Recovering Missing Monitoring Data to Enhance Service Provisioning in the Edge-to-Cloud Continuum

Recovering Missing Monitoring Data to Enhance Service Provisioning in the Edge-to-Cloud Continuum

Gaetano Francesco Pittalà*, Cristian Zilli†, Nicola Di Cicco‡, Gianluca Davoli*, Alessio Sacco†
*Department of Electrical, Electronic, and Information Engineering (DEI), University of Bologna, Italy
†Department of Control and Computer Engineering (DAUIN), Politecnico di Torino, Italy
‡Department of Electronics, Information, and Bioengineering (DEIB), Politecnico di Milano, Italy

***Abstract*—Efficient service provisioning in the Edge-to-Cloud Continuum is of utmost importance for modern applications. While sensible decisions can be taken if enough monitoring data is collected, maintaining continuous telemetry data streams amidst the continuum's complexity is challenging. This paper introduces CRISP (reConstructing Resource Information for Service Placement), a solution combining data reconstruction and service placement strategies to optimize decisions despite incomplete monitoring data. CRISP utilizes Convolutional Neural Networks and Long Short-Term Memory models for data reconstruction, integrating them with a heuristic algorithm that selects nodes for service component placement. Numerical results demonstrate CRISP's efficacy in optimizing service provisioning despite missing data, contributing to enhanced resource utilization and service performance in the considered context.**

I. Introduction

In an era where network services are expected to be more pervasive than ever, the choice of where to operate the processing required by those services is particularly important. Service placement strategies have a strong impact on the resulting performance of the service, both from the point of view of users (e.g., in terms of perceived latency) and from that of the network operator (e.g., in terms of power consumed by the involved equipment).

With the plethora of devices available in the Edge-to-Cloud Continuum that can potentially be harnessed to provide services, optimized service placement becomes a nontrivial task. Numerous research efforts have been facing the challenges of service orchestration across the Continuum, proposing architectures for automated service provisioning [1], as well as strategies for the fulfillment of performance requirements [2].

Monitoring data plays a fundamental role in assisting placement strategies, enabling informed choices and dynamic management of resources. This makes the collection of such data a particularly crucial duty. Numerous factors, including network latency, intermittent connectivity, resource constraints at the edge, and hardware or software failures, can lead to losing monitoring information, and, in turn, to service degradation [3]. To mitigate the problem, various solutions can be applied at different levels [4], [5], with a multitude of approaches aiming to reconstruct missing information has been proposed [6]–[10].

Among them, Spatio-Temporal Tensor Completion (STTC) [8] and Spatio-Temporal Graph Mixformer (STGM) for traffic forecasting [9] make use of temporal patterns found in the traffic matrix data to improve the reconstruction process, with the first employing tensor factorization techniques to project information to a lower dimensional space, while the second applies spectral clustering and multi-Gaussian modeling to identify spatially similar portions of the traffic data and reconstruct it. Auto-Encoder models and their variations, for example, have been thoroughly investigated in this context [6], [11], [12], given their capability to compress latent information in the data and use it to restore noisy and/or incomplete input samples. In the case of reconstruction, convolutional architectures are preferred, not only because of their training efficiency and robustness against over-fitting phenomena, but also their superior performance with multi-dimensional data [13].

The integration of machine learning and AI-driven techniques into service placement frameworks presents both opportunities and challenges. While these approaches offer the potential for enhanced decision-making capabilities and adaptive resource management, they also introduce complexities related to model interpretability, training data quality, and algorithmic bias. In our problem, we need to predict more features regarding more nodes, thus learning both spatial and temporal patterns.

The implications of service placement decisions extend beyond immediate performance metrics, encompassing broader considerations such as energy efficiency, environmental impact, and regulatory compliance. Achieving a holistic understanding of these multifaceted dynamics is essential for designing sustainable and resilient network infrastructures.

In this paper, we propose a solution for reConstructing Resource Information for Service Placement (CRISP), presenting a combination of data reconstruction strategies and service placement decision policies that aim at maximizing optimal decisions even with incomplete monitoring data. In case information for a node is missing, we reconstruct it using a deep learning model based on Convolutional Neural Networks (CNN) (in order to extrapolate spatial patterns from the data, namely correlation between the state of different nodes) and Long Short-Term Memory (LSTM) (so as to cap-

ture tendencies in the temporal dynamics of the state of each node). The resulting set of information is employed to select which nodes to place service components on, using a simple heuristic-based algorithm. The performance of the monitoring data reconstruction and that of the service placement decisions leveraging it are evaluated in this paper, showing the benefits of the approach.

