International Journal of Information Management Data Insights 3 (2023) 100158

Contents lists available at ScienceDirect

International Journal of Information Management Data Insights

journal homepage: www.elsevier.com/locate/jjimei

Graph neural networks for representing multivariate resource usage: A multiplayer mobile gaming case-study☆

Theodoros Theodoropoulos[a,*], Antonios Makris[a], Ioannis Kontopoulos[a], John Violos[a], Przemysław Tarkowski[b], Zbyszek Ledwoń[b], Patrizio Dazzi[c], Konstantinos Tserpes[a]

[a] Harokopio University of Athens, Omirou 9, Athens 17778, Greece
[b] Orbital Knight, Aleje Jerozolimskie 96, Warsaw 00807, Poland
[c] University of Pisa, Largo Bruno Pontecorvo 3, Pisa 56127, Italy

ARTICLE INFO

Keywords:
Graph neural networks
Resource usage prediction
Deep learning
Mobile gaming

ABSTRACT

The emergence of Multiplayer Mobile Gaming (MMG) applications is intertwined with a plethora of Quality of Service and Quality of Experience requirements. Resource usage prediction can provide valuable insights into the corresponding orchestration and management process in the form of several proactive functionalities in resource scaling, service migration, task offloading and scheduling. These processes are crucial in the Cloud and Edge environments exploited by MMG applications. Thus, producing accurate resource usage predictions concerning these types of applications is of paramount importance. To that end, we propose a resource usage representation paradigm based on Graph Neural Networks (GNNs). The novelty of this approach is based on the process of leveraging the dependencies that exist among the various types of computational resources. Furthermore, we expand upon this representation approach to develop a GNN-based Encoder-Decoder model that caters to the complexities of resource usage and can provide multi-step resource usage predictions. This model is compared against numerous well-established Encoder-Decoder and Deep Learning prediction models to assess its efficiency. Finally, the proposed model is incorporated in a proactive Horizontal Autoscaling solution that manages to outperform a standard reactive Horizontal Autoscaling approach in the context of a large-scale simulation, in terms of various performance metrics, while keeping the volume of the required computational resources to a minimum. The findings of this work showcase the importance of developing novel approaches in order to represent resource usage and the numerous benefits in the context of application performance and resource consumption that may derive from such scientific endeavors.

1. Introduction

During the last decades, we have witnessed the emergence of rather demanding applications in terms of Quality of Experience (QoE) and Quality of Service (QoS) requirements. Technological paradigms, such as Multiplayer Mobile Gaming and Extended Reality (XR) applications (Makris et al., 2021a), are associated with various QoS (Theodoropoulos et al., 2022a) and QoE requirements. These two types of applications are often intertwined regarding their perspective requirements and actual architectural design (Taleb et al., 2022). The backbone of both types of applications is the ability to provide an immersive experience to the end-user. Providing acceptable levels of immersion requires low latencies and high bandwidths. Especially in the field of XR applications, these requirements are stringent. The corresponding scientific literature has shown that for an end-user experience to be acceptable, the end-to-end latency shall be less than 15ms, and the bandwidth should be able to scale up to 30 Gbps (Boos, Chu, & Cuervo, 2016). Furthermore, the inevitable emergence of faults in task processing may impose dire ramifications on the implementation of immersive experiences since they often result in service delivery disruption and thus, the desired immersion is jeopardised. Therefore, these types of applications need to be able to manifest fault tolerance characteristics. Finally, both types of applications are extremely demanding in terms of computational resources as they are associated with rendering complex 3D models, highly defined graphics, and various advanced assets. Tra-

☆ This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016509 (CHARITY) and No 871793 (ACCORDION).

* Corresponding author.
E-mail addresses: ttheod@hua.gr (T. Theodoropoulos), amakris@hua.gr (A. Makris), kontopoulos@hua.gr (I. Kontopoulos), violos@hua.gr (J. Violos), ptarkowski@orbitalknight.com (P. Tarkowski), zledwon@orbitalknight.com (Z. Ledwoń), patrizio.dazzi@unipi.it (P. Dazzi), tserpes@hua.gr (K. Tserpes).

https://doi.org/10.1016/j.jjimei.2023.100158
Received 29 June 2022; Received in revised form 25 January 2023; Accepted 25 January 2023

ditional monolithic development approaches would require incorporating the essential computational resources into the end-user equipment, thus making it expensive and bulky. In the case of multiplayer mobile gaming, this setback directly contrasts with the core principles of mobile gaming. These principles require that mobile gaming applications run on hardware to be mobile and relatively inexpensive. Cloud computing environments provide computational resources that applications can use to serve the needs of users through the Internet. The cornerstone of Cloud computing is based on the use of shared computational resources, which may be distributed over multiple locations. By enabling end-users to access the necessary computational resources remotely, the burden of computational adequacy is transferred to these remote resources, thus allowing the end-user devices to be aligned with the main concepts of mobile gaming. Unfortunately, Cloud infrastructures alone cannot fully support immersive applications associated with requirements in terms of low latency and high bandwidth. The main reason is that end devices are usually distant from the Cloud servers, thus adding processing and network overhead, resulting in high latency, low bandwidth and overall performance degradation.

A conceptual approach that combines the Cloud's benefits and the decentralised processing of services on edge devices is known as Edge Computing. Edge computing has attracted a lot of attention both from industry and academia in recent years (Hao, Novak, Yi, & Li, 2017; Hu, Patel, Sabella, Sprecher, & Young, 2015; Makris, Psomakelis, Theodoropoulos, & Tserpes, 2022; Patel et al., 2014; Satyanarayanan, 2017). It is considered a key enabler for addressing the increasingly strict requirements of the next-generation applications (Sabella et al., 2019). In general, Edge computing aims to establish decentralised topologies and allow the relocation of various computational and storage resources closer to the Edge of the network. Doing so is expected to provide service delivery and content with better response times, transfer rates, and higher scalability and availability. In addition, Edge computing significantly reduces the amount of data in transit towards remote clouds and enables data processing near the data sources. Ultimately, expanding the possibilities for more delay-sensitive and high-bandwidth applications that would not be feasible using Cloud and far remote processing alone (Makris et al., 2021b). Within the context of immersive applications, Edge computing enables the processing to take place closer to the end-user devices. By doing so, the overall end-to-end latency is significantly reduced. This inherent characteristic of Edge Computing is vital to these modern paradigms since they both present the need for low end-to-end latency. End-to-end latency is the time required for a task to be processed upon arriving at a specified computational node. Therefore, Edge Computing Infrastructures need sufficient computational resources to process tasks within the acceptable time specified by the QoS requirements. If not enough computational resources are allocated, then overheads in task execution are expected to occur.

The resource allocation process is closely intertwined with resource usage metrics such as CPU, Memory and the number of bytes sent or received throughout the network that correspond to the allocated processing nodes. Resource usage metrics may serve as indicators towards whether a set of processing nodes is adequate to handle the incoming tasks in the frame of the next time-steps or whether an orchestration tool should allocate additional computational resources (Roy, Dubey, & Gokhale, 2011). Scaling down is the reverse process and refers to the de-allocation of redundant processing nodes in order to reduce energy consumption and monetary charges. Autoscaling is the automated process that consists of the scale-up and scale-down actions that are performed, based on the incoming workloads and the pre-defined QoS requirements. The autoscaling may take place either reactively, meaning that after the degradation of a QoS metric the orchestration tool can request for a scale-up action to take place, or proactively, in order to prevent the QoS degradation from occurring at the first place. Despite their many benefits, one significant drawback of proactive autoscaling solutions is that they tend to result in the over-provisioning of computational resources. A well established heuristic approach is the Horizontal Pod Autoscaler (Vohra, 2017). This approach was introduced in the context of Kubernetes[1], where pods refer to the smallest, most basic deployable objects in Kubernetes and represent a single instance of a running process in a distributed infrastructure.

Even though there is a plethora of autoscaling solutions already in production, such as Kubernetes, unfortunately they do not incorporate the sophistication required in order to manifest the desired properties that are vital in Edge & Cloud environments. The scale-up actions should be implemented in a timely manner before a bottleneck occurs, while also minimizing deployment and startup delays. This motivated us to propose a mechanism that is capable of establishing accurate resource usage predictions that can be incorporated in a proactive autoscaling solution. The resource usage metrics for a processing node exhibit a time-series format and most of the times a non-linear behaviour, thus making the use of Recurrent Neural Network (RNN) a seemingly ideal option.

Monitoring and predicting the capacity under which the Edge nodes are operating in terms of resource usage can be a valuable piece of information in implementing the decision-making processes mentioned above in a proactive manner. Resource usage predictions that derive from time-series characteristics of historical data are an essential source of information. Such predictions are a strong indicator regarding the adequacy of the available computational nodes, either when considering a scenario in which additional workloads will arrive or as a way to predict potential QoS degradation that may occur in the near future. Publicly available monitoring tools, such as Prometheus[2], can provide the resource metrics in a stream format. These time-series streams can be leveraged to produce datasets appropriate for the prediction models' training process. Time-series forecasting has been proven to be a quite powerful tool when periodic phenomena are involved (Ensafi, Amin, Zhang, & Shah, 2022a). The dynamic characteristics intrinsic to Cloud & Edge infrastructures derive from the fluctuation of application requests and the associated workloads. The number of requests per time interval changes at different time-frames and is affected by many periodic phenomena. Thus, resource usage presents highly serial and cross-correlation values, making the use of time series approaches quite effective (Nisar & Ahmed, 2020).

Recurrent Neural Networks (RNN) (Shiva Prakash, Sanjeev, Prakash, & Chandrasekaran, 2019) that leverage time-series characteristics via Gated Recurrent Units (GRU) (Shen, Tan, Zhang, Zeng, & Xu, 2018) and Long Short-Term Memory (LSTM), can be used to predict the various resource usage metrics accurately. The typical time series models and the simple RNNs are mainly utilised to formulate only one-step predictions. Multi-step prediction approaches consist of a sequence of values that correspond to sequential time steps. The multi-step prediction in the context of time series for modelling resource usage is important because it can be leveraged to achieve enhanced granularity of resource orchestration and management compared to single-step prediction approaches. The resource orchestration and management entities can implement more sophisticated real-time strategies when leveraging multi-step insight (Theodoropoulos, Makris, Violos, & Tserpes, 2022b) because each virtual device and service function has a different deployment time. The Encoder-Decoder architectural paradigm can be used to facilitate multi-step time series forecasting. Encoder-decoder structures are Deep Learning (DL) architectures consisting of two Neural Networks interacting via intermediate representations and producing sequence-to-sequence predictions.

