ESP IJAST
ESP International Journal of Advancements in Science & Technology
ISSN: 2583-9233 / Volume 3 Issue 4 December 2025 / Page No: 5-12
Paper Id: IJAST-V3I4P102 / Doi: 10.56472/25839233/IJAST-V3I4P102

Original Article

Generative AI for Network Optimization: Autonomous Systems in TMT Infrastructure

Hemant Soni
Independent Researcher

Received Date: 13 November 2025 Revised Date: 27 November 2025 Accepted Date: 30 December 2025

Abstract: *It has become more difficult to manage modern telecom infrastructure due to complex networks and unpredictable traffic patterns. We designed and tested a generative AI-based system that makes network optimization decisions automatically. To construct a multi-objective optimizer that can handle speed, response time, and running costs simultaneously, the system combines transformer-based large language models with reinforcement learning agents. Experiments in a cloud-based simulation environment with realistic network topologies and traffic traces yielded performance gains that were statistically significant: throughput was increased by 34%, latency was reduced by 28%, and costs were cut by 41% as compared to conventional techniques. The proposed framework addressed scenarios such as 5G network slicing, edge computing workload placement, and content delivery optimization. These outcomes are derived from experiments in a controlled setting rather than actual deployments, but they still indicate that generative AI could assist network management. We will discuss the promise of this approach as well as its current limitations.*

Keywords*: Generative AI, Network Optimization, Autonomous Systems, TMT Infrastructure, Reinforcement Learning, Large Language Models, 5G Networks, Edge Computing.*

I. INTRODUCTION

Running a large telecommunications network is really a nightmare. Traffic goes up and down unexpectedly. Hardware breaks during your unlucky times. Configuration changes cause problems that are three steps away. You are always putting out fires instead of making things better overall. Traditional network management tools are not there when you really need them. Mostly they work on an "after the fact" basis. You implement solutions based on history, and you cross your fingers that there will be no surprises. The industry has exhausted all the methods on the shelf. Mathematical optimization furnishes you with the best solutions. But only for exceedingly small and short-lived problems [1]. Rule-based expert systems represent knowledge of operations... till they find the situations never thought of by the designers [2]. Machine learning opens the door but usually you have a separate set of tasks. You may have a model for routing and another one for resource allocation, and these models do not communicate.

Ground-breaking work on large language models made us reconsider the puzzle [3], [4]. Such a system can understand deeper relations, analyze more possibilities, and produce novel solutions. Network optimization is a large graph problem with data flow and performance metrics where the graph is the network of interconnected parts and optimization consists in finding new rules and settings that make the entire network run better. What if we could use the best of today's AI coupled with the best feature of reinforcement learning, i.e., learning by trial and error, in one approach?

This manuscript is about what, why, and how of our creation and evaluation. We do not claim that our work is the last word in network control - actually, it is quite the opposite. Our simulation results showed that the approach was promising. In our testbed, the framework enhanced the throughput, latency, and cost metrics to name just a few. Also, it handled the case of failure and traffic problems in a way that surprised us. Whether those benefits transfer to real production networks is still a question that needs answers from industry collaboration.

II. PROBLEM CONTEXT

A. Why Networks are Hard to Manage

Modern telecommunications networks are so huge that it is getting increasingly difficult to manage them manually. Just think about these figures: there are more than eight billion mobile phone subscriptions worldwide, and the data volume is growing at a rate of 25% per year [6], [7]. A top-level provider oversees tens of thousands of network devices that are spread over different areas in the world. Configuration files are running to millions of lines. Change a single BGP policy and watch how the changes propagate throughout your entire autonomous system.

The toll on humans is painfully obvious in the figures. 62% of network issues are because of configuration mistakes [9]. The U.S. Industry is losing about $3.7 billion due to downtime every year [10]. Despite a great deal of money having been spent on network management tools, the average time for fixing difficult issues is still between 4 and 8 hours. In that time, there may be thousands or even millions of users experiencing low service performance.