Our contributions are structured as follows:

- We propose CRISP, a solution for service allocation in the Edge-to-Cloud Continuum using partial monitoring information (Section II);
- We design a CNN-LSTM model for the reconstruction of incomplete telemetry data (Section III);
- We demonstrate that CRISP can effectively enhance service orchestration performance (Section IV);
- We summarize our main conclusions and outline future work (Section V).

II. CRISP: System Design and Overview

A conceptual architecture of the reference orchestration system is represented in Fig. 1, along with the interactions among functional elements required for service deployment. The orchestrator consists of two layers at two different levels of abstraction. The topmost one, denoted as *Service Orchestration* (SO), comprises the processes involved in the composition of abstracted service components according to requests coming from users and service placement policies. The second layer, denoted as *Resource Orchestration* (RO), comprises the processes in charge of collecting, processing, and adapting the information coming from the resources[1] in the underlying infrastructure, to provide it to the service orchestration processes. As the difference in the size of the two layers in Fig. 1 suggests, the focus of this work is primarily on the latter one, with an effort to contextualize the reconstruction of monitoring data in a service orchestration architecture, and evaluate its impact on the overall service provisioning performance.

As previously mentioned, the role of the functional elements in the SO layer is that of taking placement decisions for service components, by means of algorithms that enforce the placement policy of choice, based on information provided by the RO layer through appropriate abstraction models. The details of the functional elements that operate within the SO layer, as well as the description of the interactions between the SO and RO layers, are in line with the architecture described in [14], and are omitted from this paper as they are not required for the understanding of the data reconstruction dynamics.

The core element of the RO layer is the *Resource intelligence* element, which is tasked with the reconstruction of incomplete monitoring data using AI (more details on this in Section III), consuming information coming from the infrastructure (M3) and providing an enhanced version of it to service placement processes (M4). Gathering such data is

[1]We use the term "resource" to include all possible service allocation targets, which mainly include computing nodes distributed across the Continuum, but in general it may also include different items, e.g., programmable switches.

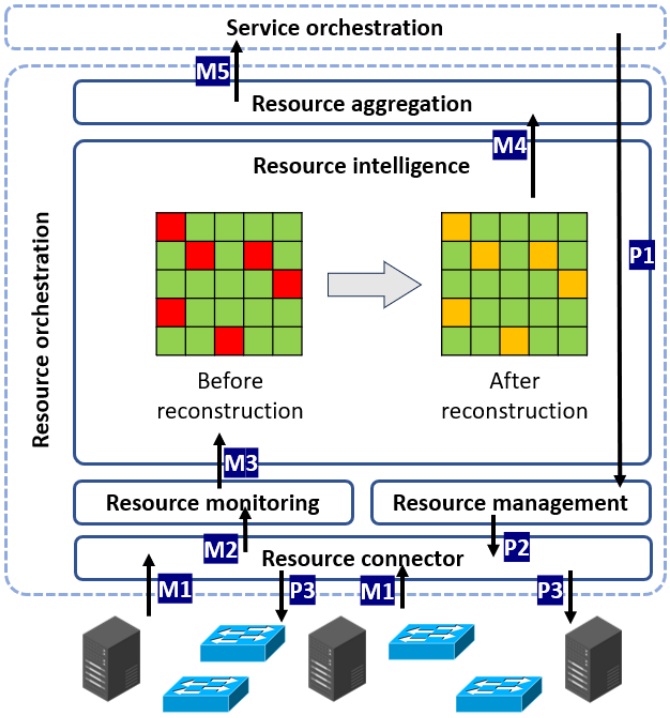

Fig. 1. CRISP reference architecture with interactions among functional elements, where M stands Monitoring and P for placement.

a job for the *Resource monitoring*, which collects information from the underlying resources handling the monitoring processes running on them (M1, M2). Conversely, supplying the processed monitoring data to the SO layer is a responsibility of the *Resource aggregator*, which provides the abstractions needed to the service orchestration processes, presenting the available resources (along with their monitoring information) as abstracted service components (M5). Once placement decisions are taken (P1), the *Resource management* element can handle the deployment and decommissioning of service components through the *Resource connector* element (P2, P3), which provides the technology-specific endpoints [15] for the orchestrator to interact with the underlying infrastructure.

