Machine Learning for Service Migration: A Survey

Nassima Toumi, Miloud Bagaa, and Adlen Ksentini, *Senior Member, IEEE*

***Abstract*—Future communication networks are envisioned to satisfy increasingly granular and dynamic requirements to accommodate the application and user demands. Indeed, novel immersive and mission-critical services necessitate increased computing and network resources, reduced communication latency, and guaranteed reliability. Thus, efficient and adaptive resource management schemes are required to provide and maintain sufficient levels of Quality of Experience (QoE) during the service life-cycle. Service migration is considered a key enabler of dynamic service orchestration. Indeed, moving services on demand is an efficient mechanism for user mobility support, load balancing in case of fluctuations in service demands, and hardware failure mitigation. However, service migration requires planning, as multiple parameters must be optimized to reduce service disruption to a minimum. Recent breakthroughs in computational capabilities allowed the emergence of Machine Learning as a tool for decision making that is expected to enable seamless automation of network resource management by predicting events and learning optimal decision policies. This paper surveys contributions applying Machine Learning (ML) methods to optimize service migration, providing a detailed literature review on recent advances in the field and establishing a classification of current research efforts with an analysis of their strengths and limitations. Finally, the paper provides insights on the main directions for future research.**

***Index Terms*—Machine learning, service migration, management and orchestration, deep learning, reinforcement learning, prediction, placement.**

I. Introduction

NETWORK evolution towards 5G and beyond (B5G) technologies has been motivated by multiple high-demanding applications such as autonomous driving, Augmented Reality (AR), and high-resolution video streaming. Indeed, these applications generate multiplied traffic volumes and are characterized by very stringent requirements for latency, throughput, network availability, but also density with the required support for increased numbers of connected users and devices, which current network technologies are unable to sustain. Thus, next-generation networks are expected to provide support for heterogeneous applications with highly precise Service Level Agreement (SLA) guarantees for latency and reliability, such as immersive Tactile Internet, haptic and mission-critical applications, while optimizing energy consumption and minimizing the carbon footprint [1], [2]. Future networks are also envisioned to achieve higher programmability by supporting increasingly granular and dynamic network behavior customization and attaining full automation and self-management through network intelligence [3], [4].

To overcome the ultra-low latency requirement, Multi-access Edge Computing (MEC) can be leveraged to deploy services on edge clouds in the vicinity of users. Indeed, the MEC concept allows computation and storage to be offloaded to edge devices or clouds that are closer to the end-users, thus decreasing service latency and improving the response time. Moving services to the edge also reduces network congestion and communication costs and accelerates content delivery through caching popular content [5], [6]. Therefore, end-users can attach to the closest edge cloud and access services with an improved QoE. However, a number of users access network services on their handheld mobile devices and can change locations during service provisioning. This mobility might increase the distance between them and their initial edge cloud, which would require passing communications through intermediary edge clouds to reach the service due to the limited coverage of edge clouds. This might lead to increased service latency and response time; even to service interruption.

To mitigate this issue and ensure service continuity, multiple works have proposed to move services along with the users, under different terms such as "Follow-Me Edge" (FME), "Follow-Me Fog", or "Companion Edge Computing" [7], [8], [9], [10]. The main idea is to migrate services to the nearest available edge or fog cloud, following the user's mobility seamlessly so that the end-user experiences minimal service disruption or degradation.

Software migration is defined as the process of relocating a virtual environment from one physical node to another by suspending the running instance on the source node, copying the disk, memory, and process states, and resuming the instance on the destination node [11]. It differs from re-instantiation in the sense that migration preserves session information for stateful services instead of creating a new session on a different service instance [12]. Besides mobility support, migration helps maintain services in case of node or link failure and prevents service degradation in case of network overload by moving services to physical nodes with sufficient resources.

Given the strict requirements of new applications, the migration process ought to be automated and optimized to reduce the impact on QoE to a minimum. Multiple parameters can

Manuscript received 21 March 2022; revised 31 October 2022 and 14 March 2023; accepted 29 April 2023. Date of publication 8 May 2023; date of current version 23 August 2023. This work was supported in part by the European Union's H2020 5G!Drones Project under Grant 857031. *(Corresponding author: Nassima Toumi.)*

Nassima Toumi is with the Communication Systems Department, EURECOM, 06904 Sophia-Antipolis, France, and also with the Department of Networks, TNO, 2595 The Hague, The Netherlands (e-mail: nassima.toumi@tno.nl).

Miloud Bagaa is with the Department of Electrical and Computer Engineering, Université du Québec à Trois-Rivières, Trois-Rivières, QC G8Z 4M3, Canada, and also with the Pouta, CSC-IT Center for Science Ltd., 02150 Espoo, Finland (e-mail: miloud.bagaa@uqtr.ca).

Adlen Ksentini is with the Communication Systems Department, EURECOM, 06560 Sophia-Antipolis, France (e-mail: adlen.ksentini@eurecom.fr).

Digital Object Identifier 10.1109/COMST.2023.3273121

TABLE I
COMPARISON OF RELATED SURVEYS AND STUDIES

Reference	Scope	Migration (Data Transfer)	Other Planning	ML Techniques
[11], [21]–[24]	Live migration technique evaluation, and impact analysis of different factors for VMs and/or containers	✓	X	X
[25]	Review of scheduling models for placement and migration of containers	✓	✓	X
[28], [29]	Systematic review on VM consolidation	X	✓	✓
[26]	Discussion of optimization strategies for service migration in MEC	✓	✓	X
[27]	Classification of mobility-triggered migration contributions in terms of optimization objectives	✓	✓	X
[30]	Comparative analysis of ML-based container orchestration approaches for resource provisioning	X	X	✓
[14], [17], [20], [31]–[35]	Overview of ML-based techniques and their use for network resource management	X	X	✓
[36]–[38]	Discussion of applications of DL to network management in MEC	X	X	✓
[39]	Classification of forecasting schemes for predictive migration	X	✓	✓
Current Survey	Survey of ML-based contributions to migration planning	✓	✓	✓

be optimized during migration planning, such as the timing of migration, destination node selection, and migration strategy. Indeed, service degradation can be minimized if the migration process is triggered proactively by predicting the user's mobility or the network load surges. Then, selecting the destination node should take into account migration costs, network bandwidth, and link latency. Finally, the migration strategy should be chosen depending on SLA requirements for service availability and multiple factors that affect the migration process duration, such as the Virtual Machine (VM) or container size, network bandwidth, or the probability of unexpected events disrupting the migration process.

One promising method that can help optimize migration is Machine Learning (ML), which has been successfully applied to different problems in multiple areas in networking [13], [14], [15], [16], and increasingly complex network management problems are being tackled thanks to Deep Learning (DL), a more advanced technique of ML that relies on Deep Neural Networks (DNN) and removes the need for manual feature engineering [17]. More recently, Reinforcement Learning (RL), another branch of ML, has gained momentum for its potential to automate network management. Indeed, it allows an agent to learn policies from previous experience and continuously improve decision making through feedback from the environment [18].

The main advantage of these AI-based methods compared to conventional approaches is the capability of deriving patterns and policies from datasets of input and output examples or from past experiences, and the ability to provide results quickly regardless of the size of the solution space. Indeed, optimization using classic algorithmic methods and mathematical models either require detailed knowledge of the problem and its dynamics to determine the optimal solution (which is not always possible for complex problems) or rely on exploration by evaluating all or a significant part of the solution space. For the latter, the computational solving time and cost can increase exponentially as the problem scales to bigger instances, which is not convenient for runtime use. Besides, AI techniques can dynamically adapt to changes in the environment, as opposed to fixed rules and static hard-coded algorithms and models. Relying on ML/RL methods allows full automation of the Life-Cycle Management (LCM) of services, thus eliminating the need for human intervention and making autonomous Zero-touch network and Service Management (ZSM) a reality [19], [20]. In this vein, service migration can benefit from ML/RL in multiple ways. For instance, using network analytics, ML can be leveraged to predict events that are likely to cause service disruptions, such as user mobility or network overload. This early prediction allows for preventive measures to be taken; in this case, to trigger migration in advance to minimize service disruptions experienced by the users. Furthermore, ML/RL accelerate decision-making when selecting the destination node and network path and reduce the migration time by choosing the appropriate migration scheme. To this end, a model can be trained to learn an efficient policy and output qualitative solutions in near real-time.

A. Related Surveys and Motivation

Table I summarizes related surveys in the literature. Previous works have studied related topics such as live VM or container migration or ML-based service orchestration but in a separate manner. Most live migration surveys focus on the data transmission phase of the process while ignoring the planning phase [11], [21], [22], [23], [24], which is essential for optimizing multiple aspects of migration to reduce the impact on QoE. Oleghe [25] provides a review of optimization schemes for the placement and scheduling of containers in a MEC environment but focuses on solutions based on traditional optimization methods such as multidimensional knapsack, Markov Decision Processes (MDP), or exact and heuristic algorithms. Similarly, Wang et al. [26] discuss optimization strategies for service migration in MEC and identify AI-based strategies as a promising tool for making efficient decisions in

future research works. Rejiba et al. [27] focus on mobility-triggered migration, and classify contributions in terms of optimization objectives such as time, cost, and success rate, but only a few ML-based solutions are mentioned, in a succinct manner. Zolfaghari et al. [28] and Dias et al. [29] provide systematic reviews on contributions to the VM consolidation process, which includes migration, and multiple works using ML techniques are mentioned in the latter. However, VM consolidation is only one of many use cases for service migration, and most of the included ML-based contributions focused on load prediction to trigger the migration process proactively. Furthermore, those surveys are limited to the trigger and placement phases, and do not include the migration strategy selection process which consists in determining the appropriate method for data transmission. Hence, this process can also be optimized using ML to minimize service interruption time. On the other hand, multiple surveys have discussed the opportunities and use cases of applying ML-based methods such as Deep Learning or Federated Learning (FL) for network management and resource provisioning in Cloud Computing [14], [17], [20], [30], [31], [32], [33], [34], [35], or with a focus on MEC [36], [37], [38]. However, migration-related contributions are ignored or briefly mentioned without deep analysis. Zhong et al. [30] perform a comparative analysis of ML-based container orchestration approaches for resource provisioning. Khan et al. [31] review ML-centric resource management approaches and identify multiple factors that can affect migration time, but without going into details on the subject, by mainly referring to works that use ML for prediction to reduce migration time. However, ML can be used to optimize migration in multiple other ways, such as placement or selecting the appropriate migration scheme. Similarly, Masdari and Khezri [39] present a survey of contributions using forecasting for predictive VM migration in Cloud Computing, where classification is provided in terms of prediction schemes.

To the best of the author's knowledge, the current survey is the first contribution that studies the application of ML-based methods to service migration optimization for all phases of the planning process independently from the virtualization technology. In contrast with previous work, this work covers service migration planning in both Edge and Cloud Computing contexts, which can be triggered by user mobility, workload, or failure. Additionally, more complex use cases where multiple instances are migrated in parallel are covered. Overall, the survey includes a taxonomy of works on all phases of service migration planning, namely, reactive and proactive migration trigger, placement, but also data transfer strategy selection.

B. Contributions and Paper Organization

This paper covers background on related technologies and the fundamentals of service migration and ML, and provides an exhaustive overview of the current research on the topic. The main contributions of this work are summarized as follows:

- The use of ML for service migration is motivated by describing multiple use case scenarios and applications.
- The main enabling technologies are identified and their impact on service migration is discussed.
- A taxonomy of the works of interest is provided and contributions are categorized into the different phases of the optimization process of service migration, with an analysis of their contributions and shortcomings.
- Insights are provided on challenges and open research issues based on the identified limitations of current works.

For the reader's convenience, the main abbreviations used in this paper are listed in Table II, and the paper's structure is shown in Figure 1, where the background knowledge sections are colored in blue, and the taxonomy sections are colored in red. The remainder of this paper is organized as follows. Section II provides an overview of the technological ecosystem of migration and its different enablers together with a set of use case examples. Section III details technical aspects of the migration process and Section IV provides a primer on ML and RL techniques. Sections V to VII review relevant contributions to migration optimization in terms of reactive and proactive triggers, service placement, and migration scheme selection, respectively. Section VIII discusses the open issues

TABLE II
LIST OF COMMON ACRONYMS

Acronym	Description
ACR	Application Context Relocation
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DNN	Deep Neural Network
DQN	Deep Q-Networks
DRL	Deep Reinforcement Learning
ETSI	European Telecommunications Standards Institute
GRU	Gated Recurrent Unit
KPI	Key Performance Indicator
LCM	Life-Cycle Management
LR	Linear Regression
LSTM	Long-Short Term Memory
MANO	Management and Orchestration
MDP	Markov Decision Process
MEC	Multi-access Edge Computing
ML/DL	Machine Learning/Deep Learning
NFV	Network Function Virtualization
NN	Neural Network
PER	Prioritized Experience Replay
PI	Policy Iteration
QoE	Quality of Experience
QoS	Quality of Service
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SDN	Software Defined Networking
SFC	Service Function Chaining
SLA	Service Level Agreement
UE	User Equipment
URLLC	Ultra-Reliable Low Latency Communications
VI	Value Iteration
VM	Virtual Machine
VNF	Virtual Network Function

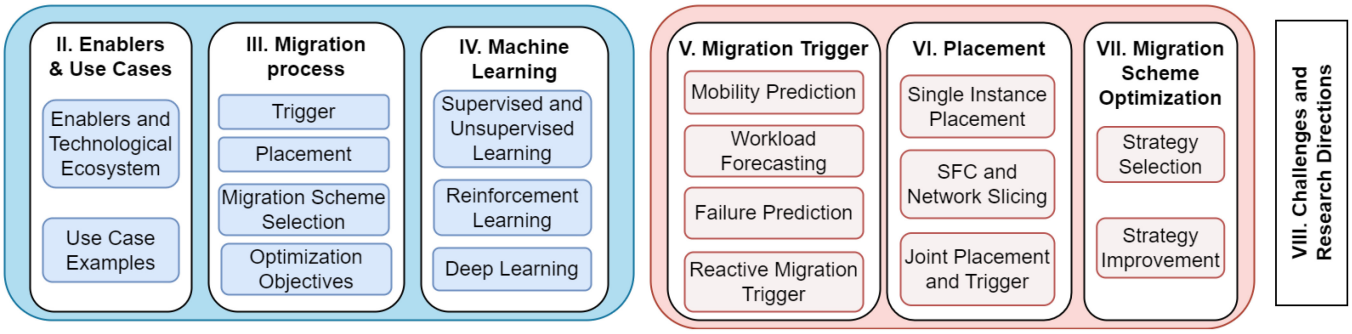


Fig. 1. The Survey Structure.

and future research directions, and Section IX concludes the paper.

II. Enablers and Use Cases

A. Enablers and Technological Ecosystem

This section provides a background on the enabling and related technologies for migration and discusses their effect on the process.

1) Network Function Virtualization: Network Function Virtualization (NFV) decouples network functions from proprietary dedicated hardware and allows the deployment of services on general-purpose servers as VMs or containers. NFV reduces capital and operational expenditures [40], facilitates service management, and allows flexible and efficient resource provisioning by dynamically scaling instances up and down depending on the evolution of their resource consumption and service demand [41]. Furthermore, virtualizing functions and services facilitates their migration to a different physical node. Two main technologies can be used for service virtualization:

a) Virtual machines: are deployed on top of hypervisors, with an Operating System (OS), kernel, and hardware-level virtualization. This allows deployments of independent VMs isolated from the host OS, thus making it possible to virtualize different guest Operating Systems on top of the host OS. However, due to the number of abstraction layers, hypervisor-based virtualization creates a high overhead, affecting overall performance.

b) Containers: Are a lightweight alternative to VMs, where the abstraction is made at the OS level instead of hardware virtualization for VMs. With containerization, the OS kernel of the hosting machine is shared between multiple isolated instances of user-space environments. This OS-level virtualization produces a lower overhead and reduces image size, making it faster to migrate containers compared to VMs [42].

2) Software Defined Networking: SDN is a paradigm that separates network intelligence from the forwarding components. It introduces a layered architecture comprising three separated planes: An application plane that formulates the desired network behavior and sends it through a northbound interface to a control plane where a logically centralized controller disposes of a holistic view of the network and translates the formulated behavior into a set of forwarding rules. Finally, a forwarding layer composed of interconnected network devices enforces the forwarding rules that are received through a southbound interface. Centralizing intelligence at the control plane reduces deployment costs and facilitates network management. Indeed, since the data plane devices only apply the forwarding rules, less intelligence is needed in the networking equipment, and cheaper devices can be used. Furthermore, the control plane can dynamically change the network behavior by updating the forwarding rules on all of the involved data plane devices in an automated manner, thus making the network programmable [43]. This dynamic management of network behavior facilitates service migration by automatically redirecting network flows to the destination node once the service instance has been resumed.

3) Microservices and Service Function Chaining: As virtualization gained popularity, the concept of micro-services emerged, which is the decomposition of services into multiple small blocks that perform simple functions, this approach improves services' modularity and resilience, adds flexibility in service management, and allows for function sharing between services [44]. Decomposing services into a set of functions creates the need for steering traffic between those functions in a certain order so that services are delivered properly; this process is referred to as Service Function Chaining (SFC) [45]. Service migration in a SFC context ought to consider additional aspects, which adds more complexity to the process. Indeed, services contain multiple Service Functions (SFs), where multiple instances of VMs and/or containers might need to be migrated. Since the service relies on all of the different functions, it cannot be resumed until the migration of all of the functions has been completed. Therefore, those functions should be migrated in a synchronized manner [46]. Moreover, the migration process should consider the link mapping for the new placements of the migrated instances and its effect on the end-to-end latency of the service.

4) Edge Computing: Edge Computing aims to provide storage and computing capabilities at the logical borders of the network in the close vicinity of users. It is a key technology for future communication networks [47] since it enables a wide range of delay-sensitive applications [48] by reducing the distance between services and the end-users, thus diminishing the end-to-end cost and latency of communications and facilitating real-time processing. Besides, moving computation

closer to the edge reduces network load and avoids transferring sensitive data. However, Edge nodes and links are resource-constrained, which means that particular attention must be paid to the amount of transferred data during the migration process to reduce network load, and avoid performance degradation [49]. Additionally, due to the time-sensitive nature of MEC applications, the migration process needs to be completed in a short time span. Two main edge-based concepts have emerged in recent years:

- *Multi-Access Edge Computing (MEC):* Formerly known as Mobile Edge Computing, MEC is a concept introduced by the Industry Specification Group (ISG) of the European Telecommunications Standards Institute (ETSI) [50], which moves the storage and computing capabilities to Edge devices in the access networks [51]. The initiative aims to provide a standardized reference architecture for MEC and create an open environment to facilitate MEC application development and integration.
- *Fog Computing:* Fog Computing was introduced as a way to extend core cloud computing capabilities into the edge, thus reducing the amount of data transmissions, and computations performed at the core cloud. Fog devices can be edge routers, switches, access points, computers, or UE devices such as smartphones. However, unlike MEC, fog nodes cannot operate independently from a core cloud [52], [53].

To support Edge applications in 5G systems, 3GPP standards [54], [55] define an architectural framework and a set of procedures that enable UEs to access Edge Application Servers (EAS) that are managed by Edge Enabler Servers (EES). In particular, the standards define the Application Context Relocation (ACR) decision process, where user context is relocated (i.e., migrated) from one EAS to another following detected or predicted UE mobility to ensure service continuity. The target EAS for ACR is selected using (predicted) UE position and specific application characteristics criteria, but the details of context transfer between the source and target EAS are left out of the scope of the standards.

