Multivariate Time Series Synthesis Using Generative Adversarial Networks

Mark Leznik
Institute of Information Resource Management
Ulm University
mark.leznik@uni-ulm.de

Patrick Michalsky
Institute of Information Resource Management
Ulm University, Germany
patrick.michalsky@uni-ulm.de

Peter Willis
BT Applied Research
Ipswich, UK
peter.j.willis@bt.com

Benjamin Schanzel
Institute of Information Resource Management
Ulm University
benjamin.schanzel@uni-ulm.de

Per-Olov Östberg
Department of Computing Science
Umeå University, Sweden
p-o@cs.umu.se

Jörg Domaschka
Institute of Information Resource Management
Ulm University, Germany
joerg.domaschka@uni-ulm.de

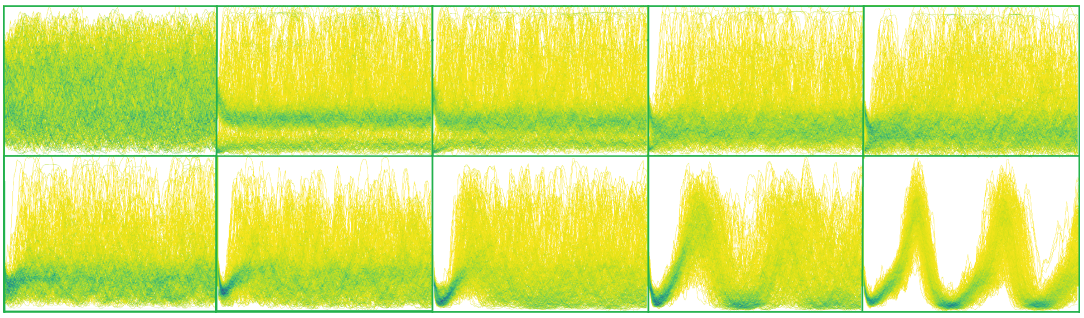

Figure 1: From left to right, top to bottom, the training process of our GAN network is depicted. The generated data is plotted as a density chart throughout the training process, showing how the network learns to reflect the fidelity of the original data.

ABSTRACT

Collection and analysis of distributed (cloud) computing workloads allows for a deeper understanding of user and system behavior and is necessary for efficient operation of infrastructures and applications. The availability of such workload data is however often limited as most cloud infrastructures are commercially operated and monitoring data is considered proprietary or falls under GPDR regulations. This work investigates the generation of synthetic workloads using Generative Adversarial Networks and addresses a current need for more data and better tools for workload generation. Resource utilization measurements such as the utilization rates of Content Delivery Network (CDN) caches are generated and a comparative evaluation pipeline using descriptive statistics and time-series analysis is developed to assess the statistical similarity of generated and measured workloads. We use CDN data open sourced by us in a data generation pipeline as well as back-end ISP workload data to demonstrate the multivariate synthesis capability of our approach. The work contributes a generation method for multivariate time series workload generation that can provide arbitrary amounts of statistically similar data sets based on small subsets of real data. The presented technique shows promising results, in particular for heterogeneous workloads not too irregular in temporal behavior.

ICPE '21, April 19-23, 2021, Virtual Event, France.
ACM ISBN 978-1-4503-8194-9/21/04.
https://doi.org/10.1145/3427921.3450257

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Modeling methodologies**; • **Information systems** → *Traffic analysis*; • **Networks** → *Network measurement*; *Network simulations*; • **Security and privacy** → *Privacy protections.*

KEYWORDS

time series synthesis; workload generation; server workload; synthetic data; generative adversarial networks

ACM Reference Format:
Mark Leznik, Patrick Michalsky, Peter Willis, Benjamin Schanzel, Per-Olov Östberg, and Jörg Domaschka. 2021. Multivariate Time Series Synthesis Using Generative Adversarial Networks. In *Proceedings of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3427921.3450257

1 INTRODUCTION

The rise of automated infrastructure monitoring solutions and system surveillance tools, as well as the boom of the Internet of Things (IoT) technologies have provide vast amounts of data over the last decade. Nevertheless, for development, evaluation and profiling purposes, researchers still rely on artificial data [16]. In the research field around cloud and edge computing, monitoring data for virtual and physical infrastructures is crucial for load balancing, load prediction and smart provisioning algorithms [28].

However, while some public data sets have been contributed and enable research around this area [7] [1], the vast majority of all data produced in these environments (i.e. by commercial cloud service providers) are generally regarded confidential for reasons of either protecting proprietary intellectual property or compliance with data regulations such as the General Data Protection Regulation (GDPR).

Synthetically generated data offers the ability to supplement or generate missing data. For these reasons, algorithm evaluations and analysis often rely on artificial data to test systems in the scale and scenarios intended.

For example, imputation, the replacement of missing values in observations, relies on generating substitution measurements, ideally while retaining the original characteristics of the data at hand.

Additionally, *what-if* analysis of systems, offering a larger amount of measurements than are available can be performed. Simply put, the question of how a workload would look with double the amount of users or machines can be answered by artificially generated data.

