Automatic Generation of Digital Twins for Network Testing

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*—The increased use of software in the operation and management of telecommunication networks has moved the industry one step closer to realizing autonomous network operation. One consequence of this shift is the significantly increased need for testing and validation before such software can be deployed. Complementing existing simulation or hardware-based approaches, digital twins present an environment to achieve this testing; however, they require significant time and human effort to configure and execute.**

This paper explores the automatic generation of digital twins to provide efficient and accurate validation tools, aligned to the ITU-T autonomous network architecture's *experimentation subsystem*. We present experimental results for an initial use case, demonstrating that the approach is feasible in automatically creating efficient digital twins with sufficient accuracy to be included as part of existing validation pipelines.

***Index Terms*—Digital Twin, AutoML, Network Testing, Autonomous Network**

I. INTRODUCTION

Autonomous networks represent the holy grail of network and service management, aiming to achieve self-configuring, self-optimizing, and self-healing capabilities with minimal human intervention [1]. Recent industry motivation for autonomous networks is driven by increasing complexity of networks, particularly with the advent of 5G and beyond, which demand more agile, scalable, and efficient management [2].

Traditional network management, which relies on human expertise and manual configurations, struggles to keep pace with the rapid advancements in network technologies and the diverse requirements of emerging services [3]. Recent technologies have enabled software-based control of the network, paving the way for approaches such as Machine Learning (ML) [4] approaches to support dynamic and intelligent network operations. These technologies facilitate the network programmability, allowing real-time adaptation to changing conditions, resource optimization, proactive fault resolution, and enabling more granular control and optimization tailored to specific performance requirements.

However, given that only 39% of global software projects succeed [5], holistic software deployment across the network exposes national critical infrastructure to significant levels of risk. Thus, telecommunication networks, like other safety-critical industries (e.g. aviation), require a significant validation/testing pipeline for network control software before it is considered for production deployment. This pipeline consists of multiple validation tools (VT), such as simulators, scripts, testbeds, or pre-production staging environments, where each VT has a trade-off between execution time (speed), accuracy, and configurability Figure 1, leading to complex, inefficient, and often ineffective software validation (section II). This excludes the time taken to prepare, configure, or schedule for each VT. This level of human involvement in the validation is a clear barrier to the telecoms industry's shift toward software-based operations and fully autonomous networks.

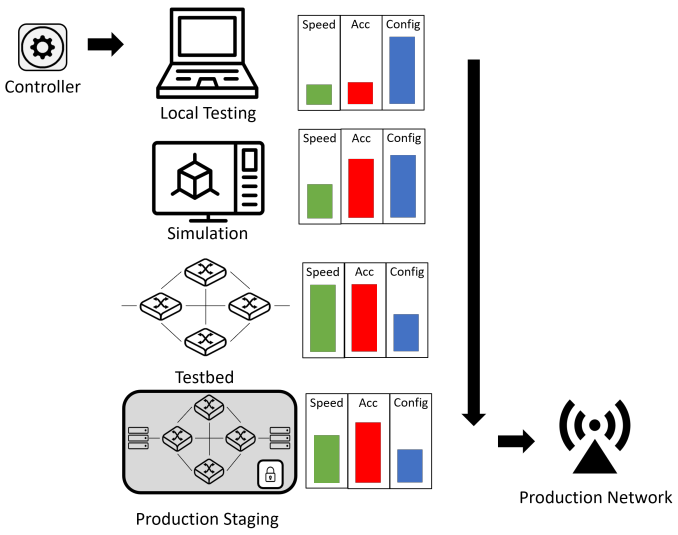


Fig. 1. Telco Software Validation Tool Pipeline

Research on the potential of *Digital Twin* (DT) [6] technology has attracted widespread attention due to its ability to simulate end-to-end services through comprehensive resource management and planning [7]. This capability enables DT as a potential testing tool and enhances evaluation efficiency. However, based on current research, the development of DTs still faces the challenge that the generation process relies heavily on domain-specific expertise.