5) Orchestration and Life-Cycle Management: Service orchestration provides a set of policies and procedures meant to guarantee the service's QoE and optimize the operator's objectives from the deployment of the service to its deletion. It ensures the proper service creation, operation, and deletion while satisfying the service KPIs by performing monitoring, anomaly detection, and elasticity and fault management procedures such as scaling or service migration to prevent or recover from service disruptions [56]. For efficient service orchestration, the decision-making process of different LCM operations needs to be optimized to determine the best course of action. Decisions should be made quickly for a short response time. Furthermore, orchestration policies should be dynamic to adapt to changing service requirements and unexpected events in real-time.

Although traditionally mathematical models and algorithms are used for optimization problems (e.g., Integer Linear Programming, different heuristics, or Game Theory), most of these proposals fall short on scaling to bigger or more complex problem instances, as the processing time exponentially

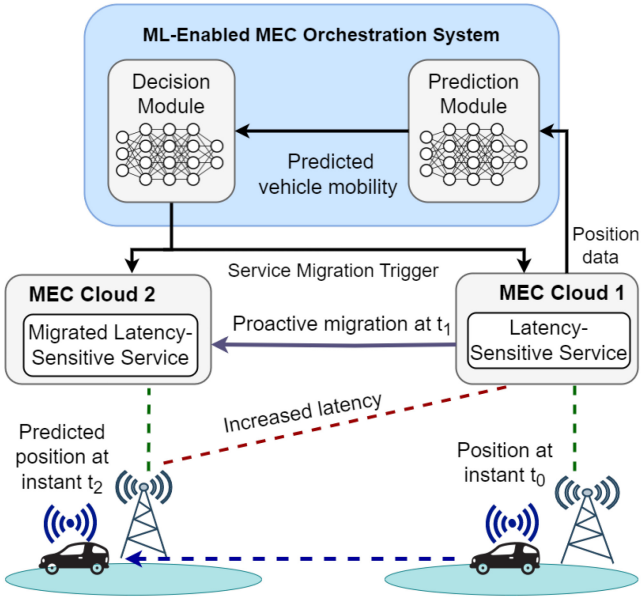


Fig. 2. Mobility Use Case in Multi-access Edge Computing.

increases with the size of the solution space. On the other hand, ML is gaining popularity as an efficient method to tackle more complex problems and is being applied to multiple research areas, including network and service management. Indeed, ML is expected to play a crucial role in accomplishing the objectives of B5G networks by enabling real-time decision-making and adaptive Zero-touch Service Management [57], [58]. In this vein, the 3GPP standards introduce the Network Data Analytics Function (NWDAF) in the 5G core architecture [59], which can be used to train ML models using data collected from other components of the architecture, and provide analytics. NWDAF predictions can then be used by the 5G system to trigger and optimize operations such as ACR for Edge applications [60]. Subsequently, as a part of the LCM process of services, and as shown in the use case examples in Section II-B, migration can also benefit from ML optimization.

B. Use Case Examples

To illustrate applications of ML for service migration, multiple use case examples are described in the following.

1) Mobility in MEC: The example in Figure 2 depicts a ML-enhanced Follow-Me Edge [61] system. End-users in mobility are connected to Edge Clouds using handheld devices and consuming services such as infotainment, high-definition video streaming, or interactive applications like cloud gaming. To reduce latency and transmission costs, these services are hosted on the edge clouds closest to the users. Since the users are mobile, they might move away from the Edge Clouds they were initially attached to, which results in increased response times, and service disruption. Therefore, the services must be dynamically migrated to the nearest edge cloud at all times, following the user's mobility patterns.

However, the migration process consumes time, as data needs to be transmitted to the destination edge cloud, impacting the duration of service disruption for the user. This

disruption can be minimized by using ML to trigger the migration process proactively, relying on the historical positions as well as velocity data to train a ML model that predicts the user's path, estimates the moment when migration will be required, and the best target edge cloud node. Using that information, migration can be triggered to the selected edge node in the predicted trajectory early enough to reduce service degradation time. Similarly, ML can be used for clustering by identifying services that are simultaneously used by groups of users and applying the same mobility patterns for migration.

2) Cloud-Edge Offloading: Deploying services at the edge reduces communication latency and cost and preserves the confidentiality of sensitive data. However, edge servers dispose of limited storage and computation capabilities and cannot host compute-intensive applications. In contrast, cloud servers are resource-sufficient, but placing services in the cloud increases the end-to-end communication latency and cost. Thus, the best choice for placing a service depends on application needs that might vary in time. In consequence, an efficient Cloud-Edge offloading scheme should adapt to changes in user application needs in real-time by moving services to cloud servers when the applications require more resources and to edge servers when lower latency becomes a priority. However, service migration has a cost and also causes service interruptions; hence, ML could be used to predict the changes in user demands in the short but also mid-term future. If spikes or dips in resource demand are predicted to be very short, or if too many fluctuations are predicted in a certain interval, it might be worth evaluating whether the multiple successive migrations would cause more disruptions than offloading the service for a limited number of times. Based of those predictions, a decision model can then be trained to schedule offloading operations with the objective of minimizing impact on the service's QoE over time.

3) Load Balancing: ML can also be used for load balancing to optimize resource allocation by moving service instances depending on service request evolution to avoid overloading or underloading Edge Clouds. For example, ML can predict surges in network traffic that can lead to service overload, and if the current physical node does not dispose of sufficient resources to scale the service up, the latter can be proactively migrated to a different hosting node with enough resources to host the scaled-up service. In contrast, ML can be used for consolidation by predicting the reduction in resource usage. Based on those predictions, the decision can be made to free underused physical nodes by relocating services to physical nodes hosting other services in a way that minimizes the number of used servers and reduces energy consumption.

4) Fault Tolerance: Industry 4.0 relies on Industrial IoT (IIoT) to automate and optimize the operations of the future smart factories by deploying IoT sensors to collect industrial data in real time, and transmit it to geo-distributed MEC nodes for analysis and decision-making [62]. However, factory automation use cases require a high level of reliability [63], which means that efficient mechanisms are needed to predict, detect and mitigate node failures. Similar to the previous use cases, ML can be used to anticipate failures by predicting early signs ahead of time and proactively migrating services from the potentially faulty node to a healthy one, which can reduce service interruptions compared to reactive measures. Furthermore, using past and present experience, RL models can help select the physical node candidates that are currently the most reliable for migrating services with strict requirements, such as Ultra-Reliable Low Latency Communications (URLLC). RL models will also be able to dynamically adapt to changes and events in the network by continuously updating their decision policy, which is an advantage compared to static failure mitigation policies.

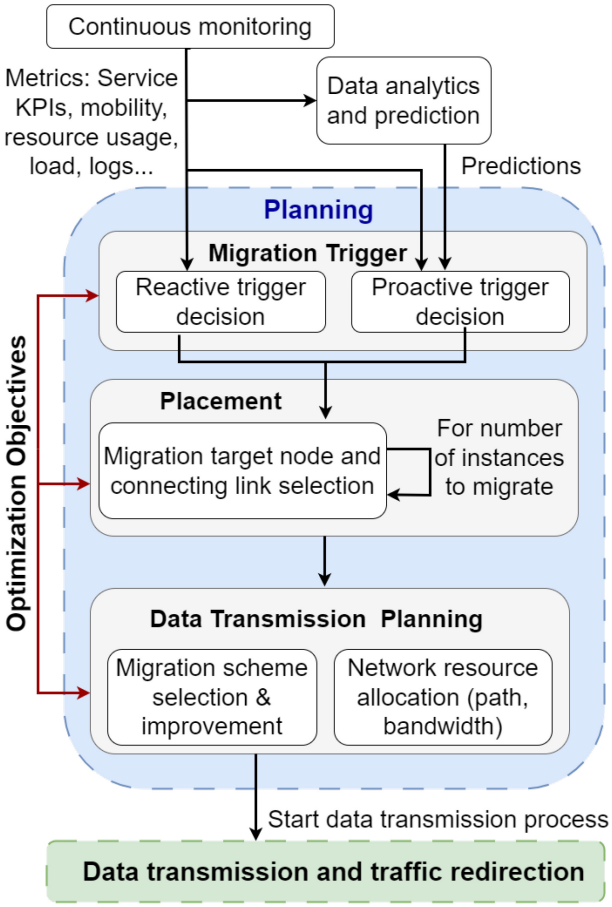


Fig. 3. Migration Process Steps.

III. Migration Process

As illustrated in Figure 3, the service migration process comprises two separate phases. The first is a planning step, where monitoring data is processed, and decisions are made regarding the moment when migration should be triggered, the destination node, the transfer link, and the migration scheme that should be used. Optimizing this process minimizes service disruption by forecasting migration triggers and proactively starting the migration process before service degradation is experienced, by selecting the best-suited node and link mapping for the migrated service instance, and by choosing the optimal migration strategy that minimizes QoE degradation for the service [64].

The second phase applies previous decisions: when the original instance is stopped, the state is effectively transferred to

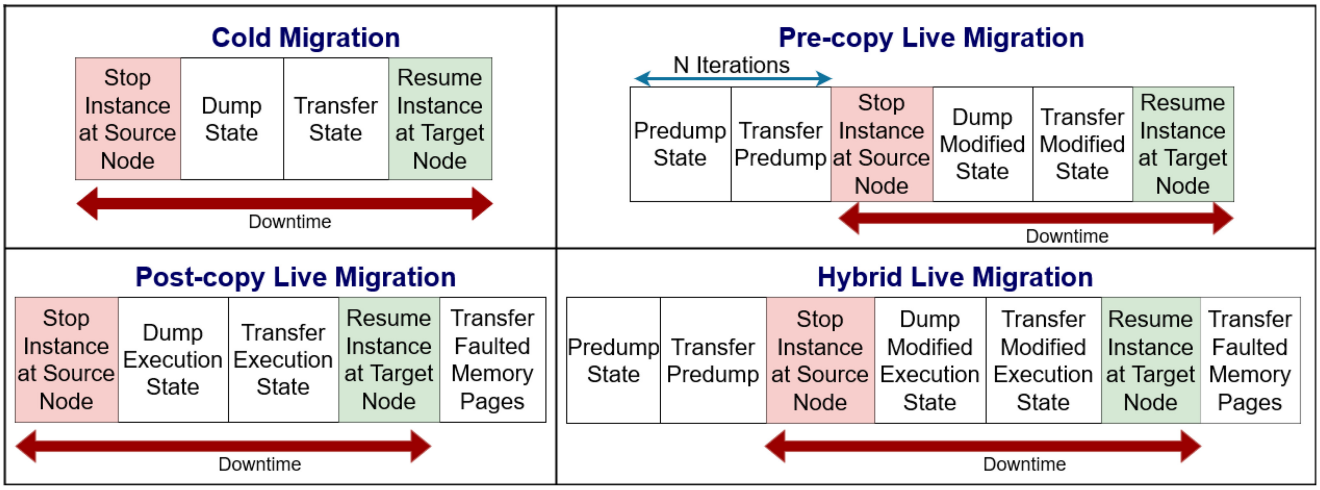


Fig. 4. Overview of the Different Migration Strategies.

the destination node, where the instance is resumed before performing service traffic redirection. This phase is out of the scope of this paper, as the surveyed contributions focus on the decision process in migration.

A. Trigger

The migration trigger process determines the appropriate moment when the service migration should start. Service migration can be performed in a reactive way if triggered manually or automatically after observing a degradation of QoE, or proactively if this degradation is predicted in advance, thus reducing or even preventing impact on QoE. Multiple events and factors can trigger automated service migration [65], [66], [67]:

- Mobility, when changes in the recorded positions of a group of User Equipments (UE) of interest imply that a significant portion of users of the service is moving away from the service's current placement nodes in a similar mobility pattern.
- Resource availability, when the request load increases significantly, and a scale-up of the service is required, but the hosting nodes do not dispose of sufficient resources. The monitored metrics, in that case, are the request load, resource capacity, and the percentage of resource usage.
- Reliability, in case of a major service disruption caused by hardware failure, or performance degradation observed through service availability time, and application-specific QoE KPIs.
- Security, if unusual system behavior indicates that the hosting infrastructure has been compromised.
- Consolidation based on resource usage distribution, to reduce power consumption by moving services from under-loaded nodes.

B. Placement

During the placement process, the destination node for the migrated instance is selected, and in the case of a SFC, the inbound and outbound virtual links of the migrated instances also need to be mapped again. This process is very similar to classic node and link mapping, which has been extensively explored in the literature [68], but as detailed in Section III-D, multiple migration-specific optimization objectives and constraints need to be taken into account additionally.

C. Migration Scheme Selection

Once the migration timing and target have been selected, the VM/container files and current state need to be saved and transmitted over the network to the destination node according to a selected scheme. Then, the service is resumed, the user traffic is redirected to the new instance, the original instance is deleted, and the resources are freed on the source node.

Multiple migration techniques have been proposed for both VMs and containers. Each method targets different metrics and has its own advantages and drawbacks. Thus, the appropriate technique depends on the service's constraints and priorities, as well as multiple factors that can affect migration, such as the VM/container size, network bandwidth, or page dirtying rate [22], [24], [69]. Furthermore, certain methods require additional parameters to be optimized, for instance, the number of iterations for the pre-copy live migration, as shown in the following.

Service migration methods can be classified into two types: *stateful* and *stateless*. Stateless migration concerns services that have no running state and do not keep any session information, meaning that services can be simply deployed by starting new instances on the destination nodes. Stateful migration, however, is more challenging, as it requires state information to be preserved and transferred to the destination node to resume the service. Most contributions in the literature focus on stateful migration since stateless migration only consists of stopping the instance at the source node and restarting it from scratch at the destination node without preserving the current running state. Therefore, the scope of this survey is limited to stateful migration due to its complexity and challenges.

Multiple strategies have been proposed for stateful migration as shown in Figure 4:

1) Cold Migration: Where the original instance is stopped/frozen, then its state is dumped and transferred over the network, and the new instance is resumed on the destination node once the state has been completely transferred. Using this approach, the service downtime is equal to the total migration time, and the service is unreachable during the whole migration process.

2) Live Migration: Which aims to minimize downtime, by keeping the instance running during most of the data transmission process. Multiple methods can be used:

With the *pre-copy* method [70], the whole state is saved and iteratively transferred. At each iteration, the memory pages modified during the previous iteration (called dirty pages) are copied to update the instance state. Once a predefined number of iterations has been reached, the instance is stopped, and the remaining dirty pages are transferred. However, a number of memory pages might be transferred several times if they are modified during multiple iterations, which increases the amount of transferred data and the total migration time that becomes non-deterministic. The pre-copy method is the most commonly used live migration technique and is implemented on most popular hypervisors such as VMware ESXi, Microsoft Hyper-V, Xen, and KVM [71].

In contrast, the *post-copy* method [72] differentiates between execution state and memory pages by migrating them in separate phases. The instance is first stopped at the beginning of the process, and only the execution state is transferred. The instance is then resumed, and the memory pages are transferred only once, when required by the new instance during its runtime, thus reducing data transmission load and cost. However, service disruption might be caused by missing pages during the instance execution, and resources at the source node are not released when the new instance is resumed due to the incomplete transmission process. In fact, the original instance must be kept on the source node until all of the faulted pages have been transferred, otherwise causing loss of data, which means that resources are reserved both at the source and destination nodes for a longer period.

The *hybrid* method aims to combine both previous methods [73]. It first dumps and copies the execution and memory state for a limited number of iterations to reduce the network overhead compared to the pre-copy method. Then, the instance is stopped, and the modified execution state is transferred to the target node before resuming the instance. Then finally, the faulted memory pages are transferred on demand. Compared to the post-copy method, the number of faulty pages is significantly reduced since most memory pages were transmitted during the pre-dump phase, and only the dirtied pages require transmission.

More recently, new contributions introduced another level of granularity by proposing *application-aware* approaches that consider application types and context. One proposal is to categorize transferred data according to their access frequencies, and the least accessed data would then be migrated in priority, as the probability of dirty pages and re-transmission is lower [74]. Another proposed optimization approach is to identify the core state needed by the application to initialize, and migrate it in priority (or in advance when possible) to reduce downtime [75]. Additionally, multiple SDN-based frameworks have also been elaborated to coordinate application state transfer by dynamically selecting the appropriate network paths for data transfer flows [76], or through flow prioritization [77]. Other frameworks aimed to reduce the volume of transferred data using distributed databases [78], or using compact *statelets* [79].

D. Optimization Objectives

Additional to the service's KPIs that should be maintained to avoid SLA violations, all of the steps in migration planning should optimize metrics that are specific to the migration process:

- Migration cost, which encompasses the cost of transmitting data over the network links, the additional cost of deploying services on the destination nodes and redirecting traffic, and possible penalties from SLA violations occurring during the migration process.
- Total migration time, which is the duration from the moment when migration is initiated, to the moment when the data has been completely transmitted to the destination node, and deleted from the host node.
- Service downtime, the duration for which the service is unreachable, which is critical for URLLC services.
- Network load, which is the amount of data transmitted over the network during migration, which is an important metric in bandwidth-constrained networks.
- Energy, which is the amount of energy that has been consumed due to the migration process. Additional to data transmission energy consumption, the original instance is kept running on the source node until the instance at the target node is ready, thus consuming energy at both nodes.

IV. Machine Learning

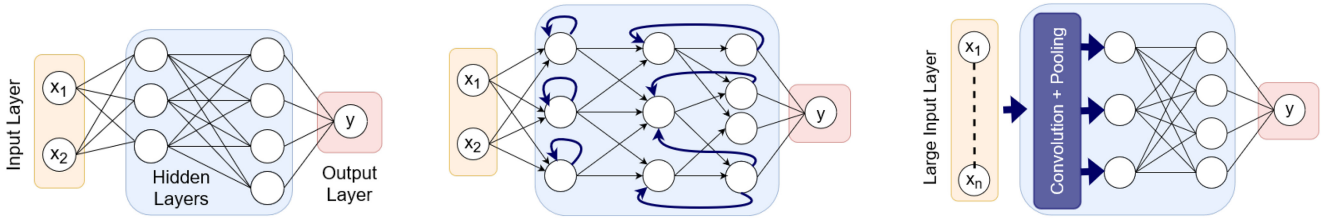
Owing to substantial advancements in computing capabilities and the growing volumes of generated and available data, recent years have witnessed a growing interest in ML and its applications in a wide range of research areas. Indeed, ML techniques give computing systems the ability to derive policies, recognize patterns and extract features without being explicitly programmed for those tasks; the learned knowledge is then applied to process new data. ML techniques are generally divided into three types: *Supervised*, *Unsupervised*, and *Reinforcement Learning*.

A. Supervised and Unsupervised Learning

In Supervised Learning, pairs of input x and ground-truth output y are provided to train the model, then, the model is used to predict the value of y for a different set of inputs while minimizing a loss function that penalizes prediction errors [80]. In Supervised Learning, input data must be labelled, which requires a pre-processing step to extract features, which is generally performed manually, and requires a detailed knowledge of data. The main applications for Supervised Learning are classification, where one class is selected among a finite set, and regression, where the

TABLE III
DEEP LEARNING TAXONOMY

DL Method	Concept	Main Algorithms	Use Cases	Use for service migration
Recurrent NNs	The output of a layer can be fed back to its input, or the input of a previous layer	LSTM, GRU	Data sequence processing, time-series-based prediction.	Predicting when migration will be needed.
Convolutional NNs	Use convolutional and pooling layers to reduce input dimension and extract features.	CNN	Problems with large input dimensions such as computer vision	Migration trigger prediction and placement decision.
Deep Reinforcement Learning	Use feedback from the environment to improve the decision policy of the NNs.	DQN, DDPG, A2C, A3C	Learning decision policies	Learning a policy for migration target selection.