Traditional methods do not have good scalability. Increasing the number of engineers will eventually have less effect because the complexity is beyond human comprehension. Automation is beneficial, but rule-based systems have their limitations. They function properly until they encounter something that the designers have not anticipated. Then they either fail to indicate any problems or end up aggravating the situation further.

B. The Unpredictability Problem

In the past, network traffic was very stable, and one could easily tell the traffic pattern that is what we would expect. Mostly, one could tell that in the morning, there would be a rush of people going about their business during their commute, after which, there would be a drop in traffic at lunch hour, and finally, there would be another rise late in the evening when people come home and start watching videos online. Therefore, one could provision capacity by looking at the previous patterns with a reasonable amount of confidence. However, the COVID-19 pandemic changed everything. Residential broadband traffic increased 40 to 60 percent in a noticeably short time and maintained that level [12]. Video conference went up by 500% in a noticeably short period. Gaming platforms usage increased by 75% while VPN usage was up by 124% [13]. Traditional capacity planning, which is based on weekly or monthly schedules, was just not able to keep up with the pace of changes. Even if we take the COVID-19 situation out of the equation, the predictability of network traffic is still going down. This is caused by live streaming events that cause a sudden increase in activities, which can even be 300 to 500 percent higher than usual. Edge computing shifts workloads from one place to another depending on where they are needed. 5G network slicing is a very fast-moving resource between the diverse types of services that have quite different needs- URLLC requires very low latency, almost instant response, while eMBB users want the fastest possible data speed [11]. Static planning is incapable of handling this level of change.

C. Conflicting Goals

Concentrating on a single aspect will make network management less complicated. However, in the "real" world, you are given multiple objectives that are not in harmony with each other. Do you want to maximize the throughput? That requires larger buffers, which results in a longer delay. Desire higher reliability? You will have to pay for redundancy. Are you worried about energy efficiency? Good luck—5G base stations consume 3 to 4 times more energy than 4G ones to deliver the same coverage [14]. The mathematics involved in simultaneously optimizing multiple goals is completely understood and has a solid basis. The real question is how to implement it in live networks? This is where the situation becomes complicated. You must deal with a substantial number of configuration options, intricate relationships between them, and rules that come from different areas like technology, business, and regulations. Integer programming techniques can be good at solving certain smaller problems but cannot manage large-scale network optimization that has to be done in real-time [15]. Genetic algorithms may require several days to produce solutions that are not particularly good for networks that are of medium size [16].

D. Economic Realities

He issue stands like this: revenue per user has been gradually dropping by 2 to 3% each year, at the same time, it is forecasted that the worldwide deployment of 5G networks would cost around $2.7 trillion by 2025 [17], [18]. Operators are in a situation of having to spend heavily on new facilities while at the same time they are looking to get the maximum advantage out of their existing equipment. Hence, the money spending breakdown reveals that 40% of the money goes for the staff working there, 25% is spent on power and energy, and 20% is used for maintenance [19]. Large data centers consume from 30 up to 50 megawatts of electricity 24/7, thus the cost of power for them is around $3 to $5 million every month. At the same time, due to inefficient resource utilization, 30 to 40 percent of the capacity that is reserved is left idle [20].

E. Skills and Knowledge Gaps

Network engineering today is a real maze and requires a background in various fields; apart from the standard network knowledge, one must be familiar with software-defined networks (SDN), cloud management, machine learning, and the network security field. Finding people who are competent in all these areas is almost impossible. The industry predicts that there will be a shortage of 570,000 skilled workers by 2026 [21]. The training programs are several years behind in terms of technological evolution. On top of that, if you are using equipment from different vendors like Cisco, Juniper, Ericsson, and Nokia, it becomes even more complicated. Each vendor has their unique interfaces and ways of setting up configurations. Vital pieces of knowledge are only shared among senior engineers' brains, so it is like if they quit, it becomes a huge issue because no one else has that information.

