# Fast-Fourier-Forecasting Resource Utilisation in Distributed Systems

1st Paul J. Pritz
*Department of Computing*
*Imperial College London*
London, United Kingdom
paul.pritz18@imperial.ac.uk

2nd Daniel Perez
*Department of Computing*
*Imperial College London*
London, United Kingdom
daniel.perez@imperial.ac.uk

3rd Kin K. Leung
*Department of Computing*
*Imperial College London*
London, United Kingdom
kin.leung@imperial.ac.uk

***Abstract*—Distributed computing systems often consist of hundreds of nodes (machines), executing tasks with different resource requirements. Efficient resource provisioning and task scheduling in such systems are non-trivial and require close monitoring and accurate forecasting of the state of the system, specifically resource utilisation at its constituent machines. Two challenges present themselves towards these objectives.**

**First, collecting monitoring data entails substantial communication overhead. This overhead can be prohibitively high, especially in networks where bandwidth is limited. Second, forecasting models to predict resource utilisation should be accurate and also need to exhibit high inference speed. Mission critical scheduling and resource allocation algorithms use these predictions and rely on their immediate availability.**

**To address the first challenge, we present a communication-efficient data collection mechanism. Resource utilisation data is collected at the individual machines in the system and transmitted to a central controller in batches. Each batch is processed by an adaptive data-reduction algorithm based on Fourier transforms and truncation in the frequency domain. We show that the proposed mechanism leads to a significant reduction in communication overhead while incurring only minimal error and adhering to accuracy guarantees. To address the second challenge, we propose a deep learning architecture using complex Gated Recurrent Units to forecast resource utilisation. This architecture is directly integrated with the above data collection mechanism to improve inference speed of the presented forecasting model. Using two real-world datasets, we demonstrate the effectiveness of our approach, both in terms of forecasting accuracy and inference speed.**

**Our approach resolves several challenges encountered in resource provisioning frameworks and can also be generically applied to other forecasting problems.**

***Index Terms*—Load Forecasting, Data Collection, Communication Efficient, Fourier Transforms, Complex Gated Recurrent Units, Deep Learning**

## I. Introduction

Distributed systems usually consist of hundreds or thousands of nodes (machines). Efficient management of such systems is often challenging and requires collecting and forecasting of the utilisation of resources such as CPU and memory of the machines of the system. In practice, resource over and under provisioning are common and overall resource utilisation is often poor, leading to a waste of computational resources or the violation of service level agreements [1]. Data collection in some distributed systems is further hindered by communication constraints, especially in systems that are not interconnected by a high-bandwidth network. For instance, sensor networks or networks involving edge devices may suffer from significant communication constraints. This leads to two concrete challenges as follows. First, a central controller that manages scheduling and resource allocation needs to collect monitoring data from all nodes in the network in a communication-efficient manner. Second, the scheduler requires an efficient forecasting model. Efficiency in this context encompasses both accuracy as well as inference speed.

Due to the aforementioned communication constraints it is often detrimental to send all of the collected data to the central controller, forcing the local machines to reduce or compress data before transmission. To address this challenge, we present a data reduction mechanism based on Fourier transforms that can significantly reduce the communication overhead of transmitting monitoring data. Our experiments on real-world data in Section V demonstrate that communication savings in excess of 60% can be achieved while only incurring minimal error in the transmitted data and achieving prediction accuracy comparable to our benchmark model. The proposed methodology can be combined with lossless compression algorithms and exhibits error bounds, which we derive in Section III-B.

As a forecasting model, we propose the use of a deep learning architecture based on complex Gated Recurrent Units (cGRU). In real-world systems, deep learning models are often not practical since training of such models and their use for inference tends to be computationally expensive. The data collection mechanism we present in this paper, however, can be directly combined with our proposed forecasting architecture to improve both training and inference speed, which we demonstrate in Sections IV and V, using synthetic and real-world datasets.

The methods presented in this paper are developed specifically with distributed systems in mind. However, the individual components can be easily applied to different time series forecasting problems. The integration of the proposed data processing and inference techniques with scheduling and

Paul Pritz acknowledges the financial support by the Computing Department at Imperial College London. Thanks are also due to Tiffany Tuor of Imperial College London and Shiqiang Wang of IBM Research for their insightful discussions.

further system management is left for future work. Our main contributions are as follows:

1) We propose a communication-efficient algorithm for time series data transmission in distributed systems, using a batched data transfer protocol with a Fourier transform based mechanism for data reduction.
2) We show that our data transmission algorithm can be readily applied to improve the inference speed of recurrent neural networks, specifically complex Gated Recurrent Units.
3) We propose a deep learning architecture for forecasting resource utilisation in distributed systems that achieves state-of-the-art forecasting accuracy.
4) We conduct extensive experiments using both real-world and synthetic datasets that demonstrate the effectiveness of our proposed methodology.