We underline that end-to-end service provisioning incurs numerous limitations of technological and administrative nature. Technological issues are addressed by common APIs that allow monitoring and management processes to access diverse resources. Administrative issues include the fact that different tenants are unlikely to allow access to their domains to a centralized orchestrator entity. This should be addressed by agreements on the different parties to exchange information and management rights to portions of their infrastructures, of shared deployed services. This is out of the scope of this paper, in which we assume to operate on the portion of the infrastructure owned by a single tenant.

Orchestration processes in the SO layer make placement decisions based on the available information on underlying resources, according to the requirements posed by service requests and by context-dependent policies. Although it might be argued that this is an interesting application for AI/ML –

Algorithm 1: Service placement with Load Balancing policy

Input : set R of computing resources, $R = \{r_i \mid i \in \mathbb{N} \wedge i \leq \text{amount of comp. resources}\}$; current status s of each resource as a collection of metrics, $s(r) = \{\text{CPU, RAM, etc.}\}, \forall r \in R$; service placement policy p;

Output: service placement decision d

```
if R = ∅ then
    d ← block
else if |R| = 1 then
    d ← the only available resource
else
    Create the ordered set R_O^load by sorting elements of
      R by their occupation metrics, in lexicographic
      ordering (CPU, then RAM, then disk, then
      bandwidth)
    d ← the first element of R_O^load
```

and we agree, having investigated that in [16] – in this work, we want to isolate the effect of ML-based reconstruction of incomplete monitoring data, so we employ a simple heuristic to implement a placement decision policy. We refer to this policy as *Load balancing* (LB), as it aims at evenly distributing the computational load given by a series of service deployments across the available resources. The service placement algorithm implementing the LB policy is illustrated in Alg. 1. In case there are no available resources or only one feasible choice, the placement decision is straightforward. Otherwise, the result of the algorithm is based on an ordering of the possible service deployment targets (i.e., the computation resources) in terms of their computation metrics (e.g., CPU, RAM, etc.), with the final choice falling on the resource that is the least active at the moment.

III. Reconstruction Model

In this paper, we specifically address the problem of estimating future resource usage starting from partially available information. While the field of information reconstruction is rich in solutions of different natures, going from mathematical and statistical approaches to machine learning and deep learning models, the input data is typically modeled as a multi-dimensional vector, either in the form of tensor-like structures or time series. Examples of mathematical models for spatial reconstruction like LMAFit [17] and LRTC [18] represent a general approach for a wide range of (matrix) data completion and estimation problems based on the low-rank matrix factorization technique. Recent machine learning architectures allow relaxing such assumptions and can learn either temporal or spatial patterns in input data. Given this flexibility and the heterogeneity of data in our system, we consider a deep learning solution, and in particular, our model is a combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks.

CNNs [19] are a sensible choice for multi-dimensional data (e.g., images), because of their reduced computational complexity and their innate regularization capability compared to fully connected networks. These properties are a natural consequence of two key factors: *(i)* the replacement of fully connected layers with strided convolutional layers, which not only reduce the number of trainable parameters but also efficiently encode spatial patterns and *(ii)* pooling layers, which further compress the hidden information.

LSTM networks [20] are a subclass of Recurrent Neural Networks (RNN) engineered to address the problem of the vanishing gradient [21] common to traditional recurrent architectures. They achieve their objective by adding gated units responsible for holding and controlling the flow of information during the back-propagation process. The flexibility of LSTMs in capturing long-term dependencies makes them a powerful tool in various domains, ranging from sequence modeling to time series prediction. In this latter case, LSTMs predict future values based on historical data, with an approach that is applicable in finance, weather forecasting, and stock price prediction [22], [23].

Convolutional LSTM is the result of combining the building blocks of CNN and LSTM. This architecture suits the case at hand because of its inherited properties and spatio-temporal nature, being able to capture both the dynamism across time and the spatial correlation of the node metrics. Such a model is: *(i)* trained to predict the resource state matrix of all nodes, provided the same information from previous time steps is available, and *(ii)* used to reconstruct the missing measurements.