(a) Feed-forward Neural Networks (b) Recurrent Neural networks (c) Convolutional Neural Network

Fig. 5. Relevant Neural Network Architectures.

output is a continuous number. Algorithms from the regression class such as Linear Regression (RL) are mainly used for the prediction and recommendation systems. Conversely, Unsupervised Learning takes unlabeled data as input, and infers data distributions by detecting similarities between data points [81]. It is mainly used for clustering, by grouping similar data points into a pre-defined number of homogeneous classes, outlier detection to isolate anomalies, and feature extraction to automate the pre-processing step of Supervised Learning [82]. Both techniques can be combined in Semi-Supervised Learning by using both labeled and unlabeled data for training [83].

B. Reinforcement Learning

Reinforcement Learning is inspired by human learning, where the agent learns behavior through its interactions with the environment by evaluating selected actions according to the observed outcomes.

Given an observation of the environment's current state s_t, the agent outputs an action a_t, then receives the new state of the environment s_{t+1}, and a reward or penalty r_t depending on the outcome of that action. Thus, through a feedback loop with the environment and by exploring the set of possible actions, the agent learns a policy that aims to maximize its cumulative reward by selecting the best-evaluated action for each given state based on previous experience. The agent can also dynamically respond to observed changes by continually adjusting its policy. When the state space for a problem is finite, the state action transitions can be modeled using a MDP, and the optimal policy can be constructed through basic Dynamic Programming (DP) methods such as Policy Iteration (PI) or Value Iteration (VI) when the state transition probabilities are known or using more sophisticated model-free RL algorithms such as Q-learning, SARSA, or Monte-Carlo where policies are improved and evaluated through direct interactions with the environment [18].

C. Deep Learning

A more advanced field of ML uses Neural Networks (NN) for training in more complex problems. NNs are composed of multiple layers of neurons that are interconnected using weighted links: The input layer receives input data, the output layer returns the inference results, and intermediate layers are called *hidden* layers. The number of hidden layers in a NN defines its depth, and NNs that comprise more than one hidden layer are called Deep Neural Networks (DNN). Each link between neurons of successive layers is characterized by weights a and b. During inference, for each value z received as input by a neuron from the previous layer, the neuron applies the weights a and b associated with that link ($a * z + b$). The weighted inputs for each neuron are then summed, and the result is passed through a non-linear activation function such as *sigmoid*, *tanh*, or *ReLU* before being transmitted to the linked neurons in the following layer [17], [84]. Inference is usually performed by forward propagation through the network layers, and learning is performed by updating the link weights using a backpropagation mechanism by calculating a loss function derivative. Multiple methods for DRL can be found in the literature (i.e., Feedforward, Recurrent, and Convolutional NNs, Deep Belief Networks, Generative Adversarial Networks, Deep RL). For the reader's convenience, Table III includes a taxonomy of the relevant ML methods.

1) Feed-Forward Neural Networks: Or Multi-Layer Perceptron (MLP) is the default architecture for NNs, where the output of each layer is directly fed to the next one, without additional computation as shown in Figure 5(a).

2) Recurrent Neural Networks (RNN): Introduce recursion in the hidden layers by feeding the output of a layer back into previous layers as shown in Figure 5(b). RNNs are particularly useful for processing sequences of data, such as text, speech, or time series, but suffer from a *vanishing gradient* issue, which can be solved using Long-Short Term Memory (LSTM) [85] units composed of memory cells, input, output and forget gates or Gated Recurrent Units (GRU) [86] composed of the reset and update gates. These architectures allow the RNN to efficiently learn long-term dependencies.

3) Convolutional Neural Networks (CNN): Illustrated in Figure 5(c), are generally used to process inputs with larger size or dimensionality, such as images. Complexity in those cases is reduced by adding two types of layers to the architecture: convolutional layers that compress input size by extracting features and pooling layers that combine the previous layer's input by computing the maximum or average value [87].

4) Deep Reinforcement Learning: Deep Reinforcement Learning (DRL) methods rely on using DNNs for RL by training them using feedback from the environment, where the DNN gradually adjusts its weights depending on the returned rewards and penalties [88]. Once the model has been properly trained, it can provide near-optimal solutions in a short time, which makes DRL particularly suitable for real-time service orchestration and management [34]. Popular DRL approaches include value-based methods such as Deep Q-Networks (DQN) [89], and policy-based methods such as Advantage Actor Critic (A2C) [90], Asynchronous Advantage Actor Critic (A3C) [90], or Deep Deterministic Policy Gradient (DDPG) [91]. Figure 6 illustrates the difference between value-based and policy-based Actor-Critic methods. Value-based methods are mainly used for discreet action spaces, where the Neural Network takes the state as input and outputs the Q value, which corresponds to the expected cumulated reward for each possible action. In comparison, the Actor-Critic methods are employed for continuous action spaces and use two separate neural networks: The actor network's input is the state, while the outputs are the exact action values. The critic provides the Q value for that state and action pair. While the actor network generates the actions that should be enforced in the environment, the critic network's role is to evaluate (i.e., criticize) the actions by measuring the value function of those actions, so that they can be adjusted accordingly.

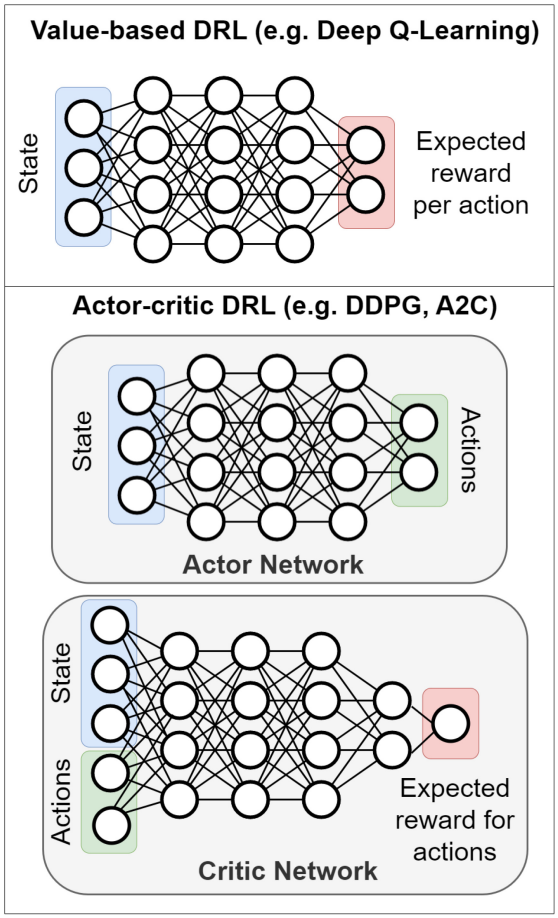


Fig. 6. Comparison between value-based and Actor-Critic Deep Reinforcement Learning algorithms.

V. Migration Trigger

As discussed earlier, the migration trigger is the step in the planning process to determine the appropriate moment for starting service migration. Migration can be triggered reactively by taking into account the current service state or proactively by forecasting different events that can potentially provoke QoE degradation in services, such as user mobility, workload peaks, or hardware failures. In this survey, prediction contributions are classified in terms of look-ahead window size as the following: short term for a few seconds up to less than 5 minutes, mid term for up to a couple of hours, and long term for higher.

A. Mobility Prediction

User mobility prediction has been extensively studied in the literature due to mobile operators' need to anticipate future user movements and trigger handover or radio resource reservation in a proactive manner [105], [106], [107]. As MEC brought computation closer to users, and as shown in the use case example in Section II-B1, service migration became a requirement for user mobility support in next-generation networks. On the other hand, proactively starting the migration process can reduce QoE degradation but requires an accurate prediction of the user's future positions. Indeed, as shown in the study by Gilly et al. [108], mobility prediction errors have a linear effect on the average delay for MEC users post-migration.

Thus, multiple works have included ML-based mobility prediction into their migration schemes to improve performance, as illustrated in Table IV.

In terms of input data used to train the algorithms, the datasets differed depending on whether the objective was to predict user [94], [96], [97], [101], [104] or vehicle mobility [98], [99], [100], [102], [103], or both [92]. Contributions that targeted vehicular mobility mainly relied on recorded taxi traces from cities such as San Francisco and Rome, or software-generated samples, while user mobility leveraged various mobility traces spanning months to multiple years. One

TABLE IV
SUMMARY OF MOBILITY PREDICTION REFERENCES (SECTION VII-A)

	Ref.	Prediction target	Method	NN Architecture	Dataset	Results
Short term	[92]	Future app. specification (includes user position)	Matrix Completion [93]	-	Combination of 3 datasets for vehicular and pedestrian mobility (2011,2017)	Estimation error of $< 20\%$
	[94]	Edge cloud visit probability of subscribers	Mixture Density NN [95]	4 layers: 512, 128, 12, RMSProp	Geolife, pedestrian GPS coordinates of 182 users over 5 years (2007-2012)	Accuracy > 95 %
	[96]	Distribution of mobile users and resource demands	ConvLSTM	2 ConvLSTM layers with 200 filters, each followed by batch normalization	GPS of 100 taxis in Rome over one month (2014)	MSE less than 5%, ConvLSTM outperforms LSTM with attention
	[97]	Future sequence of user locations	LSTM	Embedding and projection layers: 1024 neurons. Hidden layer: 256 neurons	Human mobility traces from CRAWDAD NCSU data set (2009)	Average latency reduced by 15%
	[98]	Density distribution of users	LSTM and GRU	2,3, and 4 layers of 50 neurons each	Vehicle trace generated using SUMO	LSTM accuracy around 85%
	[99]	Handover probabilities to neighboring eNBs	GRU-based RNN, CNN, Markov Chain	6 layers: 2 GRU layers of 20 neurons, 2 convolutional with batch normalization, 2 dropout layers	300 vehicle traces generated using SUMO	Accuracy of over 95%
	[100]	User's next cell	CNN	2 convolution layers with 128 filters then 2 fully connected	San Francisco taxi dataset (GPS of 536 taxis over 30 days in 2009)	QoS violations reduced by 15%
Short and mid-term	[101]	Single and group user trajectory prediction	LSTM	Set using Neural Architecture Search and Transfer Learning	GPS coordinates of 100 users over 2 years (2009-2011)	Accuracy of 60-90%
	[102]	Vehicular cloud availability to user before migration is required	LR combined with NNs	One layer of 200 neurons	54000 samples of software-generated vehicular traffic	Precision of 93%
Mid term	[103]	Vehicular User route trajectory	SVM	-	Generated, observations from 5 users over 60 days	Very high accuracy (almost 100%)
Long term	[104]	User positions	Factor Graph Learning	-	Dataset of locations and social behaviour of a 100 users on a social network app for 20 days	94% Precision, 7% more than SVM

observation is that apart from the New York taxi mobility traces [109] used in [92], all of the other datasets comprise less recent traces that have been collected between 2007 and 2014.

A few contributions combined mobility data with additional information to improve prediction quality. Indeed, the solution presented in [104] is able to achieve similar performance with data from only 20 days by also considering the social behaviour of users, which provides additional insights on their future mobility. Similarly, Rago et al. [96] predict the distribution of mobile users on available edge clouds, but also their resource demands in terms of communication and computing. The authors argue that this joint prediction allows the MEC orchestrator to perform predictive resource allocation more accurately by accounting for changes in resource demands.

Looking at the window size for prediction, it can be noticed that most of the surveyed papers targeted a short term window size by predicting the position of the user or vehicle in the next few time steps, with fairly high accuracy values reaching 80 to over 95%. However, short term prediction, especially when targeting the immediate next time-step, would not be sufficient to perform migration early enough to prevent disruption of services.

A majority of the surveyed contributions used NN and RNN-based ML methods such as LSTM, GRU, and Convolutional LSTM (ConvLSTM) [110] with a variety of Neural Network architecture configurations in terms of depth and number of neurons. Zhao et al. [101] set the hyper-parameters of the LSTM prediction agent (i.e., NN depth, layer size, connections) using the RL-based Neural Architecture Search framework [111], and by using Transfer Learning (TL) from previously tested parameter configurations as a *teacher* to accelerate the learning process. Fattore et al. [98] also test multiple values for the number of layers for LSTM and GRU models, and conclude that for the specific input dataset and prediction target, LSTM with 3 layers provided the best results, while training significantly faster than the GRU equivalent architecture (20 to 120% depending on depth).

A few papers combined mobility prediction with migration target selection, by using the future position of users to determine the closest edge clouds and select the most appropriate one for migration. To cope with cases where the predicted positions are not accurate, Dalgkitsis et al. [100] introduce a *confidence level* metric for the predicted next edge clouds, depending on which the service instance is either directly migrated to only one edge cloud, or duplicated and placed on multiple edge clouds; then only one instance is kept post-handover. In the same vein, Labriji et al. [99] predict handover probabilities to neighboring eNBs that are assumed to be co-located with MEC Edge platforms hosting services, which means that the migration destination can be inferred from the obtained predictions. At each handover event, multiple strategies are used for service migration: (1) full replication, where the VM is migrated to all of the neighboring MEC nodes, (2) a single replication, where the target is the one with the highest handover probability or (3) a proposed strategy that optimizes the number of replications.

On the other hand, longer term predictions are more challenging compared to short term but allow migration planning ahead of time, as well as migration scheduling optimization. By forecasting a bigger part of the user or vehicle's trajectory, it may be possible to select more optimal targets that the user can attach to for a longer period instead of only migrating to the cloud immediately next to the source. In the long run,

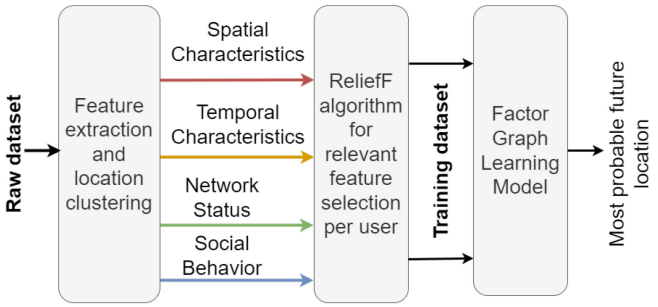


Fig. 7. Relevant feature extraction and social data correlation for user position prediction in [104].

such strategies would help reduce the number of migrations and thus minimize the impact on QoS.

Kuruvatti et al. [103] used Support Vector Machines (SVM) to predict mid-term vehicular routes, which is not as granular as GPS positions, but considering that road maps are known, it can still be sufficient to foresee whether or not the vehicle is expected to leave the coverage area of its current serving edge cloud. However, the dataset used to train the SVM model was comprised of observations of only 5 users, which is not sufficiently representative. As illustrated in Figure 7, Wu et al. [104] combine mobility information and social data to forecast future positions for service migration in MEC using a Factor Graph Learning coupled with selective feature extraction using the ReliefF [112] algorithm to reduce complexity. As mentioned earlier, by correlating mobility and social data, the proposed solution is able to achieve performance for long term prediction that is similar to those obtained by other solutions for the short term, with a significantly smaller training dataset.

B. Workload Forecasting

Effective workload forecasting allows the early detection of future over-loaded hosts that might not be able to support VM scaling, or under-loaded hosts, in which case a consolidation process would allow for more energy-efficient service placement.

The work conducted by Ricardo et al. [115] proposes a methodology for evaluating the impact of ML-based overload prediction methods on VM migration performance. Multiple methods based on Linear Regression, Neural Networks, LSTM, Simple Moving Average (SMA) and SVM algorithms are evaluated using performance metrics for migration such as energy consumption, SLA violations, and performance degradation. The results show the importance of accurate workload prediction for increased performance in the VM migration process. The contributions that used ML to perform workload forecasting for service migration are listed in Table V.

The input datasets used to train the prediction models are mainly based on real-world traces. In particular, many contributions used datasets pulled from the CoMon monitoring system [136], which provides realtime monitoring data from the PlanetLab platform, a real-world network that comprises over 1000 nodes spanning over 40 countries. However, the CoMon project is not active anymore, an interesting alternative would be the Measurement Lab (M-Lab) project [137], which is a similar open distributed server platform that provides detailed measurement data. A few works also used measurements from large clusters made available by companies such as Google [128], [130], Hadoop [126], and Ali-Baba [117], or from operator backbone links [120], [132], [133]. A minority generated synthetic samples from simulations or from existing signals.

It can also be noticed that different from mobility prediction that used traces collected over multiple years, workload prediction uses input traces that are much smaller. Indeed, traffic loads usually have a cyclic nature, where service demands fluctuate depending on the time of the day and also differ between weekdays and weekends.

The main prediction targets are future CPU usage and traffic load, with the exception of [117], [118], and [130] that also considered memory and bandwidth usage.

Similar to mobility prediction, workload forecasting contributions also mainly focus on the short term time window. Among those short term contributions, RNN-based methods are used the most, with different variants and optimizations. In [129], different LSTM variants were tested (LSTM, bi-LSTM, bi-LSTM+, CAT-LSTM) with and without attention, with similar results for all configurations. Duggan et al. [131] analyze the maximum number of time-steps for which the prediction accuracy is acceptable. The authors train an RNN using the Back-Propagation-Through-Time (BPTT) algorithm instead of the vanilla backpropagation and observe that the model can accurately predict CPU consumption 15 minutes into the future. To maximize prediction accuracy, Mason et al. [127] use RNNs to predict host CPU consumption where RNN weights are determined using swarm and evolutionary optimization algorithms such as Particle Swarm Optimization (PSO) [138], Differential Evolution (DE) [139] and Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [140].

On the other hand, and despite their simplicity, statistical and regression-based methods such as Simple Moving Average, Autoregressive Integrated Moving Average (ARIMA), simple/multiple Linear Regression, Support Vector Regression (SVR), and Locally weighted linear regression (LWR) (which don't require an intensive training phase) achieve similar, sometimes better performances than more elaborate DL methods such as NNs and LSTM for the short term time window. One exception is the work done in [125] where location-aware Multivariate LSTM performs better than location-unaware ARIMA. However, NN performance is bound to the specific architecture/hyperparameter configuration that has been implemented, as well as the input dataset. Thus this conclusion cannot be generalized. Nevertheless, Regression-based methods require less computation and demonstrate a good performance overall, which makes them an interesting alternative to NN-based models for short term prediction.

In terms of NN architecture, it can be seen that generally, RNNs with one hidden layer, sometimes with as little as 3 neurons, can have a similar performance as a NN of 3 layers with 100 neurons each on the same dataset (PlanetLab).

