CloudBruno: A Low-Overhead Online Workload Prediction Framework for Cloud Computing

Vinodh Kumaran Jayakumar*, Shivani Arbat†, In Kee Kim†, and Wei Wang*
*The University of Texas at San Antonio, Computer Science, rvn028@my.utsa.edu, wei.wang@utsa.edu
†University of Georgia, Computer Science, {sga64681, inkee.kim}@uga.edu

***Abstract*—Accurate prediction of future incoming workloads to cloud applications, such as future user request count, is critical to proactive auto-scaling, and in general, critical to the cost-effectiveness of cloud deployments. However, designing a generic predictive framework that can accurately predict for any types of workloads is difficult, especially when the workload is dynamic and can change to a pattern that has not been observed in the training data sets. However, existing workload prediction solutions typically rely on complex machine learning models, which require comprehensive training data, making it difficult for them to handle dynamic workloads. Moreover, the training of existing workload prediction solutions are also expensive in terms of both time and computing resources.**

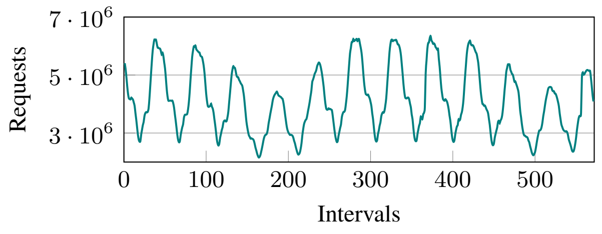
This paper presents a generic and low-cost online workload prediction framework, called CloudBruno, which combines the more accurate LSTM models with less expensive but fast SVM models to achieve high accuracy and low training overhead. When compared to existing predictors, CloudBruno had at least 8.8% lower error than existing deep learning-based predictors for a highly-dynamic workload that does not have comprehensive training data (i.e., has changes unknown to training data). For workloads with comprehensive training data, CloudBruno's error was at most 2.5% higher than optimized deep learning-based predictors. More importantly, CloudBruno can effectively execute on a free cloud CPU, allowing it to be used as an online workload predictor without additional cost.

***Index Terms*—Cloud Workload Prediction, Long Short-Term Memory, Support Vector Machine, Online Prediction Framework, Auto-Scaling**

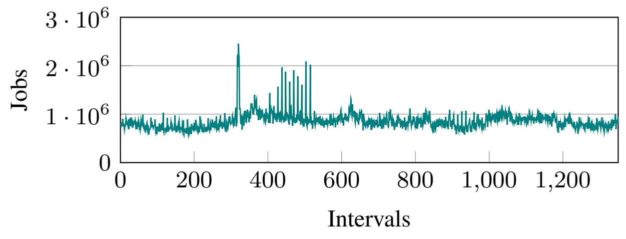
I. Introduction

Auto-scaling is the key technology used to realize the full potential of cloud computing's cost-effectiveness [1]–[3]. With auto-scaling, a cloud deployment can dynamically increase or reduce its resource allocations to handle the increased or reduced workload with both satisfying performance and low cloud usage costs. Therefore, proactive auto-scaling is usually a more desirable solution, as it can scale the virtual machine (VM) or container allocation before the workload change actually happens, avoiding the performance degradation that may be experienced in reactive auto-scaling [3], [4].

A key requirement for proactive auto-scaling is the ability to predict the future incoming workloads, such as the number of jobs or the number of user requests that will arrive in the next hour [5]. The accuracy of this prediction, in turn, determines the effectiveness of proactive auto-scaling. That is, the inaccurate prediction may lead to resource over- or under-provisioning, which either incurs unnecessary cloud usage costs or causes performance degradation [6].



(a) Wikipedia: user request count every 30 min.



(b) Google Cluster: job count every 30 min.

Fig. 1. Traces of two workloads with different patterns.

However, creating accurate workload predictors (or predictive models) for cloud computing is challenging. The main difficulties lie in the large variety of cloud workloads and the various workload changes within a workload. Fig. 1 shows the workload traces for Wikipedia [7] and Google's cluster [8]. The two workloads shown in Fig. 1 are drastically different – although the Wikipedia workload is seasonal and more stable, the Google cluster workload is more random and has a sharp increase (e.g., traffic spike) in jobs numbers in the middle of the trace. To handle these two difficulties, existing cloud workload prediction approaches typically rely on complex neural networks (NN), such as Long Short-Term Memory (LSTM) [9]–[12] and Transformer [13]–[15], to build and optimize predictive models for each workload to handle various workload changes. However, the use of complex NN models has three limitations,

1) The first limitation is the requirement of comprehensive training data sets. Due to the nature of NNs, training accurate NN models requires a comprehensive workload history that contains most of the potential workload patterns/changes as training data. That is, the potential workload changes after deployment should be (mostly) *known* at the training time. This requirement, however, is difficult to meet in practice. To address this limitation,

prediction models usually need to be retrained online (i.e., after deployments) once workload changes are observed [6].
2) The second limitation is the long training time. Complex NN models and large training data sets imply a long training time. Moreover, to achieve high accuracy for various types of workloads, existing predictive frameworks also require extensive model optimization (e.g., hyperparameter tuning) to find the best model for a workload, which further prolongs the training time. This extended training time makes existing frameworks impractical for online retraining. This is especially true for frequently-changing workloads, where a model may become obsolete before it finishes retraining if another workload change happens during this retraining.
3) The last limitation is the expensive hardware required to train complex NN models. Modern complex NN models typically require powerful CPUs, or even GPUs, to train. However, powerful CPUs and GPUs can be expensive to rent in the cloud. Hence, online retraining with expensive cloud CPUs/GPUs will increase cloud deployment costs.

This work aims to build a cloud workload prediction framework with the following characteristics: 1) *generic*, the framework can generate predictors for various workloads with comparable accuracy than optimized NN models with comprehensive training data; 2) *dynamic*, the framework can retrain its models online after deployments to adapt to workload changes that are unknown before deployment; 3) *low-overhead*, the retraining can be done in a short amount of time to accommodate frequent workload changes; 4) *low-cost*, the retraining should execute effectively with cheaper cloud resources.

This paper presents the design of CloudBruno[1], a generic online workload prediction framework. CloudBruno combines two prediction models with different characteristics, LSTM and Support Vector Machine (SVM) [16]. LSTM models have higher accuracy for cloud workload prediction than SVM models. However, LSTM is also more expensive and slow to train. By combining the predictions of LSTM and SVM, and by properly selecting their retraining frequencies, CloudBruno can achieve high accuracy, low overhead, and low cost.

More specifically, in CloudBruno, both LSTM and SVM models are periodically retrained to adapt to workload changes. However, they are retrained at different frequencies – the faster SVM model is retrained more frequently, whereas the LSTM model is retrained less frequently. Moreover, a third tournament predictor is trained to predict whether the SVM model or LSTM model will be more accurate based on their past accuracy, and the predicted more-accurate model will then be used to make predictions for the future workload. This third classifier is retrained at the same frequency of the SVM model. By combining SVM and LSTM, CloudBruno can enjoy the high accuracy of LSTM for workloads during their (relatively) stable phases but can also quickly adapt to workload changes through the use of SVM.