The input of the model represents a time series of resource usage for the nodes in the system. In further detail, at each time interval, our model receives in input a sequence of the last k matrices dimensioned as $4 \times N$, where for each node we store 4 different metrics and N is the number of nodes in the cluster. Thus, each of the four rows represents a specific metric for the usage, in order: CPU, RAM, disk, and bandwidth. In conclusion, the input of the model is a multidimensional array $k \times 4 \times N$. Empirically, we found that using the three most recent pieces of historical information, i.e., $k = 3$, provides a good prediction accuracy. Increasing values of k led to negligible performance gains, at the cost of substantially higher training times and resource requirements. Thus, we set this value as default in the experiments.

IV. Illustrative Numerical Results

In this section we quantify the ability of CRISP to estimate missing values and then, using these values, to allocate services in a satisfactory way, showing that:

- the CNN-LSTM model adopted in CRISP produces smaller errors than CNN models;
- CRISP leads to better orchestration performance than the considered baselines, both in terms of service allocation

accuracy and efficacy, facilitating the deployment of services over large scale dynamic scenarios.

A. Evaluating the features reconstruction

We benchmark the CNN-LSTM model and compare its performance to a CNN regression model for our monitoring data reconstruction task. The CNN-LSTM model consists of three stacked convolutional LSTM layers, having, respectively, 64, 32, 32 filters sized 3×3, and a final convolutional layer needed for *(i)* decoding the hidden state and *(ii)* matching the shape of the output to the shape of the state matrices. The CNN model represents a different approach to reconstruction, using partial information, i.e., incomplete matrices, about the state of the nodes at a given time in order to infer the missing values. Missing information in the data was replaced with placeholder values. Empirically, we found the best composition performance-wise for CNN to be as follows: two convolutional layers having 64 and 32 filters sized 3×3, two pooling layers, two fully connected layers with 128, 64 units and a 4 units wide output layer, which allows for predicting the usage of CPU, RAM, disk and bandwidth at a particular node. Both CNN-LSTM and CNN were trained and evaluated on data following a 75%-25% train-test split. All evaluations of the reconstruction models are led using the Mean Absolute Percentage Error and Mean Squared Error metrics, defined as follows:

$$\text{MAPE} = \sum_{i=0}^{N-1} \frac{|y_{t_i} - y_{p_i}|}{|y_{t_i}|}, \qquad \text{MSE} = \sum_{i=0}^{N-1} (y_{t_i} - y_{p_i})^2, \quad (1)$$

where y_{t_i} stands for the i-th observed value, y_{p_i} corresponds to the i-th predicted value, and N denotes the total number of samples under consideration. MAPE offers an intuitive description of the performance of the models, while MSE allows for evaluation from a different standpoint, placing more emphasis on the occurrence of large errors. Fig. 2 and 3 show the error values of both models for increasing input sizes, or node counts, and for different percentages of missing data from the input matrix of the CNN model (namely 10% and 20%). From our tests, CNN-LSTM is by far the best performer, producing errors one order of magnitude smaller than CNN in any condition. Therefore, we choose this architecture as a component of the following analysis.

B. Evaluating the placement strategy

To assess the effect of the reconstruction of monitoring data on the overall service orchestration performance, we need to define criteria for the evaluation, as well as reference baselines. We define as "optimal service placement decision" the one taken by the SO layer when it is provided with the complete, uncorrupted set of monitoring data, arguing that the service placement processes are reasonably expected to perform best in that case. Additionally, we define the Service Allocation Failure Probability (SAFP) as

$$\text{SAFP} = \frac{N_B + N_M}{N_T}, \quad (2)$$

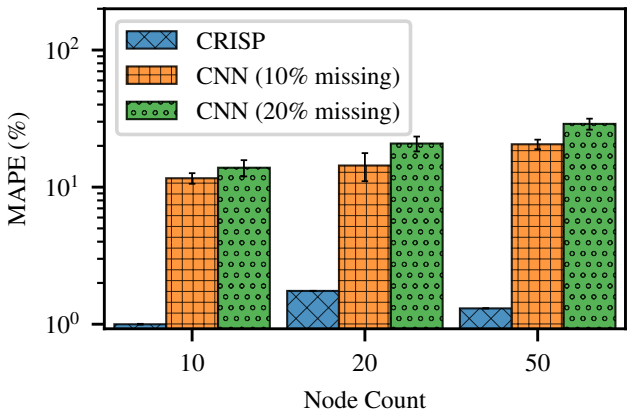

Fig. 2. Mean absolute percentage error of CRISP compared to CNN-based spatial reconstruction for different node counts.

