electronics

Article

Technical Study of Deep Learning in Cloud Computing for Accurate Workload Prediction

Zaakki Ahamed [1,*], Maher Khemakhem [1], Fathy Eassa [1], Fawaz Alsolami [1] and Abdullah S. Al-Malaise Al-Ghamdi [2]

[1] Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University (KAU), Jeddah 21589, Saudi Arabia
[2] Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: zluthufi@stu.kau.edu.sa

Abstract: Proactive resource management in Cloud Services not only maximizes cost effectiveness but also enables issues such as Service Level Agreement (SLA) violations and the provisioning of resources to be overcome. Workload prediction using Deep Learning (DL) is a popular method of inferring complicated multidimensional data of cloud environments to meet this requirement. The overall quality of the model depends on the quality of the data as much as the architecture. Therefore, the data sourced to train the model must be of good quality. However, existing works in this domain have either used a singular data source or have not taken into account the importance of uniformity for unbiased and accurate analysis. This results in the efficacy of DL models suffering. In this paper, we provide a technical analysis of using DL models such as Recurrent Neural Networks (RNN), Multilayer Perception (MLP), Long Short-Term Memory (LSTM), and, Convolutional Neural Networks (CNN) to exploit the time series characteristics of real-world workloads from the Parallel Workloads Archive of the Standard Workload Format (SWF) with the aim of conducting an unbiased analysis. The robustness of these models is evaluated using the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) error metrics. The findings of these highlight that the LSTM model exhibits the best performance compared to the other models. Additionally, to the best of our knowledge, insights of DL in workload prediction of cloud computing environments is insufficient in the literature. To address these challenges, we provide a comprehensive background on resource management and load prediction using DL. Then, we break down the models, error metrics, and data sources across different bodies of work.

Keywords: deep learning; workload prediction; cloud computing; machine learning

Citation: Ahamed, Z.; Khemakhem, M.; Eassa, F.; Alsolami, F.; Al-Ghamdi, A.S.A.-M. Technical Study of Deep Learning in Cloud Computing for Accurate Workload Prediction. *Electronics* **2023**, *12*, 650. https://doi.org/10.3390/electronics12030650

Academic Editors: Andrea Prati, Luis Javier García Villalba and Vincent A. Cicirello

Received: 22 December 2022
Revised: 18 January 2023
Accepted: 24 January 2023
Published: 28 January 2023

1. Introduction

The past decade has witnessed a rapid escalation in demand for upscale computational devices [1]. The progression of technology during this period has facilitated the emergence of highly evolved and advanced computing paradigms, cloud computing being a prominent part of it [1,2]. Cloud computing offers ease of access to a range of virtual resources, platforms, and software as services [2,3]. Due to its responsiveness, scalability, and efficiency, it is widely used by organizations and private as well as public sectors [2,4].

One major requirement in offering high-performance cloud computing services is to ensure efficient resource management (RM). RM is essentially the process of handling the release and obtention of virtual resources [1,3]. Furthermore, effective RM results in the lowering of the cost and energy consumption of cloud Data Centers (CDCs), which in turn results in reduced CO_2 emissions [3,5]. However, modern cloud computing systems are highly scalable and cater to a large number of users. This factor makes RM particularly challenging due to its complexity [6]. Under RM auto-scaling is an attribute that enables

dynamically adjusting an application's resource aptitude [7]. Auto-scaling can be divided into two main categories, reactive and proactive methods.

Reactive methods: In reactive methods, the system adjusts the workload by scaling the computational resources up or down. Its actions are based on an explicit preset threshold from the runtime environment [3,7–9]. However, one key setback is that the system has to reach its operating threshold before the auto-scaling mechanism comes into play. Therefore, reactive methods might handle unexpected surges of workload poorly [3,7].

Proactive methods: These methods follow a predictive approach, where the system is capable of actively adapting its behavior based on the forecast of future occurrences. Thus, pre-empting the obtainability of adequate resources prior to the occurrence of actual workload [3,7,9,10]. Efficient proactive methods can help in reducing the cost and improving performance by ensuring minimal inert resources [3]. Figure 1 illustrates the common auto-scaling techniques under reactive and proactive methods [7].

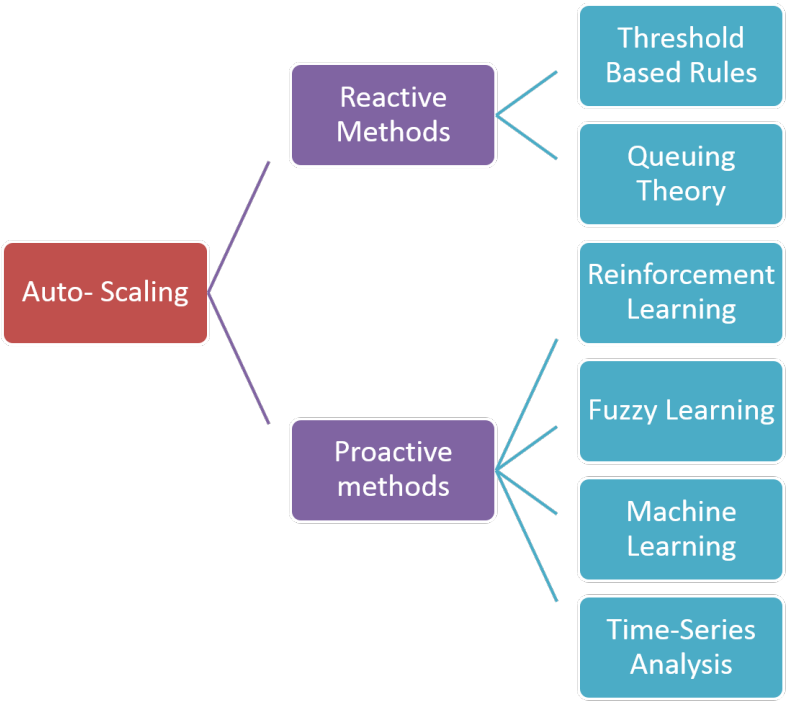


Figure 1. Common auto-scaling techniques for reactive and proactive methods.

Based on previous studies [3,7,10,11], proactive methods have proven to be more effective when it comes to load prediction. Additionally, various incidents such as Service Level Agreement (SLA) violations, over-provisioning, under-provisioning or a combination both can take place due to the dynamic nature of the cloud environment. Over-provisioning happens when the CSP allocates or reserves excessive computing resources to accommodate the peak time surges. Under-provisioning happens when the allocated resources cannot cater to the existing demand, thus affecting the Quality of Service (QoS). Thus, to mitigate and minimize the occurrence of such events, taking a proactive approach to auto-scaling resources would be appropriate. The prediction of requirements via proactive and dynamic decision-making ensures appropriate resource provisioning. Such methods ensure a direct relationship between the existing resource provision and the demand for future load based on historic data of resource usage [11,12].

Deep Learning (DL) has made a resurgence in modern applied research. The key factor is that unprecedented quantities of data are generated from modern computing systems. DL models perform well when extracting patterns from complicated multidimensional data, making them ideal for this scenario [13,14]. Therefore, it is not surprising that DL has seen popularity in research related to cloud computing as they are witnessing widespread adoption [4,15–18]. However, the quality of the model can be affected when a large amount

of data is fed into it [19,20]. The work conducted in this domain handles data in different ways, lacking uniformity. Issues in the data would compromise the quality of the models, finally resulting in biased results. While some studies focus on machine learning for workload prediction, we find that studies focusing on DL specifically are lacking, despite its popularity.

The impact of data quality has not been deeply investigated despite its importance in the output of the models. Only a few works exist in this domain. One such example is the Data Quality Toolkit [21] proposed by Gupta et al. They proposed an automated solution for investigating weaknesses in datasets. While it is a valid and valuable contribution, an argument can be put forth that it does not cater to specific domains such as text or speech. The application of data cleaning depends on the context of the problem. For example, data representing cloud workloads cannot be cleaned in the same manner as an image dataset as the feature representation would have different weightage in each scenario. We conducted an analysis using high-quality data focusing directly on the cloud computing domain. Furthermore, Gupta et al. presented only a limited demonstration of the application in model training. We trained our data on contemporary deep learning models, which gives a platform for robust analysis. We believe that these bodies of work are in the right direction for attaining robust quality data for machine learning applications.

This article offers a detailed analysis of load prediction schemes based on DL techniques. Accordingly, two research questions are considered: (1) How efficient are the existing models of DL for load prediction in the cloud environment? (2) What is the performance of DL models working with real-world datasets in an unbiased manner? The main contributions of this article include:

- A brief overview of load prediction, highlighting the challenges and requirements.
- A review of existing work of deep-learning schemes used for load prediction in cloud environments.
- A detailed comparison of the workload prediction models in terms of the DL technique applied, the dataset used, and other factors.
- A technical analysis of real-world datasets using DL models such as Recurrent Neural Networks (RNN), Multilayer Perception (MLP), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN).
- A discussion on the research challenges and future research areas.

The remainder of this article is structured as follows: Section 2 discusses the preliminaries such as metrics, challenges, and objectives related to workload prediction. Section 3 gives a detailed literature review on DL methods employed for workload prediction and Section 4 presents a comparative analysis of the existing literature. Section 5 discusses the methodology for the technical analysis. Section 6 discusses the results and metrics. Section 7 explains the findings of the analysis, while the paper concludes in Section 8. Section 9 discusses possible future directions.

2. Background

Load prediction enables the appropriate allocation of resources by predicting future load based on historic load data and actual load data, as illustrated in Figure 2. This section provides a brief insight into the background of the load prediction in the cloud paradigm and discusses (1) the key challenges related to load prediction, (2) the primary objective that should be achieved during load prediction, (3) types of load predictors, (4) types of datasets commonly utilized to conduct experiments with load predictors, and lastly (5) the commonly applied prediction error metrics and evaluation criteria that need to be considered during load prediction. Each of these points has been briefly discussed in the subsection below.

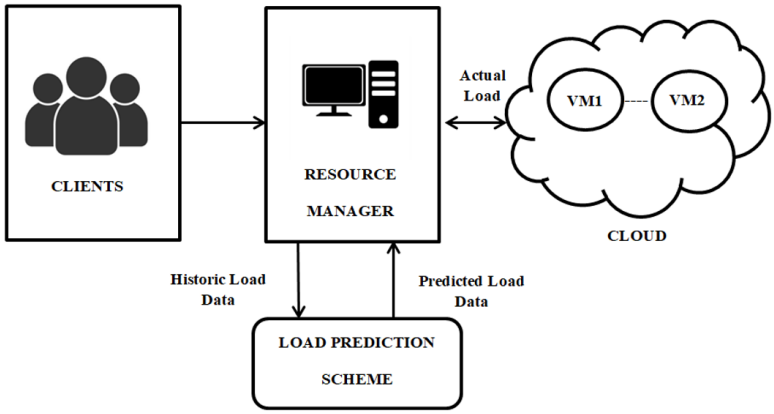


Figure 2. Load Prediction Process Overview.