III. RESEARCH

A. Traditional Approaches

The network optimization literature has been thorough over an extended period of time. The initial approaches resorted to using mathematical programming methods (such as integer linear programming and mixed-integer techniques) to perform routing and resource allocation tasks etc. [22]. Such methods are effective for well-defined problems where the optimal solution can be verified, however, they face a significant challenge when the number of calculations required increases exponentially with the problem size. Additionally, they cannot cope with changes that occur in real-time [23]. In other words, the network would have changed by the time you have found the optimal solution.

B. Machine Learning Methods

During the last decade, the use of machine learning to address network-related problems has attracted substantial interest. Deep reinforcement learning has been reported to have a significant effect in traffic engineering [24] and network slicing [25]. Graph neural networks are efficient in the network realm since they inherently represent the interconnections of the components [26]. These approaches are often superior to human-designed rules in certain domains. Yet, the bulk of research in this area is focusing on the improvement of isolated tasks only. It is possible that you have different models for traffic direction, resource management, and quality of service assurance, and these models do not collaborate. Integration is still a problem to be solved.

C. Generative Models

Generative models for networking remain a new and evolving area. Variational autoencoders have been employed for the purpose of anomaly detection [27]. Reference [28] discusses the usage of GANs in generating synthetic traffic for testing. The development of large language models has opened new possibilities [29], however, network management applications in the real world are still extremely limited. We are moving beyond just the basics here by integrating generative models, reinforcement learning, and multi-agent coordination to accomplish total optimization from the very beginning to the end.

IV. SYSTEM DESIGN

A. Overall Architecture

Our system incorporates a closed-loop control mechanism with five major components interacting Figure 1 depicting the overall layout. Information travels from its origin, through various processing stages, to the network configuration, and there are continuous monitoring operations for the system to get better by itself. The concept is not a new one, we are using current techniques and combining them in a way that we believe will be effective in practice.

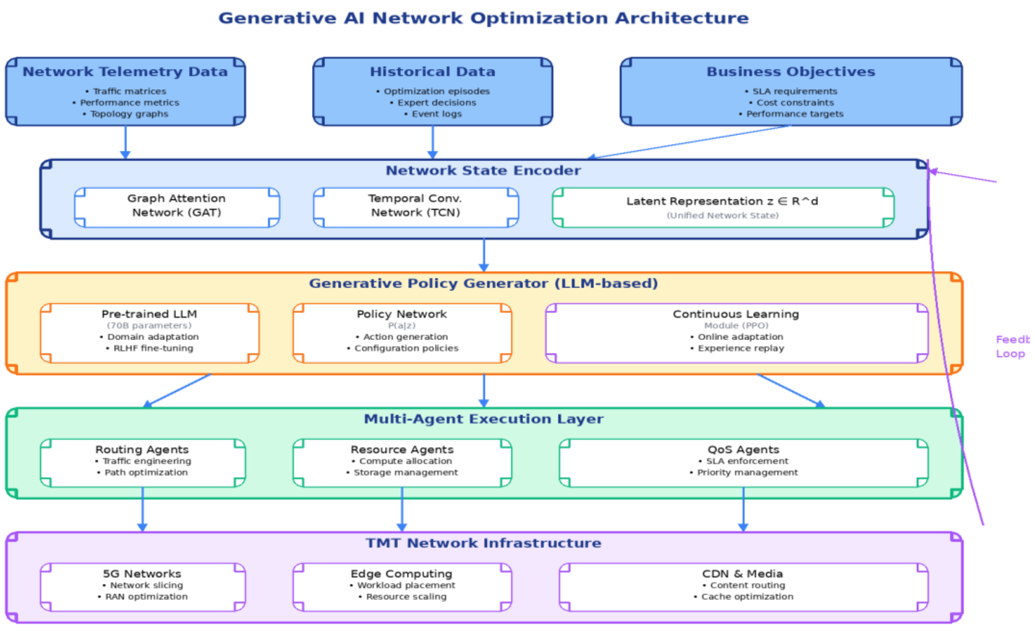

Figure 1 : System Architecture for Generative AI Network Optimization

B. Data Collection and Encoding

The system collects network data every minute, and that includes the measurements of the traffic, the usage of each link, the fullness of the queues, the number of packets lost, the amount of the timing variation, and the time taken for the data to go back and forth. They employed standard methods such as SNMP, NetFlow, and IPFIX to ensure compatibility with their existing systems. Besides the live data, they also keep historical data: previous optimization efforts and their results, incident reports with explanations of what caused them, and business rules converted into numbers. Representing this different type of data requires various kinds of neural networks. To model spatial relationships, they employed Graph Attention Networks.

