Automated Digital Twin Generation for Network Testing: A Multi-Topology Validation

Shenjia Ding
School of Computing Science
University of Glasgow
Glasgow, UK
2788155d@student.gla.ac.uk

David Flynn
School of Engineering
University of Glasgow
Glasgow, UK
david.flynn@glasgow.ac.uk

Paul Harvey
School of Computing Science
University of Glasgow
Glasgow, UK
paul.harvey@glasgow.ac.uk

***Abstract*—With the exponential growth of data traffic, ensuring reliable and efficient network testing has become critical throughout the design, implementation, and management operations. As testing becomes essential to ensure that changes in configuration or traffic conditions do not degrade user experience, current testing practices rely heavily on manual configuration and simulators. This reliance lead to time-consuming, difficult-to-scale, and expert-dependent processes. To address these limitations, our work explores the role of Automated Machine Learning (AutoML)–based automatically generated Digital Twin (DT) in network testing to enable rapid and scalable testing across diverse network conditions. By integrating this approach with a network service controller for configuration optimization, our results evidence an improvement that DT-enabled testing achieves high accuracy while being approximately 25,000 times faster than simulator-based testing. The implications of these findings, suggest that automated DT generation through AutoML can reduce dependence on manual modeling, allow DTs to adapt to diverse test scenarios, and enhance scalability for complex network.**

***Index Terms*—Digital Twin, Automated Machine Learning, Network Testing**

I. Introduction

As more people rely on networked services, the total number of Internet users is projected to grow from 3.9 billion in 2018 to 5.3 billion by 2023, at a Compound Annual Growth Rate (CAGR) of 6 percent, highlighting the urgent need for scalable, reliable, and adaptable communication networks [1]. This escalating pressure on the network has transformed the design of Network Management and Operations (MANO) systems, where software now plays a central role [2]. However, this evolution also brings new challenges: software must undergo testing to guarantee safe and reliable operation. Such testing processes are often manual and time-consuming, leading to limited testing speed and reduced evaluation efficiency.

Traditional testing tools (Section II-B) struggle to meet the needs of highly dynamic networks [3]. These tools often require extensive manual effort in both configuration and use, are limited in scalability, and have low efficiency on comprehensively addressing the wide range of parameters and test combinations encountered in real-world systems [4]. As a result, the testing process is not only time-consuming but also slow and inefficient, particularly when applied to complex network topologies [5].

Digital Twin (DT) technology [6] has emerged as a promising solution for network software validation. A DT is a digital replica of a physical system or process that enables real-world simulation, monitoring, and optimization. Early adopters of Digital Twins include sectors such as the built environment. With emergent applications in healthcare, energy networks etc for both design, planning and operational decision support [7]. In the context of network services, DTs can provide a flexible testing environment by simulating diverse configurations, predicting outcomes, and supporting decision-making without disturbing the live system.

Despite their potential, a key limitation of DTs lies in the inefficiency and manual nature of their creation. For each testing scenario, such as latency prediction or QoS optimization, large amounts of representative data must be collected and new models must be trained, often with bespoke architectures [8], [9]. For instance, Machine Learning (ML) is used as a support technology within Software-Defined Networking (SDN) platforms to detect vulnerabilities and monitor network activities [10]. In wireless networks, ML has been employed for traffic classification, particularly to test unseen data during the testing phase [11]. The lack of automation forces operators to either retrain models frequently or maintain multiple DT various, reducing practicality and scalability [12]. As a consequence, the overhead of maintaining, validating, and deploying several DTs causes both time and computational costs to increase substantially, which in turn constrains the practicality of *ML-based* DTs in dynamic or large-scale environments.

To overcome the human-centric process of creating and maintaining DTs, we propose the *automatic generation of digital twins* (Section III). These DTs then support network software validation and testing, where each DT is tailored to specific network operational scenarios. The proposed approach (i) reduces the reliance on manual model design, (ii) improves the scalability of DT-based testing, and (iii) accelerates the validation process across diverse and complex network scenarios. By introducing automation into the generation process, we enhance both the efficiency and adaptability of DTs, making them a more practical tool for modern network service management.

To substantiate this approach, the paper is organized as follows. Section II reviews network testing platforms and Digital

Twins. Section III presents our AutoML-based DT framework. Section IV covers the experimental setup, Section V analyzes performance, Section V-D demonstrates DT applications, and Section VII concludes with future directions.