2.1. Load Prediction Challenges

The process of load prediction is a complex problem to tackle. If not carried out properly, it could lead to a range of issues from SLA violations to the over-allocation of resources and under-allocation of resources, thus causing the wastage of resources, higher losses, and a lower performance of the cloud environment. Hence, it is essential that the prediction scheme is reliable and efficient and is capable of accurately forecasting the load to avoid such issues. However, some key challenges that need to be considered during this process are cost, data granularity, pattern length and complexity [3,10,22]:

1. Cost: The expense of computational resources to run and train the prediction schemes.
2. Data granularity: This is the level of detail for a particular data structure. The preliminary phase of designing the prediction scheme is to define the resources that need to be monitored, and to determine the next sampling window. The coarse-grained (long-term sampling) prompts the scheme to lose its potency. The fine-grained (short-term sampling) can potentially comprise factors that are not beneficial. This increases the intricacy of the scheme to capture it; in addition, this can also increase the cost of data collection.
3. Pattern length: This is common for the pattern length in prediction schemes to be constant. A sliding window is utilized to extricate and obtain the pattern with a fixed length. Therefore, the limitations of the pattern length limit the scheme to a particular pattern length, thus preventing the scheme from learning intricate representations from the data.
4. Complexity: To accurately forecast the workload, prediction schemes require composite computational means. Hence, to ensure effective prediction, the time and space requirements of the prediction scheme should not be substantial.
5. Historical load data: The quantity of specific historic data or insufficient historic data to train the prediction scheme has a major effect on load prediction.

2.2. Main Objective

The following are the foremost objectives that need to be achieved in an efficient and reliable load prediction scheme [22].

1. Adaptability: The prediction scheme should adapt to changes, as the cloud environment is dynamic and constantly evolving. Thus, the prediction scheme must be capable of learning the workload behavior changes to exploit the underlying patterns, in turn reducing the chances of forecasting errors.
2. Proactiveness: The load prediction scheme should be proactive, and should forecast future demand before the occurrence of load fluctuations. This allows the RM sufficient time to offer suitable resources, thus avoiding the time spent on Virtual Machine (VM) migration and provisioning.

3. Accuracy: The accuracy of the prediction scheme depends on past data and the nature of the load. The prediction scheme examines and analyzes historic load data and learns the behavior of the workload. The schemes are assessed by the accuracy of their prediction and evaluated based on how it is closer to the actual values.

2.3. Types of Workload Dataset

Two types of workload data can be obtained and used for the evaluation of prediction schemes—data sets from actual data centers versus simulated workload datasets where synthetic data are generated by recreating the data center environment within a virtual environment. The actual dataset can be retrieved from a regular cloud platform, or obtained from benchmark datasets such as NASA, Planet lab, Google trace, Calgary, Saskatchewan Alibaba Cluster Trace, and Dilma. Key characteristics that can be retrieved from such data include CPU usage, memory, bandwidth, number of requests, etc. [4,14–17,23–27].

2.4. Evaluation Metrics

Generally, load prediction schemes are evaluated based on error metrics and certain evaluation criteria such as cost, success, profit, and performance in terms of accuracy [22]. These factors differ from one scheme to another, depending on the scope of the scheme and the outcome the scheme is designed to offer. However, the commonly used error metrics and evaluation criteria are depicted in Figure 3.

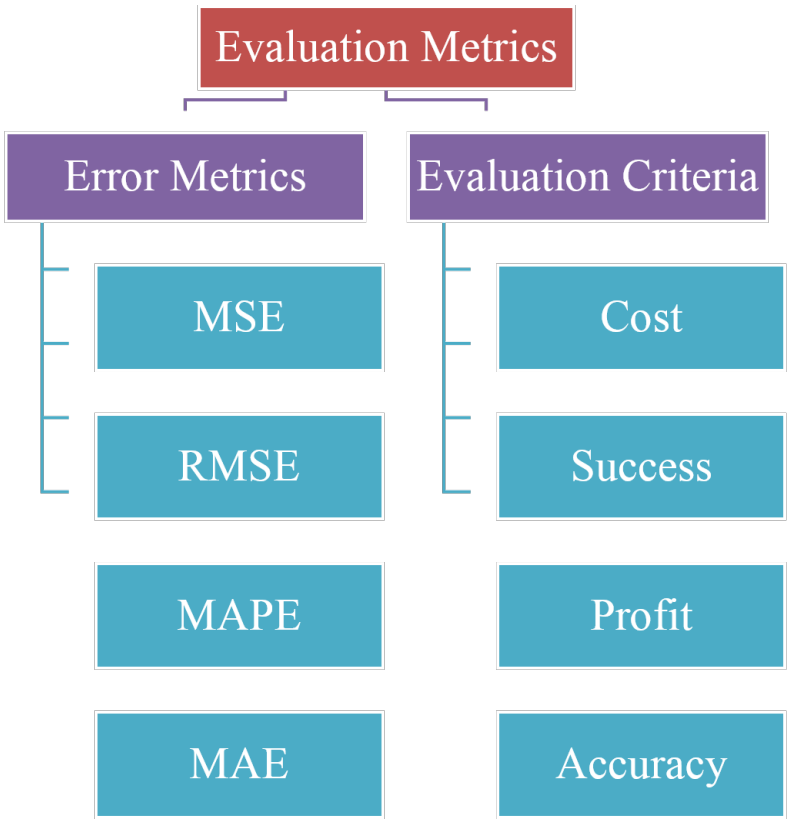


Figure 3. Commonly used evaluation metrics.

1. Error metrics: This is used to evaluate the variance between the actual behavior and the forecasted behavior of the application in different methods. The deviation metrics directly deliberate the deviation amid the actual value and the forecasted value [22]. Figure 3 depicts some of the commonly utilized error metrics when it comes to evaluating load prediction schemes.
2. Cost: Errors in prediction could cause SLA violations. Hence, the cost metrics are applied to estimate the cost derived from the prediction error [3,22].

3. Success: This defines the level up to which the scheme can accurately predict the future activities of the application. The success rate is described as the proportion of the number of accurate approximations to the total amount of approximations [3,22].
4. Profit: The profit rate is calculated depending on the profits acquired from renting out the resources, hence avoiding resource wastage and SLA violations. Thus, the profit metrics are utilized to calculate the profit rate of the cloud service provider [3,22].
5. Accuracy: The prediction schemes are assessed based on the accuracy of the forecasted results. Hence, the schemes whose outputs are closer to the actual values are more consistent and reliable [22].

3. Deep Learning for Load Prediction

Several DL-based workload prediction systems have been proposed in the literature [4,14–17,23–27]. In this section, we give a detailed analysis highlighting their contributions as well as the DL techniques utilized for the load prediction scheme. The taxonomy of the commonly used load prediction schemes applied in the cloud environment based on the type of algorithms utilized in the load prediction process is illustrated in Figure 4. Further, we have highlighted the strengths and weaknesses of the discussed studies in Table 1.

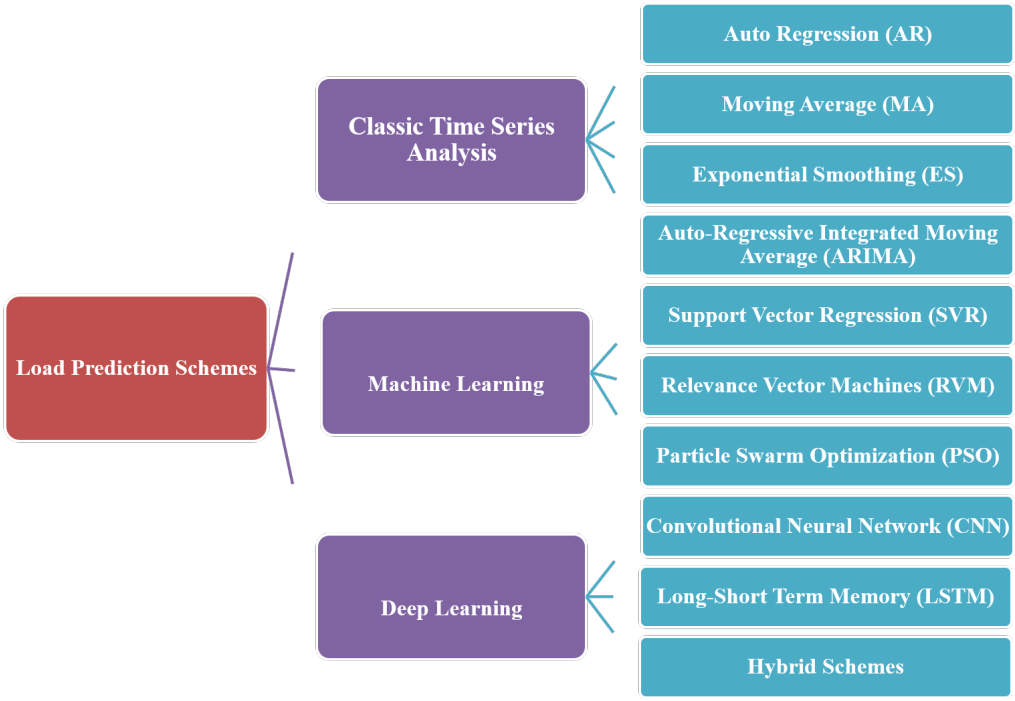


Figure 4. Taxonomy of load prediction schemes.

Additionally, we would like to mention that when it comes to the classic time series analysis schemes, such as Auto Regression (AR), Moving Average (MA), etc., these schemes presume fixed behavior and linear dependency among the time-series samples. Although such schemes have been widely utilized previously, they were unable to provide accurate forecasts in the long run and during high demand or highly erratic time series [4,23]. Although modern applications in time-series schemes are more accurate and provide more efficient results, classic schemes lack these aspects. Schemes such as Particle Swarm Optimization (PSO), Support Vector Regression (SVR), and Relevance Vector Machines (RVM) are techniques that are expansively applied to predict load in the dynamic cloud environment, addressing the shortcomings of the first-generation schemes. However, such techniques require the prudent regulation of the parameters. Their forecasting performance is based on selecting the top parameters [4,23].

DL schemes are currently popular and upcoming techniques when it comes to load prediction in cloud environments as they offer greater benefits than the traditional machine learning techniques, in a dynamic and complex environment [4,23]. Hence, a detailed analysis and survey of the literature on DL-based studies is done in the following paragraphs.

In [4] a hybrid scheme is proposed that applies Generative Adversarial Network (GAN) to forecast exceedingly unstable and chaotic cloud workload. The scheme employs a multistep-ahead method that utilized the composite and nonlinear dependencies amongst sequential samples of the workload time series. This results in substantial advancements in the accuracy of workload prediction which can be beneficial for RM decisions. The two key components of the E2LG scheme are the stacked LSTM method capable of forecasting low-frequency Intrinsic Mode Functions (IMF) units. Whereas, the GAN architecture is capable of forecasting high-frequency IMF units.

To utilize the double-layer stacked LSTM Network, initially; the sliding window method is employed to assemble the historic sub-sequence as an input for the scheme. Due to the double-layer neurons, the scheme looks into the subsequent sample for a training sub-sequence and attempts to learn the effect fluctuations in the input time series on the following sample. By applying this process during the training stage, a trained stacked LSTM network can be created which can then be utilized to effectually execute one step-ahead forecasting for each sample in the testing process of the scheme for individual IMF. Lastly, to acquire the final forecasting value, all predictions for the entire IMF and residual are added together to create the final value.