The retraining process of CloudBruno includes hyperparameter optimization (search), which allows CloudBruno to be generic (i.e., predicting various types of workloads with high accuracy). The hyperparameter search space and search iterations are also selected in a way to balance the accuracy and retraining time/cost.

We first evaluated CloudBruno with 14 workload configurations from five different (representative) application models in clouds. The evaluation results show that CloudBruno had an average error of 20.5%, which was only 2.5% higher than existing more complex workload prediction frameworks trained with extensive and comprehensive data sets. Furthermore, when applied to a highly-dynamic workload, CloudBruno had at least 8.8% less error than other NN frameworks and 5% less error than another online prediction framework. When applied in Google Compute Engine, CloudBruno managed proactive auto-scaling reduced average job turnaround time by up to 7% than complex NN frameworks, showing that CloudBruno's better accuracy can translate into real performance benefits.

CloudBruno could also quickly retrain its models even using a free cloud CPU. On average, CloudBruno could retrain an SVM model every 3.4 seconds and an LSTM model every 23.5 minutes, which was at least $4.0\times$ faster than existing NN frameworks. This fast retraining on a free cloud CPU allows CloudBruno to be applied with limited or even no additional monetary cost.

The contributions of this paper include:

1) The design of CloudBruno, a generic online workload prediction framework that combines automatically optimized LSTM and SVM models to provide high accuracy, low overhead, and low cost predictions for a variety of workloads with dynamic changes.
2) A thorough evaluation of CloudBruno with 15 workload configurations to demonstrate that CloudBruno can indeed provide high accurate predictions for various workloads with low training cost.
3) A case study to demonstrate that the highly-accurate predictions from CloudBruno can further improve the performance of auto-scaling on real public clouds.

The rest of this paper is organized as follows: Section II formally formulates the workload prediction problem and presents our motivation. Section III presents the detailed design of CloudBruno. Section IV evaluates the accuracy and training time of CloudBruno. Section V presents a case study of auto-scaling with CloudBruno. Section VI discusses the related work. and Section VII concludes the paper.

II. Problem Definition and Motivation

A. Problem Definition

In this work, we define the workload to a cloud application as the number of incoming jobs or user requests within an interval. For example, for a website deployed to the cloud, if the number of user requests that access this website every

[1]We use this name from Bruno (Madrigal), a character in the Disney movie "Encanto". He has the ability to foresee the future.

TABLE I
TRAINING TIME AND PROCESSORS USED FOR TRAINING BY PRIOR NN STUDIES. THE LSTM AND BILSTM TIMES WERE REPORTED BY THEIR ORIGINAL PAPERS. WGAN-GP TIMES WERE MEASURED BY US.

Prior work	Type of Model	Reported Training Time	Training Processor
LoadDynamics [10] (2020)	LSTM	about 3 hours	16-core Intel Xeon 8153
Bi et. al [17] (2021)	BiLSTM	70 to 100 mins	Unknown
Arbat et. al [13] (2022)	WGAN-gp Transformer	1 to 5 hours	GTX 2080 Ti

second is 30, then its workload is 30 requests/sec. As a second example, for a high-performance computing (HPC) cluster deployed to the cloud, if the number of jobs arriving at the cluster is 50 per hour, then its workload is 50 jobs/hr. For simplicity, we refer to the number of jobs/requests during an interval as *request rate*.

For a deployed cloud application, there will be a trace of request rates observed in the intervals during which it executes. Without loss of generality, let the real request rate at time interval t be w_t, and the predicted request rate at time interval t be p_t. After executing for some time, there will be a trace of request rates starting from time interval 0 to interval $t-1$, i.e., $\{w_0, w_1, \ldots, w_{t-1}\}$. Our prediction problem then can be determined as,

Workload Prediction Problem Definition
Input: Past request rates: $w_0, w_1, \ldots, w_{t-1}$
Output: Prediction for request rate at interval t: p_t

The core issue of this prediction problem is to determine the predictive model, M, that takes the past request rates and produces an estimation p_t. Our framework, CloudBruno, is designed to generate the model M for various workloads at each new interval. Note that, to ensure high generality over various cloud applications, the feature used in our prediction problem is only the past request rates. Moreover, although the input can contain all past request rates, real models may only use a fraction of these rates. That is, only n records of past request rates, $\{w_{t-n}, w_{t-n+1}, \ldots, w_{t-1}\}$, will be actively used for prediction. The value of n also needs to be determined as part of model training and optimization.

B. Motivation

Because of the large variety of workload patterns and potential workload changes, prior work typically employed complex NN models. Table I lists several recent workload prediction methodologies from prior studies with various NN models, including LSTM [10], Bidirectional (BiLSTM) [18]. and WGAN-gp Transformers [13]. Table I also provides the longest training time and the processor used for training for some of these models, which shows that these models typically require hours to train on powerful CPUs or GPUs. Such long training times make these models impractical for real-world cloud applications because online model retraining should quickly adapt to workload changes. The requirement of powerful CPUs/GPUs can also increase the deployment cost. To overcome these limitations, in this work, we explore the possibility of designing a workload prediction framework that can retrain its models every few seconds/minutes on the least expensive cloud CPUs.

III. THE DESIGN OF CLOUDBRUNO

This section presents the design of the CloudBruno framework. The main idea of CloudBruno is the tournament prediction of two types of machine learning (ML) models, one *fast* and one *slow*. The fast-training model provides quick adaptation to (sudden) workload changes, whereas the slow-training model provides higher accuracy predictions when the workload is in a (more) stable phase.

A. Machine Learning Background

Machine Learning Models. CloudBruno includes two types of ML models, the SVM and LSTM models. CloudBruno employs SVM [16] to quickly adapt to a workload change. SVM is a statistical learning model that is primarily used for classification by determining a decision boundary (hyperplane) within a set of data. SVM is also frequently adapted to do regression, where the decision boundary is used to approximate the regression curve. Although SVM is not designed specifically for time series regression, prior work has shown that it is very effective for single workloads [6], [19].

However, despite SVM's reasonable accuracy for cloud workload prediction, its accuracy is still lower than complex NN models when training workload can better represent the workload during deployment. Therefore, to achieve high accuracy for workloads in their stable phases, CloudBruno also employs LSTM models. LSTM is a type of NN, designed for sequential data, such as the time series data observed in cloud workloads. For each inference, a LSTM model not only produces a prediction but also produces hidden states, which are passed back to the LSTM model for the next prediction [9]. These hidden states keep track of additional information in a sequence of data that is useful for interference, such as its long-term or short-term trends in time series. Note that prior work has also shown that LSTM is usually more accurate than SVM, although its training time is considerably longer, especially when model optimization is required [10].