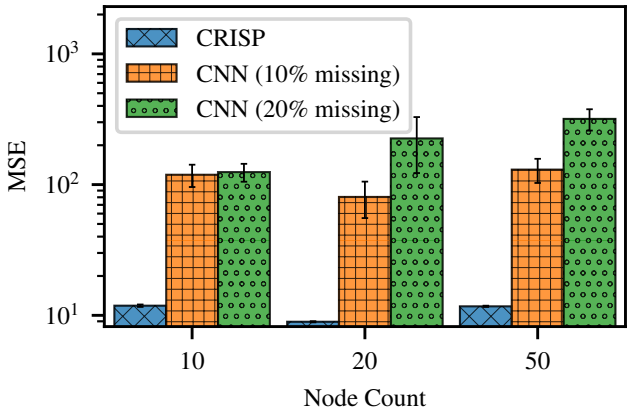

Fig. 3. Mean squared error of CRISP compared to CNN-based spatial reconstruction for different node counts.

where N_B is the number of blocked service requests (i.e., requests that the orchestrator decided to block due to perceived unavailability of resources), N_M is the number of mismanaged service requests (i.e., requests that the orchestator accepted, but ended up being served by nodes that were not actually available), and N_T is the total number of service requests offered to the system.

Lastly, we define two baseline strategies concerning the usage of monitoring data by the placement processes, namely *Empty* and *Past*. With the former, all resource nodes for which incomplete monitoring information is available are considered inactive (computationally "empty"), and can therefore be allocated computing tasks for a service. With the latter, vacancies in the monitoring data are filled with the most recent reliable information on that specific resource. The *Empty* strategy may seem imprudent, as intuitively it would make more sense to refrain from mapping services onto resources without any valid monitoring information. However, we have included it as it reduces the probability of blocking a request, at the cost of increasing the occurrences of sub-optimal deployments.

We expect the results to be influenced by multiple factors,

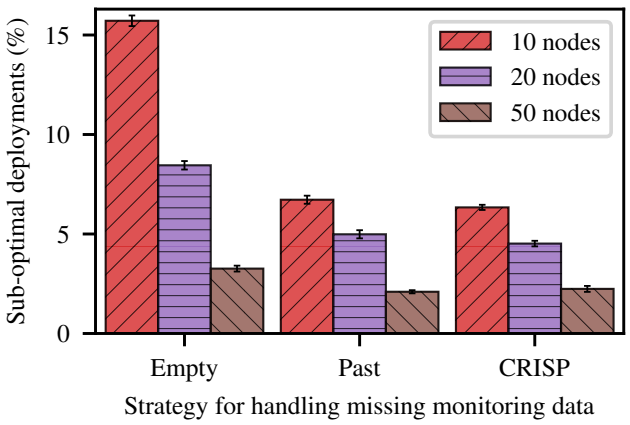

Fig. 4. Performance of the service placement algorithm in taking optimal decisions (expressed as a percentage of sub-optimal decisions) for different node amounts.

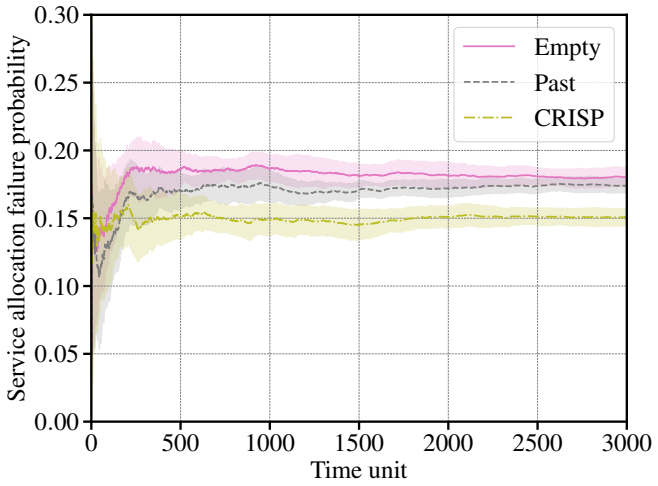

Fig. 5. Evolution of SAFP over simulated time.