Equally important, researchers working in the field of cloud and telecommunication rely on real or realistic data, which, as mentioned above is very scarce due to regulations and corporate secrecy. None of these challenges and issues regarding data availability in the cloud computing context in regards to data center design, multi-tenancy configurations and resource provisioning are new, as can be seen by work stemming back to 2011 [35]. While the cloud and resource sharing paradigm currently makes up a multi-billion dollar industry [21], the open source data problem in it is yet to be addressed properly.

In this work, we propose an approach for generating arbitrary amounts of multivariate time series workload data using a Generative Adversarial Network (GAN). We demonstrate our solution on real world production data of content delivery network (CDN) workloads and corresponding back-end traffic utilization. We further evaluate our approach using statistical parameters and time series analysis comparison of the original and generated data.

The workload definition should be briefly discussed here. In our case, we consider the workload *arisen* on a particular system from an *applied* workload to it by e.g., a user. Hence, if not specifically stated otherwise, the further use of the workload term refers to workload arisen in a given system.

Our main contributions lie in: (i) providing an approach for synthesizing arbitrary amounts of multivariate time series data; (ii) open sourcing a large amount of previously unavailable CDN data from a production environment; (iii) providing the comparative means for assessing similarity between multivariate time series data; (iv) an exemplary scenario of synthetic data use for time series prediction.

The remainder of this work is structured as follows: Section 2 provides an overview of research related to the subject, Section 3 introduces our approach. Section 4 depicts the descriptive statistics and time series analysis applied to the data for a ground-truth to synthetic data comparison. Section 5 presents and discusses our results, showing how the generated data closely mimics the underlying ground-truth data. We conclude this work with a summary and an outlook in Section 6.

2 RELATED WORK

Time series data generation is as an established discipline when it comes to approaches purely based statistics. Extrapolation, missing data interpolation and to some extent simulation can also be regarded as synthetic data generation. These three approaches can be further subdivided into model-based and imputation based techniques. Model-based algorithms perform model fitting for the missing data based on likelihood and are more suited for multivariate data, while direct imputation, being the simpler approach are used with univariate time series [3].

2.1 Simulation

Simulating the observed user or system behaviour to generate workload is mostly done for the purpose of performance analysis of hardware and software systems under certain workload scenarios. The commonality hereby is a preliminary workload model derived from the observed workload, reflecting characteristics of the user's behavior.

These characteristics can include descriptive statistical measures and probability distributions as well as measures of the burstiness and self-similarity of the workloads. An algorithm is then fitted to simulate tasks or requests in such a way that these characteristics are preserved. Several out-of-the-box tools are available for workload generation with a focus on interactive web applications (e.g. httperf [27], ProWGen [6]. A comprehensive survey of such workload generation tools can be found in [8]. While these publications and tools focus on interactive web-based applications, there are also publications focusing on data-intensive applications, e.g., MapReduce [11].

Contrary to these task-focused approaches, our work does not search to simulate the behavior of users directly, but the resource utilization metrics that result from input workloads.

2.2 Workload Generation using Neural Networks

The generation of synthetic time-series using Artificial Neural Networks (ANN) is subject to several publications [1, 5, 30]. In the following, we discuss approaches that are closely related to ours, be it technology-wise (the architecture of the ANN) or scope-wise (the application domain).

Generative adversarial networks (GANs), being a subdomain of ANNs, have been used in financial applications [36] to generate large amounts of univariate time series data while capturing long term temporal dependencies within the measurements. In the

[1] http://ai.googleblog.com/2011/11/more-google-cluster-data.html

medical domain, they have been used to generate sinusoidal electrocardiogram and photoplethysmogram single channel time series data as images [10]. The Philips database [29], consisting of medical measurements taken at intensive care units from around 200,000 patients, is also used [9] as a baseline for synthesising medical data to overcome the bottleneck of publicly available data, while also presenting a thorough evaluation framework, which is not only suitable for the specific case of medical data but could also be a suitable method for evaluating GANs in general. Lin et al. [22] have proposed the *Doppelganger* approach, utilizing so-called Wasserstein GANs to synthesize time-series data. The fidelity of the generated data is also evaluated in this work in terms of capturing short term and long term temporal correlations in the time series. Recurrent Neural Networks (RNNs) have been used for synthesizing classical music [24], which when can be transcribed as multivariate time-series.

2.3 Data Validation

Validation is a crucial aspect of workload and time series synthesis, as it defines the success of an approach. The literature here presents many different approaches dependent on the application domain and goals. Visual examination of generated samples is found [29] just as the measurement of the distribution of the generated data, for instance via the squared Maximum Mean Discrepancy (MMD) between generated and real samples [29]. Further metrics include short term and long term temporal correlations in the time series [22]. A domain-specific comparison for music generation applies metrics such as the polyphony, measuring how often at least two tones are played simultaneously [24]. In earlier work [19] we have used the entropy, approximate entropy and correlation in the evaluation part.