Hyperparameter Optimization. A key step in model training is hyperparameter optimization [20], [21], which significantly affects a model's accuracy. For LSTM, the hyperparameters may include the number of NN layers, the batch size, the history length (n as discussed in Section II-A), and the size of the internal cell activation vector (c size). For SVM, the hyperparameters may include the regularization value and kernel coefficient. Prior work has shown that proper hyperparameters are critical to the accuracy and generality of cloud workload prediction framework. That is, the predictive model for workload requires its own set of parameters and tuning [10].

Hyperparameter optimization is essentially a search process where different sets of hyperparameters are evaluated to find

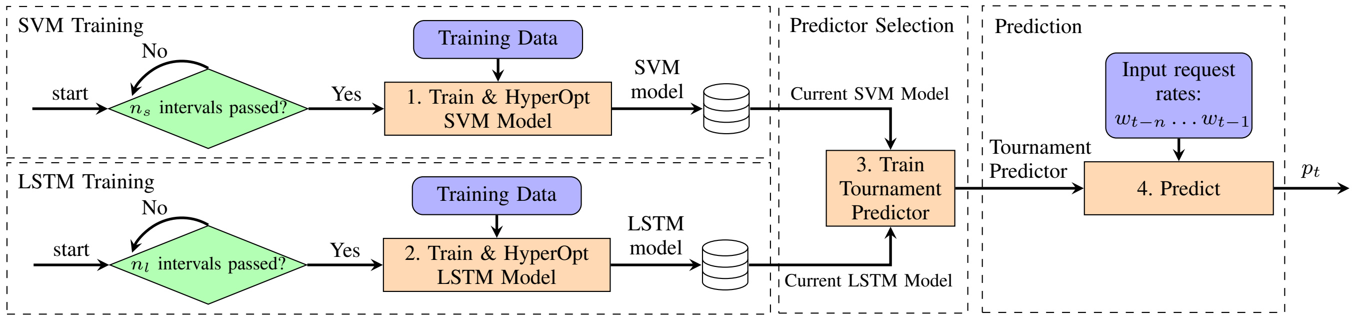


Fig. 2. The overall workflow of building a new predictor and making predictions with CloudBruno.

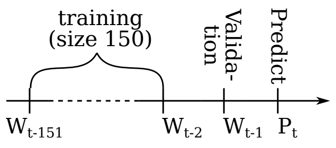


Fig. 3. The training and validation data sets, as well as the predictions for the SVM model.

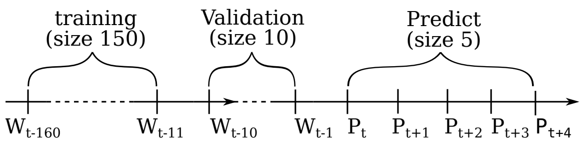


Fig. 4. The training and validation data sets, as well as the predictions for the LSTM model.

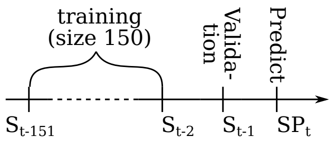


Fig. 5. The training and validation data sets, as well as the predictions for the tournament predictor model.

the best set. To reduce the search time on a large search space, contemporary research typically employs Bayesian Optimization (BO) [22] or Random Search [23] in this optimization. Nonetheless, even with better searching algorithms, the large hyperparameter search space is still one of the main reasons why some of the state-the-art cloud workload predictors are time-consuming to train. CloudBruno also employs BO-based hyperparameter optimization to (re)train individual models for each workload. However, we reduced the search space (i.e., potential values for hyperparameters) to bring down the retraining time. In fact, there is also no need for exploring a large hyperparameter configuration space, as the models only need to "learn" from the recent workload patterns.

B. Combining Two Models in CloudBruno

The predictions of SVM and LSTM models are eventually combined in CloudBruno. That is, a third *classifier* that works as an tournament predictor, which determines whether SVM or LSTM is more accurate for the current workload, and a predicted more-accurate model will then be used to predict the workload for the next interval.

The SVM, LSTM, and tournament predictor all need to be retrained periodically to handle workload changes. As LSTM models have a longer training time, they will be retrained less frequently in CloudBruno. On the other hand, SVM models can be retrained more often due to their lightweight nature [19]. As the tournament predictor also needs to be retrained every time the SVM model is updated (i.e., retrained with the same frequency as the SVM model), the tournament predictor is also configured to be an SVM classifier in CloudBruno for fast retraining. Moreover, as the prediction models are online and focused on making predictions based on the recently-seen workload patterns, there is no need to employ a large training data set.

C. The Workflow of CloudBruno

Fig. 2 illustrates the overall workflow of CloudBruno. As Fig. 2 shows, for every n_S interval, a SVM model will be retrained (Step 1). In the current CloudBruno, n_s is set to be 1, indicating that, for every interval, a new SVM model is trained (to ensure every workload change is captured). Fig. 3 shows the training, cross-valuation data sets and the prediction of a SVM model in CloudBruno. As Fig. 3 shows, the training data include 150 (intervals of) most-recent request rates. As stated previously, the training also includes hyperparameter optimization, which is conducted with one data point for validation. Because the SVM model is retrained for every interval, a trained model is only make one predictions (i.e., the P_t in Fig. 3).

For the LSTM model, it will be retrained every n_l interval (Step 2). Please note that n_l should be long enough for the LSTM models to finish training. In the current CloudBruno, n_l is set to be 5 (intervals). The smallest interval used in our evaluation is 5 minutes, so five 5-minute intervals are slightly longer than the training time of CloudBruno's LSTM model. Fig. 4 illustrates the training, cross-valuation data sets and the prediction of a LSTM model in CloudBruno. Similar to the SVM model, the training sets have a size of 150. However, the validation set has a size of 10 for LSTM because the complex LSTM model requires more tuning for better accuracy as well as the LSTM models do not need to be retrained at each interval like SVM. Moreover, because a LSTM is retrained every five intervals, it will be used to make five continuous predictions consecutively.

TABLE II
WORKLOADS USED FOR EVALUATION.

Workload Trace	Type	Interval length (mins)
Wikipedia (Wiki) [7]	Web	5, 10, 30
Grid (LCG) [24]	Scientific	5, 10, 30
Azure (AZ) [25]	IaaS Cloud	10, 30, 60
Google (GL) [8]	Data Center	5, 10, 30
Facebook (FB) [26]	MapReduce	5, 10
Combined	Synthetic	All of above at 10-min intervals

After each time a new SVM/LSTM model is trained, the tournament predictor will also be retrained using the past model's errors to determine which model is more accurate (Step 3). Here, the errors of the past 150 request rates are used, as shown in Fig. 5. In Fig. 5, S_i is a tuple denotes which model was used in the time i. That is, if S_i is $\{0, 1\}$ then the SVM is used at time i. If S_i is $\{1, 0\}$ then the LSTM is used at time i. Once the tournament predictor is built, it is then used to predict either SVM or LSTM should be used at time t as a tuple, SP_t. For hyperparameter optimization of the SVM and LSTM models, only 10 iterations of optimization are performed in CloudBruno for faster training time.