TABLE V
SUMMARY OF WORKLOAD FORECASTING REFERENCES (SECTION VII-B)

	Ref.	Prediction target	Method	NN Architecture	Dataset	Results
Short term	[113]	VM CPU usage	LR	-	5 days from Planetlab (On-demand)	2% less SLA violations than static rule
	[114]	VM CPU load	LR with error estimation	-	3 days from PlanetLab (On-demand)	SLA violations reduced to $< 1\%$
	[115]	CPU utilization	LR, NNs, LSTM,SVM Simple Moving Average	3 layers of 100 neurons	PlanetLab (On-demand)	SMA (non-trainable) and LSTM had the best performance
	[116]	Future host load	Binary decision tree classifier	-	3 days from PlanetLab (On-demand) + random	Similar to [114]
	[117]	Host usage for CPU, memory, bandwidth	Hybrid kernel based SVR	-	Ali-Baba data set (1300 machines during 12h) (2017)	Mean Absolute Percentage Error $< 1\%$
	[118]	SFC requirements in CPU, memory, bandwidth	Deep Belief Networks [119]	5 layers: 60-128-32-16-6 neurons	10k generated samples (superposition of sinusoidal and cosine signals)	RMSE < 0.16
	[120]	Traffic load fluctuation	Gaussian Process Regression [121] and change point detection [122]	-	Two 48h-long traces from internet backbone link (2018,2019)	Average prediction error of 0.13
	[123]	Future CPU usage	New Linear Regression	-	4 days of PlanetLab (on-demand)	50x less energy and SLA violations than threshold
	[124]	Resource usage patterns	Linear Regression, MLP, SVR, Decision Tree Regression, Boosted Decision Tree Regression	2 layers	5 days of PlanetLab (on-demand)	MLP had the lowest MSE ($< 2\%$)
	[125]	Location-aware workload	Multivariate LSTM	One layer of 50 neurons	San Francisco and Rome taxi traces (35 days) (2008,2014)	Accuracy improvement of $> 40\%$ compared to location-unaware ARIMA
	[126]	Node workload	Auto-regressive Modeling, SVR, NN, LWR	One layer of 10 neurons	Google Cluster trace, OLTP, Hadoop (2008,2011,2012)	LWR MSE is less than 0.3, NN performed the worst
	[127]	CPU utilization	RNN with PSO,DE, CMA-ES	One layer of 3 neurons	8 days of PlanetLab (on-demand)	MSE of 4% for 1 step ahead, 9% for 4 (PSO,DE, CMA-ES more accurate than backpropagation)
	[128]	CPU, memory, disk usage	WMA, ESA, HWM, AR, NN	One layer of 3 neurons	Google workload (2012)	Reduced energy consumption, number of migrations
	[129]	CPU Usage	LSTM, bi-LSTM, bi-LSTM+, CAT-LSTM	6 LSTM layers	12h of data generated from deployed SFCs	RMSE reduced from 29 in simple historical sequence to 7 with SFC
Short, mid term	[130]	Resource usage	Multiple Linear Regression	-	24h from Google Cluster dataset (12k machines) (2016)	Accuracy of 99% for very short term, 95% for short term, 90-93% for mid-term
	[131]	Host CPU consumption	RNN	One layer of 3 neurons	10 days of PlanetLab (on-demand)	MSE < 0.05 for up to 15 minutes
Short, mid, long term	[132], [133]	Future host capacity	DNN composed of 3D-CNNs and fully connected layers	3 layers: 3×3×3 kernel, 6×6×6 kernel, dropout	2 months of measurement from a major operator in a large metropolitan region	$<1\%$ of un-serviced demand for all time windows
	[134], [135]	Traffic volume	Deep Spatio-Temporal NN (Fusing ConvLSTMs and 3D-ConvNets)	Fusion of 3 ConvNet layers and ConvLSTM twice	7 week data trace from the city of Milan (2015)	Up to 35 % less error than ARIMA for mid and long term. In short term, ARIMA performed equally, sometimes better

It is, therefore, not necessary to use deeper neural network architectures to obtain a good performance in short term load prediction.

In contrast, mid and long term predictions have not been as extensively explored as short term. Evaluation results in [130] show that Multiple LR performs well for mid-term with an accuracy of 90-93%, while the authors of [134], [135] demonstrate that Regression methods are not efficient for long term prediction, with a mean error multiplied by a factor of 4 to 5. Instead, the proposed ConvLSTM-based solution achieves higher performance with 35% less error. Similarly, the solution in [132], [133] leverages a Deep Spatio-Temporal Neural network (D-STN) [135], which is composed of an encoder with a stack of 3D-CNNs and ConvLSTMs to forecast future host capacity, with a result of under 1% of un-serviced demands for all time windows.

Although NN-based prediction methods are more complex than Linear Regression, they prove more efficient in medium and long-term prediction, which allows better-informed decisions in the long run.

Another observation is that adding context information and correlating data can improve overall performance. Kim et al. [129] used dependencies between VNFs in SFCs to predict future CPU usage, which achieves a significant error reduction compared to using a simple historical data sequence. Nguyen et al. [125] also aimed to predict location-aware workload by using mobility data to predict future workload based on the workload of nearby edge data-centers, which improves accuracy. Bega et al. [133] provide a context-aware prediction solution by considering traffic from different types of services (i.e., Youtube, Snapchat, Facebook).

C. Failure Prediction

In high-availability systems, anomaly detection and prediction is a critical process for failure mitigation and avoidance. ML techniques and particularly DL methods have proven their efficiency in addressing this challenge [141], and as illustrated in Table VI, multiple works perform failure detection either from workload metrics or from log messages to proactively trigger migration from the future faulty nodes to healthier ones. Failure prediction is different from mobility and load prediction in the sense that the latter forecasts continuous values, and performances are evaluated by calculating

TABLE VI
SUMMARY OF FAILURE DETECTION REFERENCES (SECTION VII-C)

	Ref.	Method	NN Architecture	Prediction input	Results
Short term	[143]	Multi-agent Q-learning	-	Host performance metrics, 5 days from PlanetLab (On demand)	SLA violations reduced to 12%
	[145]	FastTree, RandomForest, SVM	-	Disk and system-level data, 1 month real-world data from a large-scale cloud (2017)	FastTree has 20% increased true positive rate compared to SVM, RandomForest
	[144]	LSTM, Bi-LSTM	2 layers of 100	Cluster performance metrics, Bitbrains dataset (2 months of CPU and memory usage in 2014)	Accuracy of 95% for CPU, 97% for memory
	[142]	Chained NNs	6 NNs with a 52 neuron hidden layer	MOGON HPC failure traces (2019)	Prediction of 73.02 % of failure events, and almost no false positive
	[147]	Bi-LSTM, Random Forest, Ranking Model	One dense layer of 128 neurons after input layer of bi-LSTMs	Fault data, 3 datasets of 1 month (2017)	92% Precision and F1 score of 75%, better than LSTM (72, 60)
	[148]	Two-phase LSTM	2 hidden layers	Cray: Logs of 4 systems over 8-12 months, 1800 to 6500 nodes (2017)	84% precision, 87% recall, 85% F1 score
	[151]	CNN	Convolution, max pool, dropout	Log data collected from Openstack testbed (600 data points)	Accuracy 95%, F1 score was 67% at best
Mid, long term	[150]	Time-Based Phrases extraction	-	Cray: Job logs from 3 systems over 8-18 months (total of 13k nodes) (2017)	$>$ 83% recall and $>$ 98% precision
	[146]	LSTM, MING, Random Forest	Set using random search for each scenario	Monitoring data from Ali-Baba DC over 6 months	Area Under the Curve 0.9

the error. In contrast, failure prediction output is a discrete value (i.e., whether or not there would be a failure during the targeted look-ahead window). In that case, it is important to reduce false negatives (measured using recall) where an upcoming failure is not detected, but also false positives (measured using accuracy/precision) that lead to unnecessary migrations. Thus, to have an accurate performance evaluation of failure prediction, it is important to measure precision together with recall or calculate the F1 score that uses both.

Another difference is the availability of training datasets. Indeed, it is difficult to obtain balanced datasets because failures are fairly rare due to the high availability requirements in computer systems. Furthermore, as pointed out by [142], there is a lack of best practices for logging failures and creating datasets.

Two types of datasets were used: performance metric traces [143], [144], [145], [146], [147], and log messages [148], [149], [150], [151].

In the first category, multiple performance metrics such as CPU and memory usage, disk-level sensor data, or system-level signals are examined to detect performance degradation or other early signs of failure to determine the nodes that are most likely to fail in the future. Li et al. [146] use monitoring data from Ali-Baba data-centers to predict failures in the short, mid, and long term. The proposed solution includes an AIOps pipeline which comprises a pre-processing phase of training data where feature engineering extracts relevant features, a ML model training phase that also includes hyper-parameter tuning, and an evaluation phase. Lin et al. [147] perform training in two phases: in the first phase, a Bi-LSTM model is trained using temporal data, and a second Random Forest model is trained using spatial data to detect node dependencies. In the second phase, a ranking model is trained using the intermediate results from the previous models to produce a ranking of nodes that are the most likely to experience failures. The obtained results surpass those of a vanilla LSTM. Haghshenas et al. [143] use a Multi-Agent ML-Based approach by training Q-learning on each server. The learned strategy aims to minimize faults but also energy consumption through consolidation.

Fewer works used log messages, which require text processing to analyze application logs and detect patterns leading to failure events. Das et al. [150] leverage the Time-Based Phrases extraction mechanism to analyze noisy system log messages and identify successive and correlated time-based events that indicate a likely node failure in supercomputers, and thus predict node shutdowns ahead of time and improve resilience. The authors also observe and discuss the effects of different types of node failures on system performance. Nam et al. [151] use a CNN model to predict future VM failures by analyzing log messages and detecting early fault messages after a pre-processing phase using log-based word embedding and pre-failure tagging techniques. The model then outputs a probability of failure after a certain time for each VM. Although accuracy was satisfactory for the CNN model, the F1 score was at 67% at best, which means that many failure events were not correctly predicted due to false negatives. In contrast, time-based phrase extraction and LSTM-based models showcased high values for both accuracy and recall, even for longer term prediction. However, it is not possible to derive conclusions due to the small number of contributions and the difference in datasets used as input.

D. Reactive Migration Trigger

In contrast with the previous subsections that mentioned prediction contributions for proactive migration, a smaller set of works tackled reactive migration by selecting the appropriate timing to trigger the process. Almost all of them employed RL methods to learn the appropriate strategy using data generated randomly [152], from experiments [71], [153], or using different mathematical models [154], [155], [156].

Elsaid et al. [71], [153] use Regression to predict live migration time, power consumption, and the network overhead of migrating a VM to a certain node depending on the VM size and workload. These predicted metrics are then used to determine the appropriate time to trigger the migration process for one or multiple VMs in parallel, which reduced migration time by up to 50%.

In [157], the authors propose a migration decision algorithm that determines whether to trigger the migration or not for each time-step in a mobility in MEC scenario, which allows a reward improvement of up to 75% compared to a static decision policy. In addition, the authors also study the impact of migration costs and resource demands on the decision strategy. Duggan et al. [154] investigate the network resource aspect of migration, and propose a solution to schedule VM migrations during peak loads. In this context, a Q-Learning agent is trained to determine the migration timing (migrate or delay) for a group of VMs from over-utilized hosts depending on bandwidth availability to transfer migration data.

Zangiabady et al. [155] propose a self-adaptive service migration mechanism where the decision is made to either trigger service migration or not with the goal of minimizing the number of migrations while increasing the number of accepted service requests.

Peng et al. [158] propose a DRL-based decision strategy learning method for service migration in MEC-enabled vehicular networks. The model is trained using DQN to decide whether a service should be migrated following the user's mobility, while taking into account the cost and QoS requirements. The obtained decision strategy allows further QoS improvements compared to the VI solution in [61]. Furthermore, the work studies the effect of vehicle's velocity and the memory dirty rate on the service's performance and the observed QoS metrics. Likewise, Park et al. [156] formalize service migration in MEC by considering multiple metrics such as migration cost, transaction cost, or energy consumption. In the proposed solution, a DQN model that includes convolutional layers (CNN) is trained using different layer numbers (1-3) to determine whether to migrate a service or not, depending on the distance between a user and the MEC server where the service is currently deployed. Results showed that little reward improvement was obtained (up to 3%) with no impact from the number of layers. Lan et al. [152] study reactive and proactive migration schemes with the goal of minimizing latency and resource consumption by training a DRL model using DDPG to select either a proactive or reactive migration decision and its resource allocation. The model's performance is evaluated using multiple learning rate values.

Nonetheless, reactive migration timing takes away the advantage given by prediction, which allows scheduling and starting the migration process in advance to reduce the impact on services.

E. Discussion

This section described the contributions that employed ML to predict in advance the events that trigger migration, as well as reactive approaches. The mobility prediction literature mainly focused on predicting future user positions, trajectories, or distributions to proactively migrate services in a MEC context. In workload forecasting, the surveyed works predicted the future request load or VM or host resource usage (mainly focusing on CPU) as the metrics that reflect the future workload status. Those predicted values are used to detect over-loaded or under-loaded nodes and perform proactive migration with the objective of performing dynamic resource provisioning and consolidation. In failure prediction, the studied contributions aimed to improve service availability by predicting future node failures using log messages or performance metrics. It could be seen that there are relatively few contributions that considered failure prediction. However, service failure prevention is crucial to ensure high reliability for mission-critical applications such as V2X or Industry 4.0 services, as service failures have important repercussions. Failure prediction is especially important since failure-related service migration is mainly performed proactively. Indeed, the source node might not be operational/reachable once the hardware failure has occurred. Additionally, it was noted that balanced failure log datasets are lacking, which makes failure prediction more difficult. Thus, more work is needed in that specific direction.

Moreover, for all prediction targets, most contributions relied on RNNs, and LSTM in particular, followed by different Regression methods and NNs with satisfactory results. It could be seen that Regression methods exhibited similar if not improved performances compared to RNN when predicting future short term loads, with reduced complexity. However, Regression performance dropped when moving to longer look-ahead windows, where LSTM showed robust results.

Furthermore, it was shown that deeper Neural Networks are not always more effective for short term load prediction, as models with only one hidden layer were able to achieve very high precision values. In contrast, for longer term prediction, deeper architectures were used. Nonetheless, the optimal set of hyper-parameter is different for each problem and dataset, and multiple combinations should be tested during training.

More recent models such as the attention-based Transformer [160] might also be worth exploring. Indeed, although it has been mainly used for Natural Language Processing (NLP) with popular implementations such as the BERT [161] and GPT-3 [162] models, recent applications to time-series forecasting have shown promising results [163], [164]. This method would be particularly

TABLE VII
SUMMARY OF REACTIVE MIGRATION TRIGGER REFERENCES (SECTION VII-D)

		Ref.	Input Data	Method	NN Architecture	Objective	Main findings
Load	ML	[153], [71]	From experiment	Regression	-	Time, network congestion	Timing optimization reduced migration time by up to 50%
	RL	[154]	Generated, triangular wave	Q-Learning	-	Bandwidth usage	Migration time reduced by 27%
		[155]	Generated, Poisson distribution	Q-Learning	-	Cost, request acceptance	Profit increase of up to 40%
Mobility	RL	[157]	Generated	Q-Learning	-	Cost, QoS	Total reward improvement of up to 75% compared to static policy
		[159], [61]	Generated	VI	-	Cost, QoS	Service disruption time 50-500ms depending on distance to target
	DRL	[158]	Not specified	DQN	Not specified	Cost, QoS	Improved QoS compared to [61]
		[156]	Random walk	DQN, CNN	1,2,3 layers	Cost, energy	Layer number had no impact, reward improvement of up to 3%
		[152]	Random	DDPG	One layer	Latency, resource usage, cost	Cost reduced by $>$ 50%

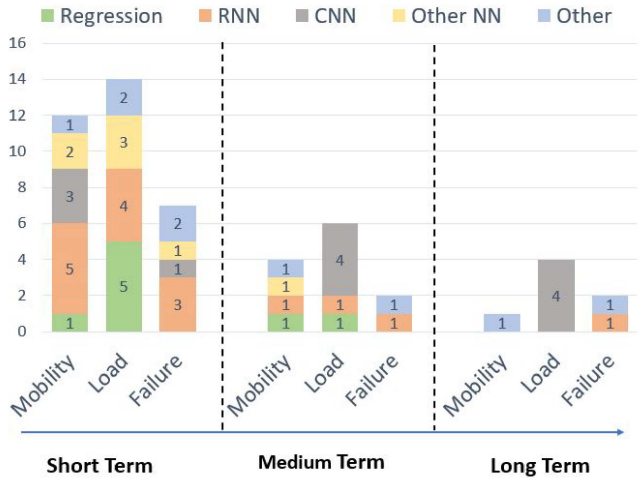


Fig. 8. Classification of the surveyed prediction works in terms of time window and used methods.

interesting for failure prediction if applied to analyzing textual log messages.

In terms of input data, it could be seen that many real-world datasets and platforms are made available for workload metrics, while failure logs and recent pedestrian and vehicular mobility traces are harder to find. Another key takeaway is that in many cases of mobility and load prediction, correlating input data with additional information such as application context, VNF dependencies, or user behaviour improved prediction results, even when using smaller input datasets. This observation concurs with the recent trend of Data-Centric AI, which advocates using smaller sets of meaningful, higher-quality data points for training AI models instead of larger ones [165], [166] for improved efficiency.

Finally, as illustrated in Figure 8, the majority of contributions for all three prediction types targeted short term time windows, which is less challenging than longer term predictions. However, the latter are important to allow migration scheduling in advance for a minimal impact on service quality. Therefore, future works should explore longer prediction time windows.

VI. PLACEMENT

Once the need for migration has been established, the next step is to select the target host for the migration process that optimizes the performance metrics. In the following, the contributions related to placement are classified into multiple categories depending on context: Single instance placement contributions are described in Section VI-A, SFC and Network Slicing works are discussed in Section VI-B, and solutions where both placement and timing are selected can be found in Section VI-C. For each category, the surveyed papers are further classified according to the migration trigger (load or mobility), and per ML method. Finally, Section VI-D provides an analysis and discussion of the featured contributions.

A. Single Instance Placement

Table VIII shows contributions that performed migration target selection for single service instances. Please note that for the sake of brevity, unless specified otherwise, the main findings for each paper are expressed in terms of improvements in comparison with static policies. It can be noticed that many contributions compared their proposed solutions with least-performing static policies such as always migrate, migrate to a random target, or no migration. However, this comparison is not sufficient for accurate performance assessment. In fact, it is preferred to use optimal decision outputs (provided by exact algorithms or ILPs) as the upper bound benchmark to determine the optimality gap for the proposed solutions.

Mobility-triggered migration in a MEC context was studied the most, with the objective of improving performance-related metrics (delay, service continuity, overhead, migration time) and reducing costs. A smaller set of contributions also considered minimizing energy consumption [173], [182], [183], [186]. In terms of input data, similar to the observation made on mobility prediction, the most recent traces used in the surveyed papers were collected in 2014.