Encoder-Decoder topologies have shown great promise in the context of surpassing other prediction models and as a result, the authors of this paper have chosen to dedicate a significant part of this paper to them. The encoder part of the Encoder-Decoder topologies receives as input a variable-length sequence and transforms it into a state with a

[1] https://kubernetes.io/

[2] https://prometheus.io/

fixed shape which can then be leveraged by the decoder part in order to formulate the desired predictions. However it is questionable whether the encoders that have been used up this point in the context of resource usage prediction can optimally encapsulate the underlying intricacies of resource usage. As a matter of fact, the main research questions that the authors attempt to address in this paper are as follows:

- *Can the use of encoder topologies that may encapsulate the dependencies that exist between the resource usage metrics assist towards establishing more accurate prediction models in the field of resource usage prediction?*
- *Can the incorporation of the subsequent prediction model in a proactive autoscaling operational pipeline significantly improve the efficiency of the underlying Cloud & Edge infrastructure across a number of performance metrics?*

The facts mentioned above served as a motivation for proposing an advanced resource usage representation paradigm that leverages Graph Neural Networks (GNNs). According to this paradigm, each monitored resource usage metric is represented as a node of a Graph Convolution Network. This novel representation approach was then expanded upon by incorporating it into a resource usage prediction model based on the Encoder-Decoder paradigm, which can produce accurate multi-step predictions. This Encoder-Decoder model is called GCN-LSTM in this paper's context. The main advantage of the proposed methodology compared to other approaches, is that the graph representation allows us to take advantage of the interdependencies between the input variables. Current proactive autoscaling mechanisms employ prediction models that forecast workloads of multiple metrics based on identified trends of their past behavior. On the other hand, the proposed approach can exploit not only the past trends, but the possible dependencies that might exist between the metrics and their respective trends. This is achieved by iteratively updating the graph node representations through the exchange of information with their neighbors. By developing a more accurate prediction model, the authors of this work aspire to introduce more sophisticated and refined proactive autoscaling solutions that can greatly enhance the performance of contemporary types of applications, such as Multiplayer Mobile Gaming, while keeping the volume of the required computational resources at a bare minimum.

Given the relatively recent emergence of this type of applications that are intertwined with a plethora of rather demanding QoS requirements, the need to facilitate them has become quite significant. As such, the findings of this work are expected to be of interest to a wide range of parties. These parties include entities such as Cloud providers, application owners and Deep Learning researchers. Through accurate resource usage prediction, Cloud providers are capable of keeping up with QoS requirements that are often stated in the form of Service Level Agreements which are legal contracts that are established between them and application owners. Furthermore, resource usage prediction may enable application owners to provide a more refined experience to end-users. Finally, the scientific community that explores Deep Learning methodologies and their potential applications on Cloud & Edge computing environments is expected to gain valuable insights on the aforementioned topics.

To that end, the four major contributions of our research are:

- the use of Graph Neural Networks for representing resource usage. Furthermore, we expand upon this representation paradigm and propose using a GNN-based Encoder-Decoder model to predict resource usage. To the best of our knowledge, we are the first to try both approaches.
- the analysis of the complexities of resource usage prediction in the context of contemporary applications that are associated with a plethora of rather demanding QoS and QoE requirements.
- the analysis of various state-of-the-art Encoder-Decoder topologies.
- the experimental evaluation that compares the GCN-LSTM model to other well-established prediction models.
- the experimental evaluation that compares a proactive scaling approach that leverages the aforementioned proposed solution against a reactive one.

The rest of the paper is structured as follows: Section 2 highlights the related work in resource usage prediction, time-series and DL methodologies. Section 3 analyses different basic and Encoder-Decoder DL prediction models. Section 4 provides an analysis of Graph Neural Networks, Graph Convolution, the GCN-LSTM model and the proposed problem formulation. Section 5 analyses the case of multiplayer mobile gaming. Section 6 describes the experimental process undertaken to evaluate the efficiency of the proposed methodologies. Finally, Section 7 draws the final remarks on the paper, reports the current limitations and suggests future directions.

2. Related work

Over the years, machine/deep learning has been successfully applied in a wide variety of applications. Tameswar, Suddul, & Dookhitram (2022) presented a hybrid deep neural network model, which combines Nature-Inspired Algorithms with DNN for enhanced prediction of software bugs. In Sridevi & Suganthi (2022), an AI-based system is developed to measure and predict suitable candidates for available jobs. Parviero et al. (2022) presented a new data-driven agent based model whose parameters can be estimated by maximum likelihood, for predicting the results of a new product or service in the market. Another research (Walid, Ahmed, Zeyad, Galib, & Nesa, 2022) examined the reasons behind the failure of undergraduate admission seekers using different machine learning strategies. The models are able to provide "risk" warnings in advance in order to advise applicants for the university's undergraduate admissions test. Chondrodima, Georgiou, Pelekis, & Theodoridis (2022) introduced a novel data-driven approach based on Radial Basis Function neural networks for predicting public transport mobility. Al-Sulaiman (2022) developed a feed-forward deep neural network to predict stock prices at a given time. In Ensafi, Amin, Zhang, & Shah (2022b), various classical and advanced time-series forecasting models are applied to predict the future sales of furniture. Yang, McEwen, Ong, & Zihayat (2020) proposed a data-driven end-to-end depression detection framework which utilizes machine learning techniques, to provide a mechanism for mental health professionals in monitoring the depressive behaviors of people. Gellert, Florea, Fiore, Palmieri, & Zanetti (2019) used Markov chains, stride predictors and a hybrid predictor in modelling the evolution of electricity production and consumption in buildings, aimed at reducing uncertainty about the demand of electricity and its production from renewable sources. Nolle, Luettgen, Seeliger, & Mühlhäuser (2022) introduced a recurrent neural network, for real-time multi-perspective anomaly detection in business process event logs. Xiong, Yu, Zhang, & Leng (2021) proposed a deep learning approach which incorporates an attention mechanism based on the attractiveness and timeliness of individual terms contained in a news article for predicting news clicks. Nguyen, Tran, Thomassey, & Hamad (2021) proposed two data-driven approaches to provide better decisions in supply chain management, a Long Short Term Memory (LSTM) network-based method for forecasting multivariate time series data and an LSTM Autoencoder network-based method combined with a one-class support vector machine algorithm for detecting anomalies in sales. Brusch (2022) utilized a combination of image analysis methods and fuzzy cluster algorithms, including support vector machines and convolutional neural networks, to predict user preferences. Liu, Mai, Shan, & Wu (2020b) proposed a text analytics deep learning framework to automatically extract patterns for prediction potentially opportunistic insider trading.

In addition, deep learning methods offer a lot of promise in resource usage prediction. Predicting the volume of data traffic on a network and the number of service requests at specific time frames is of utmost importance for an optimal resource management plan (Serhani et al.,

2020). Accurate traffic prediction methodologies are fundamental for ensuring an effective strategy for load balancing and resource allocation, thus fulfilling the QoS requirements. With the increasing development of data centres, novel large-scale network traffic prediction methods need to be developed to handle complex higher dimension properties and non-linearity (Cao et al., 2018).

To this end, several well-established algorithms have been proposed and leveraged to achieve high-accuracy predictions of network utilization. One example is the usage of the time series models, namely Autoregressive–Moving-Average (ARMA) and its variations Autoregressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA). Eramo, Catena, Lavacca, & di Giorgio (2020) minimized the operation cost by exploiting SARIMA models and by taking into account two factors: i) the Cloud resource costs which occur when non-essential resource provisioning is performed due to traffic overestimation, and ii) the QoS degradation cost which occurs when the traffic is underestimated, resulting to fewer resources than needed to be allocated and thus jeopardizing the satisfaction of the users of the data services. Liu, Guo, Li, & Luo (2020a) developed a forecasting methodology of workload based on error correction that exploits the ARMA model in combination with an Elman Neural Network (ENN). The authors initially used the ARMA model for forecasting. Then, the forecasting error of each value produced by the ARMA model in the sequence was calculated and fed into the ENN. Consequently, the ENN exploited the forecasting error sequence to correct and optimize the forecasting values.

Another example of well-established algorithms used for resource usage prediction involves deep-learning models (Duc, Leiva, Casari, & Östberg, 2019). Specifically, GRU and LSTM neural networks have been used for forecasting CPU usage Janardhanan & Barrett (2017). Moreover, comparisons between the two neural network architectures for their forecasting ability have also been made (Violos, Psomakelis, Danopoulos, Tsanakas, & Varvarigou, 2020). Violos et al. (2021b) developed a Gated Recurrent Neural Network combined with a Hybrid Bayesian Evolutionary Strategy algorithm for the resource usage prediction over the edge. Their novelty lies in using the Evolution Strategy algorithm to fine-tune the network's hyper-parameters. Furthermore, still, Violos, Pagoulatou, Tsanakas, Tserpes, & Varvarigou (2021a) included another hyper-tuning technique to develop a Convolutional Neural Network optimized for predicting the resource usage in the edge. They developed a hybrid method for optimisation that exploits particle swarm optimization and Bayesian optimization, leading to superior experimental results compared to other machine learning meta-predictors and state-of-the-art resource usage models. Fujimoto, Fujita, & Hayashi (2021) exploited the concept of reservoir computing which is suitable for handling the dynamics of time series data. They used an Echo State Network (ESN) architecture (Lukoševičius, 2012) for short-term load forecasting tasks. ESN is a type of RNN that can describe the nonlinear behaviour of temporal dynamics based on a simple learning rule. The ESN architecture enables the development of flexible forecasting models with limited computational resources, making it suitable for edge implementation. Another approach to forecasting time-series data in the Edge computing environment was proposed by Pesala, Paul, Ueno, Praneeth Bugata, & Kesarwani (2021) where an incremental forecasting algorithm was presented. Due to limited resources and processing capabilities, Edge devices cannot process vast volumes of multivariate time-series data. Therefore, the authors developed a new forecasting method called Incremental Learning Vector Auto Regression (ILVAR). ILVAR minimizes the differences in variance between the actual and the forecasted values as a new chunk of time-series data arrives sequentially, thus updating the forecasting model incrementally. Their approach was evaluated on some Raspberry Pi-2 behaving as Edge devices and was compared with the Vector Auto Regression (VAR), Incremental Learning Extreme Learning Machine (ILELM), and Incremental Learning Long Short-Term Memory (ILLSTM) methods, yielding superior results.

Even though forecasting the resource usage over a network has been widely studied, optimal resource allocation and forecasting in edge and mobile gaming is now at the forefront of research as an increasing number of mobile devices with low computing capabilities are being used every day. Only in recent years have researchers tried to tackle mobile and Edge gaming challenges. Xu, Mehrotra, Mao, & Li (2013) developed the PROTEUS system that uses regression trees to forecast future network performance. In their research, PROTEUS was used to predict the occurrence of packet loss and system delays and managed to reduce the perceptual delay in a gaming application by up to 4s. Similarly, Basiri & Rasoolzadegan (2018) developed a delay-aware cost-minimizing resource allocation framework that can satisfy the delay requirement of the connected users in simulated environments of real-time online gaming. Sharif, Jung, Razzak, & Alazab (2021) developed an Edge Computing (EC) mechanism that dynamically allocates resources by considering the nature of the incoming requests and surpassing other EC schemes for resource utilization, average response time, task execution time, and energy consumption. Violos et al. (2022b) introduced a Double Tower Neural Network architecture that is capable of predicting resource usage in Edge computing environments in order to perform proactive autoscaling. This solution managed to improve various performance metrics, however it increased resource consumption when compared to a reactive approach by about 5%. Theodoropoulos et al have explored proactive fault tolerance methodolies for Edge and Cloud computing environments that leverage Deep Learning (Theodoropoulos et al., 2022c) in the context of resource usage prediction. Despite improving various fault tolerance metrics, this approach increased resource consumption when compared to a reactive approach by 3.2%. Li et al. (2019) developed a novel methodology called GAugur that accurately predicts the performance interference among games collocated in the Cloud. The authors use a classification model to identify the QoS requirement of a game co-located with a set of other games and a regression model to predict the performance degradation of a game. Their experiments demonstrated that their methodology could improve the overall performance by 15% and increase the resource utilization by at least 20%. A summary of the approaches discussed above, organized by subject area, is presented in Table 1.