II. Background

A. Softwarization

Software-Defined Networking (SDN) [13] and Network Function Virtualization (NFV) [14] have redefined how networks are designed and managed. These approaches promote logical centralization of control and introduce "softwarization" [2] of network functions, enabling scalable and elastic operation.

Softwarization is not only a design trend but also a critical enabler of service assurance. By shifting control from rigid hardware appliances to software-based systems, networks gain the ability to be dynamically configured, monitored, and optimized [2]. In SDN, for example, operators can programmatically adapt routing policies and resource allocation to ensure Quality of Service (QoS) even under changing traffic conditions [13]. From this perspective, software-based *controllers* enhance the capability of networks to deliver consistent, scalable, and adaptive services.

Despite these advances, testing and validation remain critical bottlenecks. While the integration of software into controllers significantly strengthens the ability to ensure network service quality, it also increases the complexity of verification [15]. Current validation frameworks are largely manual and resource-intensive [16], limiting their applicability to large-scale networks [12]. This creates an urgent need for systematic, automated testing methodologies that can match the scale and dynamism of modern softwarized infrastructures.

B. Network Testing Tool and Platform

With the rise of software-based network control, testing management and orchestration software is crucial for ensuring reliability, efficiency, and compliance with performance standards. QoS frameworks use end-to-end latency to evaluate service quality, guiding network engineering and Service Level Agreements (SLAs) compliance [17]. However, both hardware and software solutions have drawbacks that increase costs.

1) Hardware Platform: Hardware testbeds are composed of purpose-built devices specifically designed for network testing. However, they incur high costs and exhibit limited flexibility, as reconfiguration for new testing or monitoring functions is often impractical [18]. As network requirements continue to evolve, these constraints hinder scalability and delay the deployment of necessary updates.

2) Software: Simulation or emulation platforms, such as ns-3 [19], offer greater flexibility and lower cost than hardware testbeds. They eliminate physical hardware requirements, allow rapid parameter adjustments, and provide reproducible environments for diverse network experiments. However, their scalability is limited: as the number of test parameters increases, configuration becomes labor-intensive and execution time grows substantially, restricting efficient exploration of large-scale scenarios.

TABLE I: Comparison of Network Testing Methods

Testing Method	Accuracy	Speed	Config Flexibility
Physical Testbed	High	Low	Low
Simulation	Medium	Medium-High	High
Digital Twin	Variable	High	Medium-High

Beyond the scalability constraints of general-purpose simulators, these limitations also extend to more specialized testing tools. For instance, in the case of OpenRASE [20], the time required per experiment—notably around ten minutes—becomes a critical bottleneck, particularly when the number of experiments increases. This demonstrates that the challenge of increasing time consumption under scaling workloads is a pervasive issue across both general and specialized network testing paradigms.

C. Network Digital Twin

In networking, DTs have emerged as powerful tools for monitoring, analysis, and management [6]. Unlike traditional simulators, a Network Digital Twin (NDT) places a greater focus on capturing the realism of actual network operations, allowing accurate replication of topologies, traffic, and behaviors. This makes NDTs especially promising as testing environments where scenarios can be validated without affecting live systems [21].

Recent studies illustrate this trend: NDTs have been applied to QoS prediction [22], and optimization [23], often enhanced by machine learning, giving rise to ML-based DTs that integrate predictive or adaptive intelligence. Deep and reinforcement learning approaches further expand these capabilities, enabling faster simulations and scalable optimization [6].

However, the practical application of ML-based DTs is often hampered by the inefficiency of the generation process. For each specific testing objective, constructing an ML-based DT requires collecting representative data and training a tailored model, demanding extensive manual effort and domain expertise. Tasks such as data interpretation, problem formulation and model selection remain largely human-driven [24]. This becomes particularly burdensome when testing requirements evolve, since each scenario demands a separate modeling cycle [9]. Moreover, current ML-based DTs struggle to quantify prediction uncertainty, producing overconfident outputs without reliability estimates [25]. The lack of confidence intervals reduces operator trust and necessitates frequent validation or retraining, as different models are often optimal for different testing scenarios [12].

There is, therefore, a clear need for methods that can streamline the creation of ML-based Digital Twins across diverse network testing scenarios.