It is worth noting that it may appear that the tournament predictor can be simply built by comparing the errors of the past SVM and LSTM models and then picking the one with the lower average error. However, during the development of CloudBruno, we learned that this simple method does not work. We suspect there are two reasons. First, because the SVM models are frequently retrained, the past errors are not from the same model, making it mathematically less correct to compute the average. Second, in some cases, the trend and variation of the errors as usually as average prediction errors. Therefore, we eventually employed SVM classifiers to build the tournament predictor.

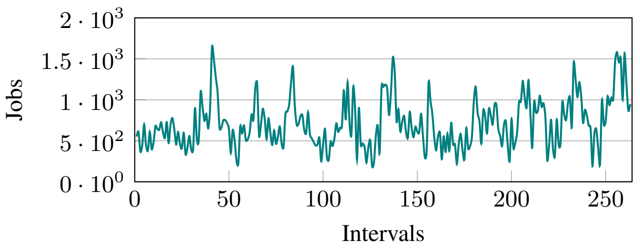
IV. Evaluation

This section reports the evaluation results of CloudBruno. This evaluation focused on the prediction accuracy and training time of CloudBruno.

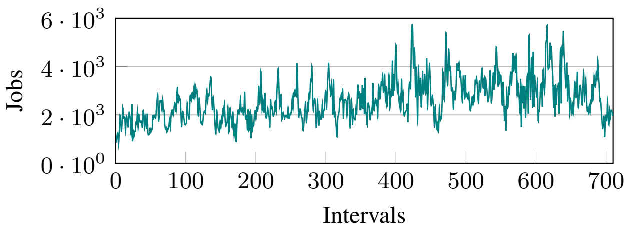
A. Experiment Setup

Workload Traces. Five real workload traces were employed in this evaluation, representing different cloud use cases, such as web applications [7], data analytics [8], [26], data center applications [25], and scientific computing [24]. Table II describes the detailed information of these workloads. For each workload, two or three interval lengths were used, ranging from 5 minutes to 1 hour. That is, a workload's request rate may represent the number of jobs/requests every 5 minutes to every hour. In total, 14 real workload traces were used in this evaluation. The large number of workloads ensured a thorough evaluation and also allowed us to examine the generality of CloudBruno. Fig. 1 and Fig. 6 visualize these workloads.

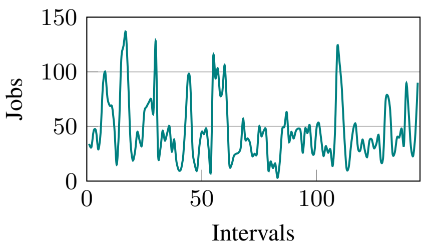
In addition to these 14 real-world workloads, we also devised a synthetic workload by combining the five workload traces. This combined workload is used to evaluate CloudBruno at the worst-case scenario – a workload with extremely varying request rates.



(a) LCG Grid workload job count every hour.



(b) Azure Cloud workload job count every hour.



(c) Facebook workload job count every 10 minutes.

Fig. 6. Traces of the LCG, Azure, and Facebook workloads used in the evaluation.

Baselines. We compared CloudBruno with four state-of-the-art workload predicting frameworks. The first framework is LoadDynamics [10], which employs LSTM and hyperparameter tuning to build predictive models for each workload. The second framework is WGAN-gp transformer [13], which employed (hyperparameter-) optimized Wasserstein Generative Adversarial Network with gradient penalty (WGAN-gp) to train transformer models for workload prediction. The third framework employs Hybrid Bidirectional LSTM (BiLSTM) models [17], [18]. The fourth framework is CloudInsight [19], [27], which is also an online prediction framework based on various statistical learning models to build predictors. These four frameworks allow us to compare with both complex offline deep learning frameworks as well as faster online framework from the literature.

Metrics. We employed mean absolute percentage error (MAPE) to evaluate the accuracy of all prediction frameworks. MAPE is calculated as average percentage differences between real request rate w_i and its predicted request rate p_i for all n predictions. MAPE is expressed as:

$$MAPE = \frac{100}{n} \sum |\frac{p_i - w_i}{p_i}| \tag{1}$$

For training overhead, the wall-clock time training is reported.

Hardware Platforms. For CloudBruno's training and workload prediction, we employed the free VM provided by Google Colaboratory[2]. This VM has a generic Intel 2GHz Xeon processor with two cores. We intentionally chose this VM as it has an even less powerful CPU than Google cloud's free-tier `e2-micro` VMs, which have a 2.3GHz generic Intel Xeon processor[3]. With such a low-end CPU, we could more strictly evaluate the training time of CloudBruno when it is used on free cloud resources. The baseline models require a longer training time. Therefore, they were trained using more powerful CPUs and GPUs in our labs – LoadDynamics and WGAN-gp models were trained on NVIDIA GTX 2080 Ti GPU, BiLSTM models were trained on NVIDIA Tesla T4 GPU, and CloudInsight was trained on eight AMD Opteron 4386 CPUs.

Hyperparameter Search. The hyperparameter search space for CloudBruno's LSTM models is shown in Table III. The Facebook workload is smaller than all other workloads. Therefore, the hyperparameter search space for it was also smaller (i.e., it is not long enough for a long history size). To reduce training time, we set the number of hyperparameter optimization iterations to 10. That is, 10 sets of hyperparameters were used in BO for CloudBruno's LSTM model. For the SVM model, the hyperparameter search space consists of different values of C (regularization parameter) and gamma (rbf kernel coefficient), where both values range from 1 to 5.

For the baseline frameworks, the hyperparameter search spaces reported in their original papers were used.

TABLE III
HYPERPARAMETER SEARCH SPACE FOR THE LSTM MODELS IN CLOUDBRUNO.

Workload	**Hist Len (n)**	**Cell Input size**	**Layers #**	**Batch #**
Wiki	[1-99]	[1-100]	[1-5]	[16-256]
LCG				
Azure				
Google				
Facebook	[1-50]	[1-50]		[8-64]

B. Evaluation of Accuracy with Real Workloads

1) Accuracy of CloudBruno: Fig. 7 reports the accuracy (MAPE) of CloudBruno (the "green" bar) for the 14 workloads described in Table II. As Fig. 7 shows, CloudBruno had high accuracy for the Wikipedia (Wiki) workloads, where the maximum error (MAPE) was only 1.3%. CloudBruno also had less than 12% error for the Google (GL) workloads. CloudBruno's errors for the Grid (LCG), Azure (AZ), and Facebook (FB) workloads were relatively higher and were mostly in the range of 20% to 40%. These relatively higher errors were partially due to the way MAPE was computed (Equation-1). That is, when the actual request rate w_i is too small, even a small absolute error may lead to a very large percentage error. As we show later with the case study, these predictions with high MAPE but small absolute errors would still yield overall system performance improvements for cloud deployments. The average error for CloudBruno for all 14 workloads was only 20.5%.