The remainder of this paper is structured as follows. In Section II, we discuss existing literature. The proposed methodology is presented in Section III, starting with the data transmission protocol before introducing the proposed Fourier processing mechanism and forecasting model. Section IV presents an illustrative experiment on synthetic data before we discuss the experiments using real-world datasets in Section V. Lastly, we conclude in Section VI and provide directions for future research.

## II. Related Work

Previous literature relevant to the approaches outlined in Section III can be broadly categorised into the areas of resource utilisation forecasting, data collection in distributed systems and complex valued recurrent neural networks.

Ample previous research studies the use of classical time series models, such as autoregressive and moving average models for load forecasting, while some have also explored neural networks and alternative models such as support vector regression. A comprehensive overview of previously studied forecasting models for cloud workloads is provided by [2]. The authors of [3], [4] and [5] propose the use of classical time series models. [4] and [5] both propose autoregressive models for load forecasting, using a simple autoregressive model and ARMA models (autoregressive moving average) respectively. [3] propose a load balancing algorithm for cloud infrastructures, as part of which they employ an exponential smoothing based forecasting method. [6], [7] evaluate neural network based approaches. [6] evaluate several neural network architectures and compare them against linear regression using simulated data. The approach proposed by [7] combines a threshold-based method for communication reduction in collecting utilisation data in distributed systems and couples this with a k-means clustering where a forecasting model predicts the centroid values to infer resource utilisation values of individual machines. Alternative modelling approaches including support vector regression, Markov chain models and exponential smoothing based models are presented by [8], [9], [10] and [11]. Several previous papers use the same datasets we use to evaluate their proposed models, enabling comparison among the different approaches [7]–[10].

Further to load forecasting models, several approaches for reducing the communication overhead in distributed systems have been proposed in the body of existing literature. A number of existing methods use a set of randomly selected nodes in the system to infer data for the remaining unobserved nodes using matrix completion [12]–[15]. The approaches in [16], [17] also use a set of randomly selected nodes, but employ Gaussian methods to infer unobserved data. For both approaches, data is only being collected for a random subset of nodes, rather than for all nodes. This not only leads to resource utilisation imbalances but may also cause deviations in accuracy between different nodes. A different approach is presented in [7], which uses a threshold based condition to determine the data transmission frequency for each node in a distributed system. Other algorithms, relying on a per-node condition, i.e. avoiding the problems of imbalance, are presented in [18]–[22]. The proposed methods apart from the one in [22] do not use Fourier transforms for data reduction and none of them use the reduced data by Fourier processing as a means to accelerate inference or model training at a central controller. The approach in [22] relies on a heuristic Nyquist rate based sampling that is somewhat less flexible than our approach. While there is very little research on the use of Fourier transforms for data collection in distributed systems, previous research has explored its uses for correlation approximation and similarity search in databases that is comparable to the idea we propose [23]–[25]. The methods they present rely on the truncation of Fourier transforms to approximate data and [25] employs an energy-based criterion, which is similar to our approach. However, none of the proposed methods are applied in the context of distributed systems nor directly combined with forecasting models.

The methods in [26]–[28] explore this combination of Fourier transforms and deep learning models, but none of them uses truncation of the Fourier transforms for the purpose of accelerating model training and inference. Specifically, [26] and [27] propose the computation of convolutions in the frequency domain to speed up the training of convolutional neural networks and the approach in [28] uses windowing to reduce the training time of recurrent neural networks where Fourier transforms serve as a pre-processing step.

To the best of our knowledge, this work is therefore the first to propose the use of Fourier transforms and truncation in the frequency domain to reduce the communication overhead of transmitting time series data in distributed system and to improve the inference speed of recurrent neural networks.

## III. Proposed Methodology

Our proposed methodology consists of three key components, namely a batched data collection algorithm, a Fourier transform mechanism for data reduction and a deep learning model that can leverage the former two components to achieve improvements in inference speed.

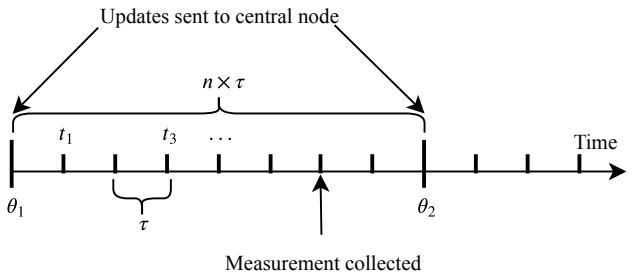


Fig. 1. Data collection and update procedure for a single node $M_i$. Measurements are collected at the discrete time steps $t$ and and the central node $C$ is updated every $n$ time steps, i.e., at update times $\theta$.