Unfortunately, the aforementioned scientific works, despite their numerous merits, exhibit certain drawbacks that this work aspires to mitigate. These drawbacks are the following ones:

- Up until now, the various attempts at predicting resource usage have been mainly focused on capturing the temporal patterns that are inherent in the input sequences. This work, however, focuses on the importance of encapsulating multi-variate input sequences in a manner that is capable of capturing both the temporal and the structural relations that are present.
- One more significant issue that is present at the majority of the aforementioned intelligent resource allocation approaches is that they often tend to result in the over-provisioning of resources in order to improve the various performance metrics. As a matter of fact, in some cases the resource usage prediction models are designed in a manner that deliberately overestimates the resource demand that is expected to take place in the near future. It is rather obvious that the greater the volume of allocated computational resources is, the easier it shall be to improve the various performance metrics such as latency, due to the fact that there are more processing nodes to handle the incoming tasks and thus the formation of task execution overheads becomes a rare occurrence. However, by increasing the volume of allocated computational resources, the operational costs are also increased. The trade-off between improving performance metrics and reducing the underlying operational costs may be regarded as the cornerstone of the resource allocation optimization problem.
- Finally, the efficiency of many of the aforementioned approaches was examined using data that do not derive from real-world application usage scenarios. During these works, the authors chose to use datasets that correspond to generic computational processes, whose

Table 1
Summary of the different approaches, classified by subject area.

Subject areas	Reference
Network Traffic Prediction/Network Performance Forecasting	(Cao et al., 2018), (Eramo et al., 2020), (Xu et al., 2013)
Reliable Resource Provisioning	(Duc et al., 2019), (Basiri & Rasoolzadegan, 2018), (Sharif et al., 2021)
Workload Forecasting	(Janardhanan & Barrett, 2017; Liu et al., 2020a)
Resource Utilization Prediction	(Theodoropoulos et al., 2022c), (Violos et al., 2021b), (Violos et al., 2021a), (Violos et al., 2022b),
Performance Prediction	(Li et al., 2019)
Short-term Demand Forecasting	(Fujimoto et al., 2021), (Lukoševičius, 2012), (Pesala et al., 2021)

underlying resource consumption patterns do not reflect the resource intricacies that associated with contemporary applications. As a result, the actual efficiency of implementing the proposed solutions in real-world use-cases comes into question.

The proposed intelligent resource allocation presented at the context of this work aims at incorporating a prediction model that is capable of accurately predicting the resource usage consumption that is expected to take place in the near future. By doing so, it is capable of making optimal scaling up decisions when it is required in order to avoid task execution bottlenecks and the degradation of QoS. On top of that, the proposed model shall be able to accurately predict decreases in resource demand and to release the appropriate computational resources in order to avoid resource over-provisioning. Finally, the proposed intelligent resource allocation approach is tested using real-world data that correspond to a contemporary MMG application, in the context of providing accurate predictions results and enhancing the resource orchestration process in a manner that boosts application performance while preventing any potential over-provisioning of resources to taking place.

3. Resource usage prediction

Concerning resource usage prediction, the scientific community has largely investigated the statistical models like Poisson, Autoregressive–Moving-Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA). However, the recent advances in DL have drastically transformed the landscape of data analytics and, as a consequence, the decision-making processes. Especially in the case of time series prediction, Recurrent Neural Networks (RNNs) tend to surpass well-established statistical forecasting models significantly. For this reason, the authors of this paper chose to focus only on DL-based prediction models. The following section consists of two parts. The first describes classic recurrent neural network models, while the second focuses on Encoder-Decoder architectures that derive from these models.

3.1. RNN topologies

Long Short-Term Memory (LSTM) is a non-linear time series model initially introduced by Hochreiter & Schmidhuber (1997) to overcome the vanishing and exploding gradients problems and the short-term memory occurring in standard RNNs when dealing with long-term dependencies. The former issues make the earlier layers of RNN incapable of being trained sufficiently. The latter makes RNN incapable of carrying information from earlier time observations to later ones, and, as a result, an RNN network tends to forget too fast what has been learnt (Violos et al., 2020). In the standard RNN, the overall neural network is a chain of repeating modules formed as a series of simple hidden networks. In contrast, the hidden layers of LSTM introduce the concepts of *gate* and *memory cell* in each hidden layer. The gates in an LSTM unit enable it to preserve a more constant error that can be back-propagated through time (Chauhan & Palivela, 2021; Patterson & Gibson, 2017).

More specifically, to establish temporal connections, LSTM maintains an internal memory cell state throughout the whole life cycle. The memory cell state interacts with the intermediate output and the subsequent input to determine which elements of the internal state vector

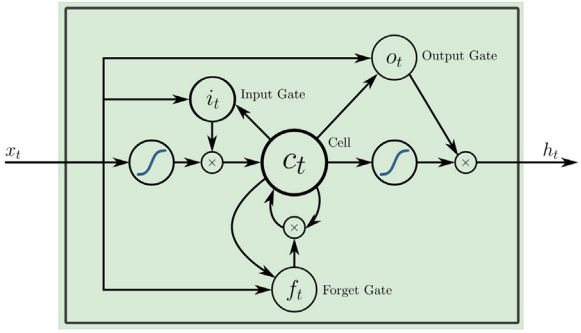

Fig. 1. LSTM unit architecture.

should be updated, maintained or forgotten on the basis of the outputs of the previous time step and the inputs of the present time step.

In addition, the LSTM structure also defines three gates: an *input gate* which controls the entry of the activations to the memory cell, a *forget gate* which is in charge of resetting the memory cells by forgetting the past input data and, finally, an *output gate* which determines the value of the next hidden state. The architecture of LSTM neworks is depicted in Fig. 1.

The Gated Recurrent Unit (GRU) model was first introduced by (Cho et al., 2014) in 2014 and represented a variant of LSTM. While LSTMs have two different states passed through the cells, cell state and hidden state, GRUs only contain one hidden state transferred between time steps. Furthermore, a GRU cell contains only two gates, *update gate* and *reset gate*. The update gate determines the amount of information stored in the previous hidden state that would be retained for the future. It is quite similar to the *input* and *forget* gate in the LSTM. However, the control of new memory content added to the network is presented only in GRUs. The model uses the reset gate to decide how much of the past information to forget. As GRU presets a simpler architecture than LSTM, it requires less computation and can be trained faster.

Bidirectional Long Short-Term Memory (BI-LSTM) neural networks are similar to the LSTM networks with the difference that the input flows in two directions. Specifically, in a standard LSTM network, the input can flow in one direction, either backwards or forward, but in a BI-LSTM, the input flows in both directions to preserve the future and the past information, namely backwards (future to past) or forward (past to future). BI-LSTM networks use two models: i) one model that learns the sequence of the input provided, and ii) a second model that learns the reverse of that sequence. Finally, the two networks are combined into one in a process called the Merge step, and it can be achieved by one of the following methodologies: sum, multiplication, averaging or concatenation which is the default methodology.

3.2. Encoder-Decoder (ED) topologies

The encoder-decoder architecture can handle inputs and outputs that are both variable-length sequences and thus is suitable for sequence-to-sequence prediction. This functionality is the result of the model's architecture. The encoder takes as input a variable-length sequence and

transforms it into a state with a fixed shape. The decoder is appropriately configured using the final states of the encoder. It is trained to generate the output based on the information gathered by the encoder. More specifically, the decoder is implemented using an LSTM model, which is trained to generate the output sequence. The initial hidden state of the decoder is the final hidden state procured from the encoder. Each part of the decoder is expected to output a value for each of the numerous future time steps being examined. For this reason, a Repeat-Vector layer is employed. Furthermore, two additional layers are incorporated, a fully connected and an output layer. The fully-connected layer interprets each time step in the decoder output sequence and sends the product to the output layer, resulting in a single-step prediction in the output sequence. For predicting the next time-steps, it is essential to wrap both the implementation and the output layers inside a time-distributed wrapper. The output provided by the decoder will be processed by the same fully-connected output layer, thus enabling the wrapped layers to be used for each time step by the decoder.

3.2.1. LSTM & Bidirectional LSTM ED

For the Encoder part of the architecture, an LSTM / Bidirectional LSTM model was utilized. This model receives an input sequence over time and produces a N element output vector which entails an internal representation of the input sequence. The size of N corresponds to the number of LSTM / Bidirectional LSTM units used. The Decoder part is also constructed by leveraging an LSTM / Bidirectional LSTM layer. Each unit that belongs to the Decoder part is designed to output a value for each of the future time steps being examined. In order to do so, a Repeat-Vector layer is leveraged.

3.2.2. CNN-LSTM ED

Feature extraction can generate meaningful information for the prediction model, thus allowing it to perform accurate predictions (Khalid, Khalil, & Nasreen, 2014). Time-series problems are not an exception in this regard. In addition, feature extraction is time-consuming, and the various methodologies vastly differ from application to application (Chauhan, Palivela, & Tiwari, 2021; Hira & Gillies, 2015). In recent years, researchers have been using convolution operations for automatic feature extraction (Elmaz, Eyckerman, Casteels, Latré, & Hellinckx, 2021),(Nasir, Khan, & Varlamis, 2021). Convolution Neural Networks (CNN) are not designed to accommodate input in the form of sequences. However, a 1-dimensional CNN layer is capable of receiving input and then learning the salient features. Both CNNs and LSTMs expect a 3-dimensional input. As far as CNNs are concerned, this design characteristic is formulated in order to be able to receive the three distinct Red-Green-Blue channels. LSTMs, on the other hand, require a 3-dimensional input corresponding to a) the number of samples, b) the number of time steps to examine, and c) the number of features. More specifically, two 1-dimensional convolution layers are utilized. The first layer reads the input sequence and projects the result onto feature maps, while the second layer receives the output of the first and performs the same function in order to amplify any salient features. Subsequently, a max-pooling layer is utilized to accumulate features from the maps generated by the previous two layers. In the final step, a flattened layer is employed to reshape the encoder output into the desired shape that the decoder can process. The CNN-LSTM architecture generally involves CNN layers for feature extraction on input data combined with LSTMs to support sequence prediction.