To illustrate the impact of the SVM predictor, we also evaluated CloudBruno with only LSTM models. That is, only using the LSTM models in CloudBruno without the SVM and the tournament predictor. The results were also reported in Fig. 7 ("CloudBruno-LSTM-Only", the "orange" bar), which showed that without SVM, LSTM-only CloudBruno had high errors. In particular, for FB-5m, the MAPE was 96%. The average error of LSTM-only CloudBruno was 33.1%, which was also considerably higher than that of CloudBruno (the average error was 20.5%). The main issue with LSTM-only CloudBruno was because the LSTM model was retrained every 5 intervals, it could not adapt to workload changes that happened within those 5 intervals. However, the faster SVM models in the complete CloudBruno are retrained after every interval, allowing CloudBruno to adapt to every workload change. These results also suggest that, for better accuracy, it is crucial that online workload predictor can adapt immediately after a workload change.

2) Accuracy Comparison with Baselines: Three of the baselines, LoadDynamics, WGAN-gp, and BiLSTM, employed complex NN models. Following the common practice in deep learning and their original papers, these three baselines were trained using 60% of the workloads, and their accuracy were also reported in Fig. 7. As Fig. 7 shows, CloudBruno ("green" bar in the figure) had higher accuracy than the BiLSTM model ("gray" bar) for all workloads. Fig. 8 reports the differences of the overall average MAPEs of CloudBruno and the baselines. Here, the overall average of a predicting framework refers to the average MAPE of all 14 workloads for this framework. As shown in Fig. 8, BiLSTM's error was 7.6% higher than CloudBruno. The main issue with BiLSTM model is that its hyperparameters were selected for the Google workload [17], making it difficult to handle all other workloads.

Fig. 7 also shows that the accuracy of CloudBruno was similar to LoadDynamics and WGAN-gp. In one case, Azure workload with 10-minute intervals (AZ-10m), CloudBruno's error was more than 11% lower than both LoadDynamics and WGAN-gp. It's worth noting that in some cases (e.g., FB-5m), CloudBruno may be worse than LoadDynamics and WGAN-gp, mainly because of the smaller hyperparameter search space in CloudBruno. Fig. 8 shows that CloudBruno's error was only 2.5% ("white" bar) higher than LoadDynamics and 0.5% ("black" bar) higher than WGAN-gp. CloudBruno's comparable accuracy to complex NN models of LoadDynamics and WGAN-gp shows **the first main benefit of CloudBruno** – it does not require prior knowledge of the workload and large training sets to provide high accuracy comparable to optimized complex NN models.

The fourth baseline, CloudInsight, was also an online prediction framework that employed multiple statistical learning models. As Fig. 7 shows, CloudBruno had better performance

[2]https://colab.research.google.com
[3]https://cloud.google.com/compute/docs/machine-types

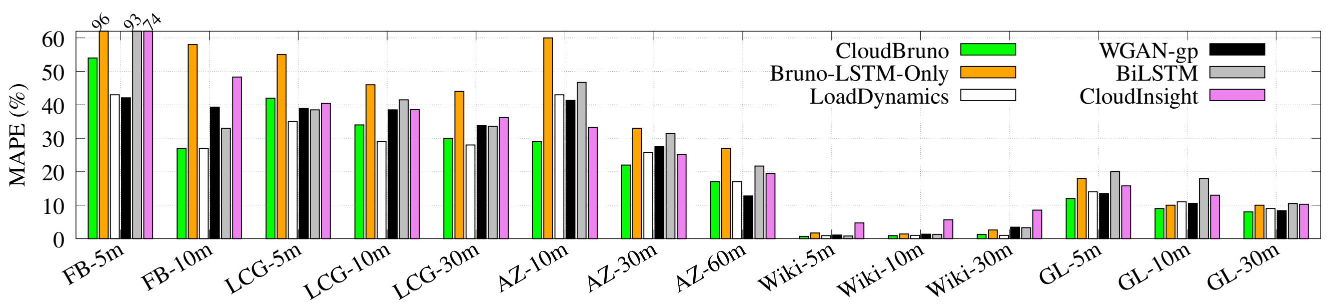


Fig. 7. Prediction errors (MAPE) of CloudBruno for 14 real workloads. Note that, WGAN-gp does not have MAPE for grid (LCG) and Wiki-5m workloads, due to insufficient memory error when training on the GPU.

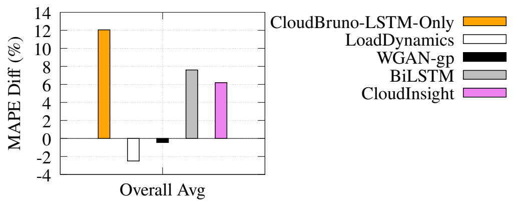


Fig. 8. The differences of the overall average MAPE between the baselines and CloudBruno. Negative differences mean lower MAPE than CloudBruno, while positive differences mean higher MAPE than CloudBruno.

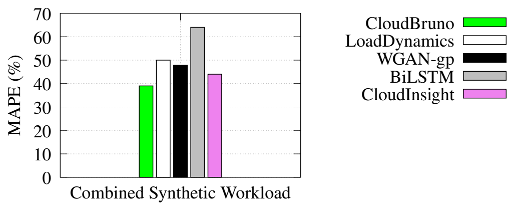


Fig. 9. Predication errors (MAPE) of the combined workload.

than CloudInsight ("violet" bar) for all workloads. On average, CloudBruno's error was 6.1% lower than CloudInsight. As an online prediction framework, CloudInsight was also optimized for low train time by employing faster statistical models, such as ARIMA and Random Forest. However, these models have lower accuracy for complex workloads with low seasonality.

C. Evaluation of Accuracy with Highly-dynamic Workload

One of the main design goals of CloudBruno is to provide accurate predictions for workloads that experience dynamic changes after deployment, i.e., for workloads with patterns that are not included in the training data sets. To evaluate CloudBruno's accuracy with such highly-dynamic workloads, we also generated a synthetic workload by concatenating/combining the five 10-minute-interval traces in Table II. As these workloads are from five different use cases, the variation in this synthetic workload is extremely high, which allowed us to evaluate CloudBruno in this extreme scenario.

Fig. 9 reports the MAPE of CloudBruno and the four baselines with this synthetic workload. As Fig. 9 shows, CloudBruno had lower error than all four baselines. Within the four baselines, CloudInsight had the lowest error. CloudBruno's and CloudInsight's better accuracy was the result of their online prediction nature. For this combined workload,

TABLE IV
THE (RE-)TRAINING TIME OF THE ONLINE MODELS IN CLOUDBRUNO.

Workload	Interval (min.)	SVM (sec.)	LSTM (min.)	Tournament predictor (msec.)
Facebook	5 min.	3.7	4	13
	10 min.	1.5	6	13
LCG	5 min.	1.4	21	13
	10 min.	3.2	38	15
	30 min.	3.5	26	15
Azure	10 min.	2.3	34	17
	30 min.	5.0	19	13
	60 min.	5.9	36	17
Wiki	5 min.	4.6	20	20
	10 min.	5.0	18	19
	30 min.	1.7	17	16
Google	5 min.	1.7	18	19
	10 min.	2.8	36	18
	30 min.	5	36	22
Average		3.4	23.5	16.43

there is little correlation between concatenated workloads. Therefore, the training data set used by the NN baselines (LoadDynamics, WGAN-gp, and BiLSTM) cannot represent the later workloads. Therefore, the NN baselines all had high errors. On the contrary, both CloudBruno and CloudInsight dynamically retrain their models to adapt to each concatenated workload, allowing them to enjoy lower prediction error. More specifically, as shown in Fig. 9, CloudBruno's error was at least 8.8% lower than the NN baselines (8.8% lower than WGAN-gp).