### A. Batched Data Collection

Let $M := \{M_1, M_2, ..., M_p\}$ be a set of $p$ nodes in a distributed system, all connected to a network that allows them to communicate with a central controller $C$. We assume that time is slotted and that the nodes $M$ observe data at pre-defined, discrete time steps. To manage the nodes $M$ and schedule computational jobs, $C$ requires information about the nodes' resource utilisation. The individual nodes collect their respective resource utilisation data locally. These form a discrete-time time series $x_{M_i}$ for each node $M_i \in M$. New observations are appended to this series as data is collected. Two protocols for data transmission are readily conceivable. Each node $M_i$ could decide whether to transmit an observation at the time it is collected. The central controller would then treat the last received data point as the current state of node $M_i$ – potentially interpolating if no update has been received for some time. [7] introduces such a methodology, where the nodes in a distributed system make an update decision at each time step. Alternatively, each node $M_i$ can collect its respective time series locally for a given number of time steps and then send it to $C$ in one batch. The central controller then has to estimate the current states of all nodes $M$ in time steps between consecutive updates using some forecasting model or method of interpolation. This batched processing approach – which our proposed methodology relies on – will be referred to as a *batched data collection mechanism*.

Before generalising to the complete set of nodes $M$, consider a single node $M_i$. Let $T$ be a set of discrete time steps at which the variable of interest is observed and define the time between two time steps as $\tau = t_j - t_i, j = i+1$. Let $\theta$ denote the update times where a batch update is sent to the central controller. The time between two consecutive updates is then defined as $n \times \tau := \theta_j - \theta_i, j = i+1, 2 \leq n$. The number of time steps $n$ between batch update times is kept constant, i.e., the batch update times are equally spaced. This is equivalent to saying: The central node $C$ will receive a batch update every $n$ time steps. This batched update mechanism is illustrated in Fig. 1. At each update step, the node $M_i$ will have collected $n$ observations, forming a time series $u \in \mathbb{R}^n$. Each observation measurement in $u$ is represented as a floating-point number. To achieve any communication savings, fewer than $n$ floating-point numbers must be transmitted to the central controller by reducing the data at the individual nodes. We propose an algorithm for achieving such data reduction in the next sub-section.

### B. Fourier Truncation

The idea behind the proposed data-reduction mechanism is as follows. A time series can be converted to its frequency-domain representation using Fourier transforms. If only a few terms of the Fourier transforms are sufficient to capture the majority of the variation of the time series then it suffices to transmit these few terms to the central controller. This way, the data batch as defined in Section III-A can be reduced to fewer than $n$ floating point numbers. We start by giving the necessary definitions for the proposed methodology, before describing two approaches for choosing the number of terms to include. The Fourier transform of the discrete time-domain signal or time series $u$ is denoted by $U$ and given by

$$U_f = \sum_{i=0}^{n-1} u_i e^{-j(2\pi/n)if} \,. \tag{1}$$

$U \in \mathbb{C}^n$ is a sequence of complex numbers of the same length as $u$. The Fourier transform of a real-valued time series, $u := u_0, u_1, ..., u_{n-1}$ of length $n$, where $u_i \in \mathbb{R}, 0 \leq i < n$, has the useful property of complex conjugacy, s.t.

$$u_i = u^*_{-i} \,, \tag{2}$$

where $*$ denotes the complex conjugate. Exploiting this property, only $n/2 + 1$ terms of the Fourier transforms of a real-valued time series are required to fully capture the series. Since the resource utilisation data under consideration is real-valued, we can directly use this property to reduce the data volume of the Fourier transforms to almost the same volume as the original time-domain data – recall that a complex number is represented using two floating point numbers.

To further reduce the amount of data in the frequency domain, we define a methodology referred to as Fourier truncation, which is equivalent to an adaptive low-pass filter without attenuation.

*Definition 1:* The energy of the discrete-time signal $u$ with length $n$ is given by

$$E(u) = \sum_{i=0}^{n-1} |u_i|^2 \,. \tag{3}$$

*Lemma 1:* Using Parseval's theorem and Definition 1, the energy of a signal is preserved after the Fourier transform according to

$$E(u) = \sum_{i=0}^{n-1} |u_i|^2 = \frac{1}{n} \sum_{f=0}^{n-1} |U_f|^2 \,, \tag{4}$$

where $U_f$ is the Fourier transform of $u_i$.