This paper presents an initial study on the automatic gener-

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

ation of digital twins for networks (section III). Specifically, we explore the use of automated machine learning approaches to achieve data-driven, context-specific DT creation. To the best of our knowledge, this is the first work to do so. We demonstrate the approach through preliminary results using synthetic data, showing that the approach is efficient, sufficiently accurate, and viable (section IV).

II. BACKGROUND

A. Validation in Autonomous Networks

A key element of the ITU-T autonomous network architecture [8] is the experimental evaluation subsystem, who's goal is to test and validate the software (*controllers*) responsible for the network management and operation. This concept of *validation* is one of the key concepts in the approach. To ensure the effectiveness of the testing, environment modeling plays a crucial role in autonomous frameworks [9].

B. Current Validation Tools

Current approaches to validating network software are usually broken into a linear progression from software simulations to hardware testbeds and finally production deployments, each stage presenting its own challenges, Figure 1.

1) Hardware Tools: Hardware testbeds consist of specialized purpose-built devices designed for network testing. However, these testbeds suffer from high costs and lack flexibility, as they cannot be easily reconfigured to accommodate new testing or monitoring functions [10]. As network requirements evolve, the difficulty of modifying or expanding testbeds limits scalability and slows down the implementation of updates.

2) Simulation Tools: Compared to hardware testbeds, simulation software offers greater flexibility and cost-efficiency. It eliminates the need for physical hardware, reducing both setup and maintenance costs. Unlike hardware platforms, constrained by physical limitations, simulation allows rapid parameter adjustments, enabling simulation of diverse network topologies and behaviors. In addition, simulations can execute multiple tests in a shorter time while ensuring consistent and reproducible conditions, making simulation a powerful tool for network experimentation. Common examples include OMNet++ [11], ns-3 [12], and Mininet [13].

Although existing solutions meet the environmental modeling requirements for testing, they have limitations, particularly when faced with large numbers of parameters to be tested. Specifically, the execution time and human effort required in the simulator configuration act as a barrier to exploration.

Summary: As shown in Table I, different approaches involve trade-offs between accuracy, speed, and configuration flexibility. Physical testbeds offer high accuracy but suffer from low speed and limited flexibility, while simulations provide faster, more configurable testing with reduced accuracy.

C. Network Digital Twin

A Digital Twin (DT) is digital representation of a process, asset, or physical object [15]. Unlike traditional network testing described above, DT technology offers a transformative approach, integrating diverse network behavior data, constructing accurate models, and establishing mappings between real-world objects and their virtual counterparts [6]. This enables advanced prediction, emulation, analysis, and dynamic simulation of network, significantly enhancing efficiency and scalability in network testing. DT technology provides a solution to overcome existing methods' limitations by better reflecting network behaviors and pave the way for more responsive and efficient network testing solutions.

In the context of networks, DT is already enabling the development of more efficient network control and management tools for modern communication networks [6]. Specifically, they are leveraging data from the network itself, enabling extensive testing without disrupting live systems [16] [17] [18]. The ITU-T has also highlighted several use cases where DTs serve as critical platforms for experimentation, ensuring safe, scalable, and efficient validation [19].

In this work, we use DT to describe a data-driven model of a network, corresponding to a *functional* DT [20], see section III.

D. Machine Learning-based DTs

Machine Learning (ML) is an increasingly common approach for creating DTs in network-related applications. One example is in enhancing simulation accuracy and efficiency in network environments, with techniques like deep learning and reinforcement learning (RL) enabling high-speed, high-accuracy simulators for tasks such as network routing optimization [21]. ML is also widely applied to improving Quality of Service (QoS) prediction and overall network performance, with models like Graph Neural Networks (GNNs), Spiking Neural Networks (SNNs), and ensemble models optimizing delay prediction [20] and CNN-GRU architectures being used for network capacity prediction [22]. Furthermore, ML-driven DTs support intelligent decision-making by facilitating resource management, network optimization, and real-time monitoring, as demonstrated in Deep Reinforcement Learning (DRL)-based synchronization and prediction models [23]. Finally, ML is instrumental in large-scale data processing, enabling real-time big data management and network life cycle monitoring [24], ensuring that DTs can effectively handle vast amounts of network data for dynamic and evolving systems.