including the overall amount of available resources, the distribution of service requests, the amount and characteristics of possible services, and the probability of monitoring information being lost. Regarding the former three factors, we apply the same rationale as in [16]. We consider a total of 10, 20, 50 available resources (e.g., edge nodes), and offer service requests to the orchestration system with a traffic intensity of 100 Erlang. We define four different services, each with its own profile in terms of required computational effort, in terms of CPU, RAM, disk, and bandwidth, and assume them to be uniformly distributed in $[0.15, 0.3]$, $[0.1, 0.2]$, $[0.01, 0.1]$, and $[0.01, 0.02]$ for the different services. It is worth noting that, as demonstrated in [16], the placement performance would be remarkably good also for a much larger number of resources than the one used for training. Regarding the probability of losing monitoring data, we assume that each resource (e.g., edge node) provides monitoring information in the form of a single packet enclosing all the metric values for a given time instant. Furthermore, we assume that any such packet is either correctly gathered by the collector processes in the RO layer, or lost entirely. This implies, in turn, that each resource is either associated with a complete set of monitoring data, or no data is available for that resource at all, for any given time instant. We implemented this by randomly selecting one node from the pool of available resources, on average every three time-steps, and completely masking the monitoring data associated with it for that time step.

We perform the evaluations in the same simulated environment used in [16], running each scenario 100 times, and providing 10000 requests to the orchestrator in each run.

In Fig. 4, we can observe that the placement algorithm always achieves better results when operating with reconstructed data, obtaining a tangible improvement (of 59.6%) when compared to the *Empty* strategy, and a smaller but still meaningful improvement (of 5.7%) when compared to the *Past* strategy. Furthermore, with this approach, the orchestrator does not need to host and maintain a registry with monitoring information related to past time instants. This is particularly relevant in situations where the number of available resources to be orchestrated grows to the amounts typical of massive Edge Computing or Internet of Things scenarios.

Moreover, Fig. 5 shows that CRISP outperforms the baseline strategies also in terms of SAFP. With the *Past* strategy, the orchestrator is not aware of departures that have taken place since the last service allocation (i.e., the last moment when monitoring data was collected), so it might deduce that the nodes are more occupied than they actually are, leading to a higher chance of blocking the service request. The *Empty* strategy makes the orchestrator perceive nodes for which it does not have monitoring information as not occupied, encouraging it to allocate services on those nodes. On its own, this would result in a lower occurrence of blocking, but it also implies that the services are not guaranteed to be executed correctly, as the selected node might, in fact, be occupied, resulting in the impossibility of actually instantiating the service.

V. Conclusion & Future Work

This paper presents CRISP, a solution to recover missing monitoring data for service orchestration algorithms. Our results show that our CNN-LSTM model accurately reconstructs this information, enhancing the efficacy of placement algorithms. By integrating data reconstruction techniques, orchestration systems can make informed placement decisions, optimizing resource utilization and system performance, especially in dynamic environments (e.g., Edge Computing).

Future research will investigate how recent learning-based orchestrators can benefit from this approach, focusing on the synergy between monitoring and application services, and evaluating the efficacy of CRISP in additional, challenging scenarios.

While obtaining real-world data is challenging, the deployment of testbeds in controlled environments is an ongoing effort, as it can facilitate the capture of more realistic

traffic patterns. By incorporating diverse and dynamic traffic scenarios, we can better assess the effectiveness of CRISP in practical settings and ensure its relevance to real-world deployment challenges.

Exploring alternative strategies beyond Load Balancing (LB) is essential to broadening the appropriateness of CRISP. While our paper primarily focuses on LB as a use case, we plan to investigate the applicability of CRISP to a broader range of orchestration strategies, such as QoS greedy scheduling or revenue-oriented service placement algorithms. Evaluating CRISP's efficacy across multiple orchestration paradigms will provide valuable insights into its versatility and suitability for different deployment scenarios.

In conclusion, future research aims to enhance traffic model realism, assess scalability and practical challenges through real-world deployments, and explore alternative orchestration strategies to broaden CRISP's applicability in dynamic environments like edge computing.