A scheme combining LSTM and RNN is proposed by [23]. It is used to solve issues such as dynamic resource scaling and power consumption in CDCs. The RNN component is made up of a combination of networks in loops; this allows the information to continue further. Each network in the loop obtains the information and input from the prior network; carries out the defined task, and generates the output. In addition, it also passes the information to the subsequent network in the loop. Some applications only necessitate recent information whereas others may require additional information from the past; commonly RNN has a delay in learning as the gap between the required prior information and the point of requirement might be too large. Hence to evade this issue the proposed scheme employed LSTM networks. As these are fabricated to avoid the long-term dependency problem that occurs in recurrent networks. As LSTM networks are capable of retaining information for a longer period. The forecasted output from the proposed scheme is then fed into the resource manager that deliberates the current state of the data center, which enables it to scale up or scale down the resources accordingly.

The LSTM encoder-decoder network with the attention mechanism proposed by [24] has three key stages. The first stage is where the scheme extracts contextual and sequential features of the historic workload data via the encoder network. In the second step, the scheme assimilates the attention mechanism within the decoder network, which enables the prediction process for batch workloads. Initially, during the encoding process, the input workload is fed sequentially into the context vector. Next, the decoding network recursively decodes the context vector, hence providing an output of the predicted sequence.

A cloud environment is prone to batch workloads, where compute-demanding tasks are typically broken down into sub-parts; hence, the latter subparts must wait for the prior subpart to be completed before it can be executed. In such an instance, each phase in the historic workload sequence would have a dissimilar impact on the existing workload. As a result, the relationship modeling amongst the present time step and its content, the historic workload sequence should be assigned different weights at each point instead of assigning the same weight. Since a basic LSTM encoder-decoder network is not able to do so, the attention mechanism is included to solve this by evaluating the relevance of individual parts and assigning different weights to them. A higher weight indicates more significance. Thus indicating the amount of workload that impacts the existing workload prediction.

A Deep Belief Network (DBN) scheme proposed by [15] consists of a multi-layer architecture, comprising the DBN layer that is capable of extracting high-level features

from the historic load data and the logical regression layer. The scheme follows the rapid unsupervised layer-wise training process to train the Neural Network (NN) by applying the Restricted Boltzmann Machine (RBM) to the data to reduce its dimensionality, hence preventing the error gradient from vanishing due to the increase in the number of hidden layers. The logical regression layer within the scheme is responsible for conducting the fine-tuning of the entire scheme in a supervised method. Since this scheme is specifically designed to focus on the CPU usage of VMs, the data with regard to the CPU utilization are taken from all the VMs in the cloud environment, the utilization data are from multiple prior time intervals. The input layer takes CPU utilization data as the input, and the top layer provides the predicted output. The designed model can be used to forecast the workload for a single VM or multiple VMs. The input layer at the bottom is where the CPU utilization data are fed, and the top layer provides the predicted output. The designed model can be used to forecast the workload for a single VM or multiple VMs.

In [14], a DL-based prediction algorithm (L-PAW) is proposed for workload prediction in the cloud. The proposed model uses RNN. Typically, RNN-based networks are capable of learning valuable information from the input data. As it reads, it updates all the prior information, increasing the gradients as the time intermissions increase, leading to the gradient vanishing. This causes ineffective changes in the RNN network parameters so the scheme is unable to efficiently capture long-term memory dependencies. To overcome this issue, the proposed scheme applies a top-sparse autoencoder (TSA), which is used to compress the workload data and extract essential features with a lower dimension, hence avoiding the degradation of the prediction accuracy due to redundancy and high dimensionality of the workload data. In addition, the gated recurrent unit (GRU) block is integrated into the RNN to substitute the hidden layers of a classic RNN, thus enabling the capture of the long-term memory dependencies from the historic workload input. Therefore, the incorporation of the TSA along with the GRU within the RNN leads the proposed model to attain adaptive and precise forecasts for workloads that constantly fluctuate in the cloud environment.

A method suggested by [25] tends to perform the forecast at a specific time prior to the predicted time point, hence providing an adequate period for the task scheduler to make a decision. The suggested method is an m-gap prediction that retains a gap of specific (m) time points between the input data points and the forecasted data points, thus providing sufficient time for task scheduling to take place. Furthermore, a clustering-based prediction method had been suggested to ensure all the workload patterns are captured within all the heterogeneous tasks. The clustering-based method is capable of clustering all the tasks with comparable workload patterns into groups for training to create a specific model; the equivalent model is then used to forecast its workload. Two types of clustering methods were used: Prototype-Based Clustering (PBC) and Density-based clustering (DBC). The use of these leads to a higher prediction accuracy [25].

A new deep RNN approach was proposed by [26] that incorporates the Savitzky–Golay (S-G) filter along with LSTM networks (SG-LSTM). Initially, the proposed scheme performs a logarithmic function before the task sequence smoothing. This is mainly done to diminish the standard deviation. Then, the S-G filter, also known as least square polynomial smoothing, which is a data smoothing mechanism that is used to remove any noisy modules while maintaining the original signal's peak and width, is applied. The key condition for forecasting task time series is to encapsulate the historically varying patterns. Within the proposed scheme, the LSTM network substitutes the neurons within the hidden layer of the RNN with a memory block. Individual memory blocks comprise single or multiple memory cells along with three kinds of gates known as forget, input and output gates. These gates can be opened or closed to regulate if the prior network circumstance at the output layer brings about a threshold to add to the existing layer [26]. Through the operation of the gates, LSTM memory cells are capable of realizing the intricate relationship amongst features within a task time series, which includes both long terms and short terms. Furthermore, the scheme employs a Backpropagation Through Time (BPTT) algorithm

to prevent the gradient exploding, which can occur due to the non-negligible learning of the parameters enclosed in LSTM. These operations within the proposed model result in improved accuracy when forecasting task time series within CDCs.

In [17], a storage workload prediction scheme (CrystalLP) is proposed. The key functionalities performed by the scheme include: gathering workload data, data pre-processing, time-series prediction, and data post-processing stages [17].

In the workload gathering or collection phase, the time-series data that comprise the workloads at various periods are gathered. Since the proposed scheme is specifically designed for a single-phase workload prediction, for each prediction procedure the input data are the data contained by the history horizon, and the output is the forecasted workload at the subsequent period. During the pre-processing phase, various normalization functions are performed on the workload traces by utilizing a pre-processing component that is capable of segmenting the sequential data with a permanent sliding window dimension. The component then carries out a scaling transformation to create appropriate data to be used as the input. Within the data post-processing phase, an LSTM network is utilized to predict the time series, as LSTM networks can capture composite workload patterns, thus making it suitable for capturing the inner associations amongst the historic and future values. Furthermore, to obtain efficient model training outcomes, CrystalLP employs Stochastic Gradient Descent (SGD) and Adam optimizer to train the prediction scheme, hence enabling the provision of more accurate forecasts.

A DL scheme based on Canonical Polyadic Decomposition (CPD) has been proposed by [16] to forecast the cloud workload for industry informatics. The proposed scheme comprises a stacked auto-encoder that is capable of learning significant features of the workload data. Next, the CPD is applied to contract the parameters considerably to enhance the scheme's training efficacy. However, CPD can only be applied for tensor (a matrix or vector that characterizes all types of data) decomposition; thus, to overcome this, the stacked auto-encoder is converted to a tensor format using a bijection. The CPD is then used in the tensor stacked auto-encoder to compress the parameters [16]. To train the parameters of the tensor-stacked auto-encoder, the scheme proposes an effective learning algorithm that can be applied directly to the compressed parameters to enhance the training efficacy. The proposed scheme is then employed for the workload prediction of VMs in the cloud environment.

An integrated forecasting technique was proposed in [27]. The scheme comprises a Savitzky–Golay filter, which is a data smoothing filter that is well-known for its least square polynomial smoothing. The main goal of this filter is to eliminate the noise while preserving the width and peak of the signal and further smooth out the non-stationary workload time series during the data pre-processing phase. Then, the proposed model employs Wavelet decomposition, specifically the Haar wavelet method, to attain significant details and trends for various workload time series, as it is capable of reducing varying features of a workload series to enhance the prediction accuracy. In addition, the wavelets can realize information in the data at various scales of resolution. Furthermore, the proposed model utilizes Stochastic Configuration Networks (SCN) to build a randomized learner model following the supervised approach. This enables the distinction of statistical aspects for the workload time-series trends and detailed features, thus resulting in a quicker learning speed and higher prediction accuracy.

The strengths and limitations of the aforementioned studies are tabulated in Table 1.

As evident in the above table, each model has strived to improve on a combination of higher accuracy, strong adaptability, faster learning, and dealing well with high-dimensional data. Since it is not possible to have a perfect system, the trade offs are generally related to having more processing power, time for training, and power consumption.

Table 1. Comparison of strengths and limitations of the DL models.

Ref	Model	Strengths	Limitations
[4]	E2LG	• Improves the prediction accuracy. • Reduces complexity and nonlinearity of prediction model in each frequency band. • Can exploit the long-term nonlinear dependencies in high frequency • Algorithm maintains its prediction efficiency in long-term multistep-ahead prediction scenarios.	• Imbalance between the generator and discriminator causing overfitting
[23]	LSTM-RNN	• Higher prediction accuracy	• Difficult to train (because they require memory-bandwidth-bound computation)
[24]	Attention based LSTM encoder-decoder	• Increase the accuracy of the long-term prediction method • Effective in mitigating error amplification in the long-term prediction.	• Requires twice the training of the single-layer network. • Calculation is roughly three times that of a single-layer model, requires more epochs to converge. • Extra cost of processing for a large model.
[15]	Deep Belief Network	• Improved accuracy of the CPU utilization prediction	• Training is more challenging
[14]	L-PAW	• High prediction accuracy measured by MSE for highly autocorrelated, highly periodic and highly random workloads • Strong adabptability • Suitable for High-dimensional and highly-variable cloud workloads	• Slow convergence
[25]	Clustering-based prediction method	• Higher prediction accuracy	• Difficult to scale to larger datasets • User must specify the number of clusters.
[26]	SG-LSTM	• Accurate prediction of task time series	• Higher network capacity and training required. • Long time to learn complex dependencies.
[17]	CrystalLP Framework	• Higher performance	• Increase in required network capacity and training.
[16]	CPD	• Good speed on training • Low values for RMSE and MAPE metrics	• Privacy concerns -require the domain knowledge. • Less performance in video recognition and robot control, and lesser accuracy (compared to deep reinforcement learning).
[27]	SGW-SCN	• Better accuracy • Faster learning speed	• Higher resource consumption

4. Comparative Analysis

This section provides a brief comparison of the models for workload prediction discussed in Section 3 based on the following areas, highlighting the key similarities and differences:

- The architecture type employed in designing the scheme;
- The simulation environment that is employed;
- The resource factors forecasted by the schemes to identify the sustained load;
- The type of dataset utilized for each scheme;
- The error metrics that are used to evaluate each scheme.

Figure 5 depicts the classification of the commonly employed aspects in terms of architecture, environment, datasets, features analyzed and metrics used amongst the models assessed.