III. Automated ML-based DT Generation by AutoML

To overcome the challenges of manual design and time efficiency, we explore the use of automated machine learning

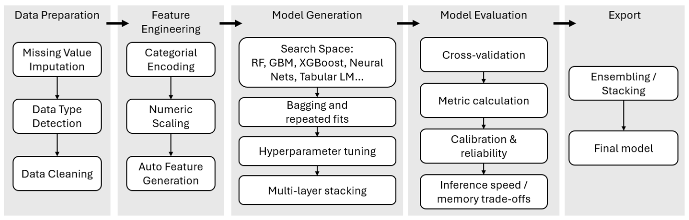

Fig. 1: AutoML framework illustrated with AutoGluon [26]

(AutoML [26]) in the automatic creation of ML-based DTs. By automating the selection of optimal architectures and hyperparameters, AutoML reducing human involvement and enables scalable, adaptive testing across complex topologies.

By reducing training costs, automating model comparison, and lowering technical barriers for practitioners, AutoML provides a practical pathway toward automated ML-based DT construction and validation.

To realize this vision, our experimental design was structured to evaluate three complementary perspectives. First, to ensure that the DT approach can scale beyond simplified scenarios, experiments were conducted on complex topologies to assess the ability of DTs to capture diverse path behaviors. Second, to examine robustness, Gaussian noise was injected into the simulator-generated datasets to emulate real-world measurement uncertainties, enabling analysis of how the accuracy of network DT is affected under imperfect data conditions. Third, the generated DTs were integrated into a controller to validate their utility for parameter optimization, where the objective was to minimize end-to-end latency. Together, these three stages guarantee that the evaluation covers scalability, adaptability, and practical applicability of DTs in realistic network management settings.

IV. Experimental Design

Building on our proof of principle for a simple 4 node diamond topology [27], we now extend the work to consider more complex topologies and scenarios with various background and application traffic shown in Figures 2 and 3.

A. Workflow

Our workflow is structured into a 3 stage approach: data collection, model training, and evaluation. First, network performance data is generated using simulation (Section IV-C), ensuring that controlled conditions are represented. Next, predictive models are trained using the AutoML framework on the full dataset, eliminating the need for manual model selection and hyperparameter tuning. Finally, the trained models are evaluated and integrated into an optimization framework to identify parameter configurations that minimize latency. This step-by-step process ensures both accuracy and efficiency, providing a comprehensive pipeline from dataset generation to optimization.

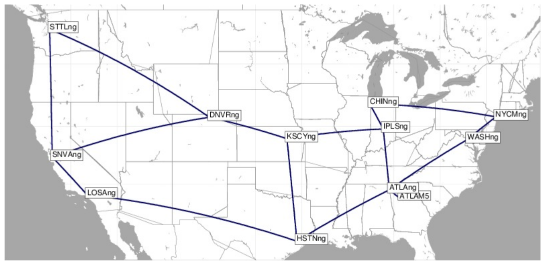

Fig. 2: USA topology [28]

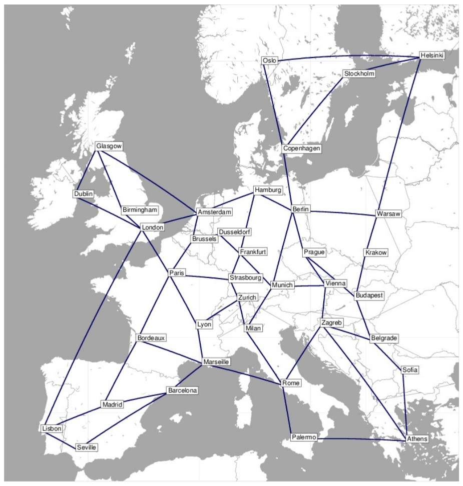

Fig. 3: EUR topology [28]

B. Network Topology

To evaluate the proposed approach under complex scenarios, experiments utilize network topologies from two sources: the Abilene network from SNDlib [28] (Figure 1, hereafter referred to as the USA topology) and the cost266 network (Figure 2, hereafter referred to as the EUR topology). The USA topology consists of 12 nodes and 15 links, while the EUR topology comprises 37 nodes and 57 links. Compared with the simple diamond topology, these configurations introduce significantly more routing paths, increasing the number of configurable network parameters to be optimized from 12 to 114.