Moreover, CloudBruno's error was also 5% lower than CloudInsight, mainly due to its use of SVM and LSTM, which can better handle workloads with little seasonality. These results illustrate **the second main benefit of CloudBruno** – it can provide high accuracy prediction for *unknown* workload changes before deployment.

D. Evaluation of CloudBruno's Training Time

1) Training Time of CloudBruno: Table IV reports the detailed training time of all the components in CloudBruno for each workload configuration. As reported in the table, the training of the SVM prediction models was very fast, with a maximum training time of only 5.9 seconds and an overall average training time of only 3.4 seconds. Similarly, because the tournament predictor is also an SVM model internally, its

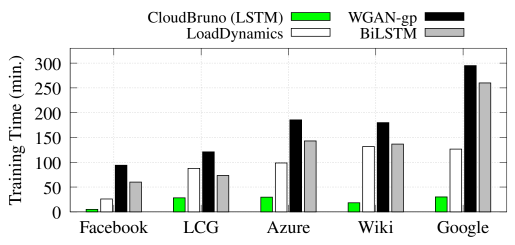


Fig. 10. Training time of CloudBruno and NN baselines. Note that, the hardware used for training is reported in Section IV-A

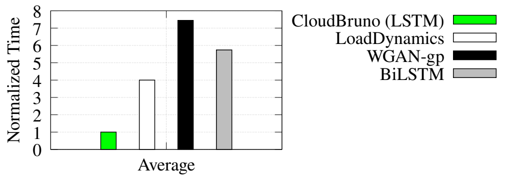


Fig. 11. Overall average training time of CloudBruno and NN baselines. Averaged over all 14 workloads and normalized to CloudBruno

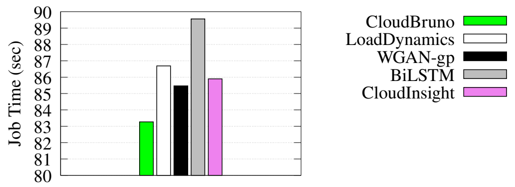


Fig. 12. Average job turn-around time for Google Cloud auto-scaling case study.

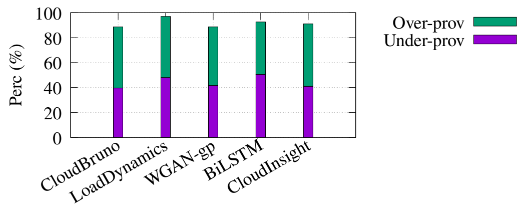


Fig. 13. Under-provisioning and over-provisioning rates for the proactive auto-scaling case study.

training is also fast – the maximum training time was only 22 microseconds, and the overall average training time was 16.4 microseconds. These fast training times show that it is feasible to retrain a new SVM model for every interval, as long as the interval length is longer than 5.9 seconds.

The training of the online LSTM models took longer, compared to the SVM models, as expected. The maximum training time of the online LSTM models was 38 minutes, and the overall average training time was 23.5 minutes. Although these training times were considerably longer than the SVM models, they were still fast enough to retrain new online LSTM models within 5 intervals for each workload configuration. For example, the longest training time, 38 minutes, was observed with LCG workload at 10-minute intervals. Five 10-minute intervals (i.e., 50 minutes) were still longer than the 38-minute training time.

Although the training data set size and hyperparameter optimization iteration count were the same for all workload configurations, the actual training times for each workload configuration were still different. This difference in training time was due to the hyperparameter optimization process. Difference workloads have different optimal sets of hyperparameters. Hence, the hyperparameter optimization under BO searched different parameter sets for different workloads – if more complex LSTM model hyperparameters were searched during the optimization, then the training time would increase.

It is worth noting that the low training time of CloudBruno was achieved with a free cloud CPU. These results illustrate **the third benefit of CloudBruno** – it can be used as an online workload predictor without additional cloud usage cost.

2) Training Time Comparison with Baselines: Fig. 10 reports the training time for CloudBruno and the three NN baselines. Due to space limitation, only the average training times for each of the five workload traces are shown. For example, for the Google workload, the average training time for the 5-min., 10-min., and 30-min. intervals is shown for each predicting framework. Moreover, only the LSTM component's training time of CloudBruno was shown, and the faster SVM training time (shown in Table IV) is omitted due to the y-axis scale.

Fig. 10 shows that CloudBruno had a significant training time reduction against the baselines for every workload. Fig. 11 shows that CloudBruno's overall average training time was 4.0× faster than LoadDynamics, 7.4× faster than WGAN-gp, and 5.7× faster than BiLSTM. This significant training time reduction is because CloudBruno employs efficient training data set size, space-efficient hyperparameter optimization, and the use of faster SVM and tournament predictor. Moreover, it is worth noting that CloudBruno achieved similar accuracy to these NN baselines even with such high training time reduction.

The fourth baseline, CloudInsight, employed fast statistical learning models. Therefore, CloudInsight also have low training time comparable to the SVM component of CloudBruno. However, as shown in Fig. 7, CloudInsight had higher errors than CloudBruno.

V. Case Study

To demonstrate the benefit of CloudBruno for predictive auto-scaling, we also conducted a case study on Google Compute Engine (GCE) with an auto-scaling policy. We used the Azure-30m workload in this case study as it was from an IaaS cloud (Azure VMs), which fits the IaaS cloud GCE. The 30-minute interval was chosen (instead of 10-minute or 60-minute intervals) to balance the workload length and the number of requested VMs per interval. Moreover, Azure-30m is also chosen because CloudBruno's MAPE for this workload is close to the overall average MAPE for CloudBruno for all 14 workloads.

The auto-scaling policy for this case study works as follows. Near the end of each time interval, CloudBruno is used to predict the number of jobs that will arrive in the next interval. Then, a group of VMs will be created proactively for the incoming jobs. The number of VMs in this group is the same as the predicted job count. When the next interval starts, the incoming jobs are assigned to the proactively created VMs. If there are more jobs than predicted, then additional VMs are created on-demand to execute these jobs. This process repeats for every interval until the whole workload is processed.

We used the *In-Memory Analytics* benchmark from Cloud Suite [28] as the job. General-purpose *e2-medium* VMs were also used execute the jobs. Besides CloudBruno, we also evaluated the four baselines in this case study.

