, wrong instructions use the number -1). At the same time, the environment component generates the next embedding vector (the next observation information).; d. The data set collects and stores the current observation information, the current instruction, the current reward, and the next observation information as a sample in its buffer.; e. Repeat the above steps to collect a large number of samples into the data set. Some samples are positive (instructions are correct), others are negative (instructions are wrong); f. Use algorithms and datasets to train operators; g. The above steps can be performed multiple times until the operator is able to generate 100% correct instructions.

4.2. Critical loss function

The critical loss algorithm proposed in this paper is improved on the basis of QRDQN algorithm. The accuracy of the original QRDQN algorithm is around 80%, which means that about 20% of the output instructions are wrong. This is due to the following two characteristics of the neural network itself:

1) The neural network will output calculated data in the statistical sense, whose value represents the correct probability of a certain result. The higher the value, the greater the probability that the result is correct; 2) In the optimization process of the neural network, its calculation parameters will be fine-tuned, which brings the volatility of the output value and affects its accuracy.

4.3. Solution

In this project, we set two target values to distinguish correct and wrong instructions. These values range between 0 and 1, with a gap between them to effectively differentiate. Avoiding the boundaries of 0 and 1 is crucial due to precision issues. We assign 0.7 as the target value for correct instructions and 0.3 for incorrect ones. The difference between the neural network's output and these targets is termed loss. Critical losses, which affect instruction optimization significantly, are given greater weight during total loss calculation and network optimization, improving overall accuracy.

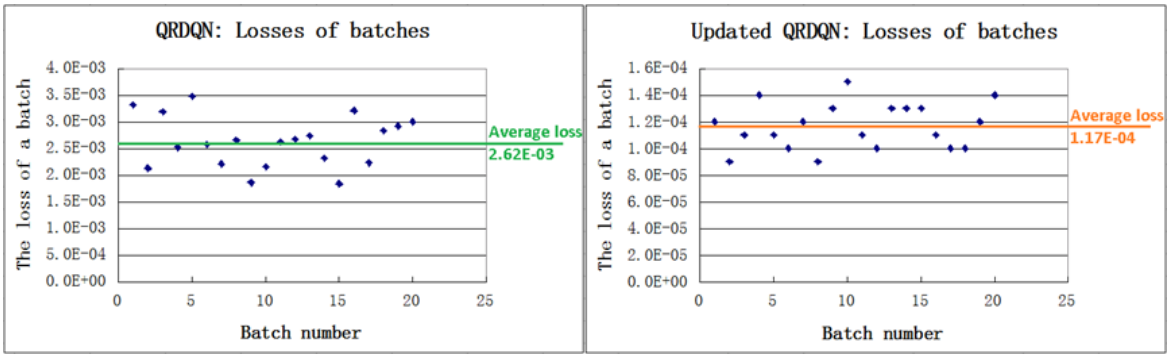


Figure 2. Ratio before and after critical loss

In the process of algorithm optimization, a neural network is configured to output 7 numerical values ranging between 0 and 1, representing the evaluation scores of instructions. The average of these scores determines the quality of an instruction, with higher averages indicating better instructions. Following the application of optimal instructions to the computer network, their effectiveness is evaluated. Instructions are categorized as correct or incorrect based on their outcomes. To calculate critical losses, four values are considered:

1) The target evaluation score for correct instructions is set at 0.7. Scores equal to or greater than 0.7 are associated with non-critical losses; 2) The safety threshold for the average evaluation score of correct instructions is 0.56. An average score equal to or greater than this value is also associated with non-critical losses; 3) The target evaluation score for incorrect instructions is set at 0.3. Scores equal to or

DOI: 10.54254/2755-2721/64/20241370

less than 0.3 are associated with non-critical losses; 4) The safety threshold for the average evaluation score of incorrect instructions is 0.44. An average score less than or equal to this value is associated with non-critical losses; 4) All other losses are classified as material losses. The Adam optimization algorithm is employed to optimize the neural network based on critical loss analysis. This optimization leads to a significant reduction in average loss, from 2.62E-03 to 1.17E-04, representing a 95.5% decrease. Further fine-tuning results in a 100% accuracy rate for output instructions (figure 2).

5. Conclusion

The intelligent operator, implemented as a neural network model, exhibits remarkable accuracy, achieving 100% accuracy across 28 issues and 237 operations per network, totaling over 2.8 million operations without error. Detailed output from these tests is available in the project documentation on GitHub. With processing times ranging from 1 to 5 seconds per network inspection and repair, the intelligent operator offers efficient solutions to a diverse range of computer network issues. It is noteworthy that initial loading of the operator takes approximately 10 seconds, and printing detailed information for all steps requires an additional 10 seconds[11-12]. In conclusion, this paper has successfully realized an AI-powered network operator, demonstrating its efficacy in efficiently addressing a variety of computer network issues. The integration of intelligent operators with computer networks and neural networks presents a promising and challenging avenue for future research and application in network management and optimization.

References

[1] Garrahan, James J., et al. "Intelligent network overview." IEEE Communications Magazine 31.3 (1993): 30-36.

[2] Robrock, Richard B. "The intelligent network-Changing the face of telecommunications." Proceedings of the IEEE 79.1 (1991): 7-20.

[3] Gong, Y., Huang, J., Liu, B., Xu, J., Wu, B., & Zhang, Y. (2024). Dynamic Resource Allocation for Virtual Machine Migration Optimization using Machine Learning. arXiv preprint arXiv:2403.13619.

[4] Zhang, Y., Liu, B., Gong, Y., Huang, J., Xu, J., & Wan, W. (2024). Application of Machine Learning Optimization in Cloud Computing Resource Scheduling and Management. arXiv preprint arXiv:2402.17216.Wang, Yong, et al. "Construction and application of artificial intelligence crowdsourcing map based on multi-track GPS data." arXiv preprint arXiv:2402.15796 (2024).

[5] Hosny, Manar I., and Christine L. Mumford. "The single vehicle pickup and delivery problem with time windows: intelligent operators for heuristic and metaheuristic algorithms." Journal of Heuristics 16 (2010): 417-439

[6] Xu, Z., Gong, Y., Zhou, Y., Bao, Q., & Qian, W. (2024). Enhancing Kubernetes Automated Scheduling with Deep Learning and Reinforcement Techniques for Large-Scale Cloud Computing Optimization. arXiv preprint arXiv:2403.07905.

[7] Xu, X., Xu, Z., Ling, Z., Jin, Z., & Du, S. (2024). Comprehensive Implementation of TextCNN for Enhanced Collaboration between Natural Language Processing and System Recommendation. arXiv preprint arXiv:2403.09718.

[8] Song, B., Xu, Y., & Wu, Y. (2024). ViTCN: Vision Transformer Contrastive Network For Reasoning. arXiv preprint arXiv:2403.09962.

[9] Zhou, Y., Tan, K., Shen, X., & He, Z. (2024). A Protein Structure Prediction Approach Leveraging Transformer and CNN Integration. arXiv preprint arXiv:2402.19095.

[10] Zhang, Chenwei, et al. "Enhanced User Interaction in Operating Systems through Machine Learning Language Models." arXiv preprint arXiv:2403.00806 (2024)

[11] Ni, Chunhe, et al. "Enhancing Cloud-Based Large Language Model Processing with Elasticsearch and Transformer Models." arXiv preprint arXiv:2403.00807 (2024).