C. Dataset Collection

With USA and EUR topologies, datasets were generated using the ns-3 simulator [19], simulating six application types: TCP file download (100 MB), web browsing, video streaming, and their pairwise combinations, with latency as the primary metric. To emulate realistic conditions, all applications operated alongside background traffic consisting of continuous TCP downloads occupying 60% of available bandwidth. The selected applications capture diverse traffic behaviors—throughput-oriented (download), bursty and delay-sensitive (web), and sustained with strict latency and jitter constraints (video). Pairwise combinations further reflect the mixed-traffic dynamics typical of real-world networks.

Link bandwidths ranged from 25–125 Mbps and queue sizes from 25–125 packets. Latency measurements were recorded as the primary performance metric. For the USA topology, latency was collected exhaustively across all node pairs, while the EU topology sampled 190 representative city pairs, reflecting the practical constraints of real-world datasets, where exhaustive measurements are often unavailable [29].

D. Dataset

The USA and EU datasets are constructed to model end-to-end latency between geographically distributed cities, differing in scale and input dimensionality. For both datasets, input features consist of the queue size and bandwidth of each network link, capturing the core network parameters that determine latency. The output labels correspond to the end-to-end latency for all city pairs, measured in milliseconds.

The USA dataset contains 30 input features and 132 output variables, representing all city-pair latencies, while the EU dataset contains 114 input features and 190 output variables. Six application scenarios were simulated to represent diverse traffic behaviors: file download (App1), video streaming (App2), web browsing (App3), and their pairwise combinations (Apps 4–6).

For model training, both datasets were randomly split into training, validation, and test sets. The complementary design of these datasets allows evaluation under both small- and medium-scale networks and challenges models to generalize under different network conditions.

E. Noise Injection

Latency measurements in real networks are inherently noisy due to factors such as congestion, background traffic, and measurement errors. Training models only on noise-free data can overestimate performance and reduce generalization. Therefore, we introduce noise into the target values to evaluate model robustness under realistic conditions and to assess whether the model can still provide accurate point predictions.

Gaussian noise is commonly used to model aggregated random disturbances because, by the central limit theorem, the sum of many independent small errors tends to be normally distributed. It provides a controlled and unbiased way to vary the signal-to-noise ratio, making it suitable for evaluating model robustness under realistic measurement uncertainty.

F. Model Training

The generated datasets were used to train predictive models with AutoGluon [26], an automated machine learning framework for tabular data. AutoGluon automates preprocessing, model selection, and ensemble construction, combining multiple models from a diverse model zoo via bagging and stacking to produce robust predictions. In this study, it reduces expert intervention by automatically selecting and combining models.

G. DT-based Network Controller Testing

To demonstrate how ML-based DTs can move beyond stand-alone predictors toward practical testing and control, and aligned with the ITU-T autonomous network framework [30], the DTs were integrated with a network configuration *controller*. The controller identifies parameter configurations that minimize end-to-end latency, addressing the ongoing challenge of dynamic network optimization where parameters such as bandwidth and queue size must adapt to changing conditions. Two evaluation modes were considered: direct simulation using ns-3 and predictive evaluation using DTs.

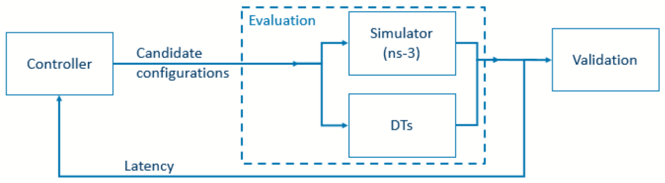

Fig. 4: Experimental design and practical workflow

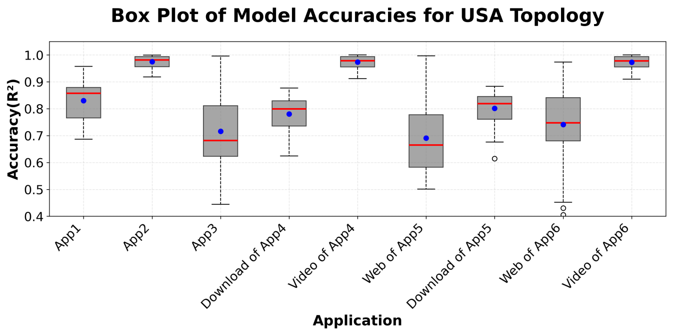

Fig. 5: Box plot for Model Accuracies of multi application prediction on paths of USA topology

V. Performance Analysis on Original and Noise-Augmented Data

A. Performance in ideal conditions

Model performance was evaluated using the coefficient of determination (R^2), which measures the proportion of variance in the observed latency explained by the model. A higher R^2 score indicates greater predictive accuracy.