2.4 Summary

The majority of ANN-based work on this topic is very recent. It is motivated by the success of GANs in the field of image generation [4]. Yet, while showing promising results, many approaches are often limited either in the output length or the dimensionality of the data. Further, the lack of guidelines in respect to validation and evaluation of the results must be noted as a hindrance for both reproducibility and more importantly, generalizability. This research question is also present in the image generation field, where metrics such as the Inception Score and Frechet Inception Distance have been introduced to tackle the issue [4].

3 METHODOLOGY & APPROACH

A GAN consists of two artificial neural networks: a *discriminator* and a *generator*, competing in min-max game [13]. A discriminative network D is trained to distinguish real from fake data by being alternately presented with sequences from both classes and thereby fitted to assign labels for real and fake data. The competing generator network G produces fake sequences by drawing from a uniform or normal distribution and generating synthetic data. This process is visible in Fig. 1, where the initial results of the generator are just noise, but after several epochs the distribution of the training data is more and more apparent. It then tries to fool D in each training step by presenting fake data together with the real class label. By this means, the generator learns, over the course of training, to generate sequences that D is not able to distinguish from real data. Fitting the described networks in an adversarial setting allows to generate an arbitrary amount of time series data that is statistically similar to real data.

The training goal of a GAN is to estimate the probability distribution of the training data and to generate synthetic samples drawn from that distribution. Hence, the goal of our GAN is to learn the probability distribution of the real production workload and be able to generate statistically similar time series data. The ability of the GAN which in our case is specifically modeled to operate on time series data is to learn and reflect important features of real data highly depends on the quality of the training samples. A GAN is considered successful if the generator is able to capture the distribution of the training data but is not explicitly trained on this goal [13].

If the empirical probability distributions of the training samples are too diverse, the training can fail, so that the GAN only learns to generate samples from a single probability distribution, neglecting the other distributions. This failure scenario is called mode collapse wherein the GAN fails to capture the diversity of the training data [31].

We solve the problem of mode collapse by training separate GANs for sequences of different distributions. While this implies a higher computational overhead, we tackle this by introducing an automated containerized workflow.

We use a reproducible training approach we have devised in [32], by utilizing container wrappers for both code and data. This enhances the reproducibility and reusability of the codebase and allows for cross platform training as well as versioning of the data sets used. By extent, since our approach is domain agnostic, this also increases the generality.

There are several other viable solutions to circumventing the mode collapse in GANs. Lin et al. [22] have opted for the use of a Wasserstein GAN [2] to solve this problem, we have seen a very large training time per network even when compared to our separation approach. Additionally, mode collapse can also be tackled by so-called unrolled GANs, as shown in [23].

The GAN developed we use in our work is derived from existing architectures found in publications with similar goals. Specifically, as discussed in Section 2, [9] and [24] present promising architectures for the generation of continuous time series data. In both of these publications, RNNs are incorporated to capture temporal dependencies in the training data and reflect these in the generated sequences.

Preparation of real data sets to effectively serve as training samples is the first step in our approach. In the following the process of synthesizing multivariate time series data is described, beginning with the preprocessing of the data at hand, followed by the GAN used for the data generation.

3.1 Preprocessing

Typical preprocessing steps include the transformation of raw data into the desired format, filtering of incomplete samples, and scaling of features to a defined range.

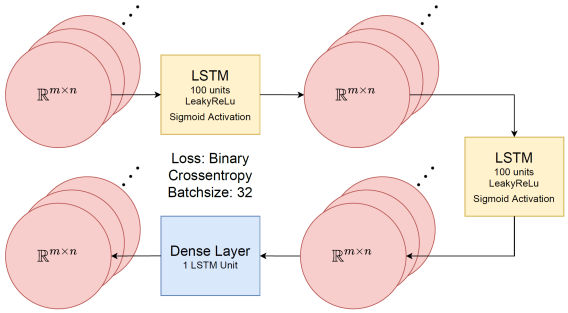

Figure 2: Generator G

Data normalization is applied to fit samples into a $[0, 1]$ range. This is a crucial preparation step for many applications of ANNs and significantly accelerates the gradient descent process [34]. The length of the input sequences is a fixed parameter of our developed GAN model, hence, the training data need to consist of samples with an equal number of time-steps. In the case of too long input samples, several options are available, the input can be either chunked into smaller portions, or if feasible without any loss of data quality, a sampling rate reduction can also be performed. These preprocessing steps are incorporated as an essential part of the overall approach to generate synthetic workload traces. This process does not significantly increase overall computation time as it is only being performed if the input data changes (we present runtime numbers in Section 5).

3.2 A Generative Adversarial Network for Multivariate Time Series Synthesis

In the following, we outline the GAN architecture and the structure of its input and output data. The discriminator (shown in detail in Figure 3) used in this approach consists of two layers of Long Short-Term Memory (LSTM) units, followed by an output layer of one LSTM unit for the final classification, resulting in a single scalar value $y \in [0; 1]$.