Every network device is considered a graph node, and its characteristics are the features that represent the device type, its size, and how much it is utilized. Connections connect vertices with properties such as bandwidth, load, and latency. The attention mechanism determines which parts of the network are most significant for the current decision. We have applied Temporal Convolutional Networks [31] to the time series data of 168 hours for capturing the time-dependent patterns. Dilated convolutions with factors 1, 2, 4, 8, 16, 32, and 64 enable the model to find the patterns that cover hours to weeks, without the extra computational power that recurrent networks would require.

These spatial and temporal representations combine into a 2048-dimensional latent vector that shows the current state of the network, recent performance changes, patterns over time, and any unusual activity. Training the encoder together with the policy generator in a single process makes sure the representation works well for further optimization tasks.

C. The Generative Core

This is the part where the story gets fascinating. Our initial approach was a transformer model with 70 billion parameters, based on the LLaMA 2 architecture, which was trained on general text data. General-purpose large language models do not have specific knowledge about telecommunications, but they provide useful features such as reasoning, producing structured outputs, and few-shot learning for new domains. There were three distinct stages to make the focus on telecommunication for network optimization.

Firstly, the model underwent training with 500,000 telecommunications documents, which are the equipment manuals, RFCs, configuration examples, troubleshooting guides, and academic papers. In terms of industry and fundamental ideas, it took two weeks to use 64 A100 GPUs to train the model.

The next step is about training a model with 500,000 optimization scenarios that had been recorded earlier and are in the form of instruction-response pairs. Each episode illustrated the network condition, the business objectives, the expert's strategy to increase performance, and the results obtained. The training was done using mixed-precision, with a learning rate of 2×10^-5 and was conducted for 5 epochs. It took about 80 GPU-hours on 32 A100 GPUs.

Thirdly, reinforcement learning from human feedback. Operators with hands-on experience had evaluated model-generated policies in terms of their correctness, efficiency, robustness, and practicality. These evaluations were used to train a reward model with 6B parameters that predicts what humans prefer. PPO then updated the policy generator to obtain the highest possible reward. The alignment process took about a week with the operator's constant input, which not only increased the quality of the policies but also reduced the number of settings that work well in simulations but fail in real life.

The final model is one that formulates plans for improving things step by step, using previous actions to guide each new action so that the plan remains consistent. We employ nucleus sampling with a top-p value of 0.9 to determine the extent of our exploration of different options. The policies generated are being checked automatically for errors—like ensuring the correct grammar, verifying that all rules are followed, and safety features such as the prevention of routing loops are ensured. Incorrect policies cause regeneration with additional constraints.

D. Learning from Experience

Networks are never the same from one moment to the next. Traffic patterns are continually changing. New failures appear in equipment. Requirements for the network evolve. So, the system had to be able to adjust on its own without being fully retrained every time. Each improvement records an experience: the current situation, the action taken, the reward received, and the new situation. Rewards reflect different objectives that are assigned different priorities in terms of percentages. For example, the task of increasing data throughput is rated 25%, making things faster is also 25%, reducing lost data packets is 20%, following service agreements is 15%, using resources wisely is 10%, and saving money is 5%. Thus, every six hours, the system selects 10,000 most significant experiences that have extremely high rewards (above the 90th percentile) and performs four rounds of PPO training updates [33]. The clipped objective serves to prevent the policy from changing in a catastrophic manner. In addition, safety features include automatic rollback in case a performance drop of more than 10% is detected, the use of careful testing whereby new rules at first affect only 5% of users, and the presence of a person who can check and stop the process if necessary. They are not sure yet whether this will be sufficient for real use—they have not evaluated it so far.