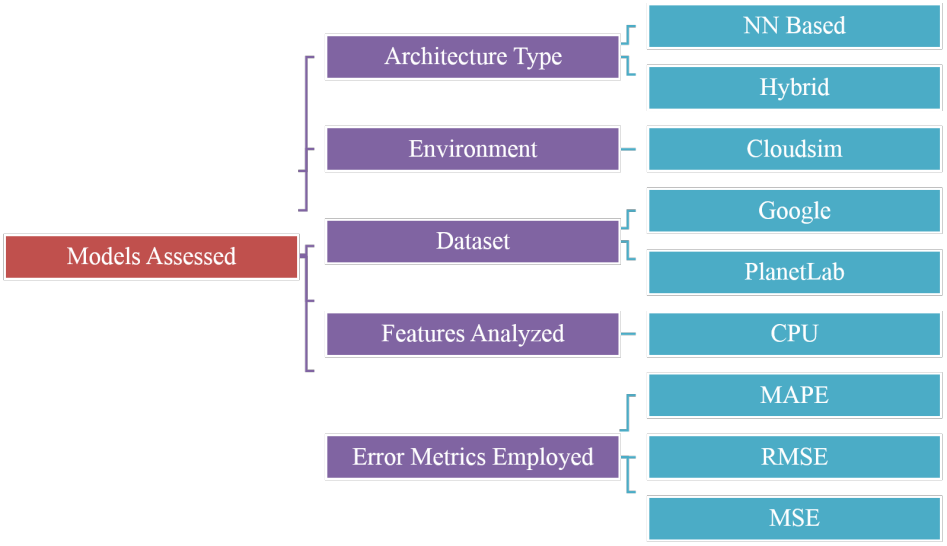


Figure 5. Classification of commonly employed aspects.

4.1. Architecture Type

Based on the models discussed and compared in Table 2, most of the workload prediction schemes either employ NN-based schemes, such as RNN, CNN, and DBN, or Hybrid schemes. NN-based schemes are employed mainly because such schemes consist of multiple layers or networks that enable the capture of intricate features from the input data; furthermore, the multilayer architectures enhance the learning of the model, which could lead to higher accuracy in workload prediction. Whereas, amongst the models that employed hybrid schemes, the hybrid was a combination of LSTM and a type of NN. LSTM is included within the hybrid schemes as they are capable of retaining long-term information; hence, a combination of NN and LSTM further improves the learning aspects of the model and hence results in a more efficient model that offers accurate workload predictions.

4.2. Environment

As shown in Table 2, among the models assessed, most of the models do not provide sufficient information on the simulation environment utilized to simulate the proposed models. However, certain models that have provided this information utilize Cloudsim [28] as the simulation environment to run the experiments simulations of the proposed models, mainly because Cloudsim is an open source framework that offers a generalized and extensile environment to run simulations on cloud computing infrastructures, hence enabling us to test how the proposed models would respond in a similar environment.

Table 2. Comparison of Architecture type and Simulation Environment of the DL models.

Scheme	Architecture				Environment	
	LSTM	NN	Hybrid	Others	CloudSIM	Other/NA
[4]			✓			N/A
[23]			✓			✓
[24]	✓					N/A
[15]		✓			✓	
[14]		✓				N/A
[25]				✓(m-grap prediction)		N/A
[26]			✓			N/A
[17]	✓					N/A
[16]				✓(CPD)	✓	
[27]		✓				N/A

4.3. Dataset

As made evident by Table 3, the most commonly employed datasets are the Google cluster trace and Planet lab datasets from Cloudsim. These datasets were chosen mainly due to the ease of access, as Google Cluster trace datasets are publicly released workload traces. They comprise running information from a large number of machines in Google CDCs. Whereas, Planet Lab datasets can be retrieved from Cloudsim. Furthermore, these datasets are actual datasets, hence providing actual workload-related information and sufficient workload information that can be used to train the proposed models efficiently.

Table 3. Comparison of the dataset and analyzed features of the DL models.

Scheme	Dataset				Features Analyzed				No. Of Requests
	Google	NASA	Planet Lab	Alibaba	Others	CPU	Memory	Task Time Series	
[4]					✓				✓
[23]		✓			✓				✓
[24]				✓	✓	✓			
[15]			✓			✓			
[14]	✓		✓	✓	✓	✓			
[25]	✓					✓	✓		
[26]	✓							✓	
[17]					✓		✓		
[16]			✓			✓			
[27]	✓							✓	

4.4. Analyzed Features

Based on the workload prediction models assessed, Table 3 shows the key features taken into consideration, including CPU, Memory, No. of requests, and task time series. However, out of the ten models, almost half, around 46 percent of the assessed models, focus on the CPU factor, as illustrated in Figure 6. This is mainly because it is essential for CDCs and cloud service providers to maintain balance and optimize computational and hardware resources, to meet the QoS standards and also ensure cost-effectiveness. Hence, the CPU information in the workload dataset can help train the workload prediction model to improve its efficacy in providing accurate results, which can aid in efficient resource provisioning.

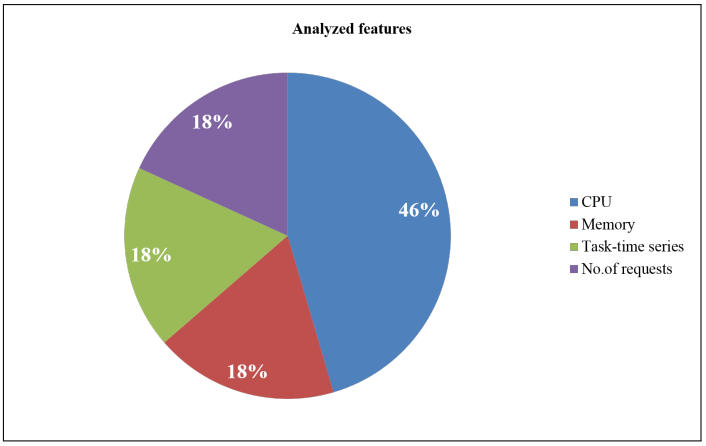


Figure 6. Representation of the features analyzed.

4.5. Metrics Employed

Error metrics function as a measure of accuracy between predicted values compared to the ground truth. Table 4 displays the error metrics commonly employed to evaluate the prediction accuracy. These include the Mean Absolute Percentage Error (MAPE) [29] and Root Mean Squared Error (RMSE) [30], followed by Mean Squared Error (MSE) [31]. However, some models use a combination of two or more evaluation metrics to evaluate the respective models. The MSE error metric was chosen for its propensity to be sensitive to outliers. They work well in assigning more weight to them. RMSE and MSE are very similar in that the former is simply the squared value of the latter. Both can represent positive and negative values. RMSE is more useful when the overall impact is disproportionate to the actual increase in error. Despite this, the similarity in function means studies would be more effective not using them together and by picking a different error metric. MAPE is popular in regression problems since it suits relative variations well. However, it is best-suited for positive data and problems where large errors are not expected. Out of the chosen models, six of them have used a minimum of two error metrics in their evaluation, lending more credibility to their findings.

Table 4. Comparison of error/accuracy metrics used in the DL models.

Scheme	Metrics					
	MAPE	**RMSRE**	**MSE**	**RMSE**	**MAE**	**Others**
[4]	✓	✓				✓
[23]			✓			
[24]				✓	✓	✓
[15]	✓					✓
[14]			✓			
[25]						✓
[26]				✓		✓
[17]	✓			✓	✓	
[16]	✓			✓		
[27]			✓			

5. Methodology

The detailed analysis of the previous section makes it evident that almost all existing Machine Learning (ML)/DL models employ singular data sources and do not take into account the importance of using unbiased data. In the bodies of existing work in this

domain, it is apparent that the output of machine learning models is highly impacted by the quality of data [32–35]. The exponents of this state that the real-world value of an ML model is directly proportional to the data it has been trained on. Therefore, it is crucial that we build methods and frameworks that enable us to assess if data are ready for training and deploying machine learning models. The effectiveness, precision, and complexity of machine learning tasks are significantly influenced by the quality of the training data. The collecting, aggregation, or annotation stages of data processing can still introduce errors or inconsistencies. Data must be profiled and evaluated in order to determine whether they are suitable for machine learning activities; otherwise, erroneous analytics and unreliable judgments may follow. The majority of the effort by researchers is centered around enhancing the quality of models and it is evident that less attention has been given to data quality. Data scientists can spend less time fixing errors to improve model performance by evaluating the quality of the data using intelligently designed metrics and creating matching transformation operations to address the quality gaps.

Table 2 shows the analysis of the datasets of Google Cloudtrace, Alibaba, NASA, Planetlabs, and proprietary datasets. Due to the unique setup of each of these data centers, the data being recorded lacks uniformity among different datasets. Each researcher has a different method for engineering the features for training. We believe that uniformity is a critical factor for a fair comparison. Various issues such as missing data, inconsistent data, erroneous data, lapses in the logging period, and unrepresentative user behavior should be taken into account [19,20]. For this reason, we have used the datasets from the Parallel Workloads Archive [36].

These workload logs are collected from large-scale parallel systems in production use in various places around the world. The unique feature of these logs is that all of them are formatted in the Standard Workload Format (SWF) format and programs would only need to parse a single format and can be applied to multiple workloads. Furthermore, the authors have taken steps to clean the data of errors and inconsistencies [36]. Table 5 lists the important data fields available in each of the datasets in the archive.

Table 5. Data fields of the Parallel Workload logs chosen for analysis

Data Fields	Feature in Dataset	Description
Job Number	jobID	Unique value representing jobs submitted for processing.
Submit Time	submission_time	In seconds. The earliest time the log refers to is zero and is usually the submittal time of the first job.
Wait Time	waiting_time	In seconds. Job's submit time vs the time at which it actually began to run.
Run Time	execution_time	In seconds. The wall clock time the job was running (end time minus start time). Values are rounded.
Number of Allocated Processors	proc_alloc	The number of processors the job uses.
Average CPU Time Used	cpu_used	Both user and system, in seconds. Average over all processors of the CPU time used.
Used Memory	mem_used	In kilobytes. Average per processor.
Requested Number of Processors	proc_req	Number of processors requested by the user.
Requested Time	user_est	This can be either runtime (measured in wallclock seconds), or average CPU time per processor (also in seconds).

Out of a myriad of fields, the ones considered most important were chosen. the `Job Number` field allows each entry to be uniquely identified. Temporal data can be inferred from the `Submit Time, Wait Time, and Run Time` fields. The `Number of Allocated Processors` and `Requested Number of Processors` fields are not always the same and the system might not always possess the resources requested by the user. `Average CPU Time Used` differs from the other time fields mentioned above since certain CPU operations can be parallellized for quicker processing. `Requested Time` is usually the upper boundary of time a user estimates that their job might need [37].

5.1. Experiment Setup

The archive in [36] contains 40 datasets. For our analysis, we have chosen the following Table 6.

Table 6. Datasets chosen for analysis from the Parallel Workloads Log.

Name	From	To	Months	CPUs	User Runtime Data	Memory Data
ANL Intrepid	Jan 2009	Sep 2009	8	163,840	✓	
CEA CURIE	Feb 2011	Oct 2012	20	93,312	✓	
UniLu Gaia	May 2014	Aug 2014	3	2004	✓	✓
MetaCentrum2	Jan 2013	Apr 2015	28	8412	✓	✓
CIEMAT Euler	Nov 2008	Dec 2017	110	1920	✓	✓
KIT FH2	Jun 2016	Jan 2018	19	24,048	✓	✓

The datasets listed in Table 6 were chosen on the following basis -

- The most recent datasets (between 2008 and 2018).
- The duration of data collection (between 3 months to 110 months).
- The number of CPUs present (from 2004 and 163,840).
- The scope of the available data (user runtime and memory data)

We believe that the above criteria would cover a wide range of scenarios when it comes to patterns to be extracted and learned The robustness of the DL models can be tested this way

The results of the preliminary analysis conducted on the dataset using the `Pandas` library [38] are tabulated in Table 7. It is evident that `df_index` and `jobID` are interchangeable in their function as unique identifiers for the jobs submitted for processing. Some datasets contained `df_index` but all the datasets contained `jobID`. Therefore, `df_index` is omitted from further analysis. In machine learning, distinct values are preferred. `jobID` is completely distinct as each row has a unique value. The rest of the table presents the percentage of distinct values. Minor distinct is for fields with less than 3% distinct values. The Pearson correlation technique [39] revealed skewed values. These outliers have the potential for creating noise in the machine learning model but cannot be ignored as spikes in usage is not uncommon. Some columns have been filled with constant values as a placeholder. Since they will not add any benefit to the training, they would also be omitted. The main feature used for prediction would be `proc_alloc` which describes the number of processors allocated for the particular job. This is justified since CPU usage is a major factor in determining resource provisioning for datacenters [40–42].