Fig. 12 reports the average turnaround time for each predictive framework in this case study. The turnaround time refers to the time between the job arrives and the job finishes execution. As Fig. 12 shows, CloudBruno had the lowest job turnaround time, whereas the BiLSTM framework had the longest turnaround time. More specifically, CloudBruno's average turnaround time was 3% faster than the best baseline WGAN-gp and 7% faster than the worst baseline BiLSTM. These turnaround time results are generally in-line with the prediction errors reported in Fig. 7, where CloudBruno has the lowest error, whereas BiLSTM has the highest error.

To further analyze the VM scaling behaviors of different prediction frameworks, we also collected the under-provisioning and over-provisioning rates for this case study, which are reported in Fig. 13. Here, the under-provisioning and over-provisioning rates refer to the percentages of intervals that experienced under- or over-provisioning (i.e., fewer than or more than required VMs). Fig. 13 shows that the total (sum) of under- and over-provisioning rates were the lowest for both CloudBruno and WGAN-gp, which corroborates the results in Fig. 12 where CloudBruno and WGAN-gp had the two lowest average turnaround times. Nonetheless, CloudBruno had a lower under-provisioning rate than WGAN-gp. Hence, CloudBruno had fewer on-demand VMs created, leading to a better turnaround time than WGAN-gp.

VI. Related Work

Many studies have been proposed for workload prediction because of the benefit of predictive (proactive) auto-scaling. This section provides a review of these models.

Time series models. As workloads are naturally time series, various time series models have been employed in this prediction. Mistral was a cloud management system that employed a time series model, ARMA, to predict future workload [29]. Roy et al. also employed ARMA in their predictive auto-scaling solutions [5]. Another commonly used time series model is ARIMA, which has been employed for VM consolidation [30] and auto-scaling [31]. There is also a group of studies that employed weight moving average (WMA) models for workload prediction [32]–[36]. Woods et al. employed Fast Fourier transform (FFT) model and a discrete-time Markov chain for workload prediction [1]. As shown by prior work [6], time series models are good at predicting workloads with good seasonality and/or relatively stable autocorrelation. However, they have high prediction errors for non-seasonal workloads, which are typically for cloud computing (such as the workloads used in our evaluation).

Statistical learning models. Cortez et al. presented workload traces from Microsoft Azure and employed Random Forest and Extreme Gradient Boosting Tree models to predict various characteristics in these traces [25]. There is also a group of work employed Linear Regression (LR) for workload prediction [2], [37]–[39]. Wrangler is a cloud management system that aims at improving the performance of straggler tasks, and it employs SVM models to predict if a task with straggle [40]. Similarly, Khan et al. employed Hidden Markov Model [41]. Similar to the time series models, these statistical learning models are usually not generic and cannot handle all types of workloads [6]. Moreover, similar to all other machine learning models, these statistical learning models also require representative workloads as training data sets, and hence, may have trouble handling unknown workload patterns.

Multi-predictor and ensemble models. As a single type of time series and statistical learning model has difficulty achieving generality, prior workload also considered employing multiple types of predictive models. Baig et al. employed Random Forest to predict the best models from LR, SVM, Gradient Boosting, and Gaussian Process [42]. Similarly, Loff and Garcia proposed to use k-Nearest Neighbors to select the best predictor [43]. Jiang et al. employed a mixture of time series (AR/MA), SVM, neural network, and genetic programming in their prediction [44]. Herbst et al. also employed four groups of time series models, such as moving average, ARIMA, and exponential smoothing [45]. Liu et al. proposed to first classify workload types and then employ different predictors based on the classification [46]. CloudInsight is an ensemble framework that can combine the prediction results of any type of models [27] and can be viewed as a generic extension of prior multi-predictor/ensemble solutions. CloudBruno is inspired by these prior studies. However, as shown in our evaluation, the use of only time series and statistical learning models provided lower accuracy than CloudBruno.

Deep learning models. The rise of deep learning has also inspired many studies to employ neural network (NN) models. Several studies have employed various types of LSTM models, such as ANN [47], multivariate LSTM [11], parallel LSTM [12], BiLSTM [18], LSTM Encoder-Decoder [48], LSTM with Savitzky-Golay (S-G) filter [49], and Bi-directional and Grid LSTM [17]. Jayakumar et al. also employed LSTM for workload prediction [10]. However, they showed that hyperparameter optimization is a key to achieving generality and high accuracy across various workloads for LSTM models. Zhang et al. employed a stacked autoencoder for workload prediction [50]. Kumar et al. employed offline-trained multi-layer NN models, whose prediction results were adjusted based on online feedback [47]. Arbat et al. employed transformers trained with WGAN-gp for workload prediction, which was shown to be more accurate than LSTM models [13]. Although deep learning models can provide high accuracy for a large variety of workloads, they usually require representative training data sets, making it difficult to handle unknown workload patterns, as shown in our evaluation. Moreover, our evaluation results also illustrated that our online model, Cloud-

Bruno, has similar accuracy to NN models even when they had enough training data. At last, many complex NN models are usually time-consuming and require more expensive hardware to train, while CloudBruno can be trained with low-cost CPUs.

VII. Conclusion

In this paper, we presented the design of CloudBruno, an online workload predictor for cloud auto-scaling. CloudBruno combines the predictions of the more accurate LSTM models and the less expensive SVM models to achieve both high accuracy and low cost. CloudBruno has at least 8.8% lower error than the existing complex neural network (NN) based framework for highly-dynamic workloads whose future workload changes are *unseen* in training data. CloudBruno also had comparable accuracy as NN frameworks for workloads with comprehensive training data. More importantly, CloudBruno can effectively execute on a free cloud CPU, allowing it to be used as an online workload predictor without additional cost.

Acknowledgment

This research was in part supported by the U.S. Department of Agriculture (USDA), under award number 2021-67019-34342, as well as by National Science Foundation (NSF), under grants, 2155096, 2202632, 2221843, and 2215359. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the USDA and NSF. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems. In *ACM Symposium on Cloud Computing (SoCC)*, Cascais, Portugal, October, 2011.

[2] Timothy Wood, Ludmila Cherkasova, Kivanc M. Ozonat, and Prashant J. Shenoy. Profiling and Modeling Resource Usage of Virtualized Applications. In *ACM/IFIP/USENIX 9th International Middleware Conference (Middleware)*, Leuven, Belgium, December, 2008.

[3] Marco A.S. Netto, Carlos Cardonha, Renato L.F. Cunha, and Marcos D. Assuncao. Evaluating Auto-scaling Strategies for Cloud Computing Environments. In *2014 IEEE 22nd International Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, Paris, France, September, 2014.

[4] Laura R. Moore, Kathryn Bean, and Tariq Ellahi. Transforming Reactive Auto-Scaling into Proactive Auto-Scaling. In *The 3rd International Workshop on Cloud Data and Platforms (CloudDP)*, Prague, Czech Republic, April, 2013.

[5] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting. In *IEEE International Conference on Cloud Computing (CLOUD)*, Washington DC, USA, July, 2011.

[6] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling. In *IEEE International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, USA, June, 2016.

[7] Erik-Jan van Baaren. Wikibench: A Distributed, Wikipedia-based Web Application Benchmark. *VU University Amsterdam*, 2009.