3.2.3. Hybrid LSTM ED

This model (Theodoropoulos, Maroudis, Violos, & Tserpes, 2021) utilizes a bidirectional and an unidirectional LSTM. The input layer is a bidirectional LSTM. Then, a unidirectional LSTM layer is stacked on top of the bidirectional one. The bidirectional layer will provide one hidden state output for each time-step in a 3-dimensional form which is then utilized as input by the unidirectional layer. The model can exploit the temporal correlations present in the various time series in a more sophisticated way than the classic models. In addition, as multiple layers are utilized, the features of the input sequence can be more robustly represented. The same design logic is implemented in the decoder part in order to mirror the encoder morphology. Instead of the basic LSTM model used in the previously explored decoders, the hybrid model utilizes a bidirectional layer stacked on top of a unidirectional layer. This structural symmetry enables the decoder to reconstruct the underlying temporal motifs of the input sequence properly.

3.2.4. Hybrid LSTM attention ED

This architecture (Violos, Theodoropoulos, Maroudis, Leivadeas, & Tserpes, 2022a) also leverages two Self-Attention layers on top of the recurrence-based ones. The first one, located at the Encoder part, receives as input the output produced by the bidirectional LSTM layer. The output of this layer shall be leveraged as input by the unidirectional LSTM layer. The second Attention layer is located at the Decoder part. It receives the output of the unidirectional LSTM layer as input, and its produced output shall be leveraged as input by the bidirectional LSTM layer. This architecture uses Attention mechanisms inside the encoder and the Decoder parts, respectively. Furthermore, in this particular architecture, the Attention layer is leveraged in a manner that aims to enhance the ability of the aforementioned Hybrid bidirectional-unidirectional LSTM structures to encapsulate the various temporal dependencies.

4. Graph neural networks

The modus operandi of the Encoder-Decoder topologies heavily relies on accurately encapsulating the various dependencies found in the input sequence. In order to do so, the different types of encoder entities leverage various feature extraction techniques. Unfortunately, the structural characteristics of the encoders mentioned above prevent them from adequately encapsulating the latent interdependencies created among the various variables that form the input sequence. This inability introduces significant limitations in the case of multivariate forecasting. The multivariate forecasting paradigm assumes that each input variable is correlated with the prediction produced. Subsequently, it is safe to conclude that specific correlations are formed among the variables that form the input sequence. These limitations dictate the need to re-examine the encoder entities' structural characteristics to establish architectures that can exploit these dependencies more efficiently.

Graphs are data structures that hold great expressive power for what concerns the encapsulation of relationships among various entities (de Fernando, Pedronette, de Sousa, Valem, & Guilherme, 2022). During the last decades, the scientific community has witnessed the emergence of Graph Neural Networks. The modus operandi of Graph Neural Networks is based on encapsulating the spatial dependencies among the various nodes that constitute graph-like structures in an advanced manner. According to the Graph Neural Network paradigm, each node's state depends on its neighbours' states. The goal of Graph Neural Networks is to capture this type of spatial dependency. To that end, numerous types of Graph Neural Networks have been proposed. Despite the structural differences that are inherent in the various types of Graph Neural Networks, all of them carry out the same functionality but leverage different methods in order to establish it. The functionality shared across all of them is the encapsulation of a node's abstract representation via the process of passing information from its neighbours to the node itself. This process can be achieved via information propagation, message passing or graph convolution.

4.1. Graph convolutions

Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016) can generalise classical Convolutional Neural Networks (CNN) in a manner that is compliant with graph-structured data. According to the GCN

paradigm, each node extracts feature information from its neighbours and from itself. Each node is assigned a dedicated feature vector. These values are then passed through a mean, average or max function, and the end-product is provided as input to a fully connected neural network. Since the concept of graph convolution is integral to the main concepts explored within the context of this work, it is of paramount importance to analyse the specifics of this concept.

Given an undirected graph G with N nodes and E edges (u_i, u_j) and the following matrices:

- an adjacency matrix $A \in R^{N*N}$: this matrix could be either weighted or binary. It represents the "relations" that are established among the various nodes.
- a degree matrix $D_{i,i} \in R^{N*N}$: this matrix equals $\sum_j A_{i,j}$. Only the diagonal elements of this matrix are non-zero values. Each diagonal element corresponds to a node, and its value signifies how many "relations" this node has.
- a feature matrix $X \in R^{N*C}$: C is the dimension of each feature vector. The feature matrix consists of various feature vectors.

Since A is not designed to contain the "relation" that a node forms with itself, it is necessary to create an updated $\overline{A}$ matrix that entails this feature. To that end, one must add the identity matrix I to the A matrix, thus performing $\overline{A} = A + I$. As a result of this process, it is required to use an updated D matrix that will be referred to as $\overline{D}$. For each node to extract the feature representations of its neighbours, one has to multiply $\overline{A}$ and X. The product is a feature matrix $Y \in R^{N*C}$ that contains the aggregated feature representations. In other words, the aggregated feature representation of each node is the summation of the various feature vectors of its neighbours and its own. Then it is essential to scale Y according to the degrees of the nodes. The next step is to pass the aggregated feature representations through the function as mentioned earlier. In the context of this work, we chose the *average* function, and as such, the same one will be used in this analysis for consistency reasons. In order to produce this function, one has to create the $\overline{D}^{-1/2}$ matrix. Each element in the $\overline{D}^{-1/2}$ matrix is the reciprocal of its corresponding element following the $\overline{D}$ matrix. The scaled version of $\overline{A}$ is referred to as $\hat{A}$, and it equals $\overline{D}^{-1/2} * \overline{A} * \overline{D}^{-1/2}$. $X * \hat{A}$ is the scaled aggregated feature representation. The scaled aggregated feature representation is then multiplied by X, and the result is provided as input to a fully connected neural network, thus producing the final feature representation. The final feature representation is calculated in Eq. 1.

$$FeatureRepresentation = ReLU(\hat{A} * X * W) \tag{1}$$

The GCN paradigm leverages an additional trainable weight matrix that is referred to as W. Furthermore, $ReLU$ corresponds to the Rectified Linear Unit activation function[3].

It is worth mentioning that the number of GCN layers that can be stacked on top of each other corresponds to the number of hops in terms of neighbours that can be performed each time the aggregation process is carried out. For instance, when leveraging a GCN consisting of only one convolution layer, the nodes only have access to their immediate neighbours in terms of aggregating representation information.

4.2. GCN-LSTM

From its conception (Gori, Monfardini, & Scarselli, 2005) until nowadays, there have been many variations to the Graph Neural Network paradigm. The most drastic differentiation that has been recorded among these variations is in regards to whether the Deep Learning methodologies are being applied on static or dynamic graphs. Currently, two paradigms facilitate the use of Deep Learning methodologies on dynamic graphs. The first one is referred to as Discrete-time dynamic graphs (DTDG), and the second one is referred to as Continuous-time dynamic graphs (CTDG) (Rossi et al., 2020). The authors of this paper chose to implement the DTDG approach. According to the DTDG paradigm (Yu, Yin, & Zhu, 2018), a dynamic graph structure can be represented as a sequence of snapshots of a static graph taken during different time intervals. A DL model designed to be implemented on dynamic graphs can be viewed as an Encoder-Decoder. The encoder is a function that formulates mappings of a dynamic graph to node-specific embeddings. The Decoder leverages as input these embeddings in order to produce predictions.

Long short-term memory networks have been successfully used in numerous resource usage prediction endeavours to encapsulate the temporal characteristics of time series data. Unfortunately, the graph structure characteristics inherent in the multivariate time-series format were not considered in these works. In this paper, a Graph Convolution Network layer is embedded in a Long Short Term Memory layer to predict resource usage utilization. This amalgamation of spacial and temporal DL layers will be referred to as GCN-LSTM in the context of this paper. The architecture of the GCN-LSTM model is depicted in Fig. 2. The GCN layer is leveraged in order to extract the structural characteristics of the resource usage graph. This layer produces an intermediate representation product referred to as Feature Representation. The Feature Representation is then leveraged as input by the LSTM layer. The LSTM layer is used to capture the temporal characteristics of resource consumption at the graph snapshot level. Finally, the LSTM layer that acts as a decoder shall provide the desired predictions. This architectural paradigm closely resembles the CNN-LSTM topology explored in a previous section. The only difference is that the ability of the Graph Convolution paradigm to encapsulate the relations between the node variables shall enable richer representations and, consequently, more accurate predictions. Up to this point, there has been only another attempt at establishing multivariate time-series forecasting using Graph Neural Networks (Wu et al., 2020) that leverages a different architectural paradigm. Our approach was chosen to be aligned with the other Encoder-Decoder topologies being examined in this paper's context.

4.3. Problem formulation

One of the main scientific contributions of this work is the formulation of the Cloud and Edge resource usage prediction problem in a manner that is aligned with the format of graph structures. Cloud and Edge infrastructures can be modeled as complex systems that consist of N processing nodes. Each processing node exhibits a resource consumption behavior based on its hardware, as well as the number and characteristics of the incoming tasks that are being offloaded to this specific node for processing. The authors of this paper propose the following problem formulation to facilitate the representation of resource usage in a manner that is compliant with the Graph Neural Network paradigm; According to our proposed representation methodology, each resource usage metric corresponds to a distinct node of a graph. The resource consumption behavior of each processing node may be regarded as a feature matrix and is represented by the set $B = \{b_1, b_2., b_b\}$, where b_b indicates the b^{th} feature, where $1 \leq b \leq B$. In the context of this work, each feature matrix consists of three resource usage metrics examined. These resource usage metrics are the percentage of the server CPU load, the percentage of the server memory usage and the amount of data received and sent over the network. The proposed topological representation is depicted in Fig. 3. As stated in the previous subsection, the authors of this paper chose the Discrete-Time Dynamic Graph approach in regards to representing resource usage in a dynamic manner. At each predefined time-interval a new snapshot of the feature matrix B is being established. B_t refers to the feature matrix that corresponds to the resource consumption that is present at a specific processing node, at a specific timestamp t.

Each timestamp corresponds to a snapshot of a static graph that consists of three nodes. The relations among these metrics are found by formulating the appropriate correlation matrix. This correlation matrix

[3] https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

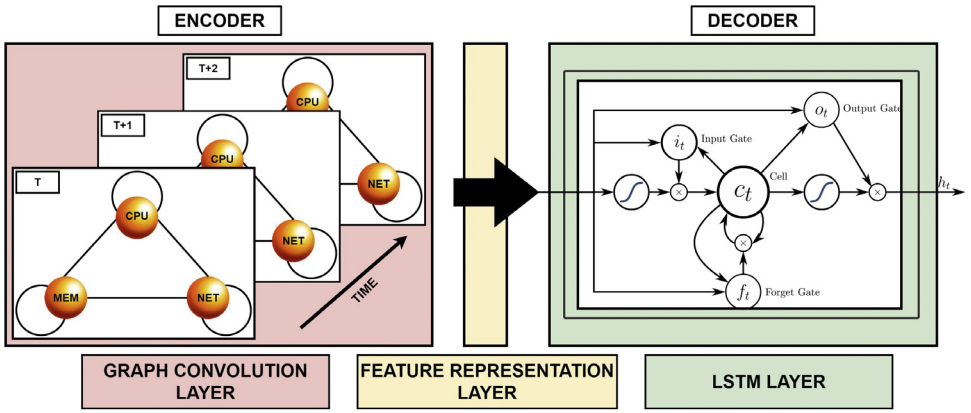

Fig. 2. GCN-LSTM architecture.