Figure 5 and Figure 6 summarizes the average prediction accuracy across the six application scenarios for both USA and EUR topologies. The box plots display the interquartile range, with the median as the central line and whiskers indicating variability; blue dots denote mean R^2 values. AutoGluon achieved high accuracy, ranging from 0.7161 to 0.9766 on the USA dataset and from 0.7175 to 0.9765 on the EUR dataset. Video streaming (App 2) attained the highest accuracy in both cases ($\approx$0.976), reflecting its stable traffic pattern, whereas web browsing (App 3) showed greater variability (USA: 0.7161; EUR: 0.9718), underscoring the difficulty in modeling bursty traffic flows.

Forecasting accuracy for web traffic is slightly lower than for download and video due to its high variability and bursty, user-driven nature. In contrast, the stability of download and video flows enables consistently higher accuracy. Despite this variability, the ML-based DTs perform well across all applications, demonstrating their ability to capture diverse network behaviors.

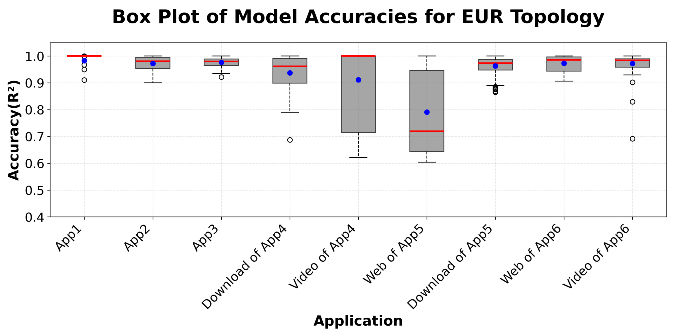

Fig. 6: Box plot for Model Accuracies of multi applications prediction on paths of EUR topology

Overall, integrating ns-3–based dataset generation with AutoML produced an automated and efficient DT generation process, delivering strong accuracy while eliminating manual feature engineering and model tuning, thus accelerating DT development and reducing technical effort.

B. Performance in noisy conditions

In addition to the clean datasets directly from ns-3, we introduced Gaussian noise into the latency measurements to more closely approximate real-world conditions and understand the ability of AutoML to handle this. While simulation provides precise values, actual network measurements are often affected by random fluctuations, hardware limitations, and transient congestion [31].

TABLE II: Average performance on Noisy dataset for USA topology

USA	App 1	App 2	App 3	App 4	App 5	App 6
R²	0.8788	0.8715	0.6850	0.8650	0.7221	0.8414

TABLE III: Average performance on Noisy dataset for EUR topology

EUR	App 1	App 2	App 3	App 4	App 5	App 6-
R²	0.2133	-0.0076	0.0354	0.09334	0.1477	0.0310
MAE	0.0861	0.0813	0.0830	0.0807	0.0795	0.0778

The noisy datasets were subsequently used to retrain the models with AutoGluon, employing the same training procedure as in the clean case. Performance was evaluated using R², enabling direct comparison between the clean and noise-augmented scenarios. Table II show that although they experienced a moderate reduction in predictive accuracy, they continued to achieve strong performance overall. This indicates that in this case, ensemble-based AutoML methods are not only robust to data imperfections but also capable of learning stable predictive models from datasets that more closely resemble real-world measurements.

For the EUR topology, as shown in Table III, R² went from 0.9 on the original data test set to close to 0 on the noised data set. However, the mean absolute error (MAE) remained relatively low, indicating that the model's predictions were still numerically close to the observed outcomes. This suggests that although the noisy labels limit the model's ability to explain overall variance in the dataset, the model can still deliver accurate point predictions within the noise margin. Since prediction accuracy at the individual sample level is more relevant for our task than explaining variance across the entire dataset, the model remains useful under noisy conditions.