The output of the last hidden recurrent layer is directly fed into the single-unit output layer of the discriminator. Both LSTM layers

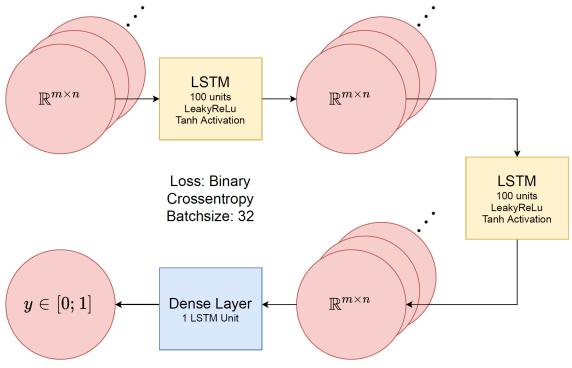

Figure 3: Discriminator D

consist of a n by m (where the former is the number of time-steps and the latter the number of features) input shape.

Likewise, the generator, depicted in Figure 2, consists of two recurrent layers. It is initialised with Gaussian noise. One fully-connected output layer follows the last recurrent layer with one unit per time step. Having two recurrent layers and an additional fully-connected output layer shows better results in our experiments. The network consists of two bidirectional LSTM layers with the same n by m shape as the generator, a global average pooling over the m features, followed by a dense output layer.

The input and correspondingly output length of the GAN is limited only by the utilized hardware, and currently allows for the input of ca. 15000 steps for univariate time-steps. The addition of another dimension to the data divides the number of time-steps by the dimension accordingly.

The length of approximately 20,000 time-steps of the data set, as outlined in Section 5 ,imposes very high computational requirements for the execution of the GAN training. In our setup, one neuron per layer is used for each time-step of the sequences (cf. Figures 2 and 3). By that means, the discriminator and generator networks would need to be 20,000 units in width. With such a large network, the training process could not be executed in a time-efficient manner, if at all. Therefore, the training samples are downsampled in a preprocessing step to a feasible length.

4 COMPARING SYNTHETIC AND REAL TIME SERIES DATA

The focus of our work lies on the generation of multivariate time series data. Sequences of resource utilization metrics are synthesized which have similar statistical characteristics when compared to real sequences. The measurement of such statistical similarity is covered in the following. It outlines which statistics are computed and explains their semantics. Features from the field of descriptive statistics and time-series analysis used in the literature [12, 17, 18, 37] are described.

4.1 Descriptive Statistics

Summarizing the values of variables in the sense of descriptive statistics allows inferring insights about their distribution and tendencies. We use the mean and the standard deviation for this purpose.

It should be noted that the analyzed time series are samples from an unknown distribution and the means are calculated on these samples, it is the sample mean that is evaluated, and thus is an approximation of the true mean value of the unspecified distribution (i.e. the population mean) [25].

To gain a better understanding of possible outliers in the data and get a measure of how far values spread from the average in general, the standard deviation σ is calculated.

4.2 Time-Series Analysis

Time series data cannot be assumed to be independently distributed. It must instead be assumed that values, observed at any point in time, depend to some degree on previously observed values [33]. This characteristic of time-series data leads to statistical measures that explore temporal dependencies in and across sequences.

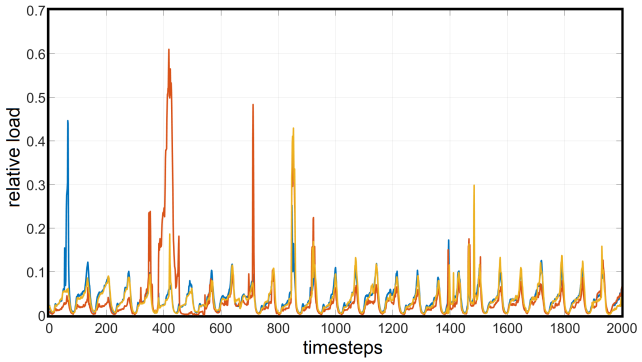

Figure 4: 2,000 downsampled timestamps of the original cache served measurements for 3 inner-core sites at BT's network.

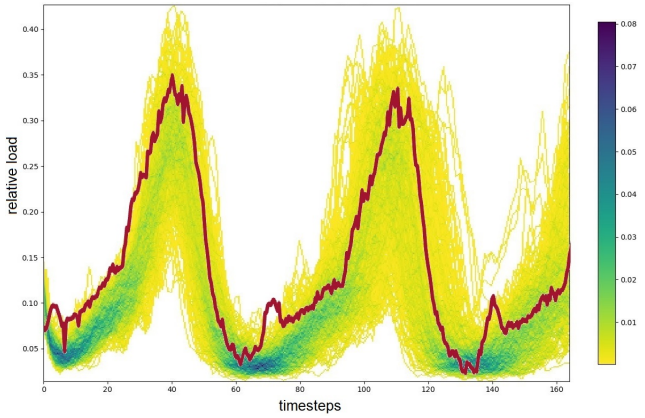

Figure 5: A density plot of 160 time-steps of 100 time series plotted together to illustrate our model capturing the seasonality of the cache serving patterns in the data. Darker colour indicates more data points. Superimposed in red, is an original measurement of the training data as a comparison.