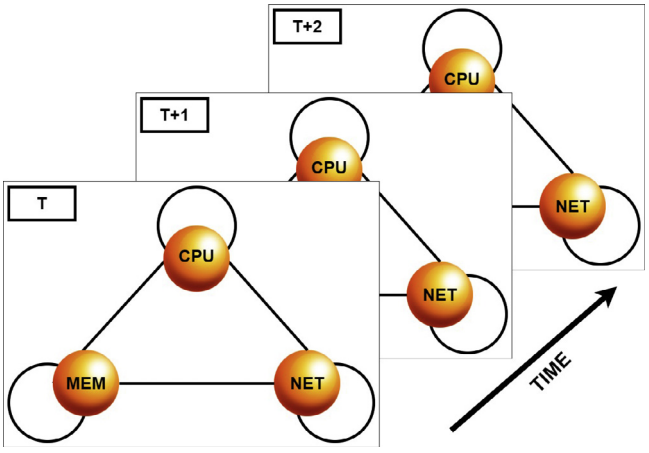

Fig. 3. Resource Usage Representation using Graph Neural Networks.

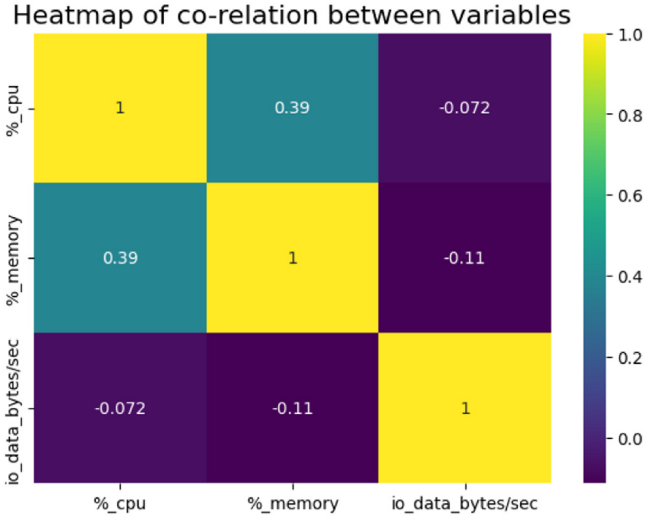

Fig. 4. Co-relation matrix between the three resource usage metrics.

is then used to create the adjacency matrix A that was mentioned previously in this section. The process of constructing the adjacency matrix A, by using a correlation matrix and a threshold value, is described in Algorithm 1. The correlation matrix was created using historical time-series data. The specifics of this data will be explored later in the Experimental Evaluation Section. The correlation matrix between the three resource usage metrics is depicted in Fig. 4. As one can see in this figure, there is a significant correlation between the memory and CPU variables, while the amount of data received and sent over the network does not seem to affect the rest of the resource usage metrics. The adjacency matrix was constructed in a way that reflects these relations. Each input consists of i B_t sets, where i corresponds to the length of the input sequence. Each input is represented by the set $I = \{i_1, i_2., i_i\}$, where i_i indicates the i^{th} time-step of the input sequence, where $1 \leq i \leq I$. The challenge we address is to predict future CPU values based on previously recorded CPU, Memory and Network values. Each produced CPU prediction can be regarded as a set $P = \{p_1, p_2, p_p\}$, where p_p indicates the prediction that corresponds to the p^{th} time-step, where $1 \leq p \leq p$. Encoder-Decoder topologies are ideal for implementing sequence-to-sequence modeling. In the context of this work, length of the input sequence is equal to i time-steps and the length of the output sequence is equal to o time-steps. The GNN-based Encoder-Decoder model is based on the concept of receiving as input the last i resource consumption behaviour values and producing an output that corresponds to the next o anticipated CPU values. Furthermore, I_t and O_t refer to the input and output that correspond to a specific processing node, at a specific timestamp t. This model requires a training process based on a set T, where $T = \{(I_1, O_1), (I_2, O_2), \ldots, (I_t, O_t))\}$ consists of pairs of resource consumption behavior inputs and their perspective CPU predictions outputs. With the training process we can find a function $f(r) : R^U \rightarrow T$ that can predict the next o CPU values. The aforementioned function is the proposed GCN-LSTM model.

5. Use case: Multiplayer mobile gaming

While the multiplayer games market relies merely on PC and console games, it still struggles to have a significant presence in the mobile gaming market. Multiplayer experiences on mobile devices are usually simplified to turn-based gameplay types or even faked (like in.io types of games). The mobile gaming market slowly leans towards maturity. In a few years, game developers will be able to provide a solid user experience for the players attributed to Mobile technologies: mobile devices, and mobile connectivity, as these have recently reached a level that allows mobile game developers to explore multiplayer types of games. New multiplayer solutions, like multiplayer frameworks and game server hosting solutions, are created. Nowadays there are several significant ready-to-use solutions on the market, the most important are Google Firebase[4], AWS GameLift[5] and Unity Gaming Services[6]. These solutions are relatively new on the market, but they already offer a wide range of services for building multiplayer games. Considering the effort of the companies in providing these solutions, it can be assumed that they will soon become a solid competition in the market.

[4] https://firebase.google.com
[5] https://aws.amazon.com/gamelift
[6] https://unity.com/solutions/gaming-services

In casual and hyper-casual games, achieving a positive Return On Investment (ROI) is very hard since the margin between the user acquisition cost and per-user income is minimal. Compensating that with an investment on a pretty expensive backend server only increases the difficulty. Developers tend to create game systems where one of the client's devices also plays the role of the game host. This solution limits the number of concurrent users as it comes at the cost of the computation performance of a single device. Unlike PC or console games, mobile gaming players are incredibly demanding; they expect the applications and games they use to run immediately, without delay or waiting for launch. Mobile games are very often launched only for a short period of time, on the way to work or school, during breaks and generally in between activities. Therefore, the time of launching a mobile game and the delays in its operation become much more critical than in the case of PC or console games. The market is still relatively young, and there are no go-to solutions for all these issues. Thus, backend service providers and game developers are further exploring solutions to overcome these. In the case of real-time mobile multiplayer games, network communication is the critical element of the whole game system. It is crucial to provide minimal server response time so the player will have an impression of uninterrupted gameplay. This will allow the user to immerse into the game's world. Additionally, the game system must be able to handle multiple users while ensuring complete and rapid synchronization of all game entities, environment elements and players.

In an attempt to achieve low response times and a seemingly uninterrupted game experience, several techniques could be exploited, such as the prediction of resource utilization, e.g., the user is expected to demand more resources (CPU, RAM requirements) and the system should proactively be able to deliver. A high-accuracy resource prediction algorithm means that the system can effectively know the demands of a user beforehand, thus allowing for more resources to be used by the user at a specific time. Therefore, the synchronization of resources per user is optimized, and the users have the impression that minimal or no delays occur while playing the game. As such, multiplayer gaming is better synchronized, and the system can handle more users per game session.

Horizontal Autoscaling refers to the process that enables Cloud & Edge infrastructures to be scalable in terms of providing resources on demand in order to ensure the QoS objectives. In other words, Horizontal Autoscaling enables the underlying infrastructure to operate without the occurrence of phenomena such as under-provisioning or over-provisioning of resources. Horizontal Autoscaling is an integral part of the resource orchestration process, in terms of adding or removing processing nodes in response to changes that occur in the context of various monitored metrics. Processing nodes can be physical machines, virtual machines, containers or pods. The Horizontal Autoscaling process is based on the use of rules to add or remove the required processing nodes in accordance with appropriate thresholds following the replication methodology. The scaling decisions can be conducted in a reactive or a proactive manner. In this work, we experimentally compare a proactive approach that leverages the proposed GCN-LSTM model against a reactive one, since reactive approaches are the most widespread Autoscaling methods in the Cloud & Edge computing domains. Reactive Horizontal Autoscaling approaches are designed to add processing nodes if a current metric exceeds a certain threshold value. These threshold values correspond to resource usage metrics such as CPU, Memory, or Network or workload related metrics such as the number of application requests. In this work we chose to use the CPU, since it is the most widely used metric in the context of contemporary orchestration framework, such as Kubernetes.

Proactive Horizontal Autoscaling approaches are designed to respond to metric predictions that are provided by dedicated forecasting models. These changes have a direct impact on the required utilization of various computational resources. By closely monitoring the current resource utilization on the available processing nodes, it is possible to form accurate estimation regarding the computational burden that is expected to take place during the next time-steps. Based on these predictions, the burden can be alleviated in a proactive manner by allocating additional computational resources. It is of paramount importance to highlight the fact that the Virtual Machine or pod replication process requires a start up time that may span from few seconds to few minutes, depending on factors such as the virtualization mechanisms, the application and the underlying infrastructure. Thus, the aim of proactive approaches is to surpass the reactive ones, by reducing or even avoiding the delays that are associated with start-up times and the subsequent QoS deterioration.

6. Experimental evaluation

6.1. Model implementation, frameworks and dataset

All the examined DL models are implemented in Python 3 using NumPy, pandas, statistics, Scikit-learn, SciPy, Scikit-Optimize, TensorFlow 2 and its higher-level API Keras. The environment we used is the Jupyter notebook. In order to examine the efficiency of the proposed solution, 2 datasets and 1 large-scale simulation were used. The first dataset that was used to train and evaluate the various DL models was provided by Orbital Knight[7]. Orbital Knight is a Polish independent game studio focused on making high-quality mobile games and provides several metrics for resource usage prediction in multiplayer mobile gaming. The ORBK game system consists of three elements: Game Server, Mobile Application/game and Game Servers Status DB, as illustrated in Fig. 5.

In order to gather the resource utilization metrics, Orbital Knight built a testbed, in which 32 simulated players (bots) were simulated on a separate machine and connected to the game server. Data collection ran continuously for around 4.5 hours, with 2 seconds intervals. The game server has been equipped with an additional dedicated module that allows the collecting of the necessary data:

- timestamp of each data sample (date-time)
- the current percentage of the server CPU load (%)
- the current percentage of the server memory usage (%)
- the amount of data received and sent over the network (bytes/sec)

Data collection begins as soon as the client app connects to the game server. Peers are pinged every 2 seconds and the received values are recorded. After each 2 minute time interval the collected ping data are sent to the online database and the 2 second pinging process continues. This way one can easily gather a large number of samples without the threat of data overflow in the client application. In addition, it is also guaranteed that the data sent during each single run is small enough to continue the measurement without any delay. In order to establish maximum possible reliability the LiteNetLib[8] library was utilized as it is proven to deliver valid and stable results. Furthermore, AWS DynamoDB[9] was used as a database system to gather and store measurement results as it also provides stable and reliable services.