C. Discussion

The different behavior between the USA and EUR datasets can be partly explained by their structural differences. The USA dataset contains 30 input features and 132 outputs, corresponding to the complete set of latencies between all city pairs. This full coverage creates redundancy in the data, as many outputs share underlying structural relationships. Even when Gaussian noise is added, these consistent patterns remain detectable, and in some cases the noise may even act as a form of regularization, preventing overfitting and improving generalization. In contrast, the EUR dataset consists of 114 inputs and 190 outputs, where the outputs represent latencies from a randomly selected subset of city pairs rather than the full set. This partial and irregular coverage reduces redundancy and weakens the underlying signal, making the dataset more sensitive to noise. Moreover, the higher input dimensionality in the EUR dataset increases complexity, allowing the model to fit noise rather than true structure. Consequently, while the USA dataset remains robust under noisy conditions, the EUR dataset experiences a sharper decline in R², with the model retaining low absolute errors but failing to explain much of the variance.

D. DT-based Controller Testing

To evaluate efficiency, we conducted a comparative experiment between the ML-based DT and the ns-3 simulator within the same optimization framework. The network controller, implemented using a Genetic Algorithm (GA), was tasked with minimizing end-to-end latency by tuning bandwidth and queue size configurations. GA was selected for its ability to handle the large, nonlinear, and non-differentiable search space of 114 parameters. In this setup, two optimization modes were tested: one using ns-3 for direct simulation and another using the trained DT for predictive evaluation. This comparison enables quantitative analysis of computational efficiency, showing how the DT can achieve similar optimization outcomes as ns-3 but with dramatically reduced evaluation time.

The times taken for the controller to converge on a stable solution were 33 hours for ns-3 and 4.78 seconds respectively. In both cases, the controller successfully identified parameter configurations that achieved the same minimum end-to-end latency (precisely matched within 0.1 s). This significant time saving for model-based DT evaluation for a solution space of by 114 parameters demonstrates the power of automatically generated DTs to support network validation and testing.

VI. Future Work

Future work will focus on validating the DT's update mechanisms and cost, assessing performance in real-time network environments, and conducting a comparative study of model selection across diverse network scenarios. These efforts aim to enhance prediction accuracy, robustness, and scenario-specific adaptability for practical deployment.

VII. Conclusion

In this paper, we proposed a method for the automated generation of ML-based DTs to accelerate testing and improve efficiency in the network control software. Our approach reduces the dependence on manual modeling, adapts to diverse test scenarios, and improves scalability for complex networks.

The experimental findings confirm the framework's effectiveness, showing robustness and adaptability, even with the introduction of noise to the dataset. Also, the DT-based testing demonstrated substantial performance gains, converging on a stable solution approximately 25,000 times faster than the conventional simulator approach while maintaining high prediction accuracy. These results underscore the potential of automated DT generation as a practical, scalable, and cost-effective solution for future network testing and management.

Acknowledgment

This work is supported by EPSRC EP/Z533221/1.

References

[1] C. Annual and I. Report, "Cisco Annual Internet Report (2018–2023)," tech. rep., 2018.

[2] A. Barakabitze and A. Hines, "Network Softwarization and Virtualization in Future Networks: The promise of SDN, NFV, MEC, and Fog/Cloud Computing," in *Multimedia Streaming in SDN/NFV and 5G Networks*, pp. 99–118, Wiley, 12 2022.

[3] Y. Li, X. Yin, Z. Wang, J. Yao, X. Shi, J. Wu, H. Zhang, and Q. Wang, "A Survey on Network Verification and Testing With Formal Methods: Approaches and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 940–969, 1 2019.

[4] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A Survey on Large-Scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 891–917, 4 2017.

[5] O. Flauzac, E. M. Gallegos Robledo, and F. Nolot, "Is Mininet the Right Solution for an SDN Testbed?," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 12 2019.

[6] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Network Digital Twin: Context, Enabling Technologies and Opportunities," *IEEE Communications Magazine*, vol. 60, pp. 22–27, 5 2022.

[7] P. D. E. Baniqued, P. Bremner, M. Sandison, S. Harper, S. Agrawal, J. Bolarinwa, J. Blanche, Z. Jiang, T. Johnson, D. Mitchell, E. J. L. Pulgarin, A. West, M. Willis, K. Yao, D. Flynn, M. Giuliani, K. Groves, B. Lennox, and S. Watson, "Multimodal immersive digital twin platform for cyber–physical robot fleets in nuclear environments," *Journal of Field Robotics*, vol. 41, pp. 1521–1540, 8 2024.