While ML models often focus on direct decision-making, DTs are generally designed to serve as neutral physical and functional mapping. Despite this distinction, their role under some specific cases [21] [22] [23] demonstrates a potential capability that can be used to support the *generation* of DTs. Recent studies have shown that ML can effectively support the generation of DTs by modeling complex, dynamic behaviors [7], [25]. As such, ML-enabled DTs are emerging

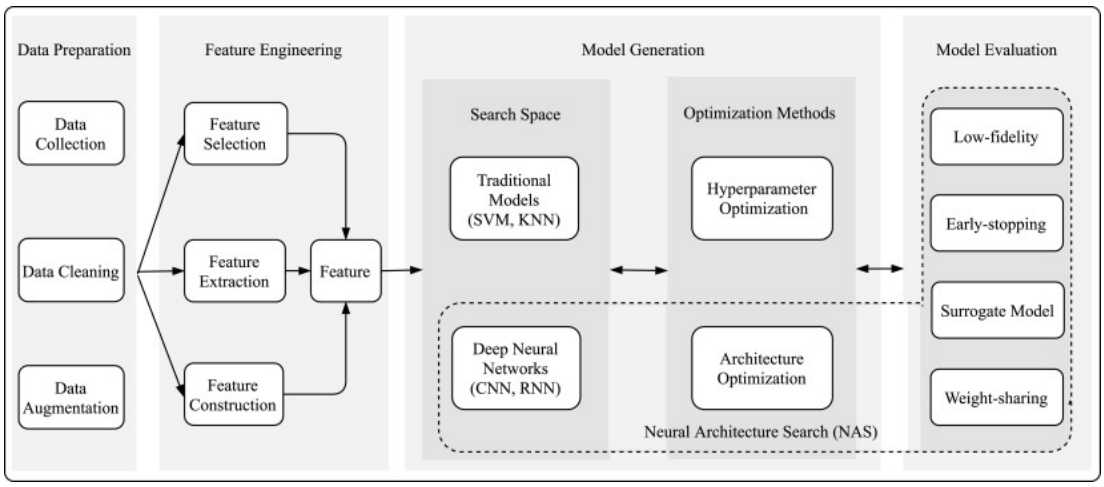


Fig. 2. Framework of AutoML [14]

as powerful tools for anomaly detection, adaptive testing, and performance optimization in autonomous networks.

E. Problem

Despite its potential, the broader application of ML in DT generation faces several barriers. Model development often requiring expertise that many network engineers lack limited and insufficient practical guidance [26]. The rapid evolution of ML also demands continuous learning, making tests highly knowledge-intensive and resource-demanding.

Key challenges include: data quality, where noise, missing values, and anomalies can significantly degrade prediction accuracy; feature engineering, where poor feature selection can lead to overfitting, bias, and poor generalization; and model selection and optimization remain complex, requiring extensive experimentation and computational resources, with advanced models often introducing further risks of errors.

III. AutoML-enabled Automated Generation

To address the challenges of creating DTs, this work explores the automated generation of DTs using Automated Machine Learning (AutoML). The goal is to be capable of handling various testing scenarios and processing the associated data with minimal manual effort. AutoML automates key steps, such as data preprocessing, feature engineering, model selection, and hyperparameter tuning (Figure 2) significantly reducing development time and human expert involvement.

By simplifying the training process, AutoML makes ML more accessible to ordinary developers and domain experts. Traditional model development is often labor-intensive, involving manual tuning and iterative testing, which hinders scalability. AutoML streamlines this process by efficiently evaluating multiple models and configurations, accelerating model generation while maintaining high performance.

As AutoML reduces training costs, lowers technical barriers for users, and automates comparison of multiple models' performance, we believe that AutoML may help automate the process of generating DTs for validation.

Unlike current *system-level approaches* [27] to DTs, which aim for holistic representations, we instead focus on what we term *Unit Twins*, with the aim of capturing specific network functionalities or components in a particular context to enable testing and validation. However, a network testing environment often requires evaluating multiple functionalities simultaneously, meaning that several Unit Twins may be needed to serve within the same network. We believe that such unit twins can also enable system-level testing and validation, however, that is beyond the scope of this work and left for future consideration. Like any other VTs (Figure 1), we consider that DTs would be integrated into the current testing and validation pipeline, as opposed to a perfect, stand-alone tool.