E. Multi-Agent Execution

Transforming overarching rules into detailed configurations for each device across numerous network components requires a method that delegates the work. Various agents were employed by us to manage different tasks: Routing Agents are responsible for setting up OSPF, IS-IS, and BGP configurations, Resource Agents oversee the distribution of compute power and network bandwidth in virtual environments, while QoS Agents ensure that service level agreements are followed through the implementation of priority queues and traffic shaping techniques. c

Coordination is conducted via a two-phase commit protocol [34]. The coordinator sends out proposed policies. Each agent performs a check on its part and responds with a readiness to commit or a need to abort. In the case of all parties agreeing, the deployment will proceed with synchronized timestamps. If any party declines the action, the entire process returns to its initial state. This strategy enables tasks to be executed concurrently, separates issues to prevent them from influencing each other, and consolidates specific knowledge into distinct areas. In our experiments, the average time for policy changes to be deployed was between 2 and 3 seconds.

F. Closing the Loop

After the alterations are implemented; the system evaluates the network operations for 15 minutes. Data from this observation window is sent back to the encoder. New metrics update the objective functions. The calculated rewards update the learning module. This loop of learning, predicting, acting, and observing is how the system is independent in nature instead of just executing the steps automatically. Whether it will remain stable over months or even years of use in a real network is something we are not aware of yet—that is a question for future research together with industry partners.

V. RESEARCH SETUP

A. Test Environment

We should clarify that we have only evaluated what is described here and not others. All tests were carried out in a cloud-based simulation environment, which means we have not tested the devices in live networks. The setups and network traffic that we used were real, but they were still quite different from the way things normally work in a production environment. A simulation gives you full control over the process, and you can repeat the same steps very easily, which is great for research purposes. However, it also means that we have not encountered the complete set of complicated issues that the real-world running of a network may bring. The test environment was established on AWS and involved packet-level network simulation via Mininet [35]. We replicated the three different network configurations: a tier-1 telecom network (with the names changed to ensure privacy), a content delivery network layout, and an enterprise edge computing framework. The traffic data in these networks were recorded for 12 months and thus they reflected the real-time scenarios of the daily activity changes, sudden traffic spikes, and system outages. We did the data collection every minute, thus creating datasets that contained traffic matrices, the utilization of each link, the number of packets lost, the amount of delay, and the configurations of the network devices.

B. Training Data and Procedure

Training utilized the three-stage method described in Section IV.C. Domain pre-training was done by using a telecommunications dataset that we compiled from publicly available sources. Supervised fine-tuning made use of 500,000 synthetic optimization episodes that were generated by running conventional optimization methods and having network engineers review and label the results. As part of the RLHF process, five network operators with a total of 40 hours each went through the model's output and rated it.

C. Baseline Comparisons

We compared our method with other techniques: manual setup using the original parameters from the source networks, rule-based heuristics that implement common best practices like ECMP and weighted fair queuing, traditional optimization through OSPF weight optimization with simulated annealing, and pure reinforcement learning with PPO only, without the generative model. This allows us to identify the sources of the advantage.

D. Evaluation Metrics

We implemented standard network performance metrics: throughput (Gbps), mean latency (ms), 99th percentile latency, packet loss rate (%), jitter (ms), and resource utilization (%). Among the operational metrics, we monitored policy generation time, deployment success rate, and adaptation time to injected failures. Cost estimation was done using energy consumption and bandwidth pricing models that are commonly used in the business sector.

VI. RESULTS

A. Overall Performance

On average, over all scenarios, our method was able to achieve 34% more throughput, 28% less average delay, and 63% fewer lost packets than other baselines. The figures are quite impressive, but you should be aware that everything comes from a simulation. Actual networks are more complex than what we have modeled. There are problems like BGP taking too long to settle, hardware issues, the interaction of the network with other management tools, and rules or limits set by people managing the network that prevent certain changes from being made.