Precursor works have introduced using RL for service placement in migration by formulating the problem as a MDP, and finding the optimal decision strategy

TABLE VIII
SUMMARY OF SINGLE-INSTANCE PLACEMENT REFERENCES (SECTION VI-A)

		Ref.	Input data	Method	NN Architecture	Objective	Main findings
Mobility	ML	[167]	Generated	E-ConvNets, E-ANN, CNN, ANN	1 to 6 layers	Delay, overhead	E-ConvNets had the best performance, for 3 and 4 layers, optimality gap of less than 10% (compared to ILP)
	RL	[168]	San Francisco taxi (2009)	Modified PI	-	Delay, Cost	Average migration cost reduced by 85%
		[169]	Generated, Shortest Path Map-Based Movement Model	PI	-	Long-term costs, QoS	Average user-perceived latency reduced by 15%
		[170]	Generated	PI, VI, in-place VI	-	Cost	Up to 5% reward increase compared to [171]
		[172]	Shanghai taxi (2007)	Q-Learning	-	Delay, cost	Up to 60% less delay
		[173]	Simulated	Q-Learning	-	Cost, energy, processing time	Revenue increase of up to 40%
	DRL	[174] [175]	Smartphone mobility (2014)	DQN	Not specified	Service continuity, cost	Percentage of successful migrations up to 20% more than multi-attribute algorithm.
		[176]	Random	DQN	Not specified	Cost	Average cost reduction by 3% compared to greedy
		[177]	Generated	DQN	3 layers of 15 neurons	QoS	Received data increased by 6%
		[178]	Generated	DQN	Not specified	Delay	45% less delay than no relocation, 20% less than static policy
		[179]	Random	A3C	Not specified	Long-term costs, QoS	Long term benefits increased by 70%
		[180]	Shenzen taxi (2014)	PPO, LSTM	Not specified	Cost	Cost 20% higher than optimal solution
		[181]	Shanghai mobile user data (2014)	Dueling DQN	2 layers of 3 and 10 neurons	Delay, cost-	Average cost 20-40% less than DQN
		[182]	Geolife: Beijing user data with different applications (2007-2012)	DRQN	LSTM layer instead of fully connected	QoS, delay, cost, energy, failure	Delay reduction of over 50% compared to static, 20 to 40 % compared to DQN depending on app, migration cost divided by 2-3x
		[183]	Generated	DQN, Double DQN	2 layers, size is 2x input size +1	Delay, cost, energy	Average delay reduced by 15%, DDQN outperforms DQN
		[184]	Linear movement	DQN	3 layers of 20 neurons	Generic	Performance close to greedy
	MADRL	[185]	Random	Double DQN, 3 MEC agents, one centralized	2 layers of 256 neurons	Delays, cost	Average delay close to optimal solution with ILP
		[186]	Shanghai taxi (2007)	DDPG, one agent per MEC Server	Actor: 2 layers of 512 and 128 neurons, Critic: 2 layers of 512 and 256 neurons	Delay, energy	MADRL takes less time, reward close to centralized Actor Critic by over 90%
		[187]	Generated, MBPR model	IDQN, CNN One agent per Service Entity	3 convolutional layers and 3 fully connected	Delay, cost, migration time	Up to 40% less service delays than static policy, up to 30% less than classic DQN
Load	ML	[188]	Generated	Neural Network	9 dense layers, 8 batch normalization layers of decreasing size (330-100) neurons	Downtime	Delay close to optimal performance
	DRL	[120]	Backbone traffic (2018,2019)	DQN	One hidden layer of 20 neurons	Cost, resource overload	DQN with PER outperforms stepwise optimization and vanilla DQN
		[189]	Generated	Policy Gradient	2 layers of 1000 and 61 neurons	Performance, cost	Delay reduction of 2%

based on Policy-Iteration (PI) [169], [170], [190], [191] or Value-Iteration (VI) [61], [159], [170] to construct a decision strategy that selects the best migration target for each state. However, not all RL problems can be solved through iteration. Indeed, these methods require the state space of the problem to be finite, which is not always the case, and it is not always possible to know the exact transition probabilities between states. Thus, other works have used Q-Learning [172], [173], which is a model-free RL method that does not require knowledge of state transition probabilities. Nevertheless, Q-Learning is a tabular method that relies on state-space exploration until convergence, which requires multiple state visits and action explorations for each state for the Q-table values to converge. Therefore, it is not scalable as it cannot handle larger state and action spaces. To cope with large state and action spaces, researchers resorted to DL techniques by training Neural Networks to select policies and approximate future rewards with algorithms such as DQN or Actor-Critic methods.

DQN was used by multiple contributions in the context of mobility in MEC [174], [175], [176], [177], [181], [182] sometimes using enhancements that accelerate convergence and improve efficiency such as Prioritized Experience Replay (PER) [192], or dueling DQN [120], [181]. Chen et al. [182] use Deep Recursive Q-Networks (DRQN) models that incorporate LSTM layers into the vanilla DQN to better process

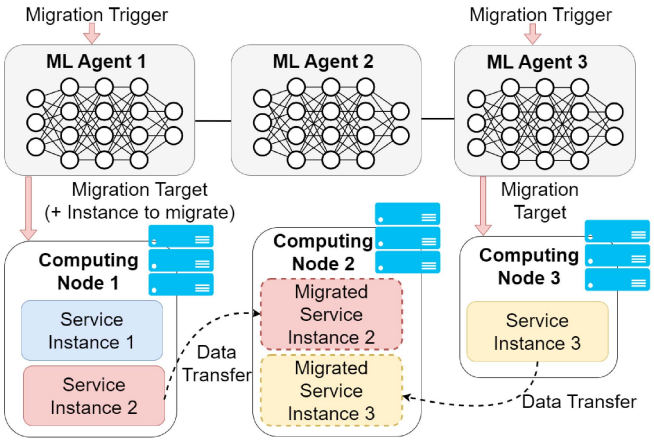


Fig. 9. Multi-agent DRL architecture for service migration.

historical data of user position to select the migration target of task offloading based on predicted user mobility in MEC. Additionally, multiple types of applications are considered (AR, infotainment, health, and video surveillance). DRQN coupled with PER and application context awareness resulted in a delay minimization of 20 to 40% compared to vanilla DQN, and migration costs were divided by a factor of 2 to 3.

Actor-Critic methods such as DDPG, PPO, A2C, or A3C train an actor-network to select the action for a given state and a critic network to approximate the value functions. Ren et al. [180] train a DRL model using Proximal Policy Optimization (PPO) [193] while using LSTM for the policy network to extract time-series features. The model is initially trained offline using service log data from a simulated environment, then used in an online environment without updating its parameters to avoid performance degradation. The observed experiences are recorded and used to train new offline models that would periodically replace the current model. To enhance performance, the authors propose a different definition of input state to reduce its dimensionality and decompose the action space by making the agent select the target node for the migration of one VM at a time. Compared to an optimal solution from ILP, the proposed PPO and LSTM combination achieves a migration cost that exceeds the optimal value by 20%.

To reduce complexity and training delays, Multi-Agent DRL (MADRL) solutions were proposed by deploying decision agents per MEC server [185], [186], or per Service Entity [187] to distribute decision-making without loss of performance. Figure 9 illustrates a Multi-Agent scenario where each computing node hosts its own ML agent. For all of the service instances that are hosted by the node, the ML agent makes decisions to select the service instance to be migrated, and determine its migration target.

On the other hand, there are comparatively very few contributions that used ML-based solutions besides RL and DRL. Emu and Choudhury [167] use Ensemble DL algorithms, namely E-ConvNets and E-ANN with the objective of minimizing delay and migration overhead. A number of models are simultaneously trained using different sets of data from the training set, and their outputs are later accumulated into a common layer. Evaluation results show that E-ConvNets had the best performance, and achieved a 10% gap with the optimal solution from ILP.

Unfortunately, many contributions did not specify the NN architecture details used by their proposed solutions, which makes evaluation and reproducibility more difficult. From available information, vanilla DQN-based solutions used smaller architecture with 1-3 layers of 3 to a maximum of 20 neurons, but the performance did not significantly surpass greedy solutions. Double DQN architectures increased layer size, with average delay results approaching optimal for a multi-agent setting. Emu and Choudhury [167] test different depth values (1-6) for Ensemble CNNs, and observe that 3 and 4 layer-architectures achieve the best performance, which is close to optimal. The Neural Network proposed in [188] also achieves close to optimal delays for load-triggered migration with a deeper architecture of 9 dense layers, followed by batch normalization layers of decreasing size (330 to 100 neurons).

B. SFC and Network Slicing

As detailed in Table IX, multiple works have also explored service migration in a SFC [190], [194], [195], [198], [199], [200], [201], [202], [203] or Network Slicing [196], [197] context. As stated earlier, those scenarios add another layer of complexity to the migration process with the additional consideration for the correlations and links between VNFs. Furthermore, in the SFC context, another decision is to select which VNFs should be migrated and which ones should remain on their current hosts. The proposed solutions employed various ML methods, ranging from classic PI [190] or Q-Learning [198], to DL with RNNs [194], or DRL methods such as DQN [196], [197], [199], [200] and Actor-Critic methods [195], [197], [202], [203]. Actor-Critic methods present the advantage of converging faster than DQN. As demonstrated in [195], a DDPG agent converges 6 to 10 times faster than DQN, with a 10% cost improvement.

Different SFC lengths are considered. For their simulations, the authors of [190] and [199] use 3 types of SFC, with a maximum length of 3 VNFs. Using PI, the solution proposed by Eramo et al. [190] achieves a total cost close to ILP, while keeping the execution time in a range of seconds as opposed to the ILP that can reach multiple hours for longer episodes. Chen et al. [199] use Multi-Agent DQN to attain rewards that are 85% close to ILP.

When the migration process is triggered due to insufficient resources, the new instance at the target node requires a scale-up. In this vein, Yao and Chen [198] propose a model for selecting migration targets and scaling for migrated VNFs, and evaluating SFCs with a length of 2 to 6 VNFs. Results showcase an important performance degradation for SFC lengths of 3 and over using Q-Learning, which suggests that Q-Learning might not scale well to longer SFC instances. In comparison, in [194], SFCs comprising 2 to 5 VNFs are migrated using Attention-based GRU, with a consistent performance achieving a reward that is only 10% below optimal.

TABLE IX
SUMMARY OF SFC/NS PLACEMENT REFERENCES (SECTION VI-B)

		Ref.	Load input	Method	NN Architecture	Objective	Output	SFC/NS context	Main findings
Load	ML	[194]	Generated	Attention-based GRU	Not specified	Migration frequency	VNF + target	SFC length of 2-5 VNFs	Reward 10% below optimal with attention
	RL	[190]	Generated, Cycle-stationary model	PI	-	Energy, loss of revenue	Target	3 SFC types, max length of 3 VNFs	Total cost close to ILP, execution time 1-5s while ILP is 22-5457s
	DRL	[195]	UK ISP backbone (On-demand)	DDPG	Mix of fully connected and normalization	Cost	VNF	SFC length not specified	Cost reduced by 30%, 10% compared to DQN, converges 10 times faster
		[196]	Generated	Dueling DQN	2 layers of 128 neurons	Resource cost, service interruption	New config.	Network slicing: 5 nodes per slice	Long term revenue increased by 50% compared to greedy
		[197]	Random	DQN, A2C	2 layers of 64 neurons for DQN, 64 and 256 neurons for A2C	Operation time, resource usage	Target	Network slicing	A2C performance remains high (under 2% failure), DQN declines for more apps
		[198]	Generated, Poisson process	Q-Learning	-	Delay, energy	Target + scaling	SFC length of 2-6 VNFs	Total delay 20% lower than random selection
	MADRL	[199]	Summer olympics streaming system (2008)	DQN, one agent per node	Not specified	Long-term profit	VNF + target	3 SFC types, max length of 3 VNFs	Reward 85% close to ILP better than single agent DQN
		[200]	Real Uunet traffic (On-demand)	DQN, one agent per node	Not specified	Energy, overhead, network cost	VNF + target	SFC length of 3 VNFs	Cost reduced by 10%
Mobility	RL	[201]	San Francisco taxi (2008)	Dyna-Q	-	Resource usage	Target	SFC length not specified	Average user latency reduced by up to 60%
	DRL	[202]	Generated	A3C	Not specified	Cost of Re-configuration	Target + Traffic routing	SFC length not specified	Reduced average orchestration delay compared to DQN
		[203]	Generated	A2C	Not specified	Delay, cost	VNF + target	SFC length not specified	Average delay reduction of 20-80% compared to static, particularly for high speed movements

Architecture-wise, available information is not sufficient to compare references and results.

In contrast with the previous sub-section, most SFC and NS placement references tackle load-triggered migration. Interestingly, despite the wide availability of real-world load datasets (as discussed in Section V-B), most contributions used generated, sometimes random load data.

C. Joint Placement and Trigger Timing

This section discusses the contributions that incorporated the time dimension to the placement process of migration by learning policies for jointly selecting when and where the service should be triggered. A majority of references relied on Q-Learning and DQN to generate decision policies.

This approach presents the advantage of performing both decisions in one step, which has the benefit of reducing computation time. Furthermore, the availability of viable migration targets can impact the decision to trigger migration.

Yang et al. [204] use a correlation coefficient k to establish relationships between the user's historical position data of the last k time-slots and improve decisions. The problem is divided into two sub-tasks: Selecting the migration target and deciding when to trigger the process. Wang et al. [205] use three structurally identical DQNs in parallel to enable continuous model improvement, where one network makes decisions on whether to migrate services to a certain MEC server at the current time slot in real-time. The other two use the experiences and reward for policy evaluation and refinement offline, and periodically update the decision network with new weights. This method allows simultaneous decision-making and training, which results in migration policies that adapt to changes in trajectory or velocity.

Liu et al. [206] propose a distributed framework for joint task migration in MEC. Multiple DRL agents are trained using Counterfactual Multi-Agent (COMA) Policy Gradient [207], where independent actor-networks are based on GRUs, and a shared critic network is based on classic feed-forward NNs. Consequently, completion time is reduced by 10 to 25% compared to a single Actor-Critic agent.

Wahab et al. [208] design an application-aware approach for the placement and adjustment of VNFs at each time slot depending on the environmental changes such as user mobility or service demand peaks. First, to reduce placement complexity in large-scale systems, the authors propose a clustering technique based on K-medoids that partitions the network into multiple clusters based on selected performance metrics (energy efficiency, high bandwidth capacity, CPU$\cdots$). Then, if a SLA violation is expected, the migration target is selected from the cluster that best satisfies the SLA of the application being migrated. This approach achieves an average cost that is only 10% higher than the optimal value from ILP, while improving computation time by up to a 100 folds.

Once again, information on the Neural Network architecture is generally missing, and most contributions use different input datasets, with results evaluations against static decision policies. Thus, no comparison can be made.

D. Discussion

The contributions described in this section provided the means to select the migration target of one or multiple VNFs

TABLE X
SUMMARY OF JOINT TRIGGER AND PLACEMENT REFERENCES (SECTION VI-C)

		Ref.	Input data	Method	NN Architecture	Objective	Main findings
Load	RL	[191]	Planetlab (on-demand)	PI	-	Energy, SLA violation, cost	14% cost reduction compared to greedy
		[171]	Shenzen taxi (2014)	Q-Learning	-	Cost	Average cost ratio doubled compared to random policy
	DRL	[209]	San Francisco taxi trace (2008)	DQN	4 layers	Delay, energy, cost	30-70% cost, delay reduction compared to threshold and Q-Learning
Mobility	RL	[210], [211]	San Francisco taxi (2008)	Modified PI	-	Cost	Average cost reduced by up to 50%
		[212]	From experiment	Q-Learning	-	Cost, QoE	30% delay reduction
		[213]	Shenzen taxi (2014)	Q-Learning	-	Cost, efficiency	Cost 20-30% over optimal
	DRL	[214]	Random walk	DQN	Not specified	Cost, QoE	20-30% reward increase compared to dynamic programming
		[204]	Shanghai taxi, year not specified	DQN	Not specified	Cost	Cost reduced by 50%
		[205]	Generated	DQN	3 layers: 64,256,256 neurons	Distance, migration#	Over 60% delay reduction
	MADRL	[206]	Random walk	Policy gradient, GRU, NN	One layer of 128 GRUs	Process duration	Completion time reduction of 10-25% compared to Actor Critic
Both	ML	[208]	Generated	K-medoids	-	SLA violations	Average cost 10% higher than ILP, while up to 100x faster

using ML methods while optimizing a variety of objectives such as cost, energy, migration time, or QoE metrics. Additionally, some works considered the time dimension by selecting the time slot where migration should be triggered depending on mobility or load data.

Another observation that can be made is that most contributions have focused on the mobility-related use cases in MEC, such as Vehicular Networks, which confirms the growing interest in Edge Computing technologies in the research community. Further, motivated by the trends of using microservices and Network Slicing, multiple works are considering more complex use cases where multiple dependent VNF instances are migrated simultaneously. In that case, RL methods such as Q-Learning might not be scalable enough, and DL methods should be favored. In order to reduce the additional complexity from these use cases, Multi-Agent solutions were also proposed where the migration decision for each instance is managed by an independent agent, which ultimately reduced training and inference time, while maintaining performance levels.

Regarding ML methods, and apart from a few exceptions, it can be noticed that an overwhelming majority of contributions for placement have used RL and DRL for learning the optimal decision policy, with the Q-learning and DQN algorithms being the most popular. In the meantime, several better-performing DRL algorithms have been gaining attention. In particular, Actor-Critic algorithms such as DDPG and A2C showcased increased performance and shortened convergence time compared to DQN. Proximal Policy Optimization (PPO) would also be worth exploring further, which can be easier to tune with a robust performance, and is currently used as the default RL algorithm for training OpenAI models [215]. A comparative analysis might then be beneficial to perform an assessment and determine whether these algorithms are best suited for this problem. Besides, few contributions leveraged enhancements such as PER or Dueling DQN [216] that have proven their efficiency in improving the model's accuracy and reducing convergence time.

Apart from SFC-related references, very few contributions considered application types and context when selecting the migration target. However, different application types may have distinct requirements, which might have an effect on the optimal target to select. In fact, contributions that considered multiple application types and requirements achieved close to optimal results.

It could also be observed that many references did not disclose details on the Neural Network architecture being implemented by their respective solutions. For future works, authors are advised to provide complete information on the evaluation setup, so that experiments can be reproduced, evaluated, and built upon.

Finally, future works should make use of real-world load traces to train placement models instead of synthetic data, which might not be realistic, particularly considering that open platforms such as the M-Lab project [137] provide access to real-time monitoring traces from distributed network infrastructures.

VII. MIGRATION SCHEME OPTIMIZATION

Once the migration target has been selected, remains the decision to select the migration strategy between the pre-copy, post-copy, and hybrid-copy methods. In addition, the migration process can be improved using some enhancements that have been proposed in the literature, such as CPU throttling [217], data compression [218], or delta compression [219]. Figure 10

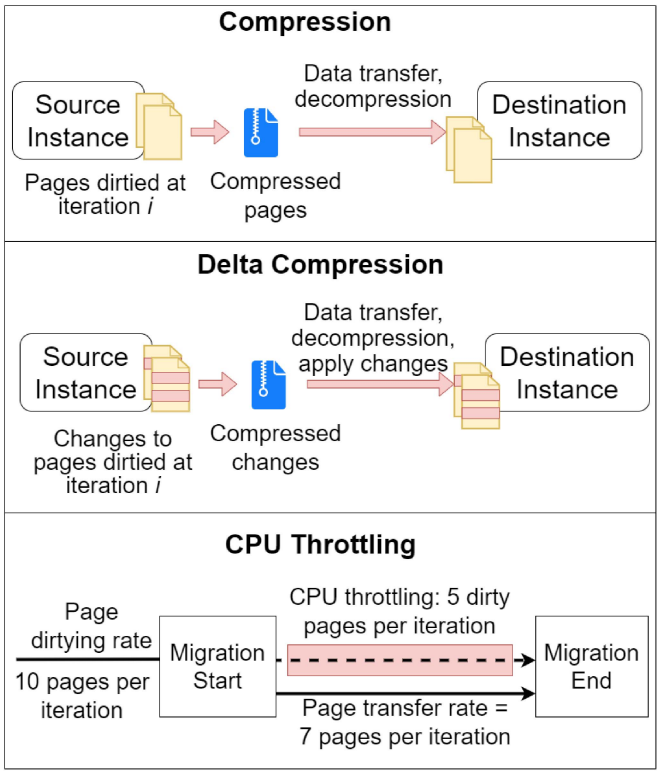


Fig. 10. Migration scheme enhancements.

depicts the aforementioned live migration enhancements; compression and delta compression are used to reduce the size of transferred data for dirtied memory pages, while CPU throttling consists in reducing CPU time for the instance to be migrated, which results in reducing its activity, and minimizes the number of dirtied pages at each iteration. This ultimately helps the migration process to converge but can negatively impact service performance. The pre-copy and hybrid methods, in particular, can be optimized by selecting the number of iterations before stopping the original instance, and scheduling the order of memory page transmission. Finally, the transmission process can also be optimized by selecting the migration path and the allocated bandwidth. The works described in the following have used ML techniques to help make those decisions and are summarized in Table XI.