Lemma 1 forms the fundamental background for our Fourier truncation methodology, since it details the relationship between signal energy in the time and frequency domain. We propose two ways of choosing the number of terms $k$ to transmit to the central controller, one using an absolute error criterion and one using a relative similarity criterion.

*Definition 2:* The series $R$ is the truncated version of the Fourier transform $U$ of $u$ that includes all terms up to term $k$, s.t. $R := U_{0 \leq i < k}\,, k \leq n$. We also define the energy of the truncated series $R$ as $E(R)$.

The truncated series $R$ only has $k$ terms. By truncating and exploiting the complex conjugacy property (Equation (2)), the number of floating point numbers that have to be transmitted is therefore reduced by $2(n-k)$.

*Definition 3:* As a measure of deviation between the original time series and the inverse of the truncated frequency-domain representation, we define the root mean squared error (RMSE) between the original time series $u$ and the truncated version $R$ as

$$RMSE(u, R) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} |u_i - \mathcal{F}^{-1}(R)_i|^2}\,, \tag{5}$$

where $\mathcal{F}^{-1}$ denotes the inverse discrete Fourier transform. We refer to this measure of error as the truncation error.

Using Lemma 1 we can impose a scale-dependent error bound in terms of the RMSE defined in Equation (5), which we detail in the following Proposition.

*Proposition 1:* The RMSE caused by the truncation of the Fourier transform $U$ of the time series $u$ is bounded by $\epsilon_{RMSE}$ according to

$$RMSE(u, R) \leq \epsilon_{RMSE}\,, \tag{6}$$

if the number of terms $k$ to include in $R$ is chosen such that

$$\sqrt{\frac{1}{n}\Big(E(U) - E(R)\Big)} \leq \epsilon_{RMSE}\,, \tag{7}$$

where $E(R)$ is the energy of $R$ and $R$ is the truncated Fourier transform of $u$, defined as $R = U_{0 \leq i < k}$, according to Definition 2.

*Proof:* From Definition 2, we define the terms of the frequency-domain representation $U$ of the time series $u$, lost in the truncation of $U$ as $L := U_{k \leq i < n}$. Using Definition 1 and Lemma 1, the energy lost in the truncation of $U$ is given by

$$E(L) = E(U) - E(R)\,. \tag{8}$$

For ease of exposition, we denote $\mathcal{F}^{-1}(R)_i$ by $r_i, 0 \leq i < n$. To prove Proposition 1, we need to show that

$$RMSE(u, R) \leq \sqrt{\frac{1}{n}E(L)}\,. \tag{9}$$

By converting both $U$ and $R$ to the time domain and using Equation (8), this can be simplified to

$$\sum_{i=0}^{n-1} |u_i - r_i|^2 \leq \sum_{i=0}^{n-1} u_i^2 - r_i^2\,. \tag{10}$$

To prove Proposition 1, it is therefore sufficient to show that Equation (10) holds. Expanding the left-hand side, we get

$$\sum_{i=0}^{n-1} |u_i - r_i|^2 = \sum_{i=0}^{n-1} u_i^2 - 2u_i r_i + r_i^2\,. \tag{11}$$

By subtracting the right-hand side of Equation (10) and simplifying, this transforms to

$$\sum_{i=0}^{n-1} |u_i - r_i|^2 - \sum_{i=0}^{n-1} u_i^2 - r_i^2 = 2\sum_{i=0}^{n-1} r_i(r_i - u_i)\,. \tag{12}$$

From Lemma 1, we know that

$$\sum_{i=0}^{n-1} r_i^2 \leq \sum_{i=0}^{n-1} u_i^2 \iff 2\sum_{i=0}^{n-1}(r_i + u_i)(r_i - u_i) \leq 0\,. \tag{13}$$

Since $u_i, r_i \geq 0$

$$2\sum_{i=0}^{n-1} r_i(r_i - u_i) \leq 2\sum_{i=0}^{n-1}(r_i + u_i)(r_i - u_i) \leq 0\,, \tag{14}$$

which proves Equation (10) and therefore Proposition 1. Since $E(L)$ is monotonically decreasing in $k$, $k$ can be chosen large enough such that Equation (7) is satisfied. ∎