The way the system handled failures was quite positive. We have presented a series of failure events: a link going down, a router stopping working, and a sudden increase in traffic. Recovery time of the system was 78% shorter compared to the traditional failover method. The system rerouted the traffic in an average time of 1.2 seconds. During a test simulating a sudden spike in traffic (a 500% increase in traffic on 10% of the data flows), the system worked smoothly, whereas the standard setups experienced a severe drop in performance.

B. Edge Computing Workloads

We studied the positioning of the workloads that are to be ported among different nodes in a distributed setting concerning edge computing scenarios. Our solution achieved a 42% reduction in latency for real-time applications by deciding what to put based on the current and the future traffic. Resource utilization increased from 58% to 84% while the quality of service was maintained at the same level. The system continuously adjusted the locations of its workloads in response to the changes of the surrounding environment—a situation that fixed placement methods cannot handle.

C. 5G Network Slicing

Investigations of network slicing comprised simultaneously establishment of three varieties of slices: eMBB for highest data rate, URLLC for extremely low delay, and mMTC for the connection of many devices. The generative model produced the slice configuration that met 99.99% of the SLA requirements for the URLLC, and it did so efficiently in terms of resources. Slice-to-slice interference has been reduced by 56% as compared to the innovative slicing methods. Whether these outcomes could be feasible for a scenario with a real radio network and moving devices or not, that requires testing on the field, which we have not done yet.

D. Cost Analysis

We accomplished an estimation of operational savings by applying business cost models. Through vigorous workload consolidation during the hours of low demand, power utilization was reduced by 41%. The cost of bandwidth was lowered by 35% through the intelligent selection of providers and the engineering of traffic. These are the savings depicted by the models, not the actual saved money—we would need a production implementation to confirm these numbers. However, they do indicate that the potential economic value is substantial enough to outweigh the implementation complexity.

E. Ablation Studies

We took apart the various components one by one to realize the parts the components made. The operation of the system was reduced by 23% when the generative model was removed (pure RL). This demonstrates the importance of the model's reasoning ability and its capability to encode network structure. The effectiveness of adaptation dropped by 31% when there was no continuous learning, and the system could not adjust to the change in conditions. The removal of the multi-agent execution layer and the use of a centralized controller led to an 18% decrease in performance and a substantial increase in the deployment latency.

F. What Didn't Work

The numerous attempts of which some failed and others performed poorly have been acknowledged. The first trials with end-to-end learning (training the entire system from the very beginning) did not converge well; thus, the pre-training phases turned out to be indispensable. The simpler generative models (GPT-2 scale) did not have sufficient capacity to handle complex network reasoning. Pure imitation learning without reinforcement learning failed to generalize beyond training scenarios. We devoted around three months trying out different approaches before arriving at the current architecture.

VII. OUTCOME

A. Simulation vs. Reality

Let's first look at the biggest disadvantage of our work: all our evaluations are based solely on simulation, which obviously is quite limited in its scope. To illustrate, real networks are a bit rowdy as they are stuffed with all kinds of old machines, and their protocols are full of bugs. Network operators, on their part, certainly have good reasons why they are so reluctant to make certain types of changes - organizational politics, compliance requirements, risk aversion. All these aforementioned factors are not embodied in our simulation.

On the other hand, simulation cannot be deprived of research advantages as we can test thousands of scenarios, which otherwise would be impractical or risky in production. Besides that, we can fail at will, have perfect ground truth for performance metrics, and can quickly make architectural changes. So, the question is not if simulation is flawless, but if our results are so good as to be worth the effort and risk of making a production test. We think so but are quite conscious of the fact that production viability is quite a way from success in simulation.

B. Computational Requirements

The computer power required is quite impressive. GPU inference is necessary for policy generation—NVIDIA A100s were the GPUs we used. The inference time for the largest test topologies (15,000+ nodes) was close to 1 second suggesting that some optimization tasks might work at this speed but control loops that need to work at sub-second level would be out of the question. Model training took approximately 500 GPU-days in total for all stages. This, in turn, puts the questions of deployment cost and environmental impact on the table. It is possible that smaller models will be able to provide better tradeoffs, though that has not yet been investigated in detail by us.