To test whether synthetic workload traces capture not only the distribution but also reflect temporal dependencies of real data, the correlation is calculated. For univariate time series, the auto correlation function may also be of assistance, however, given the focus on multivariate calculations, we opt for the correlation between the multivariate data samples. This allows to evaluate the capability of the GAN to capture dependencies between the time series and reflect those in the generated data.

To assess the common trends between the ground truth and the synthesized data, we additionally use cointegration tests, namely the Engle-Granger test [14] and the Johansen test [15]. The Engle-Granger tests for individual cointegration relationships, while the Johansen test assesses multiple relationships. Ideally, for both these tests, a p-value lower than 0.05 should reject the null hypothesis and indicate common trends between two time series. This is an essential calculation, since time series moving in different directions and following different trends, could still have a similar mean and entropy value and hence only feign similarity. Cointegration in this case helps to identify if two time series would drift apart or exhibit common behaviour.

Whether the generation approach is able to produce realistic samples is justified by the comparative evaluation of the statistical characterization. Furthermore, the described workflow is implemented with a focus on automation and reproducibility and is also containerized in a way that generator results can directly be evaluated.

In the following we first describe the data used in our approach, succeeded by the results and their evaluation using the metrics outlined in Section 4.

5 EVALUATION

5.1 CDN and Back-end Utilization Data

We open source the CDN data used in our approach as a publicly available artifact on Zenodo [20] [2]. The data includes the traffic of a Content Delivery Network from British Telecom (BT) in the United Kingdom (UK) and is described in the following.

[2]We released the data as part of an 8GB large artificial and real data compendium of cloud related data sets, the CDN data is denoted data set $D1$

The data comes from three caches located in inner-core nodes of the BT network in London, UK. The time series consists of three dimensions per measurement point, namely the load at a specific inner-core node, all measured in bits per second. The data set was sampled with an interval of 20 minutes in a time span from 2016 to 2017 leading to almost 20 thousand measurement points.

Due to its proprietary nature, only portions of data traces from 2016 to 2017 are publicly open. In the open sourced sample, timestamps are omitted and data points are divided by a peak value to give a relative traffic measure hiding the absolute traffic level. The range of this relative traffic level is between 0 and 1. The measurements, downsampled to 2,000 time-steps can be seen in Figure 4. This downsampling was performed for the purposes of visualization. For data synthesis a higher sampling rate was used.

To demonstrate the capability of our approach to capture temporal correlation between different time series, we additionally introduce the overall back-end utilization for the three inner-core nodes where the CDNs are located, namely inbound and outbound traffic measurements. The timestamps and unit of measurement are the same and also in the form of relative traffic. While this data is not openly available, this demonstrates our usecase scenario, since it offers the possibility of generating data which represents system behaviour without exposing sensitive information. Using this data we generate a three dimensional time series with ca. 4,800 time-steps in each dimension, and synthesize 100 instances of such time series, totalling 1.5 million data points. How the generation method is able to produce samples that statistically and visually fit into the real server workloads is assessed in the following section.

5.2 Results and Evaluation of Synthetic Workloads

The GAN is trained with the same parameters for 500 epochs. We first perform a visual assessment of the data using both regular plots and density plots introduced in [26], which allow us to evaluate a

large amount of time series simultaneously. We further look at the statistical evaluation of the data described in Section 4.

A look at the density chart displaying a zoomed in portion of the data in Figure 5 shows how our trained model is able to capture the seasonal patterns in the cache served data while simultaneously producing data which is similar in its characteristics but not identical. Superimposed in red is a ground truth measurement of the original training data, aiding in the comparison. The original measurement fits well within the distribution of the synthesized data. The yellow points clustered around the major blue significant hotspots show minor deviations and noise introduced by the generator. This however, does not mean this variance worsens the statistical properties of the data, as shown below.

Figure 6 depicts one of the generated series in its entire length for all three dimensions. Here, again not only the seasonal behaviour of the data is captured, but also the dependency between the time series. Higher cache serving increases outgoing traffic which is reflected in the measurements generated by our model.

The mean and the corresponding average mean values in Table 1 of the original data versus the generated data shows deviations of max. 2% between them, thus suggesting similar data points. Additionally, the standard deviation also shows no major deviations between the original data and the synthesized values, with the outgoing traffic being one exception: the higher value could indicate the introduction of some outliers and noise into the measurement, this is also supported by the slightly higher mean value.

The entropy values suggest a higher information count in the generated time series, which again, can also be attributed to noise. This is where the calculation of the approximate entropy comes in. The decrease in the approximate entropy corresponds to a decrease in the irregularity of the time series, which proves the assumption of the increased entropy being a noise component, but also speaks for a more stable data pattern as a whole.