Table 7. Preliminary exploratory analysis of the chosen datasets.

Feature	Datasets					
	ANL-Intrepid-2009-1	CEA-Curie-2011-2	CIEMAT-Euler-2008-1	KIT-FH2-2016-1	METACENTRUM-2013-3	UniLu-Gaia-2014-2
df_index	Not present	100% distinct	100% distinct	Not present	Not present	100% distinct
jobID	100% distinct	100% distinct	100% distinct	100% distinct	100% distinct	100% distinct
submission_time	99.3% distinct	77% distinct	32.2% distinct	85.6% distinct	62.6% distinct	80.2% distinct
waiting_time	Minor distinct	Minor distinct	Minor distinct	Minor distinct	Minor distinct	Minor distinct
execution_time	Minor distinct	Minor distinct	Minor distinct	Minor distinct	Minor distinct	Minor distinct
proc_alloc		SKEWED ($\gamma 1$ = 20.41500034)			SKEWED ($\gamma 1$ = 30.54379091)	
cpu_used	CONSTANT	CONSTANT		CONSTANT	CONSTANT	14.7% distinct
mem_used	CONSTANT	CONSTANT		CONSTANT	CONSTANT	13.5% distinct
proc_req		SKEWED ($\gamma 1$ = 28.98839718)			SKEWED ($\gamma 1$ = 30.54379091)	
user_est		SKEWED ($\gamma 1$ = 73.30789535)	SKEWED ($\gamma 1$ =32.67258397)			SKEWED ($\gamma 1$ = 80.07723573)
mem_req	CONSTANT	CONSTANT			SKEWED ($\gamma 1$ = 99.25888565)	CONSTANT

The experiments were run on a system with an Intel i7 processor, 16 GB RAM along with `Python(3.7)`, `Keras(2.9.0)` [43], `Sci-kit learn(1.0.2)` [44], `Tensorflow(2.9.2)` [45], `Numpy(1.21.6)` [46] and `Pandas(1.3.5)`.

5.2. Feature Engineering

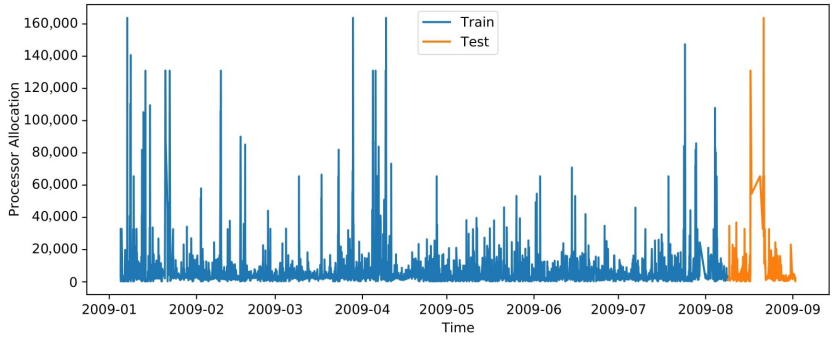
When preparing the data before using it for training, the first task was to drop columns that had constant values as mentioned in the previous section. The `submission_time` column represents the timestamp at which a job was submitted for processing in seconds. Each dataset has a field stating a `start_time` which mentions when the data collection began.

Since we are dealing with a time-series problem here, the time factor needed to be represented accurately. It was necessary to calculate the exact time a job was submitted for processing. To that end, the aforementioned `start_time` was added to each value of the `submission_time` column to produce a new column featuring a timestamp of this format—`"Year,Month,Date,Hour,Minutes,Seconds"`. Subsequent to that, the new timestamp column was set as the index column to be the unique identifier for each row while `jobID` and `submission_time` columns were dropped due to them being redundant at this point.

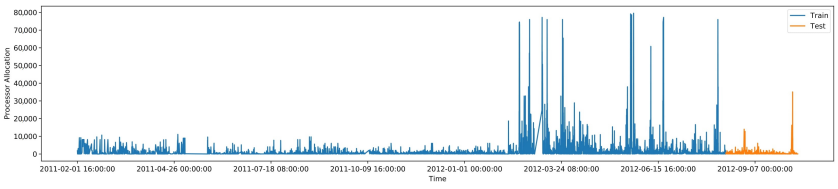
The next issue to deal with was the time interval between job submissions. For training time series problems, the interval between data points must be uniform [47]. In this case, our datapoints were not evenly spaced. Therefore, the `Pandas` package was used to resample all the values to 1-hour intervals. The rest of the values in the other columns were calculated by using the `interpolate` function.

Each of the datasets was split into training and test sets. For our analysis, 10% of each dataset was used for testing purposes as described in Figure 7 below.

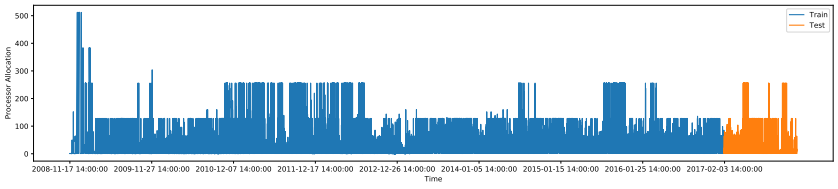
According to Figure 7, the `y-axis` denotes the processor allocation, and the `x-axis` of the graphs denotes the time period. The varied nature of each of these environments can be seen by the fact that Figure 7a has spikes of processor use going up to 160,000 while Figure 7b,d are in the range of 80,000 and 20,000 respectively. The others are relatively less comparatively even though Figure 7c,e represent valuable temporal data as they have the highest data collection window. In each of these cases, the last portion of the dataset was used for testing. Finally, the data were rescaled using the `MinMaxScaler` feature to fit between 0 and 1. Using this method, the representations and underlying patterns of data are extracted so that the training of the model is not slowed down due to a large range of values.



(**a**)

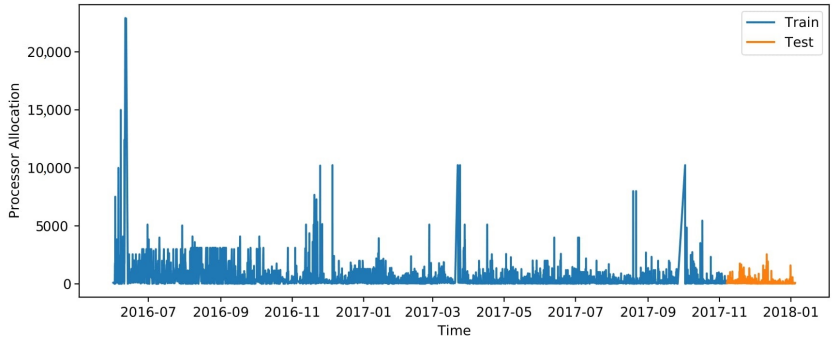


(**b**)

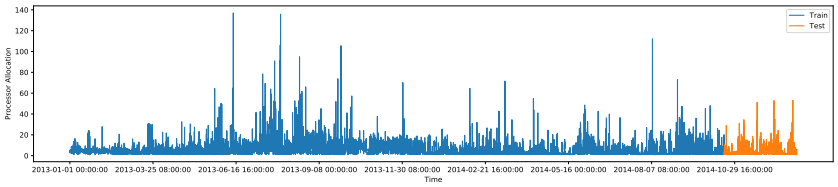


(**c**)

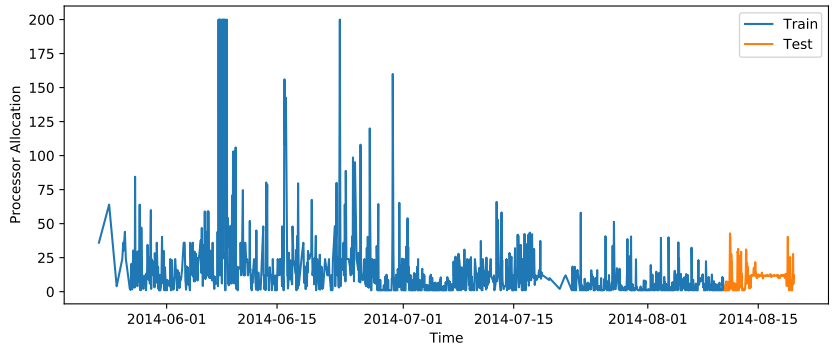
Figure 7. *Cont.*



(**d**)



(**e**)



(**f**)

Figure 7. Train/Test split of datasets representing processor allocation and data collection duration. (**a**) ANL-Intrepid-2009-1 Dataset. (**b**) CEA-Curie-2011-2 Dataset. (**c**) CIEMAT-Euler-2008-1 Dataset. (**d**) KIT-FH2-2016-1 Dataset. (**e**) METACENTRUM-2013-3 Dataset. (**f**) UniLu-Gaia-2014-2 Dataset.

5.3. Setup Deep Learning Models

In the literature review done for this paper, it was discovered that most of the work related to workload prediction is done using RNN and LSTM. For our analysis, we are comparing the more prominent DL models along with them -

- RNN
- MLP
- LSTM
- CNN

RNNs are the more popular option for time series prediction but have issues with long-term retention. LSTM is the successor for that. Along with that, we have an MLP and a CNN which would take the same data as the input.

All the datasets were trained on each of the models depicted in Figure 8 for a fair comparison. Each of the models, as shown in Figure 8a–d use the same input layers and use a generic structure for the middle layers. All of them have used a Dense layer for the final result output. Since the data points are numerous, we used a `Data generator` to feed in the data to the input layers in batches of 64 samples and trained for 100 epochs.