In the context of the experimental process, the authors of this paper chose to implement a multi-step CPU usage scenario. The prediction models receive the three latter resource usage metrics corresponding to the last ten-time steps and can provide predictions corresponding to the next ten time steps. Each time-step has a duration of 2 seconds. The use of such short time-intervals between predictions was examined in order to evaluate the proposed solution's ability to detect sudden bursts in service demand, in the context of CPU prediction. However, aside from the aforementioned sudden bursts, service demand is also influenced by periodic phenomena that may manifest across several days. In order to examine the efficiency of the proposed approach in the context of both types of phenomena, the authors of this paper perform a

[7] https://www.orbitalknight.com/
[8] https://github.com/RevenantX/LiteNetLib
[9] https://aws.amazon.com/dynamodb/

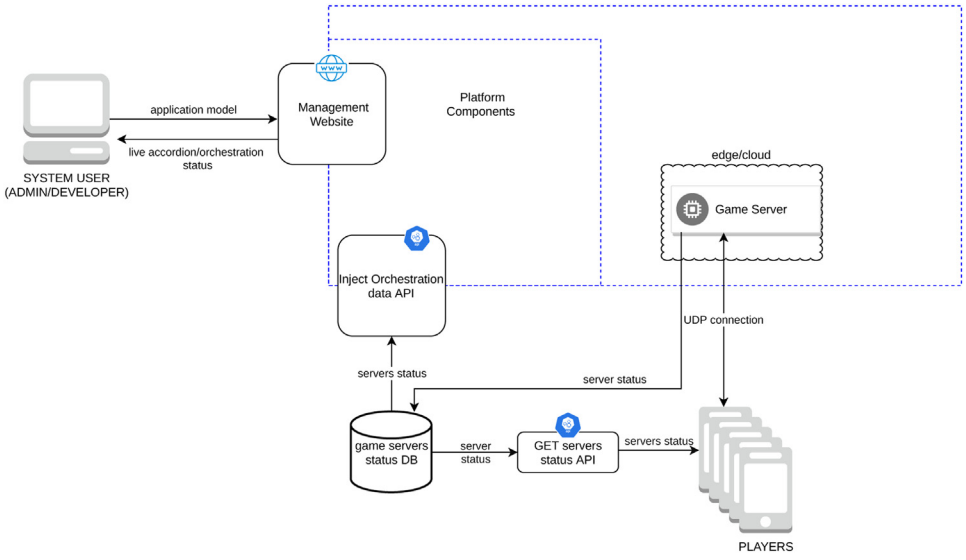

Fig. 5. ORBK game system.

large-scale experimental evaluation in a simulated cloud computing environment using CloudSim Plus[10] framework. The duration of the simulation reached the 4-day mark in order to facilitate the periodic service demand phenomena that span across multiple days. Furthermore, it included more than 1,600,000 tasks that were generated and offloaded to the available processing nodes. The tasks were generated by a mixture of Poisson probability distributions and various statistical properties that correspond to the task production rate that is present in the dataset that was provided by Orbital Knight. On top of that, in order to optimally represent the characteristics of service demand inherent in Multiplayer Mobile Gaming, application requests fluctuate throughout the day until they start to significantly increase right after the end of typical working hours, at 17:00 pm. Task production rates reach their peak at around 10:00 pm and then they start to gradually decrease.

The simulation begins with a singular available processing node and there are 9 more backup nodes for potential replication. During each moment of the simulation, numerous tasks are being produced. These tasks are then assigned to the corresponding processing nodes. The selection process in regards to which task will be sent to which computational node is being made based on the task scheduling algorithm that is being leveraged. In this work we are considering a standard Round Robin task scheduling approach. The resource usage metrics that correspond to each processing node are being collected in order to formulate the aforementioned predictions. Depending on these predictions, a corresponding signal will be sent to the broker entity of the CloudSim Plus framework. The broker serves as an orchestration mechanism that is able to allocate or de-allocate resources depending on the decision scaling paradigm that is being deployed. Each time the broker entity decides to allocate additional computational resources, there is a certain timeframe that is required for the new processing nodes to be deployed. Processing nodes can be either Virtual Machines or Kubernetes pods. Since pods are far more lightweight, they tend to be able to deployed in a far more timely manner. As a matter of fact, pods usually take seconds to be deployed, while the deployment process of Virtual Machines may take up to several minutes. It is rather apparent that the longer the deployment times are, the more drastic the effect of leveraging a proactive approach shall be, since potential task execution overheads shall be greater since it would take longer for the infrastructure to react to sudden bursts in resource demand. Nevertheless, we chose to simulate a Kubernetes pod deployment scenario, since the Multiplayer Mobile Gaming industry has already widely adopted this deployment paradigm and it shall be able to provide an adequate evaluation process in order to showcase the efficiency of the proposed approach. To that end, the deployment time for new processing nodes is set to the 3 second mark, in order to correspond to the actual time that is required to deploy a new pod that facilitates ORBK's component. Finally, there are two Horizontal Autoscaling strategies that are being examined in the context of the simulation process. The first one is a reactive Horizontal Autoscaling approach is compared against a Proactive one that leverages the proposed GCN-LSTM model in order to construct accurate CPU predictions based on which the scaling decisions are established.

The reactive approach included a decision process that takes place every 60 seconds, based on whether the infrastructure should allocate additional processing nodes or release some already allocated nodes or continue with the same topology. This decision is being made in a reactive manner and is based on its average CPU utilization recorded during the last minute. The main objective is to ensure that each processing node operates in the 40%-70% CPU usage zone in order to avoid under-provisioning and over-provisioning of resources. If the current CPU utilization exceeds the 70% upper threshold, the broker entity in CloudSim Plus decides to allocate additional nodes. If the predicted value is below 40%, the scaling mechanism decides to release the under-utilized nodes, after all of its running tasks have been completed. Because the scaling decisions take place after the resource metrics exceed the threshold, there will be a significant delay in the deployment of the newly allocated processing nodes.

The incorporation of a proactive Horizontal Autoscaling approach in our experiments requires the integration of the proposed GCN-LSTM model in the CloudSim Plus framework. In the same way as with the reactive approach, once every 60 seconds information regarding resource consumption is gathered. Furthermore, similarly to the reactive approach, the objective of this approach is to maintain the CPU utilization in the 40%-70% zone. The fundamental difference is that the proactive approach utilizes the predicted CPU values in order to proactively make the scaling decisions, contrary to the reactive approach that utilizes the ongoing CPU metrics. The GCN-LSTM model receives as input the last 10 resource consumption behavior metrics and produces a prediction that describes the CPU consumption that is expected to take place during the next 10 time-steps. Similarly to before, these resource consumption metrics include the CPU, Memory and Network demand. Contrary to the 2 second time-steps that are leveraged in the context of the resource usage prediction experiments, during the Horizontal Autoscaling experiments we chose to use time-steps that are 60 seconds

[10] https://cloudsimplus.org/

Table 2
Experimental results in terms of CPU predictions.

	RMSE	MAE
LSTM	6.007	5.049
BD-LSTM	6.735	5.300
GRU	6.507	5.129
LSTM ED	6.239	4.908
BD-LSTM ED	6.157	4.964
HYBRID LSTM ED	6.040	4.755
HYBRID LSTM ATT ED	6.134	4.952
CNN-LSTM ED	6.729	5.288
GCN-LSTM ED	**5.922**	**4.665**

long, in order to explore the long-term temporal dependencies that are present in resource demand.

6.2. Experimental results for resource usage prediction

In order to evaluate the performance of the proposed model, we used the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) metrics since they are considered the most notable metrics in evaluating time series forecasting and ML regression models. The MAE measures the average absolute deviation between the target and predicted values. Thus, it provides an insight into the magnitude of overall error that occurred because of the forecasting process, by squaring the prediction errors and then averaging the squares, the RMSE is produced. The RMSE expresses the standard deviation of the errors emphasizing the spread out of the errors, and subsequently penalizes extreme errors that occur during the forecasting process. MAE is used in cases where all the errors bear the same gravitas, while RMSE is used when the penalization of significant errors should be more severe, regardless of their frequency of occurrence. The results in terms of RMSE and MAE correspond to the average of the ten time-step related predictions. In our experiments, the current percentage of the server CPU load and the current percentage of the server memory usage were represented in percentages and the amount of data received and sent over the network in bytes/sec. Finally, these values were scaled accordingly in order for their impact to be represented in a balanced manner. The experimental results corresponding to CPU predictions regarding RMSE and MAE are depicted in Table 2. As one can see, the proposed approach produced superior results in terms of RMSE and MAE compared to the other DL models.

6.3. Experimental results for proactive horizontal autoscaling

In the previous subsection, the superiority of the proposed GCN-LSTM approach, in terms of resource usage prediction accuracy, was showcased. In this subsection, we shall continue with the evaluation outcomes of a proactive Horizontal Autoscaling approach that incorporate the aforementioned prediction model in order to conduct more efficient autoscaling decisions. To that end, we provide a comparison between the proactive approach and a reactive one. This comparison is evaluated in the context of different versions of the execution time and number of resources. The outcomes of this comparison are presented in Table 3. The execution time has been evaluated through different statistical measurements. Average Execution time (Avg. Ex. time) declares the mean time for all the tasks of the experiment. Median Ex. time refers to the middle values. Standard deviation declares how much the execution times of the tasks differ from the average value. Maximum (Max.) and Minimum (Min.) Execution time (Ex. time) refer to the maximum and minimum recorded task execution times. Two additional statistical measures are the skewness and kurtosis. Skewness indicates the symmetry of the values in execution time and a right skewed distributions is better than a left. Kurtosis indicates if the distribution of the time values is heavy-tailed or light-tailed. A rather significant evaluation metric is the tail latency. Tail latency corresponds to the 98th percentile and represents the 2% longest response times. It is a metric of paramount importance since the longest response times affect QoS in a significant way. The units in the different types of execution times and latency in the following tables are in seconds. In this subsection, we also comparatively explore the metric of active nodes which refers to the number of worker nodes that are active. This metric is intertwined with the cost of using the corresponding resources.

Incorporating the GCN-LSTM model into the Horizontal Autoscaling strategy makes it easier to speed up all tasks in general. This claim is supported by the results across all explored types of execution times. The noticeable improvement in task execution times means that tasks can be processed faster in general, regardless of their inherent computational needs. On top of that, the significantly lower standard deviation showcases that the proposed approach provides consistency in terms of execution time, thus supporting our claim regarding the need to have fewer outliers in terms of delayed tasks. This establishes a sense of determinism to the infrastructure and enhances the ability of being in control of the involved computational resources. Furthermore, the tail latency metric also highlights the efficiency of the proposed solution, since it is improved by a factor of 12%. That implies a noticeable improvement in the context of the computationally intensive tasks, which can really affect end-user experience. This claim is also supported by the impressive improvement in terms of the maximum execution time of the methods tested, where the proposed solution provides an improvement of over 38%.

Both reactive and proactive approaches have positive kurtosis. This means that both response time distributions present peaks with the reactive approach to have more peaks compared to the proactive. In addition, they both have positive skewness with the reactive approach to have a longer right tail than the proactive one. These facts show that the proactive method renders the workload execution more predictable and easily controlled by the task offloading and resource allocation mechanisms. This conclusion is also supported by the results that are presented in Fig. 6 and comparatively evaluate the Task Execution Time across the

Algorithm 1 Adjacency Matrix Constructor Algorithm.