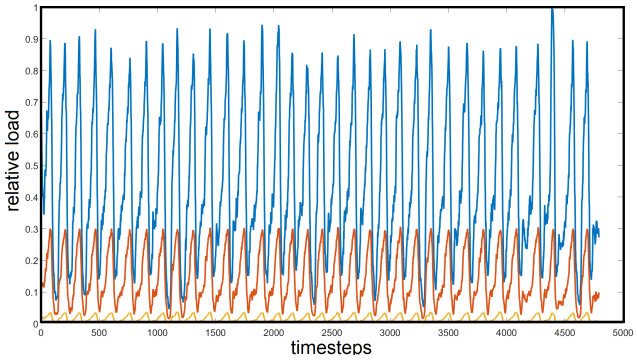

Figure 6: Synthesized cache served measurements, as well as back-end outgoing and incoming measurements for one three-dimensional time series consisting of 4,800 steps. Here again, the ability of the model to reflect the seasonal behaviour as well as the interdependency of the workload can be seen. The blue line shows the outgoing traffic, the red one the incoming and the yellow one at the bottom the cache serving.

The cointegration analysis also depicted in Table 1, shows the statistical significance of the Engle-Granger test with a p-value of 0.001 across all three time series. The same goes for the Johanson Test, which shows a lower then 0.05 p-value across all ranks. This shows that the pairs of real and synthetic data move in the same trend direction and hence, share an underlying common stochastic trend.

To further shows the viability of the data, we take a look at the correlation of the original multivariate measurements and the synthesized time series in Table 2. The correlation is calculated between the outgoing, inbound traffic and cache served traffic. The relation of the original underlying data should be kept here, e.g. that an increase in inbound traffic shall be followed by an increase in outbound traffic. This is specifically important and useful when looking at the behaviour of CDN systems and their optimizations. Here, the values are nearly identical for all measurements, namely strong positive correlation between the data.

Finally, we test the applicability of our approach with an exemplary real world use case. We use an off-the-shelf LSTM forecasting network [3] to perform predictions on the synthesized and original data. The model is first trained using an excerpt of the data, in production it then tries to predict future measurements taking into account already known (past) measurements. First, we train the network for 100 epochs using a part of the original ground truth data as training data, and then perform forecasting operations on ground truth testing data. Here the Root Mean Squared Error (RMSE) of the prediction is 0.1727. This is a measure of how far off on average the prediction was from the actually observed value, where a lower RMSE is better. We then use the generated synthetic data to train the network and perform the forecasting on the ground truth original testing data, the results of which are depicted in Figure 7. Due to the larger available amount of training data when used with the synthetical time series, the Root Mean Squared Error (RMSE) of the prediction is at 0.15893. This speaks for the statistical similarity of the synthetical data, as the prediction not only worked, but improved slightly. This demonstrates a use-case of our approach for pretraining models before production deployment using synthetic or evening out under-balanced classes with a scarce number of measurements in data sets.

5.3 Efficiency

We have evaluated and measured [4] the runtime performance of our model using two types of data. First the training and generation of the data was performed using 5000 data points of the CDN data set. Additionally, we have used randomly generated 5000 data points of time series data, which was subjected to the same tests. This was done in order to exclude the data characteristics and complexity as a factor. We have generated 5000 time-steps of univariate and multivariate three-dimensional data accordingly. The code was executed 20 times, the average measurements of all runs are presented in Table 3. On average, the training and generation of 5000 time-steps using the actual CDN data takes roughly 20-30 seconds longer.

[3] https://de.mathworks.com/help/deeplearning/ug/time-series-forecasting-using-deep-learning.html

[4] The measurements were performed on the Special Nodes of the Justus2 Cluster. https://wiki.bwhpc.de/e/Hardware_and_Architecture_(bwForCluster_JUSTUS_2)#Node_Specifications

Table 1: Mean and (average for 100 generated time series) standard deviation as well as (average) entropy and approximate entropy and the cointegration tests of the original and generated (indicated by _G) time series.

	Outgoing	Outgoing_G	Incoming	Incoming_G	Cache served	Cache served_G
Mean	0.4420	0.4618	0.1363	0.1541	0.0184	0.0174
(Avg.) Std. deviation	0.1987	0.2410	0.0754	0.0882	0.0088	0.0105
(Avg.) Entropy	7.4737	7.7003	7.6527	7.7854	7.5395	7.7628
(Avg.) Approximate Entropy	0.4331	0.2767	0.2665	0.2661	0.3443	0.2766
Engle-Granger Test (after 500 epochs)	$\rightarrow$	$p = 0.001$	$\rightarrow$	$p = 0.001$	$\rightarrow$	$p = 0.001$
Johansen Test (after 500 epochs)	$[r_0, p = 0.001$	$r_1, p = 0.0015]$	$[r_0, p = 0.001$	$r_1, p = 0.001]$	$[r_0, p = 0.001$	$r_1, p = 0.0015]$