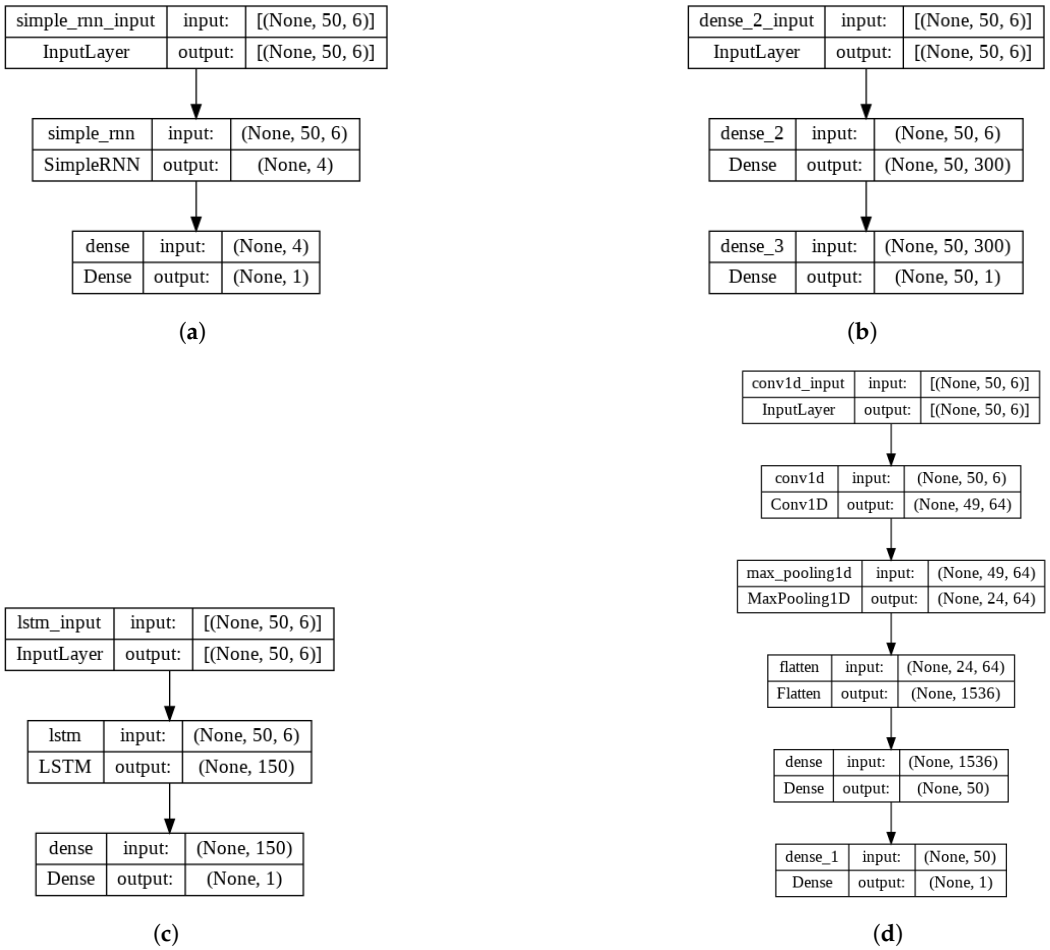


Figure 8. Structure of DL models used for training. (**a**) Recurrent Neural Network (RNN) Architecture. (**b**) Multilayer Perception (MLP) Architecture. (**c**) Long Short-Term Memory (LSTM) Architecture. (**d**) Convolutional Neural Network (CNN) Architecture.

6. Results

6.1. Metrics

For the sake of comparison, we employed the following metrics

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)

They are the more popular metrics used for statistical analysis [48].
Equation (1) presents the formula for the MAE function [48]:

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|e_i| \tag{1}$$

the Mean Absolute Error quantifies the errors between paired observations expressing the same phenomenon.

Equation (2) presents the formula for the RMSE function [48]:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}|e_i|^2} \tag{2}$$

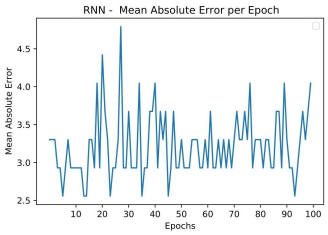
the RMSE on the other hand depicts how much the prediction errors deviate from the standard deviation. It indicates the concentration of data around the line of best fit. Since the errors are squared before they are averaged, the RMSE gives more prominence to large errors since errors are squared before they are averaged. It is particularly useful when large errors are a major issue.

The MAE and the RMSE can be utilized in tandem to find the homogeneity or heterogeneity of the sample. The greater the divide between them, the greater the variance in individual errors in the sample [49].

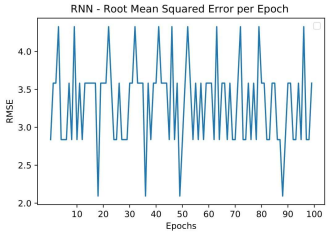
6.2. Performance Evaluation

Figures 9–12 represent the graphs associated with the RNN, MLP, LSTM, and CNN respectively. Under each main figure, there are 6 sub figures each representing the chosen datasets. The MAE and RMSE error metrics have been recorded in graph form for each of these datasets. The `X-axis` represents the training epochs while the `Y-axis` represents the relevant value for the error metric.

The results have been displayed in the following figures.

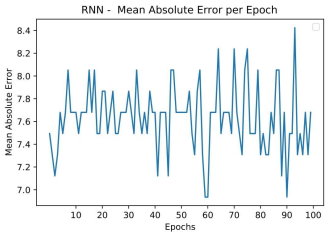


i. MAE

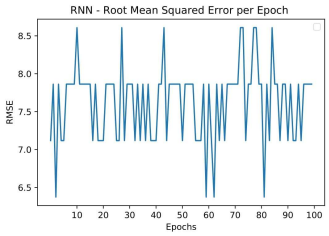


ii. RMSE

(**a**)

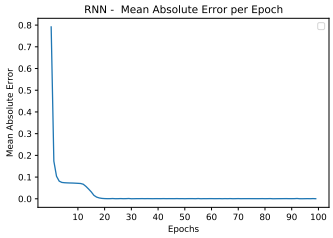


i. MAE

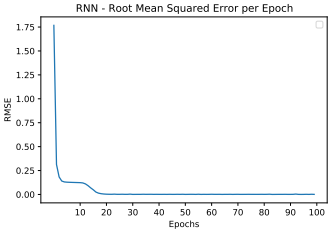


ii. RMSE

(**b**)

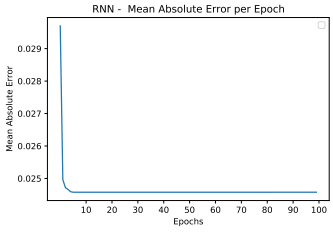


i. MAE

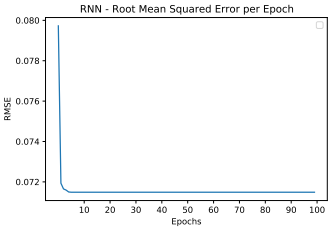


ii. RMSE

(**c**)



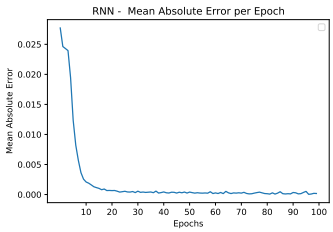
i. MAE



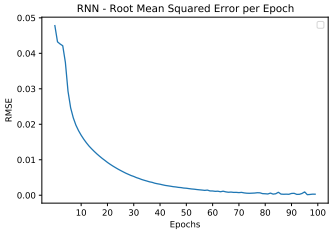
ii. RMSE

(**d**)

Figure 9. *Cont.*

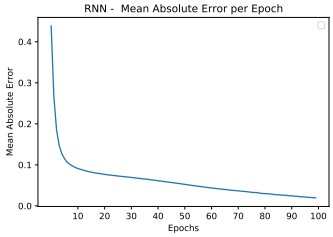


i. MAE

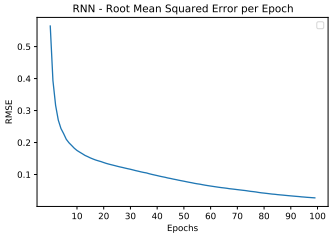


ii. RMSE

(**e**)



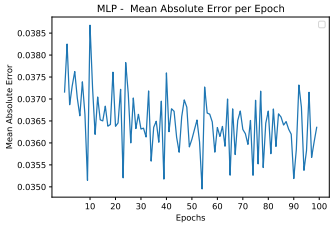
i. MAE



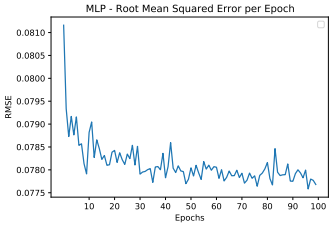
ii. RMSE

(**f**)

Figure 9. Metrics for the RNN Model. (**a**) Error Metrics for ANL-Intrepid-2009-1 Dataset. (**b**) Error Metrics for CEA-Curie-2011-2 Dataset. (**c**) Error Metrics for CIEMAT-Euler-2008-1 Dataset. (**d**) Error Metrics for KIT-FH2-2016-1 Dataset. (**e**) Error Metrics for METACENTRUM-2013-3 Dataset. (**f**) Error Metrics for UniLu-Gaia-2014-2 Dataset.

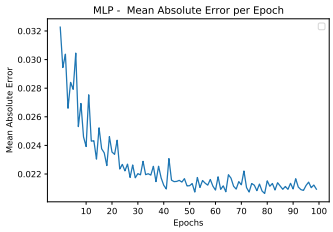


i. MAE

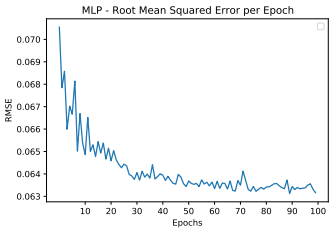


ii. RMSE

(**a**)

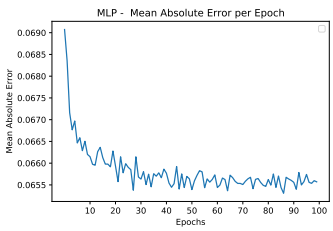


i. MAE

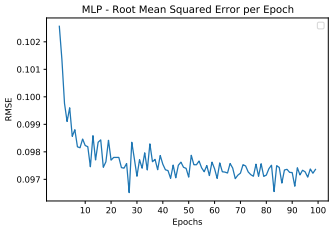


ii. RMSE

(**b**)



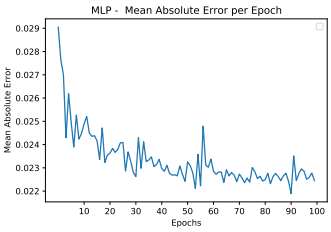
i. MAE



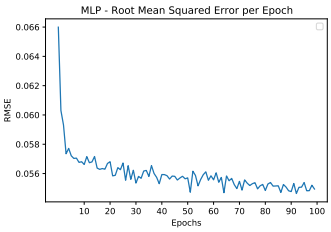
ii. RMSE

(**c**)

Figure 10. *Cont.*

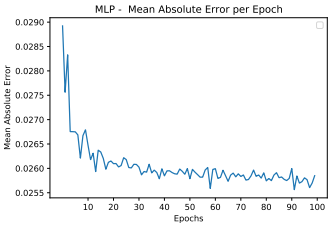


i. MAE

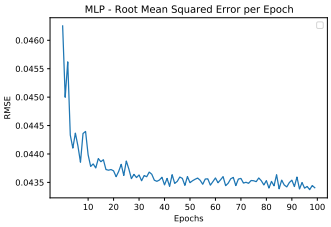


ii. RMSE

(**d**)

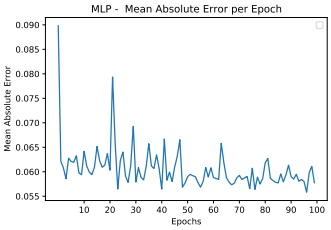


i. MAE

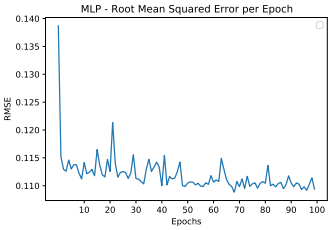


ii. RMSE

(**e**)



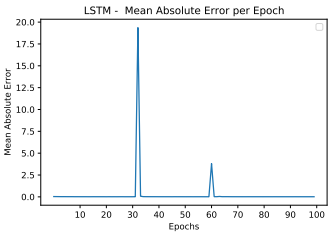
i. MAE



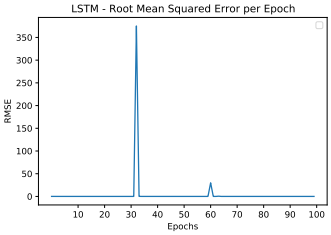
ii. RMSE

(**f**)

Figure 10. Metrics for the MLP Model. (**a**) Error Metrics for ANL-Intrepid-2009-1 Dataset. (**b**) Error Metrics for CEA-Curie-2011-2 Dataset. (**c**) Error Metrics for CIEMAT-Euler-2008-1 Dataset. (**d**) Error Metrics for KIT-FH2-2016-1 Dataset. (**e**) Error Metrics for METACENTRUM-2013-3 Dataset. (**f**) Error Metrics for UniLu-Gaia-2014-2 Dataset.