Given the relatively simple, array-based dataset used in this study (section IV), we focus on two AutoML tools: *AutoGluon* and *Auto-sklearn*, both well-suited for tabular data.

AutoGluon [28] is an open-source framework that automates the ML pipeline, supporting tasks such as classification and regression. It offers a user-friendly interface and performs well on large-scale datasets.

Auto-sklearn [29] builds on scikit-learn and focuses on traditional ML, using Bayesian optimization and meta-learning for model selection and hyperparameter tuning.

These frameworks differ in scalability, model selection strategies, and computational efficiency. We evaluate both to examine how AutoML choices influence the performance and effectiveness of DT generation.

IV. Preliminary Experiments

The objective of this initial experimental study is to evaluate the feasibility of using AutoML to automatically generate a DT of the network from data alone. Specifically, can AutoML simplify and accelerate the creation of accurate and reliable digital replicas of network environments, and produce efficient and accurate DTs of the network.

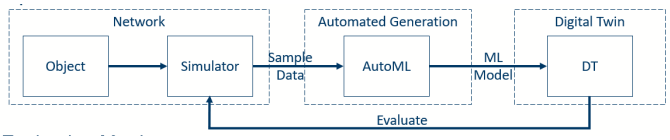


Fig. 3. Experiment Pipeline

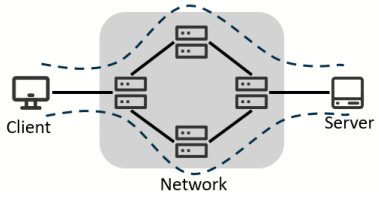


Fig. 4. Network Topology

A. Network Scenario

We consider a simple network topology, Figure 4. The network is a diamond topology consisting of two paths, where the link bandwidths and queue size in the routers are configurable. Each links bandwidth ranged from 25Mbps to 125Mbps, and each router queue size ranged from 50 to 150. This simple topology allows evaluation of how different traffic conditions impact network latency. Upon the network, a simple application was used. Specifically, a 100MB file was requested by the client to the server and then downloaded through two different network paths while adjusting configuration parameters such as queue size and bandwidth. The blue dotted line in Figure 4 shows each network path.

The objective for the generated DT is to capture the relationship between network parameters and the resulting latency on each path. Such a DT would enable testing and validation of a traffic engineering network controller.

B. Experiment Setup

1) Data Generation: Mininet [13], a network emulator, was used to create the above scenario and generate a dataset. A script was used to automate the configuration and data generation of all possible combinations of link bandwidths and router queue sizes, as well as resulting path latency times.

From this dataset, training and testing subsets were sampled. In alignment with the ultimate goal of reducing the time cost for network testing, we did not utilize the entire dataset for training. Instead, we used an interval sampling strategy to ensure that key characteristics of the network behavior were preserved. This approach was intended to balance between data volume and the retention of essential patterns, thereby enhancing the generalizability of the automated generation.

2) AutoML Training: We utilized the sampled data to train models using AutoGluon and Auto-sklearn. For AutoGluon, we configured the task as a regression problem, aligning with our objective of predicting network latency. To ensure efficiency, we imposed a time limit of 300s on the entire training process and saved all intermediate models generated.

TABLE II
RESULT OF EACH CANDIDATE MODEL IN AUTOGLUON

Model	Path 1		Path 2	
	Accuracy	Time(s)	Accuracy	Time(s)
LightGBM	99.63%	0.42	84.43%	0.41
ExtraTreeMSE	99.71%	1.94	85.02%	2.05
RandomForestMSE	99.75%	2.62	84.91%	2.33
CatBoost	99.78%	23.53	84.46%	0.71
WeightedEnsemble_L2	99.79%	25.44	91.57%	1.48
XGBoost	99.78%	0.75	91.57%	0.52
Total AutoML Time (s)	56.7		7.5	

After experimenting with several preset configurations in AutoGluon, we selected the 'good_quality' setting, which offers a balance between predictive accuracy and computational efficiency. This setting offered fast inference speeds and lower disk usage compared to the 'high_quality' option, making it suitable for applications where rapid generation are critical.