A. Strategy Selection

In the following are detailed contributions that tackled migration strategy selection by predicting the effect of choosing one of the available methods on multiple metrics, which guides decision making. Jo et al. [220] compare multiple ML techniques (Linear Regression with SVR, with and without bagging) for predicting the effect of selecting different live migration techniques and potential enhancements on multiple key metrics such as performance degradation, downtime, migration time, and resource utilization. The predicted values are fed into an algorithm that chooses the best option considering the service's SLA constraints. Ultimately, SVR with bagging provided the best performance, but multiplied learning and prediction times compared to other methods. This work has been used as a reference for multiple subsequent contributions that reused the same methods and/or evaluation metrics.

Indeed, Motaki et al. [221] use SVR and K-Nearest Neighbour Regression to predict the effect of using pre-copy, post-copy, and the enhancements explored in [220] on additional target metrics such as SLA violation time and migration overhead. Similarly, Cho et al. [222] also use SVR to predict the effect of those methods on migration cost in terms of VM performance degradation, its duration, and resource overhead. Based on those predicted metrics, a policy is proposed to select the best live migration technique that reduces SLA violations. Different from previous papers that used SVR, Altahat et al. [224] employ DNNs for prediction. The DNN model is trained to output the expected migration time, downtime, the total load of transferred data, and performance degradation for VMs when the pre-copy or post-copy methods are used to perform migration. Those key metrics are predicted under varying parameters such as application type, VM size, page dirtying rate, network bandwidth, and resource utilization. This proposal outperforms multiple regression techniques used in [220] in terms of accuracy for multiple metrics and significantly lowers prediction time. However, training time remains very high compared to Linear Regression and SVR.

Some works have also applied classic Linear Regression methods for migration metric prediction. The authors of [225] use Supervised Learning (LR, SVR, and NNs) in an SFC context to predict the resulting downtime and delay for rebuilding, cold, and live migration using pre-copy of SFCs, assuming that the target host has already been selected. The evaluation explores the effects of different parameters such as VM size, network traffic load, computing capacity, and page dirty rate on downtime and delay. The predicted values are used to build a policy for selecting migration schemes. de Jesus Martins et al. [223] implement a container platform and perform multiple migration experiments using the cold and pre-copy methods in a monitored environment to collect performance data. Afterwards, the LR and Multivariable Regression methods are used to identify patterns and predict migration cost, duration, and the required bandwidth depending on the image size, OS distribution, and the migration method (cold or live using pre-copy). However, those contributions only consider pre-copy as a live migration method and do not study improvement techniques compared to the previous works.

B. Strategy Improvement

The migration process can also be improved by reducing the number of re-transmitted memory pages, which in turn reduces transmission cost and migration time. This can be done by identifying patterns of page modification and selecting the pages that should be copied at each iteration accordingly. Wu et al. [227] forecast the probability for each memory page to be dirtied in the next round using the Markov prediction model. Combined with dirtying page rate from monitoring data, those predictions are used to select the memory pages that should be migrated in priority for each iteration. The

TABLE XI
SUMMARY OF MIGRATION SCHEME SELECTION AND IMPROVEMENT REFERENCES (SECTION IX)

	Ref.	Context	Decision/Prediction Output	Method	NN Architecture	Results
Selection Section VII-A	[220]	Migration strategy selection: pre-copy or post-copy, with enhancements (compression, delta compression, CPU throttling)	Effect on performance degradation, downtime, migration time, resource usage	LR, SVR, SVR with bagging	-	Error of 5-10% for SVR with bagging
	[221]		Effect on [220] metrics + SLA violation time, migration overhead	SVR, KNN	-	KNNR outperforms SVR, 20% error
	[222]		Effect on [220] metrics	SVR	-	Average error of 10%
	[223]	Performance data collection for cold and live migration	Migration cost, duration, required bandwidth	LR, Multivar. Regression	-	80-90% accuracy
	[224]	Migration strategy selection: pre-copy or post-copy	Expected migration time, downtime, data load, performance degradation	DNNs	2 layers:32 and 16 neurons	Lower error than [220]
	[225]	Determine whether to rebuild, or perform cold or live pre-copy migration	Downtime and delay of rebuilding, cold, live pre-copy	LR, SVR, NNs	One layer of sigmoid or ReLU	NN ReLU has lowest error
Improvement Section VII-B	[226]	Improved hybrid-copy method with Switched Decision Factor	Probability of page memory writes for pre-copy iterations	Markov model	-	Up to 75% page fault reduction
	[227]	Reducing unnecessary transmissions of dirtied pages in pre-copy	Probability for memory page to be dirtied in the next round	Markov model	-	Migration time in high dirty rate similar to low dirty rate
	[228]	Multi-phase pre-copy	Future dirty pages	Auto Regression	-	Downtime reduced by up to 20%
	[229]	Bandwidth allocation adjustment to minimize transmission delays and transmission costs	Bandwidth allocation per time-step	DDPG, DQN	- DQN: 2 layers of mean(input, output) neurons - DDPG: 400 and 300 neurons	Downtime reduced to under 1s

obtained strategy reduces the memory size migrated at the first round, which, in the long run, significantly reduces total migration time for applications with high dirty rates.

Besides the memory modification prediction, a few contributions proposed modified migration schemes. Shukla et al. [228] elaborate a multi-phase pre-copy based live migration approach that minimizes page transfer and migration time. During the first phase, the VM image and the memory pages are transferred to the destination host, then in the second phase, the least modified pages are transferred. In the third phase, an auto-regressive approach is used to forecast future dirty pages and leave the transfer process of the most frequently modified pages to the fourth phase, which corresponds to the stop-and-copy step of the pre-copy live migration method.

Targeting the faulted memory page transfer phase in the hybrid live migration scheme, Lei et al. [226] present an improved hybrid-copy method for live migration, where the number of page faults is reduced without increasing migration time by introducing a Switched Decision Factor (SDF). It is defined as the moment when the process switches from the pre-copy phase of the hybrid process to post-copy. In other words, it determines the number of iterations of the pre-copy phase before switching to post-copy. The authors use a Markov model to detect the memory access pattern of different workload types (write-intensive and read-intensive) and forecast the probability of memory writes on pages for each iteration of pre-copy. Then, a set of the least likely to be modified pages is transferred during the next iteration, which reduces the number of duplicate transmissions. At each iteration, the SDF value is computed to determine whether the process should stop the iteration phase. If applicable, the memory access pattern is also predicted to proactively transfer the required pages, thus reducing the number of faulty pages once the instance has been resumed.

Another way to optimize the migration process is to manage the amount of bandwidth allocated to the data transfer, especially in case of multiple concurrent migrations and/or if the network resources are scarce. Indeed, a dynamic allocation scheme can help prevent network congestion and adapt to network load fluctuations. This motivated authors in [229] to tackle network resource usage during the migration process of SFCs, where DQN and DDPG networks were used to allocate and adjust bandwidth for SFC migrations at each time step with the objective of minimizing transmission delays, propagation, processing and queuing delays while keeping bandwidth consumption at a minimum.

C. Discussion

Compared to the other stages of the migration process, fewer contributions have been made to migration scheme selection and improvement using ML. However, optimizing this process is crucial to reduce the impact on service continuity and user's QoE. The results from the discussed references showcased the benefits of selecting the appropriate method and ordering memory page copies. Furthermore, network resource usage should be explored more thoroughly, in particular in the MEC migration use case where network resources are limited, which requires a dynamic resource allocation and

traffic routing mechanism to account for changes in traffic load.

Algorithm wise, and different from previous sections, it can be seen that the majority of the works that optimized the data transfer process mainly used Supervised Learning, and particularly Regression methods or Neural Networks to predict the effect of the migration methods and improvements on selected metrics, or to predict memory page dirtying patterns. Although NN methods improved performance compared to Linear Regression, it should be noted that training time was also significantly higher. More advanced RNN methods might also be worth exploring to detect memory page dirtying patterns. In terms of input data, most references generated data from running scheduled migration experiments or fed data collected from the environment into the model. Notably, Motaki et al. [221] aggregate datasets from multiple workload types (online transactions, multimedia, e-commerce, data, and computationally intensive applications) to emulate realistic cloud workloads.

VIII. Challenges and Research Directions

This survey presented a comprehensive review of the contributions using ML to facilitate service migration and showcased the benefits of such methods. To conclude the paper, the following section outlines the remaining challenges and discusses future possibilities for research in the area of ML for service migration in future networks.

A. Training Data and Environment

To train a ML model correctly, a training data-set must be available with a sufficient volume of realistic and sometimes labeled data. However, real training data is not always available. For example, in the mobility prediction use case, it is difficult to obtain user position data due to the sensitivity of such information. To mitigate this issue, many contributions listed in this survey use different taxi mobility data sets or generate synthetic mobility trajectories. Still, those data sets do not always reflect realistic user mobility patterns. One emerging approach is Data-Centric AI, where smaller sets of meaningful, higher quality curated data points are used for training AI models [165], [166], which provides promising results and optimizes the training process by reducing the size of training data.

In the RL case, the model learns policies through direct interactions with the environment by exploring sets of actions and observing the effects on the long-term rewards to evaluate and improve the current policy. However, for service migration, experimenting random actions during the exploration phase can lead to important service disruptions, drops in the user's QoE and SLA violations, or migration failure. A few mitigation approaches were proposed in the literature to avoid the effects of bad exploration decisions, for a *safe RL* [230], but those methods slow down the learning process. Thus, the model should be trained in a simulated environment that is identical to the real one, to output realistic rewards.

A promising direction would be the use of Digital Twins, which are defined as the virtual instances that fully describe a physical system and its features [231], and are extensively used for simulations in Industry 4.0 use cases. Indeed, a Digital Twin can emulate the environment's feedback to the actions selected from the RL agent or generate realistic data distributions to train ML models. Another possible solution for RL is the use of ML sandboxes using network simulation tools as proposed in [232].

B. DRL Model Update Post-Training

One of the advertised advantages of using RL methods for policy selection is that RL models can adapt to changes in the environment by continuously updating their policy depending on rewards from the environment. However, it can be noticed from studying DRL references in this survey that in practice (besides a few exceptions),

Once the NN model converges to a satisfactory policy, training is stopped, and the model is switched to a full exploitation mode where no exploration is performed and the policy is not updated anymore. Although it could be argued that a deployed model should not be updated online to avoid loss of performance, this practice takes away one of the main advantages of using RL.

One solution would be to train another version of the model offline and update the main online model periodically or when the current policy becomes less efficient due to important changes in the environment.

Additionally, one emerging paradigm dubbed Continual Learning [233], [234], [235], [236], [237] or Lifelong Learning [238], [239] has been gaining attention. The aim is to allow continual online Neural Network training and update by learning from new tasks, without loss of previous knowledge to preserve performance on prior tasks. To this end, one proposal is to expand Neural Networks dynamically: old neuron parameters are fixed to avoid loss of knowledge, and new neurons are added to perform incremental learning.

C. Explainability and Model Tuning

Another issue hindering the widespread use of ML for network orchestration and management is the fact that ML models are seen as opaque black boxes, which raises concerns for reliability and accountability. Indeed, critical decision systems require solid trust and performance guarantees. This need for increased transparency in AI functioning has motivated research efforts in the field of Explainable AI (XAI) [240], [241], [242], [243] which is a field that aims to make ML models more transparent, interpretable, and trustworthy by providing explanations on why the model came to provide a certain output. This can, in turn, help detect and correct model design errors that might hinder its performance.

Indeed, the performance of ML models heavily depends on the chosen architecture and hyper-parameters, such as the number and size of layers, the activation function, or the selected algorithm. Thus, those hyper-parameters should be carefully chosen to improve efficiency and accuracy. Finding the right combination of hyper-parameter values is not a straightforward process, as each problem is different, and a set of hype-parameters that proved to be optimal for a specific

problem would not work for related problems. Many methods were proposed to automate the hyper-parameter optimization process [244], [245], [246]. However, most popular search methods leverage greedy, random, or genetic approaches to explore the solution space, which can prove to be resource and time-consuming. Explainability would help understand the effect of certain settings on the model's performance and guide hyper-parameter optimization in a more intuitive manner.

D. Edge Mobility

The main idea behind edge computing is to move computations closer to the end-user with the objective of reducing communication cost, latency, and network load. An emerging trend aims to go beyond current MEC implementations and offload computation down to the user terminals by virtualizing end device resources that are underutilized and made available to authorized third parties. Multiple proposals implemented that concept under different names such as Vehicular Edge Computing [247], Vehicular Cloud Computing [102], [248], [249], Vehicular Fog Computing [250], or Vehicular Micro Clouds [251] for the specific use case of Vehicular Networks, and the more general V-Edge [252] proposal that considers all types of computation resources.

However, this concept incurs possible mobility of the computing resources, which adds another dimension to be considered in the migration planning, where the computing nodes can also change locations. Thus, both the service provider and consumer are considered to be dynamic. In the service migration context, it would mean that the service user, migration source, and target can be mobile during the process, which leads to dynamic distance and data transmission costs and duration that can affect migration time, downtime, or even cause migration failure. Therefore, to support service migration in that context, future contributions ought to take into account mobility patterns, sojourn time, velocity, and battery life of the UE providing the service. Similar to the classic migration scenarios with static source and target nodes, ML can be beneficial in the Edge mobility use case during all phases of the migration process. Indeed, for migration target selection, the ML-based decision model can be trained using additional input such as the current and predicted mobility of the target candidates and availability times as well. It can also be used during the migration trigger phase by predicting the availability time of the service's current placement nodes to trigger service migration proactively when the current node is predicted to go offline in the near future. This use case also adds new prediction inputs, such as battery consumption or historical data of device availability (when the user makes its device available for external use) to detect device availability patterns.

E. Application-Aware Migration

Among the contributions surveyed in this paper, a few works have proposed a selection mechanism for the memory pages that should be migrated in the next copy iteration by detecting page modification patterns, and prioritizing pages that are least likely to be modified. However, those works do not consider the application specifics when selecting the page copy order. Indeed, the least modified pages might not be of the highest importance for running the service; thus, migrating them in priority might not be the best choice.

An interesting future direction would be to propose service-aware selection strategies for memory page copy. Future works might use ML methods to analyze the service runtime data and classify memory pages into categories: pages that are critical for service runtime, pages without which the service can run but with a degraded performance, and so on. Consequently, memory pages that are essential for service runtime are migrated in priority, and depending on the required performance, the instance can be resumed on the target node sooner, which would reduce downtime. Further, this analysis can help detect content that is not used by the service during its runtime, and avoid its transmission to preserve bandwidth.

F. User Context-Aware Migration

A step further for application-aware migration would be to also differentiate between individual user contexts, and use ML to predict the pages that each user is likely to use. This prediction, associated with the predicted probability of the service being requested by specific users in the next time slots, can be used to prioritize user context and memory page transmission. Once the core part of the application has been migrated, this context-based approach can be used to progressively migrate the content required for each user to the target node. Once the context and frequently requested pages for a user have been successfully transferred, the user's subsequent requests are re-directed to the target node. This approach would have the benefit of reducing request load on the source instance earlier as opposed to classic migration schemes that only re-direct user requests once the migration process has been completed, which can reduce service disruptions for load-triggered migration.

IX. Conclusion

Machine Learning is a promising tool for decision-making that paves the way to Zero-touch network automation. It allows an efficient service orchestration and Life-Cycle Management using prediction and decision policy creation and improvement. Thus, it has been used by several works in the literature to improve different stages of the service migration process and guarantee service availability and user QoE.

This survey paper investigated the applications of ML to optimize the service migration process and provided a background on migration, ML, and the relevant related technologies. Furthermore, the survey elaborated a comprehensive taxonomy of the current state-of-the-art solutions for the different stages of the migration process, with a discussion of the observed trends and suggestions for future works. Finally, the limitations of current proposals were discussed, and multiple research directions and open issues were outlined to motivate future contributions to the subject.

REFERENCES

[1] V. Fanibhare, N. I. Sarkar, and A. Al-Anbuky, "A survey of the tactile Internet: Design issues and challenges, applications, and future directions," *Electronics*, vol. 10, no. 17, p. 2171, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/17/2171

[2] E. C. Strinati et al., "6G: The next Frontier: From holographic messaging to artificial intelligence using subterahertz and visible light communication," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 42–50, Sep. 2019.

[3] A. Clemm, M. F. Zhani, and R. Boutaba, "Network management 2030: Operations and control of network 2030 services," *J. Netw. Syst. Manag.*, vol. 28, pp. 721–750, Oct. 2020.

[4] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," in *Proc. ACM ANRW*, Jul. 2018, p. 20.

[5] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[6] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[7] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[8] W. Bao et al., "Follow me fog: Toward seamless handover timing schemes in a fog computing environment," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 72–78, Nov. 2017.

[9] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, "Move with me: Scalably keeping virtual objects close to users on the move," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[10] C. Puliafito, E. Mingozzi, C. Vallati, F. Longo, and G. Merlino, "Companion fog computing: Supporting things mobility through container migration at the edge," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, 2018, pp. 97–105.

[11] T. Le, "A survey of live virtual machine migration techniques," *Comput. Sci. Rev.*, vol. 38, Nov. 2020, Art. no. 100304. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013720304044

[12] H. Hawilo, M. Jammal, and A. Shami, "Orchestrating network function virtualization platform: Migration or re-instantiation?" in *Proc. IEEE 6th Int. Conf. Cloud Netw. (CloudNet)*, 2017, pp. 1–6.

[13] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Netw.*, vol. 32, no. 2, pp. 92–99, Mar./Apr. 2018.

[14] R. Boutaba et al., "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, p. 16, Jun. 2018.

[15] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, "A survey on machine learning techniques for routing optimization in SDN," *IEEE Access*, vol. 9, pp. 104582–104611, 2021.

[16] S. B. Melhem, A. Kaushik, H. Tabassum, and U. T. Nguyen, *Machine Learning for Resource Allocation in Mobile Broadband Networks*. Hoboken, NJ, USA: Wiley, 2021, pp. 123–146.

[17] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: Bradford Book, 2018.

[19] C. Benzaid and T. Taleb, "AI-driven zero touch network and service management in 5G and beyond: Challenges and research directions," *IEEE Netw.*, vol. 34, no. 2, pp. 186–194, Mar./Apr. 2020.

[20] J. Gallego-Madrid, R. Sanchez-Iborra, P. M. Ruiz, and A. F. Skarmeta, "Machine learning-based zero-touch network and service management: A survey," *Digit. Commun. Netw.*, vol. 8, no. 2, pp. 105–123, Apr. 2022, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864821000614

[21] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206–1243, 2nd Quart., 2018.