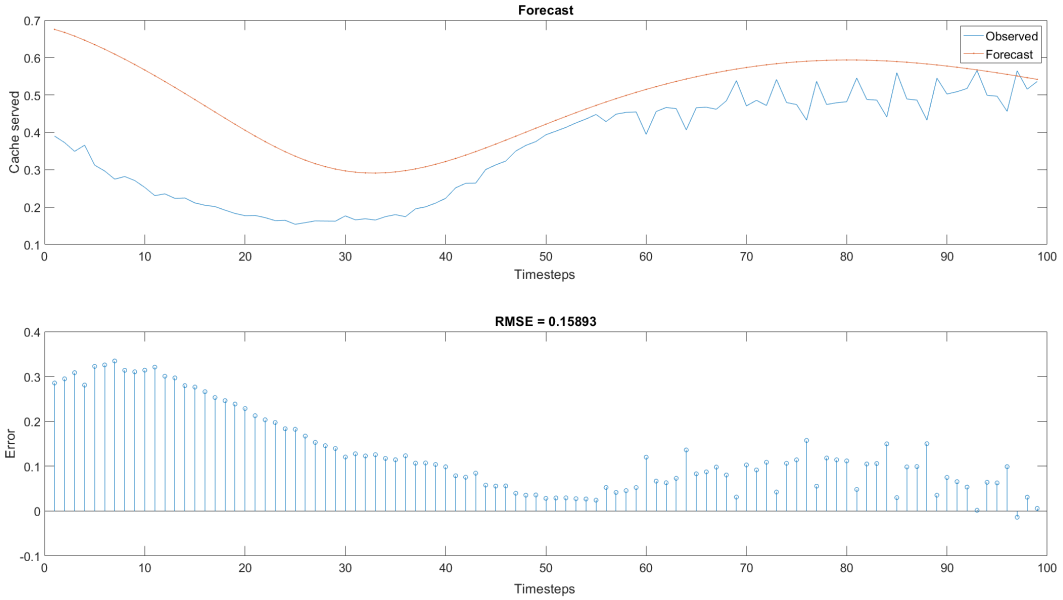

Figure 7: An exemplary forecasting application for 100 time-steps into the future, illustrated using the observed and predicted values (top), as well as the error margin measured by the RMSE (bottom).

Table 2: The correlation (left table) and average (for 100 generated time series) correlation (right table) of the original and generated (indicated by _G) time series.

	1.	2.	3.	1_G	_G	3_G
1. Outgoing, G	1	0.9810	0.9701	1	0.9839	0.9802
2. Incoming, G		1	0.9891		1	0.9882
3. Cache served, G			1			1

Table 3: The averaged runtime measurements of the data training and generation for the CDN and random data for univariate 5000 datapoints and 3 dimensional multivariate 5000 datapoints.

	5000/1 points	5000/3 points
Random data	26min 55s	27min 52s
CDN data	27min 31s	28min 11s

Hence, no significant change in the runtime can be attributed to the complexity of the data. Additionally, the measurements show that a dimensionality increase in the data only increases the runtime by ca. 2% (between 40-60 seconds longer).

6 CONCLUSION

The results of our work show that the GAN-based generation method for synthetic workloads is able to produce samples that are statistically similar to real workloads. A GAN is able to learn and reflect the characteristics of utilization workloads and generate similar samples where the similarity is measured in terms of descriptive statistics and temporal dependencies. By this standard, the synthetic data can be considered a plausible realistic representation of the original data. We also show an exemplary application of our approach to a forecasting application. The possibility of also using such a GAN solely for data obfuscation is equally viable.

The realism of synthetic workloads depends on the underlying processes that produce the considered resource utilization. Such an underlying process is the interplay of the behavior of the users

of a system and the specific resource demands of the involved applications and services. The performed statistical comparisons do not state, e.g., whether single peaks and troughs in utilizations are realistic from the behavioral point of view of a specific application or service that causes these utilizations. They do, however, measure their overall statistical similarity and show how synthetic and real samples arrange. Solving the problem of noisy and heterogeneous training data is an important direction of the GAN-based generation method. Additionally, other time series can be evaluated using the developed GAN and evaluation pipeline to test whether the chosen model parameters are suitable for data from entirely different domains. Other generation approaches can also be benchmarked using the developed method for data evaluation by replacing the GAN-based generator. The performance of the GAN can then be compared to other methods in a fixed framework.

ACKNOWLEDGMENTS

This work was supported by the Vector Stiftung. The authors thank NVIDIA for its generous donation of a TITAN V GPU. The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 40/575-1 FUGG (JUSTUS 2 cluster). The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany (grant nr. 01IS18068, SORRIR).

REFERENCES

[1] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. 2017. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 188–193.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.

[3] Neeraj Bokde, Marcus W Beck, Francisco Martínez Álvarez, and Kishore Kulat. 2018. A novel imputation methodology for time series based on pattern sequence forecasting. *Pattern recognition letters* 116 (2018), 88–96.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* (2018).

[5] Eoin Brophy, Zhengwei Wang, and Tomas E Ward. 2019. Quick and easy time series generation with established image-based gans. *arXiv preprint arXiv:1902.05624* (2019).

[6] Mudashiru Busari and Carey Williamson. 2002. ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches. *Computer Networks* 38, 6 (2002), 779–794.