For the Auto-sklearn experiments, we configured the task as a regression problem and retained the default settings for all other parameters, allowing us to evaluate the out-of-the-box performance of Auto-sklearn without extensive customization, providing a baseline for comparison with AutoGluon.

C. Result

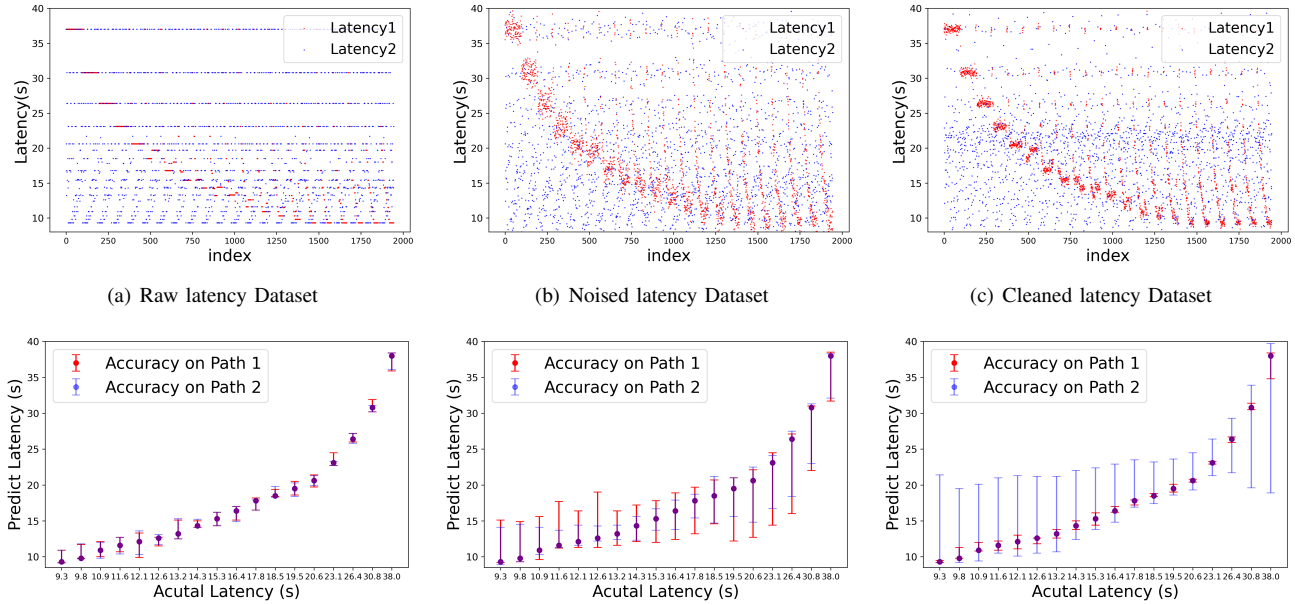
To evaluate the feasibility of automatically generating DTs via AutoML we consider accuracy of generated DT, time to generate a DT, DT execution time.

1) Accuracy: We trained multiple models using AutoGluon, leveraging its automated model selection process to identify the most suitable model. The training results, including test scores , are shown in Table II.

Here we can see the performance of different models on datasets from the two distinct paths. Notably, due to differences in the patterns of configuration variations during data generation, the dataset from Path 1 exhibits more regularity after sampling. In contrast, the dataset from Path 2 demonstrates a relatively more random distribution of these parameters. This distinction in data characteristics between the two paths provides valuable insights into how the inherent structure and variability of the datasets influence model performance.

Using the same training and validation datasets, we conducted experiments with both AutoGluon and auto-sklearn, and the results are presented in the Table III. The high accuracy achieved by both tools indicates that they are capable of generating DT effectively for the simple network topology test scenarios in our current experiments. However, determining which tool is more suitable for practical applications will require further investigation. In future work, we plan to evaluate their performance under more complex testing requirements and scenarios to provide a comprehensive comparison.