$AdjacencyMatrixConstructor(E, ResourceMatrix)$
Edge List is a list of edges where each edge corresponds to two types of computational resources. These edges correspond to all potential pairs of computational resources and thus the size of this list is equal to K*K. K is defined as the number of different computational resources that are being examined
Resource Matrix is a |K| x |T| matrix where T is defined as the number of timesteps that the available dataset consists of.
Description
The Adjacency Matrix Constructor Algorithm generates a binary value for each edge of the provided edge list. The values of the are calculated based on the results a co-relation function that are then compared against the 0.5 threshold.
Definition - Resource Matrix
$X, X \epsilon R^{|K|*|T|} \leftarrow Resource\_Matrix$
Begin
1. A: Adjacency Matrix
2. For Each $e_i \epsilon E$ do:
3. $r_j, r_k \leftarrow e_i$
4. $v_j \leftarrow X_j$ #Matrix index refers to the column of the Resource Matrix that is attributed to Resource j
5. $v_k \leftarrow X_k$ #Matrix index refers to the column of the Resource Matrix that is attributed to Resource k
6. $A_{jk} \leftarrow corelation(v_j, v_k)$
7. $if\, A_{jk} > 0.5, then : A_{jk} \leftarrow 1, else : A_{jk} \leftarrow 0$
8. End For
End

Table 3
Experimental Evaluation of Horizontal Autocaling Approaches.

Autoscaling Method	Tail latency	Avg. Ex. time	Std. Ex. time	Median Ex. time	Num. Tasks	Max. Ex. time	Skewness Ex. time	Kurtosis Ex. time
Reactive	5.610	1.767	1.944	1.539	1,624,817	47.849	9.385	129.665
Intelligent	**5.060**	**1.525**	**1.123**	**1.260**	1,624,735	**18.909**	**3.513**	**20.166**

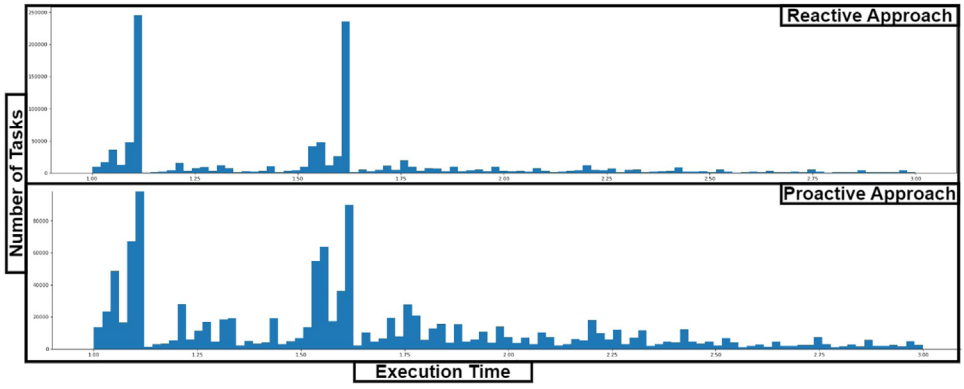

Fig. 6. Comparative analysis of Task Execution Times.

two Horizontal Autoscaling approaches. Finally, we can arrive at this same conclusion by using the average value and the standard deviation of execution time as well.

Finally, the proposed model leveraged less than 0.7% more computational resources throughout the simulation when compared against the standard reactive approach.

6.4. Discussion regarding the experimental results

The fact that the GCN-LSTM model leverages relation-based representations allows it to better encapsulate the dependencies that are formed between the variables of the input sequence when compared to the other models. This claim is supported by the fact that the GCN-LSTM model produced superior results in terms of RMSE and MAE compared to the other DL models. This is especially interesting considering that the GCN-LSTM model shares many structural similarities with the CNN-LSTM model, which produced the worst results out of all the Encoder-Decoder models. This proves that the representation capabilities of graph convolution are significantly greater than those provided by typical convolution when the representation process is performed in the context of non-structure entities. Finally, even the rest of the Encoder-Decoder models, despite being more capable of encapsulating temporal relations since they deploy a greater number of recurrent neural network layers, did not manage to surpass the overall predictive prowess of the GCN-LSTM model.

The proposed GCN-LSTM model is capable of providing accurate predictions regarding the CPU utilization of each processing node that is expected to take place during the next time-steps. These predictions are leveraged by the broker entity that performs the scaling decisions. The small errors in MAE and MSE evaluation metrics mean that the model is able to contribute towards making optimal scaling up decisions when it is necessary in order to avoid performance bottlenecks and the potential degradation of QoS. Furthermore, the model is able to accurately predict potential drops in resource demand and to subsequently release the required resources in order to avoid the resource over-provisioning phenomenon. This results to significantly reduced resource consumption compared to the other intelligent resource allocation approaches that were examined in the Related Work section (2) of this work.

The average number of utilized processing nodes throughout the simulation serves as an important indicator towards examining how accurate the proposed model is in terms of predicting potential decreases in resource demand. As stated in the Related Work section (2), it is a rather common practise for resource usage prediction models to overestimate on purpose future resource demand in order to allocate an abundance of computational resources and thus to facilitate the QoS requirements in a safer manner. The point of constructing a more accurate prediction model is significantly enhance the aforementioned performance metrics while leveraging a similar volume of computational resources. Towards this goal, we consider the proposed model a success since it leverages less than 0.7% more computational resources throughout the simulation when compared against the standard reactive approach. As a matter of fact, even this insignificant increase may be attributed to the fact that the proactive approach scales up earlier than the other approach, which results in more resources being used on average, albeit slightly.

Up until now, the various attempts at predicting resource usage have been mainly focused on capturing the temporal patterns that are inherent in the input sequences. These results highlight the importance of encapsulating multi-variate input sequences in a manner that is capable of capturing both the temporal and the structural relations that are present. The findings that derive from this work aim at expanding the body of literature that explores resource usage representation & prediction. Hopefully, this work shall serve as a stepping stone towards broadening this rather important field of study that has attracted the attention of academia & industry alike. Due to that interest, the authors of this work decided to focus their efforts towards developing a solution that caters to the intricacies of real world applications. To that end, the proposed approach was developed using real-world use-case data harvested from a contemporary MMG application.

7. Conclusions

In this paper, we proposed the use of Graph Neural Networks in the field of resource usage representations. This novel approach was further expanded upon by incorporating it into a resource usage prediction paradigm based on Graph-Based Encoder-Decoder. Furthermore, we compared numerous DL models in the context of resource usage prediction. The examined GCN-LSTM Encoder-Decoder architecture surpassed all other DL approaches in terms of RMSE and MAE. Our next steps are to find ways to enhance the representation ability of the GCN-LSTM model in terms of being able to encapsulate the temporal dependencies more robustly to provide even better results. The proposed model was then incorporated in a proactive Horizontal Autoscaling solution. This approach managed to significantly outperform a standard reactive Horizontal Autoscaling one, across a plethora of in the context of a large-scale simulation, in terms of a plethora of performance met-

rics, while keeping the volume of the required computational resources to a minimum.

CRediT authorship contribution statement

Theodoros Theodoropoulos: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Antonios Makris:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Ioannis Kontopoulos:** Conceptualization, Investigation, Writing – original draft, Writing – review & editing. **John Violos:** Investigation, Writing – review & editing. **Przemysław Tarkowski:** Data curation, Writing – original draft. **Zbyszek Ledwoń:** Data curation, Writing – original draft. **Patrizio Dazzi:** Conceptualization, Writing – original draft, Writing – review & editing, Supervision. **Konstantinos Tserpes:** Conceptualization, Writing – original draft, Writing – review & editing, Supervision.

References

Al-Sulaiman, T. (2022). Predicting reactions to anomalies in stock movements using a feed-forward deep learning network. *International Journal of Information Management Data Insights, 2*(1), 100071.

Basiri, M., & Rasoolzadegan, A. (2018). Delay-aware resource provisioning for cost-efficient cloud gaming. *IEEE Transactions on Circuits and Systems for Video Technology, 28*(4), 972–983. 10.1109/TCSVT.2016.2632121.

Boos, K., Chu, D., & Cuervo, E. (2016). Demo: Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *Proceedings of the 14th annual international conference on mobile systems, applications, and services companion*. In *MobiSys '16 Companion* (p. 94). New York, NY, USA: Association for Computing Machinery. 10.1145/2938559.2938583.

Brusch, I. (2022). Identification of travel styles by learning from consumer-generated images in online travel communities. *Information & Management, 59*(6), 103682.

Cao, X., Zhong, Y., Zhou, Y., Wang, J., Zhu, C., & Zhang, W. (2018). Interactive temporal recurrent convolution network for traffic prediction in data centers. *IEEE Access, 6*, 5276–5289. 10.1109/ACCESS.2017.2787696.

Chauhan, T., & Palivela, H. (2021). Optimization and improvement of fake news detection using deep learning approaches for societal benefit. *International Journal of Information Management Data Insights, 1*(2), 100051.

Chauhan, T., Palivela, H., & Tiwari, S. (2021). Optimization and fine-tuning of DenseNet model for classification of COVID-19 cases in medical imaging. *International Journal of Information Management Data Insights, 1*(2), 100020.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chondrodima, E., Georgiou, H., Pelekis, N., & Theodoridis, Y. (2022). Particle swarm optimization and RBF neural networks for public transport arrival time prediction using GTFs data. *International Journal of Information Management Data Insights, 2*(2), 100086.

Duc, T. L., Leiva, R. G., Casari, P., & Östberg, P.-O. (2019). Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. *ACM computing surveys, 52*(5). 10.1145/3341145.

Elmaz, F., Eyckerman, R., Casteels, W., Latré, S., & Hellinckx, P. (2021). Cnn-lstm architecture for predictive indoor temperature modeling. *Building and environment, 206*, 108327.

Ensafi, Y., Amin, S. H., Zhang, G., & Shah, B. (2022a). Time-series forecasting of seasonal items sales using machine learning – a comparative analysis. *International Journal of Information Management Data Insights, 2*(1), 100058.

Ensafi, Y., Amin, S. H., Zhang, G., & Shah, B. (2022b). Time-series forecasting of seasonal items sales using machine learning–a comparative analysis. *International Journal of Information Management Data Insights, 2*(1), 100058.

Eramo, V., Catena, T., Lavacca, F., & di Giorgio, F. (2020). Study and investigation of sarima-based traffic prediction models for the resource allocation in nfv networks with elastic optical interconnection. In *2020 22nd international conference on transparent optical networks (icton)* (pp. 1–4). 10.1109/ICTON51198.2020.9203070.

de Fernando, F. A., Pedronette, D. C. G., de Sousa, G. J., Valem, L. P., & Guilherme, I. R. (2022). Rade+: A semantic rank-based graph embedding algorithm. *International Journal of Information Management Data Insights, 2*(1), 100078.

Fujimoto, Y., Fujita, M., & Hayashi, Y. (2021). Deep reservoir architecture for short-term residential load forecasting: An online learning scheme for edge computing. *Applied energy, 298*, 117176. 10.1016/j.apenergy.2021.117176.