ACKNOWLEDGMENTS

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

REFERENCES

[1] D. Giannopoulos, G. Katsikas, K. Trantzas, D. Klonidis, C. Tranoris, S. Denazis, L. Gifre, R. Vilalta, P. Alemany, R. Muñoz *et al.*, "Across: Automated zero-touch cross-layer provisioning framework for 5g and beyond vertical services," in *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2023, pp. 735–740.

[2] M. Lashgari, L. Wosinska, and P. Monti, "End-to-end provisioning of latency and availability constrained 5g services," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1857–1861, 2021.

[3] V. Bharti, P. Kankar, L. Setia, G. Gürsun, A. Lakhina, and M. Crovella, "Inferring invisible traffic," in *Proceedings of the 6th International Conference (CoNEXT)*. ACM, 2010, pp. 1–12.

[4] J. Du, M. Hu, and W. Zhang, "Missing data problem in the monitoring system: A review," *IEEE Sensors Journal*, vol. 20, no. 23, pp. 13 984–13 998, 2020.

[5] P. Borylo, G. Davoli, M. Rzepka, A. Lason, and W. Cerroni, "Unified and standalone monitoring module for nfv/sdn infrastructures," *Journal of Network and Computer Applications*, vol. 175, p. 102934, 2021.

[6] A. Sacco, F. Esposito, and G. Marchetto, "Completing and Predicting Internet Traffic Matrices Using Adversarial Autoencoders and Hidden Markov Models," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2244–2258, 2023.

[7] Z. Liu, Z. Wang, X. Yin, X. Shi, Y. Guo, and Y. Tian, "Traffic matrix prediction based on deep learning for dynamic traffic engineering," in *IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–7.

[8] H. Zhou, D. Zhang, K. Xie, and Y. Chen, "Spatio-Temporal Tensor Completion for Imputing Missing Internet Traffic Data," in *Proceedings of the 34th international performance computing and communications conference (IPCCC)*. IEEE, 2015, pp. 1–7.

[9] H. Zhou, D. Zhang, and K. Xie, "Accurate Traffic Matrix Completion Based on Multi-Gaussian Models," *Computer Communications*, vol. 102, pp. 165–176, 2017.

[10] A. Sacco, F. Esposito, and G. Marchetto, "Resource inference for sustainable and responsive task offloading in challenged edge networks," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 3, pp. 1114–1127, 2021.

[11] A. E. Ilesanmi and T. O. Ilesanmi, "Methods for Image Denoising Using Convolutional Neural Network: A Review," *Complex & Intelligent Systems*, vol. 7, no. 5, pp. 2179–2198, 9 2021.

[12] X. Wang, Y. Chen, W. Ruan, Q. Gao, G. Ying, and L. Dong, "Intelligent Detection and Recovery of Missing Electric Load Data Based on Cascaded Convolutional Autoencoders," *Scientific Programming*, vol. 2020, p. 8828745, 12 2020.

[13] R. Memon, S. Qazi, and B. Khan, "Design and implementation of a robust convolutional neural network-based traffic matrix estimator for cloud networks," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–11, 06 2021.

[14] G. F. Pittalà, L. Rinieri, A. Al Sadi, G. Davoli, A. Melis, M. Prandini, and W. Cerroni, "Leveraging data plane programmability to enhance service orchestration at the edge: A focus on industrial security," *Computer Networks*, p. 110397, 2024.

[15] E. T. S. Institute, "Zero-touch network and service management (zsm) reference architecture," pp. 1–80, August 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf

[16] N. Di Cicco, G. F. Pittalà, G. Davoli, D. Borsatti, W. Cerroni, C. Raffaelli, and M. Tornatore, "Drl-forch: A scalable deep reinforcement learning-based fog computing orchestrator," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. IEEE, 2023, pp. 125–133.

[17] Z. Wen, W. Yin, and Y. Zhang, "Solving a Low-Rank Factorization Model for Matrix Completion by a Nonlinear Successive Over-Relaxation Algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[18] S. Gao and Q. Fan, "A Mixture of Nuclear Norm and Matrix Factorization for Tensor Completion," *Journal of Scientific Computing*, vol. 75, no. 1, pp. 43–64, 04 2018.

[19] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," 2015.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, 04 1998.

[22] P. Le Nguyen, Y. Ji *et al.*, "Deep Convolutional LSTM Network-based Traffic Matrix Prediction with Partial Information," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 261–269.

[23] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.