[22] S. Ramanathan, K. Kondepu, M. Razo, M. Tacca, L. Valcarenghi, and A. Fumagalli, "Live migration of virtual machine and container based mobile core network components: A comprehensive study," *IEEE Access*, vol. 9, pp. 105082–105100, 2021.

[23] C. Puliafito, C. Vallati, E. Mingozzi, G. Merlino, F. Longo, and A. Puliafito, "Container migration in the fog: A performance evaluation," *Sensors*, vol. 19, no. 7, p. 1488, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1488

[24] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. S. Pilli, and D. Kapil, "A critical survey of live virtual machine migration techniques," *J. Cloud Comput.*, vol. 6, no. 1, p. 23, Nov. 2017. [Online]. Available: https://doi.org/10.1186/s13677-017-0092-1

[25] O. Oleghe, "Container placement and migration in edge computing: Concept and scheduling models," *IEEE Access*, vol. 9, pp. 68028–68043, 2021.

[26] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[27] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A survey on mobility-induced service migration in the fog, edge, and related computing paradigms," *ACM Comput. Surveys*, vol. 52, no. 5, pp. 1–33, Sep. 2019. [Online]. Available: https://doi.org/10.1145/3326540

[28] R. Zolfaghari, A. Sahafi, A. M. Rahmani, and R. Rezaei, "Application of virtual machine consolidation in cloud computing systems," *Sustain. Comput. Inf. Syst.*, vol. 30, Jun. 2021, Art. no. 100524. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537921000172

[29] A. H. T. Dias, L. H. A. Correia, and N. Malheiros, "A systematic literature review on virtual machine consolidation," *ACM Comput. Surveys*, vol. 54, no. 8, p. 178, Oct. 2021. [Online]. Available: https://doi.org/10.1145/3470972

[30] Z. Zhong, M. Xu, M. A. Rodriguez, C. Xu, and R. Buyya. "Machine learning-based orchestration of containers: A taxonomy and future directions." 2021. [Online]. Available: https://arxiv.org/abs/2106.12739

[31] T. Khan, W. Tian, and R. Buyya. "Machine learning (ML)-centric resource management in cloud computing: A review and future directions." 2021. [Online]. Available: https://arxiv.org/abs/2105.05079

[32] Z. M. Fadlullah et al., "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[33] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, "Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1578–1598, 3rd Quart., 2021.

[34] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[35] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2595–2621, 4th Quart., 2018.

[36] M. McClellan, C. Cervelló-Pastor, and S. Sallent, "Deep learning at the mobile edge: Opportunities for 5G networks," *Appl. Sci.*, vol. 10, no. 14, p. 4735, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/14/4735

[37] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[38] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 38–67, 1st Quart., 2020.

[39] M. Masdari and H. Khezri, "Efficient VM migrations using forecasting techniques in cloud computing: A comprehensive review," *Clust. Comput.*, vol. 23, no. 4, pp. 2629–2658, Dec. 2020. [Online]. Available: https://doi.org/10.1007/s10586-019-03032-x

[40] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function Virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[41] "Network functions virtualization (NFV); architectural framework v1.1.1," ETSI, Sophia Antipolis, France, Rep. ETSI GS NFV 002, Oct. 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[42] J. Watada, A. Roy, R. Kadikar, H. Pham, and B. Xu, "Emerging trends, techniques and open issues of Containerization: A review," *IEEE Access*, vol. 7, pp. 152443–152472, 2019.

[43] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[44] S. Newman, *Building Microservices*, 1st ed. London, U.K.: O'Reilly Media, 2015.

[45] J. M. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," IETF, RFC 7665, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt

[46] G. Sun, D. Liao, D. Zhao, Z. Xu, and H. Yu, "Live migration for multiple correlated virtual machines in cloud-based data Centers," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 279–291, Mar./Apr. 2018.

[47] D. A. Guillen et al. "Edge computing for 5G networks—White paper." Mar. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3698117

[48] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 596–630, 1st Quart., 2021.

[49] K. Ha et al., "You can teach elephants to dance: Agile VM handoff for edge computing," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, 2017, pp. 1–14. [Online]. Available: https://doi.org/10.1145/3132211.3134453

[50] "MEC in 5G networks," ETSI, Sophia Antipolis, France, Rep. 28, Jun. 2018. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[51] Q.-V. Pham et al., "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[52] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[53] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 3rd Quart., 2018.

[54] *Technical Specification Group Services and System Aspects; Architecture for Enabling Edge Applications*, 3GPP Standard TS 23.558, Dec. 2022.

[55] *Technical Specification Group Services and System Aspects; 5G System Enhancements for Edge Computing; Stage 2, Version 18.0.0*, 3GPP Standard TS 23.548, Dec. 2022.

[56] "Network functions Virtualisation (NFV); terminology for main concepts in NFV," ETSI, Sophia Antipolis, France, Rep. ETSI GS NFV 003, Jan. 2018. [Online]. Available: https://docbox.etsi.org/isg/nfv/open/Publications_pdf/Specs-Reports/NFV003v1.3.1-GR-Terminology_for_Main_Concepts_in_NFV.pdf

[57] A. Kaloxylos, A. Gavras, D. Camps Mur, M. Ghoraishi, and H. Hrasnica. "AI and ML—Enablers for beyond 5G networks." Dec. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4299895

[58] I. Ahmad and J. Huusko, *White Paper on Machine Learning in Wireless Communication Networks* (6G Research Visions), vol. 7, S. Ali, W. Saad, and D. Steinbach, Eds. Oulu, Finland: Univ. Oulu, Jun. 2020.

[59] *Technical Specification Group Services and System Aspects; Architecture Enhancements for 5G System (5GS) to Support Network Data Analytics Services, Version 18.0.0*, 3GPP TS 23.548, Dec. 2022.

[60] G. Panek, I. Fajjari, H. Tarasiuk, A. Bousselmi, and T. Toukabri, "Application relocation in an edge-enabled 5G system: Use cases, architecture, and challenges," *IEEE Commun. Mag.*, vol. 60, no. 8, pp. 28–34, Aug. 2022.

[61] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 369–382, Apr.–Jun. 2019.

[62] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in Industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[63] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of Things Realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961–2991, 4th Quart., 2018.

[64] S. Jain, M. P. Gilesh, S. D. M. Kumar, and L. Jacob, "On the necessity of right optimizations for live migration of virtual machines," in *Proc. IEEE Region Ten Symp. (Tensymp)*, 2018, pp. 124–129.

[65] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network slice mobility in next generation mobile systems: Challenges and potential solutions," *IEEE Netw.*, vol. 34, no. 1, pp. 84–93, Jan./Feb. 2020.

[66] M. Torquato, P. Maciel, and M. Vieira, "Analysis of VM migration scheduling as moving target defense against insider attacks," in *Proc. 36th Annu. ACM Symp. Appl. Comput. (SAC)*, New York, NY, USA, 2021, pp. 194–202. [Online]. Available: https://doi.org/10.1145/3412841.3441899

[67] M. Ala'Anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: A meta-study," *IEEE Access*, vol. 7, pp. 141868–141887, 2019.

[68] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2nd Quart., 2019.

[69] S. Nathan, U. Bellur, and P. Kulkarni, "Towards a comprehensive performance model of virtual machine live migration," in *Proc. 6th ACM Symp. Cloud Comput. (SoCC)*, 2015, pp. 288–301. [Online]. Available: https://doi.org/10.1145/2806777.2806838

[70] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proc. ATEC*, 2005, p. 25.

[71] M. Elsaid, H. Abbas, and C. Meinel, "Live migration timing optimization for VMware environments using machine learning techniques," in *Proc. 10th Int. Conf. Cloud Comput. Services Sci. (CLOSER)*, 2020, pp. 91–102.

[72] M. R. Hines, U. Deshpande, and K. Gopalan, "Post-copy live migration of virtual machines," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 3, pp. 14–26, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1618525.1618528

[73] L. Hu, J. Zhao, G. Xu, Y. Ding, and J. Chu, "HMDC: Live virtual machine migration based on hybrid memory copy and delta compression," *Appl. Math. Inf. Sci.*, vol. 7, pp. 639–646, Jun. 2013.

[74] P. Bellavista, A. Corradi, L. Foschini, and D. Scotece, "Differentiated service/data migration for edge services leveraging container characteristics," *IEEE Access*, vol. 7, pp. 139746–139758, 2019.

[75] H. Ko, M. Jo, and V. C. M. Leung, "Application-aware migration algorithm with prefetching in heterogeneous cloud environments," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2324–2333, Oct.–Dec. 2022.

[76] T. V. Doan, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, "FAST: Flexible and low-latency state transfer in mobile edge computing," *IEEE Access*, vol. 9, pp. 115315–115334, 2021.

[77] L. Liu, H. Xu, Z. Niu, P. Wang, and D. Han, "U-HAUL: Efficient state migration in NFV," in *Proc. 7th ACM SIGOPS Asia–Pac. Workshop Syst. (APSys)*, 2016, pp. 1–8. [Online]. Available: https://doi.org/10.1145/2967360.2967363

[78] J. Santa et al., "MIGRATE: Mobile device virtualisation through state transfer," *IEEE Access*, vol. 8, pp. 25848–25862, 2020.

[79] L. Nobach, I. Rimac, V. Hilt, and D. Hausheer, "Statelet-based efficient and seamless NFV state transfer," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 964–977, Dec. 2017.

[80] P. C. Sen, M. Hajra, and M. Ghosh, "Supervised classification algorithms in machine learning: A survey and review," in *Emerging Technology in Modelling and Graphics*, J. K. Mandal and D. Bhattacharya, Eds. Singapore: Springer, 2020, pp. 99–111. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-7403-6_11

[81] M. Usama et al., "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65579–65615, 2019.

[82] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 907–948, Feb. 2020. [Online]. Available: https://doi.org/10.1007/s10462-019-09682-y

[83] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, Feb. 2020. [Online]. Available: https://doi.org/10.1007/s10994-019-05855-6

[84] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[86] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[87] J. Gu et al., "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317304120

[88] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: https://doi.org/10.1038/nature14236
[89] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "*Q*-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019.
[90] V. Mnih et al. "Asynchronous methods for deep reinforcement learning." 2016. [Online]. Available: http://arxiv.org/abs/1602.01783
[91] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2019. [Online]. Available: https://arxiv.org/pdf/1509.02971.pdf
[92] E. F. Maleki, L. Mashayekhy, and S. M. Nabavinejad, "Mobility-aware computation offloading in edge computing using machine learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 328–340, Jan. 2023.
[93] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2980–2998, Jun. 2010.
[94] A. L. Ibrahimpašić, B. Han, and H. D. Schotten, "AI-empowered VNF migration as a cost-loss-effective solution for network resilience," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2021, pp. 1–6.
[95] C. M. Bishop, *Mixture Density Networks*, Microsoft, Redmond, WA, USA, 1994.
[96] A. Rago, G. Piro, G. Boggia, and P. Dini, "Anticipatory allocation of communication and computational resources at the edge using spatio-temporal dynamics of mobile users," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4548–4562, Dec. 2021.
[97] C.-L. Wu, T.-C. Chiu, C.-Y. Wang, and A.-C. Pang, "Mobility-aware deep reinforcement learning with glimpse mobility prediction in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.
[98] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, "AutoMEC: LSTM-based user mobility prediction for service management in distributed MEC resources," in *Proc. 23rd Int. ACM Conf. Model. Anal. Simulat. Wireless Mobile Syst. (MSWiM)*, 2020, pp. 155–159. [Online]. Available: https://doi.org/10.1145/3416010.3423246
[99] I. Labriji et al., "Mobility aware and dynamic migration of MEC services for the Internet of vehicles," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 570–584, Mar. 2021.
[100] A. Dalgkitsis, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Data driven service orchestration for vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4100–4109, Jul. 2021.
[101] Z. Zhao et al., "Mobility management with transferable reinforcement learning trajectory prediction," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2102–2116, Dec. 2020.
[102] A. M. Mustafa, O. M. Abubakr, O. Ahmadien, A. Ahmedin, and B. Mokhtar, "Mobility prediction for efficient resources management in vehicular cloud computing," in *Proc. 5th IEEE Int. Conf. Mobile Cloud Comput. Services Eng. (MobileCloud)*, 2017, pp. 53–59.
[103] N. P. Kuruvatti, S. B. Mallikarjun, S. C. Kusumapani, and H. D. Schotten, "Mobility awareness in cellular networks to support service continuity in vehicular users," in *Proc. 3rd Int. Conf. Inf. Commun. Technol. (ICOIACT)*, 2020, pp. 431–435.
[104] Q. Wu, X. Chen, Z. Zhou, and L. Chen, "Mobile social data learning for user-centric location prediction with application in mobile edge service migration," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7737–7747, Oct. 2019.
[105] H. Zhang and L. Dai, "Mobility prediction: A survey on state-of-the-art schemes and future applications," *IEEE Access*, vol. 7, pp. 802–822, 2019.
[106] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2392–2431, 4th Quart., 2017.
[107] R. Wu, G. Luo, J. Shao, L. Tian, and C. Peng, "Location prediction on trajectory data: A review," *Big Data Min. Anal.*, vol. 1, no. 2, pp. 108–127, 2018.
[108] K. Gilly, S. Filiposka, and S. Alcaraz, "Predictive migration performance in vehicular edge computing environments," *Appl. Sci.*, vol. 11, no. 3, p. 944, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/3/944
[109] S. Moosavi, B. Omidvar-Tehrani, and R. Ramnath, "Trajectory annotation by discovering driving patterns," in *Proc. 3rd ACM SIGSPATIAL Workshop Smart Cities Urban Anal. (UrbanGIS)*, 2017, pp. 1–4. [Online]. Available: https://doi.org/10.1145/3152178.3152184
[110] 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," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1, 2015, pp. 802–810.
[111] B. Zoph and Q. V. Le. "Neural architecture search with reinforcement learning." 2016. [Online]. Available: http://arxiv.org/abs/1611.01578
[112] N. Spolaôr, E. A. Cherman, M. C. Monard, and H. D. Lee, "ReliefF for multi-label feature selection," in *Proc. Braz. Conf. Intell. Syst.*, 2013, pp. 6–11.
[113] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *Proc. 39th Euromicro Conf. Softw. Eng. Adv. Appl.*, 2013, pp. 357–364.
[114] L. Li, J. Dong, D. Zuo, and J. Wu, "SLA-aware and energy-efficient VM consolidation in cloud data centers using robust linear regression prediction model," *IEEE Access*, vol. 7, pp. 9490–9500, 2019.
[115] M. S. Ricardo, N. Goel, M. Zaman, R. Joshi, M. Daraghmeh, and A. Agarwal, "Developing machine learning and deep learning models for host overload detection in cloud data center," in *Proc. IEEE 12th Annu. Inf. Technol. Electron. Mobile Commun. Conf. (IEMCON)*, 2021, pp. 0619–0626.
[116] L. Li, J. Dong, D. Zuo, Y. Zhao, and T. Li, "SLA-aware and energy-efficient VM consolidation in cloud data centers using host state binary decision tree prediction model," *IEICE Trans. Inf. Syst.*, vol. E102-D, no. 10, pp. 1942–1951, 2019.
[117] P. Nehra and A. Nagaraju, "Host utilization prediction using hybrid kernel based support vector regression in cloud data centers," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 8, pp. 6481–6490, Sep. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821000975
[118] L. Tang, X. He, P. Zhao, G. Zhao, Y. Zhou, and Q. Chen, "Virtual network function migration based on dynamic resource requirements prediction," *IEEE Access*, vol. 7, pp. 112348–112362, 2019.
[119] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: https://doi.org/10.1162/neco.2006.18.7.1527
[120] K. Qu, W. Zhuang, X. Shen, X. Li, and J. Rao, "Dynamic resource scaling for VNF over nonstationary traffic: A learning approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 2, pp. 648–662, Jun. 2021.
[121] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, vol. 2. Cambridge, MA, USA: MIT Press, 2006.
[122] G. Comert and A. Bezuglov, "An online change-point-based model for traffic parameter prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1360–1369, Sep. 2013.
[123] N. K. Biswas, S. Banerjee, U. Biswas, and U. Ghosh, "An approach towards development of new linear regression prediction model for reduced energy consumption and SLA violation in the domain of green cloud computing," *Sustain. Energy Technol. Assess.*, vol. 45, Jun. 2021, Art. no. 101087. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213138821000977
[124] S. M. Moghaddam, M. O'Sullivan, C. Walker, S. F. Piraghaj, and C. P. Unsworth, "Embedding individualized machine learning prediction models for energy efficient VM consolidation within cloud data centers," *Future Gener. Comput. Syst.*, vol. 106, pp. 221–233, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19308969
[125] C. Nguyen, C. Klein, and E. Elmroth, "Multivariate LSTM-based location-aware workload prediction for edge data centers," in *Proc. 19th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGRID)*, 2019, pp. 341–350.
[126] B. R. Raghunath and B. Annappa, "Prediction based dynamic resource provisioning in virtualized environments," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, 2017, pp. 100–105.
[127] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host CPU utilization in the cloud using evolutionary neural networks," *Future Gener. Comput. Syst.*, vol. 86, pp. 162–173, Sep. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17322793
[128] G. J. L. Paulraj, S. A. J. Francis, J. D. Peter, and I. J. Jebadurai, "A combined forecast-based virtual machine migration in cloud data centers," *Comput. Elect. Eng.*, vol. 69, pp. 287–300, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790617315732
[129] H.-G. Kim, S.-Y. Jeong, D.-Y. Lee, H. Choi, J.-H. Yoo, and J. W.-K. Hong, "A deep learning approach to VNF resource prediction using correlation between VNFs," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, 2019, pp. 444–449.
[130] N. T. Hieu, M. D. Francesco, and A. Ylä-Jääski, "Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data Centers," *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 186–199, Jan./Feb. 2020.

[131] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, 2017, pp. 67–72.

[132] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 361–376, Feb. 2020.

[133] D. Bega, M. Gramaglia, R. Perez, M. Fiore, A. Banchs, and X. Costa-Pérez, "AI-based autonomous control, management, and orchestration in 5G: From standards to algorithms," *IEEE Netw.*, vol. 34, no. 6, pp. 14–20, Nov./Dec. 2020.

[134] C. Fiandrino, C. Zhang, P. Patras, A. Banchs, and J. Widmer, "A machine-learning-based framework for optimizing the operation of future networks," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 20–25, Jun. 2020.

[135] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep Spatio-temporal neural networks," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (Mobihoc)*, 2018, pp. 231–240. [Online]. Available: https://doi.org/10.1145/3209582.3209606

[136] K. Park and V. S. Pai, "CoMon: A mostly-scalable monitoring system for PlanetLab," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, Jan. 2006. [Online]. Available: https://doi.org/10.1145/1113361.1113374

[137] P. Gill, C. Diot, L. Y. Ohlsen, M. Mathis, and S. Soltesz, "M-Lab: User initiated Internet data for the research community," *SIGCOMM Comput. Commun. Rev.*, vol. 52, no. 1, pp. 34–37, Mar. 2022. [Online]. Available: https://doi.org/10.1145/3523230.3523236

[138] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, 1995, pp. 1942–1948.

[139] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[140] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 312–317.

[141] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surveys*, vol. 54, no. 2, pp. 1–38, Mar. 2021. [Online]. Available: https://doi.org/10.1145/3439950

[142] A. Frank, D. Yang, A. Brinkmann, M. Schulz, and T. Süss, "Reducing false node failure predictions in HPC," in *Proc. IEEE 26th Int. Conf. High Perform. Comput. Data Anal. (HiPC)*, 2019, pp. 323–332.