Using either AutoML framework, the results indicate that a DT **can** be automatically created from data which captures the relationship between network configuration parameters (queue size and bandwidth) and latency with high accuracy.



(a) Raw latency Dataset (b) Noised latency Dataset (c) Cleaned latency Dataset

(d) Accuracy (Original) (e) Accuracy (Noised) (f) Accuracy (Cleaned)

Fig. 5. Comparison of dataset processing and its impact on model accuracy. Blue values are for Path 1, red values are for Path 2.

TABLE III
BEST RESULTS OF DIFFERENT AUTOML TOOLS

Model	Accuracy on Path 1	Path 2
AutoGluon	0.99	0.94
Auto-sklearn	0.99	0.92

2) Execution Time: Execution times for Mininet and the generated DT were compared for the same network configuration based on the above scenario. The DT execution time (0.0167s) was over **500x faster** than Mininet (8.5316s). This *significantly* lower execution is a clear advantage of the DT and presents an opportunity for significant scaling in network testing to meet the demands of autonomous networks.

Furthermore, when considering the total time required—including both the training data collection via simulation and the model training process—the time-effectiveness of the DT-based approach becomes even more evident. To evaluate a large-scale dataset consisting of 194,481 samples, the DT was trained using only a carefully designed subset of 400 samples, achieving a prediction accuracy of approximately 98 %. The complete DT pipeline, including simulation-based data collection, model training and DT testing, required approximately 3.4 hours. In contrast, conducting full-scale testing using Mininet for all samples would require over 900 hours of continuous simulation time. This dramatic reduction in total execution time—by a factor of over 260×—highlights the superior efficiency of the proposed method. By significantly lowering the computational and temporal costs associated with large-scale testing, ML-enabled DTs offer a highly practical and scalable solution for evaluations in autonomous network environments.

D. Data Quality

Given the simple scenario and perfect data quality generated from the simulators, we wanted to understand how well the AutoML frameworks could handle noisy data, as would be generated from real networks, and the resulting effect on the generated DT.

1) Experimental Setup: Working with the same scenario as before (Figure 4), we now generate a dataset using ns-3; this was due to the time overhead of generating data with Mininet. The raw latency values are seen in Figure 5(a) and provide a baseline of comparison. Next, to simulate real-world imperfections, we added Gaussian noise [30] with a standard deviation of 1 and a mean deviation of 0 to the raw latencies, Figure 5(b). Finally, we applied a Savitzky-Golay filter [31] to the noised dataset to account for a scenario where a known de-noising approach can be pre-applied before AutoML, Figure 5(c).

2) Result: Figure 5(a), 5(e), 5(f) show the per-path latency predictions for the raw, noisy, and de-noised data respectfully. Red values are for path 1 and blue values for path 2. The AutoGluon framework is used.

Similar to Table II, results vary by path. Specifically, Path 1 shows a more consistent and overall decline in performance across the dataset, which can be attributed to the sequential configuration changes made during the generation of simulation data. On the other hand, Path 2 should display periodic performance fluctuations in the complete dataset but the sampling interval and Path 2's periodicity misaligned, making feature less noticeable.

The current denoising approach does not universally enhance model performance. In fact, Path 2 performance on the cleaned dataset is even worse than its performance on the

noised dataset, suggesting that the filtering process may have removed some meaningful patterns or introduced artifacts that negatively impacted the model. However, the Path 1 performance shows an improvement, with the DT outperforming its counterpart trained on the original dataset, highlighting the nuanced impact of data pre-processing on model performance.

These findings underscore the importance of tailoring data pre-processing techniques to the specific characteristics of the dataset and the problem for network testing requirements. While noise reduction by filtering can be beneficial in certain contexts, it is not a one-size-fits-all solution. Future research should focus on developing more adaptive pre-processing methods to better support AutoML tools in handling diverse and imperfect datasets.