[7] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 153–167.

[8] Mariela Curiel and Ana Pont. 2018. Workload generators for web-based systems: Characteristics, current status, and challenges. *IEEE Communications Surveys & Tutorials* 20, 2 (2018), 1526–1546.

[9] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633* (2017).

[10] Vincent Fortuin, Gunnar Rätsch, and Stephan Mandt. 2019. Multivariate time series imputation with variational autoencoders. *arXiv preprint arXiv:1907.04155* (2019), 67–73.

[11] Archana Ganapathi, Yanpei Chen, Armando Fox, Randy Katz, and David Patterson. 2010. Statistics-driven workload modeling for the cloud. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*. IEEE, 87–92.

[12] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. 2007. Workload analysis and demand prediction of enterprise data center applications. In *2007 IEEE 10th International Symposium on Workload Characterization*. IEEE, 171–180.

[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[14] Svend Hylleberg, Robert F Engle, Clive WJ Granger, and Byung Sam Yoo. 1990. Seasonal integration and cointegration. *Journal of econometrics* 44, 1-2 (1990), 215–238.

[15] Søren Johansen. 1992. Cointegration in partial systems and the efficiency of single-equation analysis. *Journal of econometrics* 52, 3 (1992), 389–402.

[16] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. 2019. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4551–4560.

[17] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. 2012. Workload characterization and prediction in the cloud: A multiple time series approach. In *2012 IEEE Network Operations and Management Symposium*. IEEE, 1287–1294.

[18] Andrzej Kochut and Kirk Beaty. 2007. On strategies for dynamic resource management in virtualized server environments. In *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE, 193–200.

[19] Thang Le Duc, Mark Leznik, Jörg Domaschka, and Per-Olov Östberg. 2020. Workload Diffusion Modeling for Distributed Applications in Fog/Edge Computing Environments. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 218–229.

[20] Mark Leznik, Rafael Garcia Leiva, Thang Le Duc, Sergej Svorobej, Linus Närvä, Manuel Noya Mariño, Peter Willis, Konstantinos M. Giannoutakis, Radhika Loomba, Héctor Humanes, Miguel Ángel López, P-O Östberg, Paolo Casari, and Jörg Domaschka. 2019. *RECAP Artificial Data Traces*. https://doi.org/10.5281/zenodo.3458559

[21] Mark Leznik, Simon Volpert, Frank Griesinger, Daniel Seybold, and Jörg Domaschka. 2018. Done yet? A critical introspective of the cloud management toolbox. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 1–8.

[22] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2019. Generating high-fidelity, synthetic time series datasets with doppelganger. *arXiv preprint arXiv:1909.13403* (2019).

[23] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163* (2016).

[24] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).

[25] João Moreira, André Carlos Ponce de Leon Ferreira, and Tomáš Horváth. 2019. *A general introduction to data analytics*. Wiley Online Library.

[26] Dominik Moritz and Danyel Fisher. 2018. Visualizing a Million Time Series with the Density Line Chart. *arXiv preprint arXiv:1808.06019* (2018).

[27] David Mosberger and Tai Jin. 1998. httperf—a tool for measuring web server performance. *ACM SIGMETRICS Performance Evaluation Review* 26, 3 (1998), 31–37.

[28] Per-Olov Östberg, James Byrne, Paolo Casari, Philip Eardley, Antonio Fernandez Anta, Johan Forsman, John Kennedy, Thang Le Duc, Manuel Noya Marino, Radhika Loomba, et al. 2017. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *2017 European conference on networks and communications (EuCNC)*. IEEE, 1–6.

[29] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific data* 5, 1 (2018), 1–13.

[30] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. [n.d.]. T-CGAN: Conditional Generative Adversarial Network for Data Augmentation in Noisy Time Series with Irregular Sampling. ([n. d.]). arXiv:1811.08295 [cs, stat]

[31] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498* (2016).

[32] Benjamin Schanzel, Mark Leznik, Simon Volpert, Jörg Domaschka, and Stefan Wesner. [n.d.]. Unified Container Environments for Scientific Cluster Scenarios. In *Proceedings of the 5th bwHPC Symposium* (Freiburg, 2019). https://doi.org/10.15496/publikation-29052

[33] Robert H Shumway and David S Stoffer. 2000. Characteristics of time series. In *Time Series Analysis and Its Applications*. Springer, 1–88.

[34] Jorge Sola and Joaquin Sevilla. 1997. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science* 44, 3 (1997), 1464–1468.

[35] Guanying Wang, Ali R Butt, Henry Monti, and Karan Gupta. 2011. Towards synthesizing realistic workload traces for studying the hadoop ecosystem. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE, 400–408.

[36] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. 2020. Quant gans: Deep generation of financial time series. *Quantitative Finance* 20, 9 (2020), 1419–1440.

[37] Richard Wolski and John Brevik. 2013. Using parametric models to represent private cloud workloads. *IEEE Transactions on Services Computing* 7, 4 (2013), 714–725.