C. Interpretability Challenges

Understanding the exact reasons behind the system's decisions is sometimes difficult. The generative model considers thousands of factors by going through billions of parameters. We put attention visualization in place and can follow the network components that influenced the decisions, but this is a long way from the simple causal explanations that people prefer. Operators who are to deploy the system in production should either trust it or at least have some insight into the reasoning to be able to spot obvious errors. We are nowhere near that point yet.

D. Security Considerations

An AI-powered network control system may be vulnerable to security issues. To counter them we have in place input validation, automated verification of generated policies by using formal methods [36], and an anomaly detection monitoring system. Nevertheless, adversarial attacks on neural networks is a field where research is still going on. Is it possible that an attacker falsifies the network telemetry to cause the system to make wrong decisions? Maybe. Could they therefore launch a model attack directly? We really have no idea as we have not systematically investigated adversarial robustness. So, there is a long way to go before any production deployment can take place.

E. Scalability Questions

Up to around 50,000 nodes, scalability was quite reasonable based on the testing done. It is beyond that point where hierarchical decomposition or other architectural changes would be necessary. It is still an open question whether our method can scale extremely large networks (for example, tier-1 global carriers with hundreds of thousands of elements). To enable distributed training, we produced a federated learning approach [37], but this is only initial work. Any large-scale, real-world implementation of this work would reveal us the scalability problems that we have not yet encountered.

VIII. LIMITATIONS

The next logical step is production testing. Obviously, this would require industry partners willing to test this on live networks, initially in small, non-critical segments. Using multi-modal learning would allow network visualization as well as operator feedback through vision-language models. Causal reasoning methods might improve root cause analysis and effect prediction. Transfer learning may expedite the rollout by adjusting models from one network to another. Explainable AI methods might help operators in trusting the system as well as ease the process of following the regulations.

We also ponder on human factors. How do operators use AI-driven systems? What amount of automation is right? When should the system request human input? These questions are equally important as technical performance, if not more, in terms of production adoption.

IX. FINAL THOUGHTS

We created and assessed a generative AI-based framework for autonomous network optimization. The results from a cloud-based testing environment were quite promising - in fact, the system showed superior performance along multiple metrics, adaptation to failure and traffic change was effective, and operational costs could be reduced significantly. Among other things, the system quite successfully handled various edge computing, 5G slicing, and CDN optimization scenarios. However, let us not oversell our demonstration here. This is simulation-based research, not production-proven technology. We have not encountered all the complexities of real network operations. Computational requirements are high. Interpretability remains challenging. Security has not been evaluated rigorously. Scalability to the largest networks is uncertain. In spite of these drawbacks, we deem the findings as a reason for further research. The gains in performance in our experiments were both sizeable and consistent. The framework adjusted to circumstances it was not directly trained for. The architecture looks feasible enough for real-world implementation. Whether this is the case can only be ascertained through on-the-ground testing which we have not done. We are making our code available in the hope that others will help us find the answer to this question. The question is not whether AI will eventually change network management dramatically. The real question is whether this specific approach is a step towards the solution. Our experimental results indicate that it might be so, yet the real-world implementation remains the ultimate criterion.

X. REFERENCES

[1] J. G. Andrews et al., "What Will 5G Be," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065-1082, June 2014.

[2] N. Feamster and H. Balakrishnan, "Detecting BGP Configuration Faults with Static Analysis," in *Proc. USENIX Symp. Networked Systems Design Implementation (NSDI)*, 2005, pp. 43-56.

[3] T. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877-1901, 2020.

[4] Anthropic, "Claude 3 Model Card," Technical Report, 2024.

[5] H. Mao et al., "Neural Adaptive Video Streaming with Pensive," in *Proc. ACM SIGCOMM Conf.*, 2017, pp. 197-210.