V. Future work

Future work will explore the scalability of generating DTs for more complex network topologies and traffic profiles, ensuring that the generated DT remains reliable across varying scenarios. Additionally, we seek to characterise the generalisability of the approach for dynamic environments, varying traffic types, and diverse network topologies.

VI. Conclusion

In this paper, we have explored the feasibility of automatically generating digital twins for network testing, aligned to the ITU-T's autonomous network architecture *experimentation subsystem*. We have presented how AutoML frameworks can be used to automatically generate DTs based on network data, addressing the human effort in the design and implementation of DTs. Initial experimental results have demonstrated the approach is feasible, even in the presence of noisy data, and provides high accuracy and very low execution times compared to equivalent simulators. Future work will explore more complex network topologies and scenarios.

VII. Acknowledgment

This work was supported by EPSRC funding EP/Z533221/1.

References

[1] ITU-T, "Architecture framework for Autonomous Networks," tech. rep., TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 9 2022.

[2] D. Lake, N. Wang, R. Tafazolli, and L. Samuel, "Softwarization of 5G Networks–Implications to Open Platforms and Standardizations," *IEEE Access*, vol. 9, pp. 88902–88930, 2021.

[3] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, "Rethinking enterprise network control," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1270–1283, 2009.

[4] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine Learning for Networking: Workflow, Advances and Opportunities," *IEEE Network*, vol. 32, pp. 92–99, 3 2018.

[5] "CHAOS REPORT 2015," tech. rep., 2011.

[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] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions," *IEEE Communications Magazine*, vol. 60, pp. 74–80, 1 2022.

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

[9] S. S. Mwanje and C. Mannweiler, "TOWARDS COGNITIVE AUTONOMOUS NETWORKS IN 5G," in *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, pp. 1–8, IEEE, 11 2018.

[10] 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.

[11] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2010.

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

[13] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp. 1–6, 2014.

[14] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 1 2021.

[15] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Digital twin paradigm: A systematic literature review," *Computers in Industry*, vol. 130, p. 103469, 9 2021.

[16] 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.

[17] R. J. Somers, J. A. Douthwaite, D. J. Wagg, N. Walkinshaw, and R. M. Hierons, "Digital-twin-based testing for cyber–physical systems: A systematic literature review," *Information and Software Technology*, vol. 156, p. 107145, 4 2023.

[18] S. Chen, M. Neubauer, and A. Verl, "A Digital Twin Platform for service-based Testing and Optimization," *Procedia CIRP*, vol. 126, pp. 260–265, 2024.

[19] ITU-T, "Use cases for Autonomous Networks," tech. rep., 10 2021.

[20] 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.

[21] Z. Wei, S. Wang, D. Li, F. Gui, and S. Hong, "Data-Driven Routing: A Typical Application of Digital Twin Network," in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, pp. 1–4, IEEE, 7 2021.

[22] Y. Shen and X. Wang, "Prediction of Carrying Capacity of Digital Twin Power Information Communication Network Based on CNN-GRU Neural Network," in *2022 4th International Conference on Smart Power & Internet Energy Systems (SPIES)*, pp. 1042–1046, IEEE, 12 2022.

[23] M. Polverini, F. G. Lavacca, J. Galan-Jimenez, D. Aureli, A. Cianfrani, and M. Listanti, "Digital Twin Manager: A Novel Framework to Handle Conflicting Network Applications," in *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 85–88, IEEE, 11 2022.

[24] J. Zhao, X. Xiong, and Y. Chen, "Design and Application of a Network Planning System Based on Digital Twin Network," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 900–904, 2022.

[25] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula, "Digital Twin for 5G and beyond," *IEEE Communications Magazine*, vol. 59, pp. 10–15, 2 2021.

[26] PyNet Labs, "Should Network Engineers Learn AI and ML?," 2023.

[27] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *Journal of Manufacturing Systems*, vol. 64, pp. 372–389, 7 2022.

[28] 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.

[29] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: Efficient and Robust Automated Machine Learning," pp. 113–134, 2019.

[30] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: removing noise by adding noise," 6 2017.

[31] R. Schafer, "What Is a Savitzky-Golay Filter? [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 28, pp. 111–117, 7 2011.