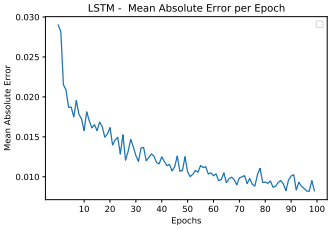


i. MAE

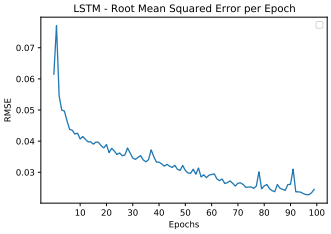


ii. RMSE

(**a**)



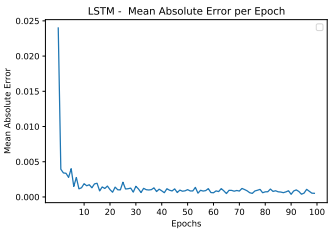
i. MAE



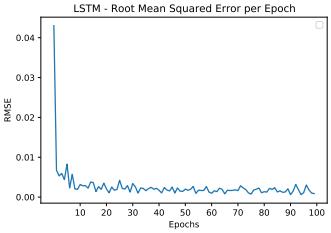
ii. RMSE

(**b**)

Figure 11. *Cont.*

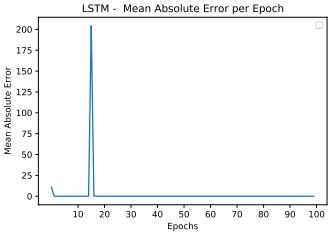


i. MAE

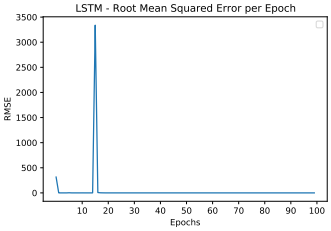


ii. RMSE

(**c**)

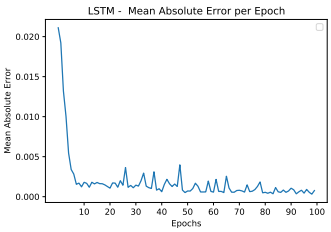


i. MAE

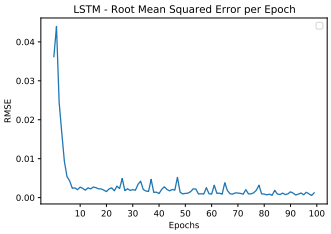


ii. RMSE

(**d**)

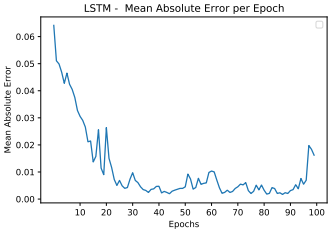


i. MAE

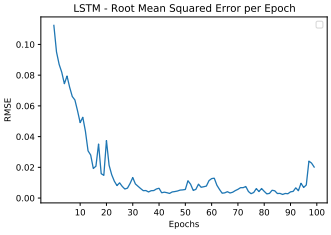


ii. RMSE

(**e**)



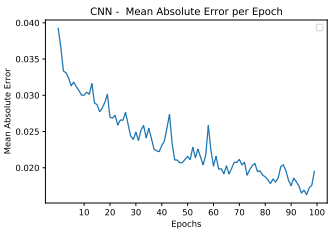
i. MAE



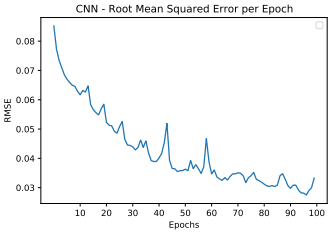
ii. RMSE

(**f**)

Figure 11. Metrics for the LSTM Model. (**a**) Error Metrics for ANL-Intrepid-2009-1 Dataset. (**b**) Error Metrics for CEA-Curie-2011-2 Dataset. (**c**) Error Metrics for CIEMAT-Euler-2008-1 Dataset. (**d**) Error Metrics for KIT-FH2-2016-1 Dataset. (**e**) Error Metrics for METACENTRUM-2013-3 Dataset. (**f**) Error Metrics for UniLu-Gaia-2014-2 Dataset.



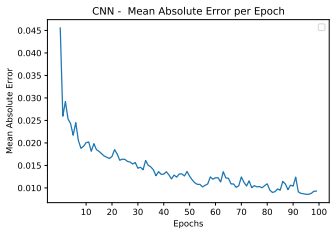
i. MAE



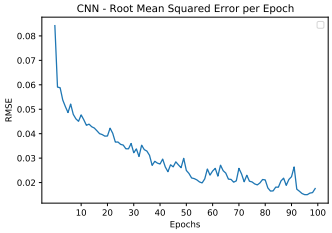
ii. RMSE

(**a**)

Figure 12. *Cont.*

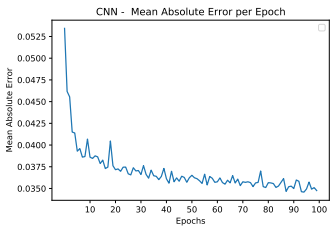


i. MAE

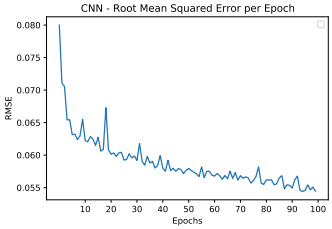


ii. RMSE

(**b**)

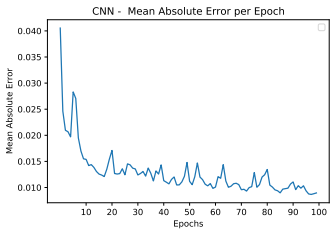


i. MAE

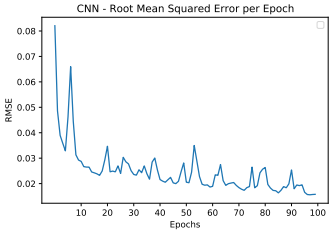


ii. RMSE

(**c**)

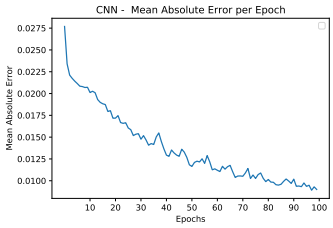


i. MAE

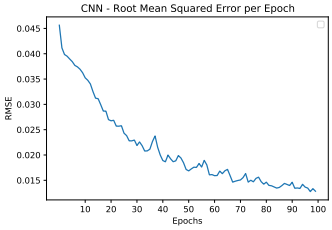


ii. RMSE

(**d**)

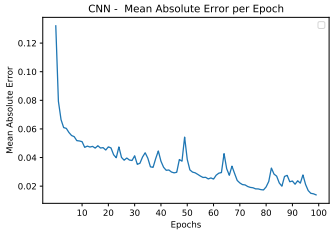


i. MAE

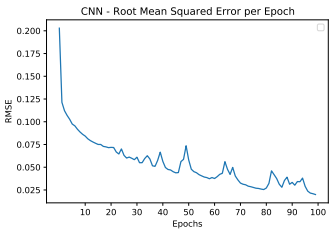


ii. RMSE

(**e**)



i. MAE



ii. RMSE

(**f**)

Figure 12. Metrics for the CNN Model. (**a**) Error Metrics for ANL-Intrepid-2009-1 Dataset. (**b**) Error Metrics for CEA-Curie-2011-2 Dataset. (**c**) Error Metrics for CIEMAT-Euler-2008-1 Dataset. (**d**) Error Metrics for KIT-FH2-2016-1 Dataset. (**e**) Error Metrics for METACENTRUM-2013-3 Dataset. (**f**) Error Metrics for UniLu-Gaia-2014-2 Dataset.

7. Discussion

The first thing of note is the fact that almost all the models start with higher MAE and RMSE values for the first 10 epochs or so. This is understandable as the neural network is attempting to optimize the weights initially.

With the exception of `i` and `ii` of datasets `(a)` and `(b)` of Figure 9 along with `i` of datasets `(a)` of Figure 10, all the other graphs indicate a general downward slope as the epochs progress. This indicates the model successfully minimizing the error metrics over successive iterations. Figure 9, being the RNN, according to our literature, has a generally weaker performance over extended periods. This could be the result of the `ANL-Intrepid-2009-1 (a)` possessing rapid fluctuations of peaks and lows over time as indicated in `(a)` of Figure 7.

As discussed in Section 6.1, the relationship between MAE and RMSE indicates the rate of individual errors in the sample. Except for `i` and `ii` of datasets `(a)` and `(b)` of Figure 9 along with `i` of datasets `(a)` of Figure 10, all the other graphs display similar trends in the error rate over the epochs. Therefore, we can conclude the individual error rate is lower since there is less variance between the MAE and RMSE values.

The LSTM model from Figure 11 shows the best results yet compared to the others; reinforcing the fact that they had time series data over a long period of time better. The RNN model from Figure 9 displays the worst performance compared to the others for the reasons mentioned above. Interestingly, the CNN model, which is usually used for interpreting graphical information, has also performed well. Going by the results represented by the graphs, LSTMs have the best performance.

8. Conclusions

This paper achieves the objective of bringing perspective to the domain of workload prediction in cloud computing environments using DL by classifying and systematically reviewing the existing literature. The findings have been broken down into cross sections detailing the taxonomy, strengths, limitations, models used, metrics used, architecture, datasets used and features analyzed.

Based on the models analyzed, it is evident that almost all the analyzed models aim to enhance the prediction accuracy by improving the model training, including adaptive strategies to improve resource provisioning or train the models more to predict multiple correlations of VMs. We also make a case for the technical analysis where the result of the DL model can be biased based on the quality of the datasets and uniformity is needed for unbiased analysis. The data from the parallel workloads archive were sanitized for the sake of removing anomalies or erroneous data that have not been the focus behind the other DL works explored in this domain. In this paper, we used datasets of the SWF format, which, to our knowledge, has not been analyzed in the context of the same DL model utilized on different datasets.