Gellert, A., Florea, A., Fiore, U., Palmieri, F., & Zanetti, P. (2019). A study on forecasting electricity production and consumption in smart cities and factories. *International journal of information management, 49*, 546–556.

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 ieee international joint conference on neural networks, 2005.: vol. 2* (pp. 729–734 vol. 2). 10.1109/IJCNN.2005.1555942.

Hao, Z., Novak, E., Yi, S., & Li, Q. (2017). Challenges and software architecture for fog computing. *IEEE internet computing, 21*(2), 44–53.

Hira, Z. M., & Gillies, D. F. (2015). A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics, 2015*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735–1780.

Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing–a key technology towards 5g. *ETSI white paper, 11*(11), 1–16.

Janardhanan, D., & Barrett, E. (2017). CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. In *2017 12th international conference for internet technology and secured transactions (ICITST)* (pp. 55–60). 10.23919/ICITST.2017.8356346.

Khalid, S., Khalil, T., & Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference* (pp. 372–378). IEEE.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. 10.48550/ARXIV.1609.02907

Li, Y., Shan, C., Chen, R., Tang, X., Cai, W., Tang, S., … Zhang, Y. (2019). Gaugur: Quantifying performance interference of colocated games for improving resource utilization in cloud gaming. In *Proceedings of the 28th international symposium on high-performance parallel and distributed computing*. In *HPDC '19* (pp. 231–242). New York, NY, USA: Association for Computing Machinery. 10.1145/3307681.3325409.

Liu, B., Guo, J., Li, C., & Luo, Y. (2020a). Workload forecasting based elastic resource management in edge cloud. *Computers & Industrial Engineering, 139*, 106136. 10.1016/j.cie.2019.106136.

Liu, R., Mai, F., Shan, Z., & Wu, Y. (2020b). Predicting shareholder litigation on insider trading from financial text: An interpretable deep learning approach. *Information & Management, 57*(8), 103387.

Lukoševičius, M. (2012). In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *A practical guide to applying echo state networks* (pp. 659–686)). Springer Berlin Heidelberg.

Makris, A., Boudi, A., Coppola, M., Cordeiro, L., Corsini, M., Dazzi, P., … Herzog, U. (2021a). Cloud for holography and augmented reality. In *2021 IEEE 10th international conference on cloud networking (CloudNet)* (pp. 118–126). 10.1109/CloudNet53349.2021.9657125.

Makris, A., Boudi, A., Coppola, M., Cordeiro, L., Corsini, M., Dazzi, P., … Pateraki, M., et al., (2021b). Cloud for holography and augmented reality. In *2021 IEEE 10th international conference on cloud networking (CloudNet)* (pp. 118–126). IEEE.

Makris, A., Psomakelis, E., Theodoropoulos, T., & Tserpes, K. (2022). Towards a distributed storage framework for edge computing infrastructures. In *Proceedings of the 2nd workshop on flexible resource and application management on the edge* (pp. 9–14).

Nasir, J. A., Khan, O. S., & Varlamis, I. (2021). Fake news detection: A hybrid CNN-RNN based deep learning approach. *International Journal of Information Management Data Insights, 1*(1), 100007.

Nguyen, H., Tran, K. P., Thomassey, S., & Hamad, M. (2021). Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International journal of information management, 57*, 102282.

Nisar, F., & Ahmed, B. (2020). Resource Utilization in Data Center by Applying ARIMA Approach. In I. S. Bajwa, T. Sibalija, & D. N. A. Jawawi (Eds.), *Intelligent Technologies and Applications*. *Communications in Computer and Information Science*. Singapore: Springer. 10.1007/978-981-15-5232-8_64.

Nolle, T., Luettgen, S., Seeliger, A., & Mühlhäuser, M. (2022). Binet: Multi-perspective business process anomaly classification. *Information systems, 103*, 101458.

Parviero, R., Hellton, K. H., Haug, O., Engø-Monsen, K., Rognebakke, H., Canright, G., … Scheel, I. (2022). An agent-based model with social interactions for scalable probabilistic prediction of performance of a new product. *International Journal of Information Management Data Insights, 2*(2), 100127.

Patel, M., Naughton, B., Chan, C., Sprecher, N., Abeta, S., Neal, A., et al., (2014). Mobile-edge computing introductory technical white paper. *White paper, mobile-edge computing (MEC) industry initiative, 29*, 854–864.

Patterson, J., & Gibson, A. (2017). *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.".

Pesala, V., Paul, T., Ueno, K., Praneeth Bugata, H., & Kesarwani, A. (2021). Incremental learning vector auto regression for forecasting with edge devices. In *2021 20th IEEE international conference on machine learning and applications (icmla)* (pp. 1153–1159). 10.1109/ICMLA52953.2021.00188.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. 10.48550/ARXIV.2006.10637

Roy, N., Dubey, A., & Gokhale, A. (2011). Efficient autoscaling in the cloud using predictive models for workload forecasting. In *2011 IEEE 4th international conference on cloud computing* (pp. 500–507). IEEE.

Sabella, D., Alleman, A., Liao, E., Filippou, M., Ding, Z., Baltar, L. G., … Schatzberg, G., et al., (2019). Edge computing: From standard to actual infrastructure deployment and software development. *ETSI White paper*, 1–41.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer, 50*(1), 30–39.

Serhani, M., El-Kassabi, H., Shuaib, K., Nujum, R., Benatallah, B., & Beheshti, A. (2020). Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven iot workflows. *Future Generation Computer Systems, 108*. 10.1016/j.future.2020.02.066.

Sharif, Z., Jung, L. T., Razzak, I., & Alazab, M. (2021). Adaptive and priority-based resource allocation for efficient resources utilization in mobile edge computing. *IEEE Internet of Things Journal*. 10.1109/JIOT.2021.3111838. 1–1

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia computer science, 131*, 895–903. 10.1016/j.procs.2018.04.298.

Shiva Prakash, B., Sanjeev, K. V., Prakash, R., & Chandrasekaran, K. (2019). A Survey on Recurrent Neural Network Architectures for Sequential Learning. In J. C. Bansal, K. N. Das, A. Nagar, K. Deep, & A. K. Ojha (Eds.), *Soft Computing for Problem Solving*. In *Advances in Intelligent Systems and Computing* (pp. 57–66). Singapore: Springer. 10.1007/978-981-13-1595-4_5.

Sridevi, G., & Suganthi, S. K. (2022). Ai based suitability measurement and prediction between job description and job seeker profiles. *International Journal of Information Management Data Insights, 2*(2), 100109.

Taleb, T., Boudi, A., Rosa, L., Cordeiro, L., Theodoropoulos, T., Tserpes, K., … Li, R. (2022). Towards supporting xr services: Architecture and enablers. *IEEE Internet of Things Journal*. 10.1109/JIOT.2022.3222103. 1–1

Tameswar, K., Suddul, G., & Dookhitram, K. (2022). A hybrid deep learning approach with genetic and coral reefs metaheuristics for enhanced defect detection in software. *International Journal of Information Management Data Insights, 2*(2), 100105.

Theodoropoulos, T., Makris, A., Boudi, A., Taleb, T., Herzog, U., Rosa, L., … Romussi, A., et al., (2022a). Cloud-based XR services: A survey on relevant challenges and enabling technologies. *Journal of Networking and Network Applications, 2*.

Theodoropoulos, T., Makris, A., Violos, J., & Tserpes, K. (2022b). An automated pipeline for advanced fault tolerance in edge computing infrastructures. In *Proceedings of the 2nd workshop on flexible resource and application management on the edge*. New York, NY, USA: ACM.

Theodoropoulos, T., Maroudis, A.-C., Violos, J., & Tserpes, K. (2021). An encoder-decoder deep learning approach for multistep service traffic prediction. In *2021 IEEE seventh international conference on big data computing service and applications (bigdataservice)* (pp. 33–40). IEEE.

Theodoropoulos, T., Violos, J., Tsanakas, S., Leivadeas, A., Tserpes, K., & Varvarigou, T. (2022c). Intelligent proactive fault tolerance at the edge through resource usage prediction. *ITU Journal on Future and Evolving Technologies, 3*(3), 761–778. 10.52953/ehjp3291.

Violos, J., Pagoulatou, T., Tsanakas, S., Tserpes, K., & Varvarigou, T. (2021a). Predicting resource usage in edge computing infrastructures with CNN and a hybrid bayesian particle swarm hyper-parameter optimization model. In K. Arai (Ed.), *Intelligent computing* (pp. 562–580). Springer International Publishing.

Violos, J., Psomakelis, E., Danopoulos, D., Tsanakas, S., & Varvarigou, T. (2020). Using lstm neural networks as resource utilization predictors: The case of training deep learning models on the edge. In *International conference on the economics of grids, clouds, systems, and services* (pp. 67–74). Springer.

Violos, J., Theodoropoulos, T., Maroudis, A.-C., Leivadeas, A., & Tserpes, K. (2022a). Self-attention based encoder-decoder for multistep human density prediction. *Journal of Urban Mobility, 2*, 100022. 10.1016/j.urbmob.2022.100022.

Violos, J., Tsanakas, S., Theodoropoulos, T., Leivadeas, A., Tserpes, K., & Varvarigou, T. (2021b). Hypertuming GRU neural networks for edge resource usage prediction. In *2021 IEEE symposium on computers and communications (iscc)* (pp. 1–8). 10.1109/ISCC53001.2021.9631548.

Violos, J., Tsanakas, S., Theodoropoulos, T., Leivadeas, A., Tserpes, K., & Varvarigou, T. (2022b). Intelligent horizontal autoscaling in edge computing using a double tower neural network. *Computer Networks, 217*, 109339.

Vohra, D. (2017). Using autoscaling. In *Kubernetes management design patterns* (pp. 299–308). Springer.

Walid, M. A. A., Ahmed, S. M., Zeyad, M., Galib, S. S., & Nesa, M. (2022). Analysis of machine learning strategies for prediction of passing undergraduate admission test. *International Journal of Information Management Data Insights, 2*(2), 100111.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., & Zhang, C. (2020). Connecting the dots: Multivariate time series forecasting with graph neural networks. 10.48550/ARXIV.2005.11650

Xiong, J., Yu, L., Zhang, D., & Leng, Y. (2021). Dncp: An attention-based deep learning approach enhanced with attractiveness and timeliness of news for online news click prediction. *Information & Management, 58*(2), 103428.

Xu, Q., Mehrotra, S., Mao, Z., & Li, J. (2013). Proteus: Network performance forecast for real-time, interactive mobile applications. In *Proceeding of the 11th annual international conference on mobile systems, applications, and services*. In *MobiSys '13* (pp. 347–360). New York, NY, USA: Association for Computing Machinery. 10.1145/2462456.2464453.

Yang, X., McEwen, R., Ong, L. R., & Zihayat, M. (2020). A big data analytics framework for detecting user-level depression from social networks. *International journal of information management, 54*, 102141.

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence*. International Joint Conferences on Artificial Intelligence Organization. 10.24963/ijcai.2018/505.