[8] Charles Reiss, Alexey Tumanov, Gregory Ganger, Randy Katz, and Michael Kozuch. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In *ACM Symposium on Cloud Computing (SoCC)*, San Jose, CA, USA, October, 2012.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997.

[10] Vinodh Kumaran Jayakumar, Jaewoo Lee, In Kee Kim, and Wei Wang. A Self-Optimized Generic Workload Prediction Framework for Cloud Computing. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Virtual Event, May, 2020.

[11] Chanh Nguyen, Cristian Klein, and Erik Elmroth. Multivariate LSTM-Based Location-Aware Workload Prediction for Edge Data Centers. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Larnaca, Cyprus, May, 2019.

[12] Xiaoyong Tang. Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network. *IEEE Access*, 7, 2019.

[13] Shivani Arbat, Vinodh Kumaran Jayakumar, Jaewoo Lee, Wei Wang, and In Kee Kim. Wasserstein Adversarial Transformer for Cloud Workload Prediction. In *The Thirty-Fourth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-22)*, Vancouver, BC, Canada, February, 2022.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *31st International Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, 2017.

[15] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, December, 2017.

[16] William S Noble. What is A Support Vector Machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[17] Jing Bi, Shuang Li, Haitao Yuan, and MengChu Zhou. Integrated Deep Learning Method for Workload and Resource Prediction in Cloud Systems. *Neurocomputing*, 424:35–48, 2021.

[18] Siddhant Kumar, Neha Muthiyan, Shaifu Gupta, Dileep A.D., and Aditya Nigam. Association Learning based Hybrid Model for Cloud Workload Prediction. In *International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, July, 2018.

[19] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Forecasting Cloud Application Workloads with CloudInsight for Predictive Resource Management. *IEEE Transactions on Cloud Computing*, 2020.

[20] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks Learning Systems*, 28(10):2222–2232, 2017.

[21] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and Optimizing LSTM Language Models. In *International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, April, 2018.

[22] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Annual Conference on Neural Information Processing Systems (NIPS)*. Lake Tahoe, NV, USA, December, 2012.

[23] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. of Machine Learning Research*, 13(Feb):281–305, 2012.

[24] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick H.J. Epema. The Grid Workloads Archive. *Future Generation Computer Systems, 24(7)*, 2008.

[25] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *ACM Symposium on Operating Systems Principles (SOSP)*, Shanghai, China, October, 2017.

[26] Yanpei Chen, Archana Ganapathi, Rean Griffith, and Randy Katz. The Case for Evaluating MapReduce Performance Using Workload Suites. In *IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Singapore, July, 2011.

[27] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. CloudInsight: Utilizing a Council of Experts to Predict Future Cloud Application Workloads. In *IEEE International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, USA, July, 2018.

[28] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, London, U.K., March, 2012.

[29] Gueyoung Jung, Matti A. Hiltunen, Kaustubh R. Joshi, Richard D. Schlichting, and Calton Pu. Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. In *International Conference on Distributed Computing Systems (ICDCS)*, Genova, Italy, June, 2010.

[30] Hao Lin, Xin Qi, Shuo Yang, and Samuel P. Midkiff. Workload-Driven VM Consolidation in Cloud Data Center. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Hyderabad, India, May, 2015.

[31] Rodrigo N. Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Transactions on Cloud Computing*, 3(4), 2015.

[32] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Munich, Germany, May, 2007.

[33] Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, and Lin Yuan. Online Self-reconfiguration with Performance Guarantee for Energy-efficient Large-scale Cloud Computing Data Centers. In *IEEE International Conference on Services Computing (SCC)*, Miami, FL, USA, July, 2010.

[34] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David E. Culler, and Randy H. Katz. NapSAC: Design and Implementation of a Power-Proportional Web Cluster. *ACM Computer Communication Review*, 41(1), 2011.

[35] Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Michael A. Kozuch. AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers. *ACM Transactions on Computer Systems*, 30(4), 2012.

[36] Eyal Zohar, Israel Cidon, and Osnat Mokryn. The Power of Prediction: Cloud Bandwidth and Cost Reduction. In *ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, Toronto, ON, Canada, August, 2011.

[37] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical Prediction models for Adaptive Resource Provisioning in the Cloud. *Future Generation Computer Systems*, 28(1), 2012.

[38] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Zexiang Mao, and Junliang Chen. Workload Predicting-Based Automatic Scaling in Service Clouds. In *IEEE International Conference on Cloud Computing (CLOUD)*, Santa Clara, CA, USA, June, 2013.

[39] Peter Bodik, Rean Griffith, Charles A. Sutton, Armando Fox, Michael I. Jordan, and David A. Patterson. Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, San Diego, CA, USA, June, 2009.

[40] Neeraja J. Yadwadkar, Ganesh Ananthanarayanan, and Randy Katz. Wrangler: Predictable and Faster Jobs using Fewer Resources. In *ACM Symposium on Cloud Computing (SoCC)*, Seattle, WA, USA, November, 2014.

[41] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach. In *IEEE International Symposium on Network Operations and Management (NOMS)*, Maui, HI, USA, April, 2012.

[42] Shuja ur Rehman Baig, Waheed Iqbal, Josep Lluis Berral, Abdelkarim Erradi, and David Carrera. Adaptive Prediction Models for Data Center Resources Utilization Estimation. *IEEE Transactions on Network and Service Management*, 2019.

[43] Joao Loff and Joao Garcia. Vadara: Predictive Elasticity for Cloud Applications. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, December, 2014.

[44] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong N. Chang. ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning. In *IEEE International Conference on Data Mining (ICDM)*, Vancouver, BC, Canada, December, 2011.

[45] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. In *ACM/SPEC International Conference on Performance Engineering (ICPE)*, Prague, Czech Republic, April, 2013.

[46] Chunhong Liu, Chuanchang Liu, Yanlei Shang, Shiping Chen, Bo Cheng, and Junliang Chen. An Adaptive Prediction Approach based on Workload Pattern Discrimination in the Cloud. *Journal of Network and Computer Applications*, 80, 2017.

[47] Jitendra Kumar, Ashutosh Kumar Singh, and Rajkumar Buyya. Self Directed Learning Based Workload Forecasting Model for Cloud Resource Management. *Information Sciences*, 543:345–366, 2021.

[48] Hoang Minh Nguyen, Gaurav Kalra, and Daeyoung Kim. Host load prediction in cloud computing using Long Short-Term Memory Encoder–Decoder. *Journal of Super Computing*, 2019.

[49] Jing Bi, Shuang Li, Haitao Yuan, Ziyan Zhao, and Haoyue Liu. Deep Neural Networks for Predicting Task Time Series in Cloud Computing Systems. In *IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Banff, AB, Canada, May, 2019.

[50] Qingchen Zhang, Laurence T. Yang, Zheng Yan, Zhikui Chen, and Peng Li. An Efficient Deep Learning Model to PredictCloud Workload for Industry Informatics. *IEEE Transactions on Industrial Informatics*, 14, 2018.