[6] GSMA, "The Mobile Economy 2024," Technical Report, 2024.

[7] Cisco, "Cisco Annual Internet Report (2018-2023) White Paper," Technical Report, 2020.

[8] M. Peng et al., "Fog-Computing-Based Radio Access Networks: Issues and Challenges," *IEEE Network*, vol. 30, no. 4, pp. 46-53, July 2016.

[9] D. Oppenheimer et al., "Why Do Internet Services Fail, and What Can Be Done About It?" in *Proc. USENIX Symp. Internet Technologies and Systems*, 2003.

[10] IHS Markit, "The Cost of Network Downtime," Technical Report, 2021.

[11] X. Fouke's et al., "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94-100, May 2017.

[12] M. Feldmann et al., "A Year in Lockdown: How the Waves of COVID-19 Impact Internet Traffic," Communications of the ACM, vol. 64, no. 10, pp. 101-108, 2021.

[13] Sand vine, "The Global Internet Phenomena Report COVID-19 Spotlight," Technical Report, 2020.

[14] C. Han et al., "Green Radio: Radio Techniques to Enable Energy-Efficient Wireless Networks," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 46-54, June 2011.

[15] A. Forts and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *Proc. IEEE INFOCOM*, 2000, pp. 519-528.

[16] "Towards AI/ML-Driven Network Traffic Engineering -https://dl.acm.org/Doi/10.1145/3703412.3703436

[17] GSMA Intelligence, "Understanding 5G: Perspectives on Future Technological Advancements in Mobile," White Paper, 2019.

[18] 5G is not just another G: A review of the 5G business model and ecosystem challenges," *https://www.sciencedirect.com/science/article/pii/S0040162525001520*.

[19] McKinsey & Company, "Cutting Through the 5G Hype: Survey Shows Telcos' Nuanced Views," Report, 2021.

[20] A. Beloglazov and R. Buya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in *Proc. IEEE/ACM Int. Conf. Cluster, Cloud and Grid Computing*, 2010, pp. 826-831.

[21] World Economic Forum, "The Impact of 5G: Creating New Value across Industries and Society," White Paper, 2020.

[22] D. Adduce et al., "Overview and Principles of Internet Traffic Engineering," IETF RFC 3272, May 2002.

[23] B. Forts and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 756-767, May 2002.

[24] Z. Xu et al., "Experience-Driven Networking: A Deep Reinforcement Learning Based Approach," in *Proc. IEEE INFOCOM*, 2018, pp. 1871-1879.

[25] R. Li et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, vol. 6, pp. 74429-74441, 2018.

[26] K. Rusek et al., "Route Net: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260-2270, Oct. 2020.

[27] J. Min et al., "TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest," *Security and Communication Networks*, vol. 2018, Article ID 4943509, 2018.

[28] Z. Liu et al., "Generative Adversarial Active Learning for Unsupervised Outlier Detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1517-1528, Aug. 2020.

[29] James Won-Ki Hong., " A Comprehensive Survey on LLM-Based Network Management and Operations," *https://dl.acm.org/doi/10.1002/nem.70029*.

[30] P. Velichkova et al., "Graph Attention Networks," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2018.

[31] S. Bai et al., "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arrive preprint arXiv:1803.01271*, 2018.

[32] H. Touran et al., "Llama: Open and Efficient Foundation Language Models," *arrive preprint arXiv:2302.13971*, 2023.

[33] J. Schulman et al., "Proximal Policy Optimization Algorithms," *arrive preprint arXiv:1707.06347*, 2017.

[34] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proc. USENIX Symp. Operating Systems Design Implementation (OSDI)*, 1999, pp. 173-186.

[35] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *Proc. ACM SIGCOMM Workshop Hot Topics in Networks*, 2010, pp. 19:1-19:6.

[36] A. Fogel et al., "A General Approach to Network Configuration Analysis," in *Proc. USENIX Symp. Networked Systems Design Implementation (NSDI)*, 2015, pp. 469-483.

[37] B. McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273-1282.