Our study has found that the LSTM model in comparison to the other models seems to have superior performance. It performs well in retaining information over lengthy time windows. We hope that our finding is a stepping stone in further enhancing the domain to enable effective and efficient resource management in cloud computing platforms. We hope the findings would help researchers in deploying DL models in production or live systems where the accurate prediction results in less resource wastage. This would lead to more efficient systems, less cost, and finally, a smaller carbon footprint when running data centers.

9. Future Work

DL is a rapidly growing field of study. While it has gained significant interest in recent years, avenues are abundant and yet to be explored. In its entirety, the models analyzed seem to have their drawbacks in terms of the limited combinations for the hybrid models, the lack of information on the simulation environment, and the limited choice of datasets mainly based on ease of access. In our future work, we will consider addressing these

matters in-depth, in the hopes of finding an efficient model that can best implement the principles of DL for resolving the challenge of efficient workload prediction. We would also consider training for larger iterations to further improve the models. We could also explore various ways of increasing the prediction window as well. All in all, the field of deep learning will not wane in research interest in the near future.

Author Contributions: Conceptualization, Z.A., M.K. and F.E.; Methodology, Z.A., M.K. and F.E.; Software, Z.A.; Validation, Z.A., F.A. and A.S.A.-M.A.-G.; Formal analysis, Z.A.; Investigation, Z.A.; Resources, M.K., F.E. and F.A.; Data curation, Z.A.; Writing—original draft, Z.A.; Writing—review and editing, Z.A., M.K., F.E. and A.S.A.-M.A.-G.; Visualization, Z.A.; Supervision, M.K., F.E., F.A. and A.S.A.-M.A.-G.; Project administration, M.K., F.E., F.A. and A.S.A.-M.A.-G.; Funding acquisition, M.K., F.E., F.A. and A.S.A.-M.A.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is a part of a project funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (KEP-PHD-21-611-42). The authors, therefore, gratefully acknowledge the DSR technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here (Parallel workloads archive): https://www.cs.huji.ac.il/labs/parallel/workload/logs.html.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DL	Deep Learning
RM	Resource Management
SWF	Standard Workload Format
NN	Neural Network
ML	Machine Learning
CDC	Cloud Data Center
SLA	Service Level Agreement
VM	Virtual Machine
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAPE	Mean Absolute Percentage Error
MAE	Mean Absolute Error
AR	Auto Regression
MA	Moving Average
SVR	Support Vector Regression
RVM	Relevance Vector Machines
PSO	Particle Swarm Optimization
CNN	Convolutional Neural Network
LSTM	Long-Short Term Memory
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
MLP	Multilayer Perception
IMF	Intrinsic Mode Functions
QoS	Quality of Service

References

1. Mustafa, S.; Nazir, B.; Hayat, A.; Madani, S.A. Resource management in cloud computing: Taxonomy, prospects, and challenges. *Comput. Electr. Eng.* **2015**, *47*, 186–203.
2. Parikh, S.M.; Patel, N.M.; Prajapati, H.B. Resource Management in Cloud Computing: Classification and Taxonomy. *arXiv* **2017**, arXiv:1703.00374.
3. Masdari, M.; Khoshnevis, A. A survey and classification of the workload forecasting methods in cloud computing. *Clust. Comput.* **2020**, *23*, 2399–2424.
4. Yazdanian, P.; Sharifian, S. E2LG: A multiscale ensemble of LSTM/GAN deep learning architecture for multistep-ahead cloud workload prediction. *J. Supercomput.* **2021**, *77*, 11052–11082.
5. Gill, S.S.; Garraghan, P.; Stankovski, V.; Casale, G.; Thulasiram, R.K.; Ghosh, S.K.; Ramamohanarao, K.; Buyya, R. Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge. *J. Syst. Softw.* **2019**, *155*, 104–129.
6. Marinescu, D.C. *Cloud Computing: Theory and Practice*; Morgan Kaufmann Publishers: Waltham, MA, USA; Elsevier: Amsterdam, The Netherlands, 2018.
7. Radhika, E.; Sadasivam, G.S. A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment. *Mater. Today Proc.* **2021**, *45*, 2793–2800.
8. Alaei, N.; Safi-Esfahani, F. RePro-Active: A reactive–proactive scheduling method based on simulation in cloud computing. *J. Supercomput.* **2018**, *74*, 801–829.
9. Bouabdallah, R.; Lajmi, S.; Ghedira, K. Use of reactive and proactive elasticity to adjust resources provisioning in the cloud provider. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications, IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, NSW, Australia, 12–14 December 2016; pp. 1155–1162.
10. Kumar, J.; Singh, A.K. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Gener. Comput. Syst.* **2018**, *81*, 41–52.
11. Vashistha, A.; Verma, P. A literature review and taxonomy on workload prediction in cloud data center. In Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 29–31 January 2020; pp. 415–420.
12. Calheiros, R.N.; Masoumi, E.; Ranjan, R.; Buyya, R. Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Trans. Cloud Comput.* **2014**, *3*, 449–458.
13. Espadoto, M.; Hirata, N.S.T.; Telea, A.C. Deep learning multidimensional projections. *Inf. Vis.* **2020**, *19*, 247–269.
14. Chen, Z.; Hu, J.; Min, G.; Zomaya, A.Y.; El-Ghazawi, T. Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *31*, 923–934.
15. Qiu, F.; Zhang, B.; Guo, J. A deep learning approach for VM workload prediction in the cloud. In Proceedings of the 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Shanghai, China, 30 May–1 June 2016; pp. 319–324.
16. Zhang, Q.; Yang, L.T.; Yan, Z.; Chen, Z.; Li, P. An efficient deep learning model to predict cloud workload for industry informatics. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3170–3178.
17. Ruan, L.; Bai, Y.; Li, S.; He, S.; Xiao, L. Workload time series prediction in storage systems: A deep learning based approach. In *Cluster Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 1–11. https://doi.org/10.1007/s10586-020-03214-y
18. Tang, X. Large-scale computing systems workload prediction using parallel improved LSTM neural network. *IEEE Access* **2019**, *7*, 40525–40533.
19. Feitelson, D.G.; Tsafrir, D. Workload sanitation for performance evaluation. In Proceedings of the 2006 IEEE International Symposium on Performance Analysis of Systems and Software, Austin, TX, USA, 9–21 March 2006; pp. 221–230.
20. Tsafrir, D.; Feitelson, D.G. Instability in parallel job scheduling simulation: The role of workload flurries. In Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes, Greece, 5–29 April 2006; p. 10.
21. Gupta, N.; Patel, H.; Afzal, S.; Panwar, N.; Mittal, R.S.; Guttula, S.; Jain, A.; Nagalapatti, L.; Mehta, S.; Hans, S.; et al. Data Quality Toolkit: Automatic assessment of data quality and remediation for machine learning datasets. *arXiv* **2021**, arXiv:2108.05935.
22. Amiri, M.; Mohammad-Khanli, L. Survey on prediction models of applications for resources provisioning in cloud. *J. Netw. Comput. Appl.* **2017**, *82*, 93–113.
23. Kumar, J.; Goomer, R.; Singh, A.K. Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. *Procedia Comput. Sci.* **2018**, *125*, 676–682.
24. Zhu, Y.; Zhang, W.; Chen, Y.; Gao, H. A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 274.
25. Gao, J.; Wang, H.; Shen, H. Machine learning based workload prediction in cloud computing. In Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 3–6 August 2020; pp. 1–9.
26. Bi, J.; Li, S.; Yuan, H.; Zhao, Z.; Liu, H. Deep neural networks for predicting task time series in cloud computing systems. In Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 9–11 May 2019; pp. 86–91.

27. Bi, J.; Yuan, H.; Zhang, L.; Zhang, J. SGW-SCN: An integrated machine learning approach for workload forecasting in geo-distributed cloud data centers. *Inf. Sci.* **2019**, *481*, 57–68.
28. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50.
29. De Myttenaere, A.; Golden, B.; Le Grand, B.; Rossi, F. Mean absolute percentage error for regression models. *Neurocomputing* **2016**, *192*, 38–48.
30. Kelley, K.; Lai, K. Accuracy in parameter estimation for the root mean square error of approximation: Sample size planning for narrow confidence intervals. *Multivar. Behav. Res.* **2011**, *46*, 1–32.
31. Sammut, C.; Webb, G.I. Mean squared error. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2010; p. 653.
32. Bandyopadhyay, B.; Bandyopadhyay, S.; Bedathur, S.; Gupta, N.; Mehta, S.; Mujumdar, S.; Parthasarathy, S.; Patel, H. 1st International Workshop on Data Assessment and Readiness for AI. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*; WSPA, MLMEIN, SDPRA, DARAI, and AI4EPT; Springer: Delhi, India 2021; pp. 117–120.
33. Jain, A.; Patel, H.; Nagalapatti, L.; Gupta, N.; Mehta, S.; Guttula, S.; Mujumdar, S.; Afzal, S.; Sharma Mittal, R.; Munigala, V. Overview and importance of data quality for machine learning tasks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 3561–3562.
34. Patel, H.; Ishikawa, F.; Berti-Equille, L.; Gupta, N.; Mehta, S.; Masuda, S.; Mujumdar, S.; Afzal, S.; Bedathur, S.; Nishi, Y. 2nd International Workshop on Data Quality Assessment for Machine Learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, Singapore, 14–18 August 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 4147–4148. https://doi.org/10.1145/3447548.3469468.
35. Gupta, N.; Mujumdar, S.; Patel, H.; Masuda, S.; Panwar, N.; Bandyopadhyay, S.; Mehta, S.; Guttula, S.; Afzal, S.; Sharma Mittal, R.; et al. Data quality for machine learning tasks. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 4040–4041.
36. Feitelson, D.G.; Tsafrir, D.; Krakov, D. Experience with using the parallel workloads archive. *J. Parallel Distrib. Comput.* **2014**, *74*, 2967–2982.
37. Cirne, W.; Berman, F. A comprehensive model of the supercomputer workload. In Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization, WWC-4 (Cat. No. 01EX538), Austin, TX, USA, 2 December 2001; pp. 140–148.
38. McKinney, W. Pandas: A foundational Python library for data analysis and statistics. *Python High Perform. Sci. Comput.* **2011**, *14*, 1–9.
39. Benesty, J.; Chen, J.; Huang, Y.; Cohen, I. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–4.
40. Ranjbari, M.; Torkestani, J.A. A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers. *J. Parallel Distribut. Comput.* **2018**, *113*, 55–62.
41. Cortez, E.; Bonde, A.; Muzio, A.; Russinovich, M.; Fontoura, M.; Bianchini, R. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28–31 October 2017; pp. 153–167.
42. Delimitrou, C.; Kozyrakis, C. Paragon: QoS-aware scheduling for heterogeneous datacenters. *ACM SIGPLAN Not.* **2013**, *48*, 77–88.
43. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 10 December 2022).
44. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
45. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: tensorflow.org (accessed on 3 December 2022).
46. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2.
47. Fulcher, B.D. Feature-based time-series analysis. In *Feature Engineering for Machine Learning and Data Analytics*; CRC Press: Boca Raton, FL, USA, 2018.
48. Vishwakarma, G.; Sonpal, A.; Hachmann, J. Metrics for benchmarking and uncertainty quantification: Quality, applicability, and best practices for machine learning in chemistry. *Trends Chem.* **2021**, *3*, 146–156.
49. Syntetos, A.A.; Boylan, J.E. The accuracy of intermittent demand estimates. *Int. J. Forecast.* **2005**, *21*, 303–314.