[8] Y. Li, X. Yin, Z. Wang, J. Yao, X. Shi, J. Wu, H. Zhang, and Q. Wang, "A Survey on Network Verification and Testing With Formal Methods: Approaches and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 940–969, 1 2019.

[9] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula, and M. Picone, "Application-Driven Network-Aware Digital Twin Management in Industrial Edge Environments," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 7791–7801, 11 2021.

[10] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 493–501, 3 2019.

[11] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Communications*, vol. 24, pp. 98–105, 4 2017.

[12] R. H. Serag, M. S. Abdalzaher, H. A. E. A. Elsayed, and M. Sobh, "Software Defined Network Traffic Classification for QoS Optimization Using Machine Learning," *Journal of Network and Systems Management*, vol. 33, p. 41, 4 2025.

[13] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, 2 2017.

[14] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-art and Research Challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, pp. 236–262, 9 2015.

[15] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. ElBadawi, "Network configuration in a box: towards end-to-end verification of network reachability and security," in *2009 17th IEEE International Conference on Network Protocols*, pp. 123–132, IEEE, 10 2009.

[16] C. Zheng, X. Hong, D. Ding, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "In-Network Machine Learning Using Programmable Network Devices: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 1171–1200, 2024.

[17] M. Adnan Khan, A. Kanwal, S. Abbas, F. Khan, and T. Whangbo, "Intelligent Model for Predicting the Quality of Services Violation," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3607–3619, 2022.

[18] R. Kundel, F. Siegmund, R. Hark, A. Rizk, and B. Koldehofe, "Network Testing Utilizing Programmable Network Hardware," *IEEE Communications Magazine*, vol. 60, pp. 12–17, 2 2022.

[19] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, pp. 15–34, Springer, 2010.

[20] T. Krishnamohan and P. Harvey, "OpenRASE: Service Function Chain Emulation," *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)[to appear]*, 7 2025.

[21] A. M. Madni, D. Erwin, S. K. Purohit, and C. C. Madni, "Digital-Twin Enabled Experimentation Testbed for MBSE," in *AIAA Scitech 2021 Forum*, (Reston, Virginia), pp. 1–12, American Institute of Aeronautics and Astronautics, 1 2021.

[22] M. Saravanan, P. S. Kumar, and A. R. Kumar, "Enabling Network Digital Twin to improve QoS Performance in Communication Networks," in *2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta)*, pp. 2151–2160, IEEE, 12 2022.

[23] P. Almasan, M. Ferriol-Galmes, J. Paillisse, J. Suarez-Varela, D. Perino, D. Lopez, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Network Digital Twin: Context, Enabling Technologies, and Opportunities," *IEEE Communications Magazine*, vol. 60, pp. 22–27, 11 2022.

[24] S. K. K. Santu, M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni, "AutoML to Date and Beyond: Challenges and Opportunities," 11 2022.

[25] A. Thelen, X. Zhang, O. Fink, Y. Lu, S. Ghosh, B. D. Youn, M. D. Todd, S. Mahadevan, C. Hu, and Z. Hu, "A comprehensive review of digital twin—part 2: roles of uncertainty quantification and optimization, a battery digital twin, and perspectives," 1 2023.

[26] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," 3 2020.

[27] S. Ding and P. Harvey, "Automatic Generation of Digital Twins for Network Testing," in *Proceedings of the 4th International Workshop on Autonomous Network Management in 5G and Beyond Systems (ANMS 2025)*, 2025.

[28] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib: Survivable Network Design Library." https://sndlib.put.poznan.pl/, 2007.

[29] X. Zhang and C. Phillips, "A Survey on Selective Routing Topology Inference Through Active Probing," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1129–1141, 2012.

[30] "Autonomous Networks -Architecture Framework," tech. rep., United Nations International Telecommunication Union, Geneva, CH, 12 2023.

[31] D. De Sensi, T. De Matteis, K. Taranov, S. Di Girolamo, T. Rahn, and T. Hoefler, "Noise in the Clouds: Influence of Network Performance }Variability on Application Scalability," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, pp. 1–27, 12 2022.