[143] K. Haghshenas, A. Pahlevan, M. Zapater, S. Mohammadi, and D. Atienza, "MAGNETIC: Multi-agent machine learning-based approach for energy efficient dynamic consolidation in data centers," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 30–44, Jan./Feb. 2022.

[144] D. D. Vu, X. T. Vu, and Y. Kim, "Deep learning-based fault prediction in cloud system," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, 2021, pp. 1826–1829.

[145] Y. Xu et al., "Improving service availability of cloud systems by predicting disk error," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, Jul. 2018, pp. 481–494. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/xu-yong

[146] Y. Li et al., "Predicting node failures in an ultra-large-scale cloud computing platform: An AIOps solution," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 2, pp. 1–24, Apr. 2020. [Online]. Available: https://doi.org/10.1145/3385187

[147] Q. Lin et al., "Predicting node failure in cloud service systems," in *Proc. ESEC/FSE*, 2018, pp. 480–490. [Online]. Available: https://doi.org/10.1145/3236024.3236060

[148] A. Das, F. Mueller, C. Siegel, and A. Vishnu, "DESH: Deep learning for system health prediction of lead times to failure in HPC," in *Proc. 27th Int. Symp. High Perform. Parallel Distrib. Comput. (HPDC)*, 2018, pp. 40–51. [Online]. Available: https://doi.org/10.1145/3208040.3208051

[149] S. Behera, L. Wan, F. Mueller, M. Wolf, and S. Klasky, "Orchestrating fault prediction with live migration and checkpointing," in *Proc. 29th Int. Symp. High-Perform. Parallel Distrib. Comput. (HPDC)*, 2020, pp. 167–171. [Online]. Available: https://doi.org/10.1145/3369583.3392672

[150] A. Das, F. Mueller, P. Hargrove, E. Roman, and S. Baden, "Doomsday: Predicting which node will fail when on supercomputers," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC)*, 2018, pp. 108–121.

[151] S. Nam, J. Hong, J.-H. Yoo, and J. W.-K. Hong, "Virtual machine failure prediction using log analysis," in *Proc. 22nd Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, 2021, pp. 279–284.

[152] D. Lan, A. Taherkordi, F. Eliassen, Z. Chen, and L. Liu, "Deep reinforcement learning for intelligent migration of fog services in smart cities," in *Algorithms and Architectures for Parallel Processing*, M. Qiu, Ed. Cham, Switzerland: Springer Int., 2020, pp. 230–244. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-60239-0_16

[153] M. Elsaid, H. Abbas, and C. Meinel, "Machine learning approach for live migration cost prediction in VMware environments," in *Proc. 9th Int. Conf. Cloud Comput. Services Sci. (CLOSER)*, 2019, pp. 456–463.

[154] M. Duggan, J. Duggan, E. Howley, and E. Barrett, "A network aware approach for the scheduling of virtual machine migration during peak loads," *Clust. Comput.*, vol. 20, no. 3, pp. 2083–2094, Sep. 2017. [Online]. Available: https://doi.org/10.1007/s10586-017-0948-7

[155] M. Zangiabady, A. Garcia-Robledo, J.-L. Gorricho, J. Serrat-Fernandez, and J. Rubio-Loyola, "Self-adaptive online virtual network migration in network virtualization environments," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 9, 2019, Art. no. e3692. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3692

[156] S. W. Park, A. Boukerche, and S. Guan, "A novel deep reinforcement learning based service migration model for mobile edge computing," in *Proc. IEEE/ACM 24th Int. Symp. Distrib. Simulat. Real Time Appl. (DS-RT)*, 2020, pp. 1–8.

[157] C. Fan and L. Li, "Service migration in mobile edge computing based on reinforcement learning," *J. Phys. Conf.*, vol. 1584, Jul. 2020, Art. no. 12058.

[158] Y. Peng, L. Liu, Y. Zhou, J. Shi, and J. Li, "Deep reinforcement learning-based dynamic service migration in vehicular networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[159] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2014, pp. 1350–1354.

[160] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, p. 19. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[161] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Ch. Assoc. Comput. Linguist. Human Language Technol.*, vol. 1. Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[162] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[163] S. Li et al., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf

[164] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 17105–17115. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c6b8c8d762da15fa8dbbdfb6baf9e260-Paper.pdf

[165] A. Ng, "Unbiggen Ai," *IEEE Spectr.*, vol. 9, Apr. 2022.

[166] M. Motamedi, N. Sakharnykh, and T. Kaldewey. "A data-centric approach for training deep neural networks with less data." 2021. [Online]. Available: https://arxiv.org/abs/2110.03613

[167] M. Emu and S. Choudhury, "Ensemble deep learning assisted VNF deployment strategy for next-generation IoT services," *IEEE Open J. Comput. Soc.*, vol. 2, pp. 260–275, 2021.

[168] W. Wang, S. Ge, and X. Zhou, "Location-privacy-aware service migration in mobile edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2020, pp. 1–6.

[169] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[170] X. Zhao, J. Liu, B. Ji, and L. Wang, "Service migration policy optimization considering user mobility for E-Healthcare applications," *J. Healthcare Eng.*, vol. 2021, Jun. 2021, Art. no. 9922876. [Online]. Available: https://doi.org/10.1155/2021/9922876

[171] S. Cao, Y. Wang, and C. Xu, "Service migrations in the cloud for mobile accesses: A reinforcement learning approach," in *Proc. Int. Conf. Netw. Archit. Storage (NAS)*, 2017, pp. 1–10.

[172] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[173] D. Wang, X. Tian, H. Cui, and Z. Liu, "Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network," *China Commun.*, vol. 17, no. 8, pp. 31–44, 2020.

[174] L. Rui, M. Zhang, Z. Gao, X. Qiu, Z. Wang, and A. Xiong, "Service migration in multi-access edge computing: A joint state adaptation and reinforcement learning mechanism," *J. Netw. Comput. Appl.*, vols. 183–184, Jun. 2021, Art. no. 103058. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521000825

[175] M. Zhang, H. Huang, L. Rui, G. Hui, Y. Wang, and X. Qiu, "A service migration method based on dynamic awareness in mobile edge computing," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2020, pp. 1–7.

[176] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Netw.*, vol. 33, no. 3, pp. 26–33, May/Jun. 2019.

[177] F. D. Vita, D. Bruneo, A. Puliafito, G. Nardini, A. Virdis, and G. Stea, "A deep reinforcement learning approach for data migration in multi-access edge computing," in *Proc. ITU Kaleidoscope Mach. Learn. 5G Future (ITU K)*, 2018, pp. 1–8.

[178] F. De Vita, G. Nardini, A. Virdis, D. Bruneo, A. Puliafito, and G. Stea, "Using deep reinforcement learning for application relocation in multi-access edge computing," *IEEE Commun. Stand. Mag.*, vol. 3, no. 3, pp. 71–78, May/Jun. 2019.

[179] X. Yuan, Y. Zhu, Z. Zhao, Y. Zheng, J. Pan, and D. Liu, "An A3C-based joint optimization offloading and migration algorithm for SD-WBANs," in *Proc. IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6.

[180] H. Ren, Y. Wang, C. Xu, and X. Chen, "SMig-RL: An evolutionary migration framework for cloud services based on deep reinforcement learning," *ACM Trans. Internet Technol.*, vol. 20, no. 4, pp. 1–18, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3414840

[181] H. Wang, Y. Li, A. Zhou, Y. Guo, and S. Wang, "Service migration in mobile edge computing: A deep reinforcement learning approach," *Int. J. Commun. Syst.*, vol. 36, no. 1, 2023, Art. no. e4413. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4413

[182] S. Chen, H. Chen, J. Ruan, and Z. Wang, "Context-aware online offloading strategy with mobility prediction for mobile edge computing," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2021, pp. 1–9.

[183] W. Chen, Y. Chen, J. Wu, and Z. Tang, "A multi-user service migration scheme based on deep reinforcement learning and SDN in mobile edge computing," *Phys. Commun.*, vol. 47, Aug. 2021, Art. no. 101397. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490721001348

[184] F. Brandherm, L. Wang, and M. Mühlhäuser, "A learning-based framework for optimizing service migration in mobile edge clouds," in *Proc. 2nd Int. Workshop Edge Syst. Anal. Netw. (EdgeSys)*, 2019, pp. 12–17. [Online]. Available: https://doi.org/10.1145/3301418.3313939

[185] A. Abouaomar, Z. Mlika, A. Filali, S. Cherkaoui, and A. Kobbane, "A deep reinforcement learning approach for service migration in MEC-enabled vehicular networks," in *Proc. IEEE 46th Conf. Local Comput. Netw. (LCN)*, 2021, pp. 273–280.

[186] Y. Dai, Q. Zhang, and L. Yang, "Virtual machine migration strategy based on multi-agent deep reinforcement learning," *Appl. Sci.*, vol. 11, p. 7993, Aug. 2021.

[187] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, Apr. 2020.

[188] D. M. Manias, H. Hawilo, and A. Shami, "A machine learning-based migration strategy for virtual network function instances," in *Proc. Future Technol. Conf.*, 2020, pp. 563–577.

[189] X. Li, N. Samaan, and A. Karmouch, "An automated VNF manager based on parameterized action MDP and reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2021, pp. 1–6.

[190] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008–2025, Aug. 2017.

[191] D. Basu, X. Wang, Y. Hong, H. Chen, and S. Bressan, "Learn-as-you-go with megh: Efficient live migration of virtual machines," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1786–1801, Aug. 2019.

[192] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. "Prioritized experience replay." 2015. [Online]. Available: http://arxiv.org/abs/1511.05952

[193] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms." 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[194] T. Hirayama, T. Miyazawa, M. Jibiki, and V. P. Kafle, "Service function migration scheduling based on encoder–decoder recurrent neural network," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, 2019, pp. 193–197.

[195] X. Bai, H. Lu, and Y. Lu, "Learning framework for virtual network function instance migration," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2018, pp. 1–7.

[196] W. Guan, H. Zhang, and V. C. Leung, "Slice reconfiguration based on demand prediction with dueling deep reinforcement learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.

[197] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "Toward using reinforcement learning for trigger selection in network slice mobility," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2241–2253, Jul. 2021.

[198] J. Yao and M. Chen, "A flexible deployment scheme for virtual network function based on reinforcement learning," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, 2020, pp. 1505–1510.

[199] J. Chen, J. Chen, and H. Zhang, "DRLEC: Multi-agent DRL based elasticity control for VNF migration in SDN/NFV networks," in *Proc. 26th IEEE Asia–Pac. Conf. Commun. (APCC)*, 2021, pp. 89–93.

[200] R. Chen, H. Lu, Y. Lu, and J. Liu, "MSDF: A deep reinforcement learning framework for service function chain migration," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2020, pp. 1–6.

[201] K. Ray, A. Banerjee, and N. C. Narendra, "Proactive microservice placement and migration for mobile edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, 2020, pp. 28–41.

[202] S. Guo, Y. Dai, S. Xu, X. Qiu, and F. Qi, "Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6010–6022, Jul. 2020.

[203] H. Zhang, R. Wang, W. Sun, and H. Zhao, "Mobility management for blockchain-based ultra-dense edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7346–7359, Nov. 2021.

[204] B. Yang, P. Han, C. Feng, Y. Liu, and L. Guo, "Service migration with high-order MDP in mobile edge computing," in *Proc. 13th Int. Conf. Commun. Softw. Netw. (ICCSN)*, 2021, pp. 206–212.

[205] C. Wang et al., "An adaptive deep *Q*-learning service migration decision framework for connected vehicles," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, 2020, pp. 944–949.

[206] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1603–1614, Jul. 2021.

[207] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. "Counterfactual multi-agent policy gradients." 2017. [Online]. Available: http://arxiv.org/abs/1705.08926

[208] O. A. Wahab, N. Kara, C. Edstrom, and Y. Lemieux, "MAPLE: A machine learning approach for efficient placement and adjustment of virtual network functions," *J. Netw. Comput. Appl.*, vol. 142, pp. 37–50, Sep. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804519301924

[209] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 712–725, Sep./Oct. 2019.

[210] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, Jun. 2019.

[211] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, 2015, pp. 1–9.

[212] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A dynamic service migration mechanism in edge cognitive computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–15, Apr. 2019. [Online]. Available: https://doi.org/10.1145/3239565

[213] Y. Wang et al., "Towards cost-effective service migration in mobile edge: A *Q*-learning approach," *J. Parallel Distrib. Comput.*, vol. 146, pp. 175–188, Dec. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303488

[214] C. Zhang and Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 96, pp. 111–118, Jul. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18329674

[215] "OpenAI." Accessed: Oct. 2022. [Online]. Available: https://openai.com/blog/openai-baselines-ppo/

[216] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. "Dueling network architectures for deep reinforcement learning." 2016. [Online]. Available: https://arxiv.org/abs/1511.06581

[217] Z. Liu, W. Qu, W. Liu, and K. Li, "Xen live migration with slowdown scheduling algorithm," in *Proc. Int. Conf. Parallel Distrib. Comput. Appl. Technol.*, 2010, pp. 215–221.

[218] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive, memory compression," in *Proc. IEEE Int. Conf. Clust. Comput. Workshops*, 2009, pp. 1–10.

[219] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth, "Evaluation of delta compression techniques for efficient live migration of large virtual machines," in *Proc. 7th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ. (VEE)*, 2011, pp. 111–120. [Online]. Available: https://doi.org/10.1145/1952682.1952698

[220] C. Jo, Y. Cho, and B. Egger, "A machine learning approach to live migration modeling," in *Proc. Symp. Cloud Comput. (SoCC)*, 2017, pp. 351–364. [Online]. Available: https://doi.org/10.1145/3127479.3129262

[221] S. E. Motaki, A. Yahyaouy, and H. Gualous, "A prediction-based model for virtual machine live migration monitoring in a cloud datacenter," *Computing*, vol. 103, pp. 2711–2735, Aug. 2021.

[222] Y. Cho, C. Jo, H. Kim, and B. Egger, "Towards economical live migration in data Centers," in *Economics of Grids, Clouds, Systems, and Services*, K. Djemame, J. Altmann, J. Á. Bañares, O. Agmon Ben-Yehuda, V. Stankovski, and B. Tuffin, Eds. Cham, Switzerland: Springer Int., 2020, pp. 173–188. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-63058-4_15

[223] R. de Jesus Martins, C. B. Both, J. A. Wickboldt, and L. Z. Granville, "Virtual network functions migration cost: From identification to prediction," *Comput. Netw.*, vol. 181, Nov. 2020, Art. no. 107429. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862031118X

[224] M. A. Altahat, A. Agarwal, N. Goel, and M. Zaman, "Neural network based regression model for virtual machines migration method selection," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2021, pp. 1–6.

[225] M. Wajahat, B. Balasubramanian, A. Gandhi, G. Jung, and S. P. Narayanan, "MERIT: Model-driven Rehoming for VNF chains," in *Proc. IEEE Int. Conf. Auton. Comput. Self Org. Syst. (ACSOS)*, 2020, pp. 139–145.

[226] Z. Lei, E. Sun, S. Chen, J. Wu, and W. Shen, "A novel hybrid-copy algorithm for live migration of virtual machine," *Future Internet*, vol. 9, no. 3, p. 37, 2017. [Online]. Available: https://www.mdpi.com/1999-5903/9/3/37

[227] T.-Y. Wu, N. Guizani, and J.-S. Huang, "Live migration improvements by related dirty memory prediction in cloud computing," *J. Netw. Comput. Appl.*, vol. 90, pp. 83–89, Jul. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517301133

[228] R. Shukla, R. K. Gupta, and R. Kashyap, "A Multiphase pre-copy strategy for the virtual machine migration in cloud," in *Smart Intelligent Computing and Applications*, S. C. Satapathy, V. Bhateja, and S. Das, Eds. Singapore: Springer, 2019, pp. 437–446. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-1921-1_43

[229] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "AI-based network-aware service function chain migration in 5G and beyond networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 472–484, Mar. 2022.

[230] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[231] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020.

[232] F. Wilhelmi, M. Carrascosa, C. Cano, A. Jonsson, V. Ram, and B. Bellalta, "Usage of network simulators in machine-learning-assisted 5G/6G networks," *IEEE Wireless Commun.*, vol. 28, no. 1, pp. 160–166, Feb. 2021.

[233] J. Xu and Z. Zhu, "Reinforced continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 907–916. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/cee631121c2ec9232f3a2f028ad5c89b-Paper.pdf

[234] O. Ostapenko, P. Rodriguez, M. Caccia, and L. Charlin, "Continual learning via local module composition," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 30298–30312. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/fe5e7cb609bdbe6d62449d61849c38b0-Paper.pdf

[235] Q. Pham, C. Liu, and S. Hoi, "DualNet: Continual learning, fast and slow," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 16131–16144. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/86a1fa88adb5c33bd7a68ac2f9f3f96b-Paper.pdf

[236] Q. Gao, Z. Luo, D. Klabjan, and F. Zhang, "Efficient architecture search for continual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 2, 2022, doi: 10.1109/TNNLS.2022.3151511.

[237] J. Xu, J. Ma, X. Gao, and Z. Zhu, "Adaptive progressive continual learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6715–6728, Oct. 2022.

[238] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022.

[239] P. Singh, V. K. Verma, P. Mazumder, L. Carin, and P. Rai, "Calibrating CNNs for lifelong learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15579–15590. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/b3b43aeeacb258365cc69cdaf42a68af-Paper.pdf

[240] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson, "Explainable artificial intelligence: An analytical review," *WIREs Data Min. Knowl. Disc.*, vol. 11, no. 5, 2021, Art. no. e1424. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1424

[241] L. Wells and T. Bednarz, "Explainable AI and reinforcement learning—A systematic review of current approaches and trends," *Front. Artif. Intell.*, vol. 4, May 2021, Art. no. 550030. [Online]. Available: https://www.frontiersin.org/article/10.3389/frai.2021.550030

[242] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowl. Based Syst.*, vol. 214, Feb. 2021, Art. no. 106685. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120308145

[243] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *J. Artif. Intell. Res.*, vol. 70, pp. 245–317, Mar. 2021.

[244] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24, 2011, pp. 2546–2554. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[245] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220311693

[246] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, J. Vanschoren, Eds. Cham, Switzerland: Springer Int., 2019, pp. 3–33. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-05318-5_1

[247] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Netw. Appl.*, vol. 26, no. 3, pp. 1145–1168, Jun. 2021. [Online]. Available: https://doi.org/10.1007/s11036-020-01624-1

[248] A. Boukerche and R. E. De Grande, "Vehicular cloud computing: Architectures, applications, and mobility," *Comput. Netw.*, vol. 135, pp. 171–189, Apr. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618300057

[249] S. Olariu, "A survey of vehicular cloud research: Trends, applications and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2648–2663, Jun. 2020.

[250] C. Tang, S. Xia, Q. Li, W. Chen, and W. Fang, "Resource pooling in vehicular fog computing," *J. Cloud Comput.*, vol. 10, no. 1, p. 19, Feb. 2021. [Online]. Available: https://doi.org/10.1186/s13677-021-00233-x

[251] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual edge computing using vehicular micro clouds," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2019, pp. 537–541.

[252] F. Dressler et al., "V-Edge: Virtual edge computing as an enabler for novel microservices and cooperative computing," *IEEE Netw.*, vol. 36, no. 3, pp. 24–31, May/Jun. 2022.