The error bound given in Proposition 1 uses a scale-dependent error metric and is thus only useful when the magnitude of the time series $u$ is known beforehand. To be able to generalise the methodology to arbitrary time series, a scale-independent error measure is more desirable. We propose the use of a percentage energy threshold to be captured in the truncated time series. We define this threshold value as $e \in [0, 1]$ and use the cumulative fraction of the time series' energy captured up to each term to choose the number of terms to include given the threshold value $e$. Let the series $S$ of cumulative sums of the energy captured in the terms of the Fourier transform $U$ of the time series $u$ be defined as

$$\begin{aligned} S_0 &= 0 \\ S_{i+1} &= S_i + E(U[i]). \end{aligned}$$

Further, denote the total energy of the time series or equivalently the maximum of series $S$ by $S_{max} := max(S)$. Then the normalised series $S^n$ with $S_i^n \in [0, 1]$, is

$$S_i^n = \frac{S_i}{S_{max}}\,. \tag{15}$$

The terms of $S^n$ are equivalent to the fraction of the signal's energy captured by the terms of the Fourier transform up to the $i^{th}$ term. If instead of imposing an absolute RMSE bound the procedure based on an energy threshold value is employed, the value of $k$ is chosen such that

$$E(R) = \frac{1}{n}\sum_{i=0}^{k-1} |U_i|^2 \geq E(u) \times e\,, \tag{16}$$

where $e$ is the energy threshold as defined previously. Inequality (16) is satisfied by choosing $k$ such that $e \leq S_k^n \wedge e > S_{k-1}^n$. The Inequality (16) can be transformed to

$$\frac{E(R)}{E(u)} \geq e\,. \tag{17}$$

1: Initialise empty list $u$
2: **while** $t \notin \theta$ **do**
3: Observe the variable of interest.
4: Append the new observation to $u$.
5: **end while**
6: $U = \mathcal{F}(u)$
7: Choose $k$ according to either of the two proposed truncation methodologies
8: $R = U_{0 \leq i < k}$
9: **Transmit** $R$ to the central controller $C$

Fig. 2. Compression and data collection algorithm executed at nodes $M$

The term on the left hand side of Equation (17) is the similarity between the original series and the truncated version in terms of captured energy. It can also be interpreted as the relative accuracy of the truncated version compared to the full series. This measure lies between 0 and 1 and is independent of the scale of the signal. In practice it is sufficient to specify some level of minimum similarity and choose the threshold value $e$ according to Inequality (17).

The algorithm, resulting from the truncation methodology in combination with the proposed batched data collection is given in Fig. 2.

Using the Fast Fourier Transform algorithm introduced in [29], the Fourier transforms of a time series of length $n$ can be computed with a time complexity of $\mathcal{O}(n\,log(n))$. The proposed truncation mechanisms require a complete pass of the Fourier transforms, adding another $n/2+1$ computation steps. Hence, the overall number of computation steps required for a single node is $\mathcal{O}(n\ \log(n))+n/2+1$, resulting in a time complexity of $\mathcal{O}(n\ \log(n))$.

### C. Forecasting by Complex Gated Recurrent Neural Networks

Our proposed methodology requires a forecasting model both for interpolation at time steps between consecutive batch updates as well as predicting future resource utilisation for system management purposes. We propose the use of complex Gated Recurrent Units (cGRU) [30] to forecast resource utilisation using the truncated frequency-domain representation of the resource utilisation time series that results from the data transmission methodology outlined in Section III-B. As a benchmark, we compare our approach against a time-domain GRU [31], which uses the complete time-domain representation of the input time series. The use of gates in recurrent neural networks has been shown to improve their ability to learn longer term dependencies [32] and GRUs implement a computationally efficient gating mechanism [31]. They are therefore well suited to the problem at hand.

Throughout our experiments, we employ a sliding window model that uses a window of historic data to forecast a predefined number of time steps into the future. The size of the window is defined as a multiple $w$ of the number of time steps between two successive batch updates and $l := n \times w$ is defined as the total number of observations in a window (in the time-domain). In the time-domain, the batches in a window can simply be concatenated to form the input time series for our forecasting model. The problem of forecasting a pre-defined number of time steps $s$ into the future can then be defined as

$$\hat{u}_{t+1}, ..., \hat{u}_{t+s} = \underset{u_{t+1},...,u_{t+s}}{\arg\max}\ p(u_{t+1}, ..., u_{t+s} | u_{t-l}, ..., u_t)\,, \tag{18}$$

where $\hat{u}_{t+1}, ..., \hat{u}_{t+s}$ are the predictions for the next $s$ time steps and $u_{t-l}, ..., u_t$ are the observations included in the input window. The concatenation of batches to form the input window is not easily accomplished in the frequency domain due to the variable length of the truncated Fourier transforms in the batches constituting a window. Hence, the problem formulation changes slightly to

$$\hat{u}_{t+1}, ..., \hat{u}_{t+s} = \underset{u_{t+1},...,u_{t+s}}{\arg\max}\ p(u_{t+1}, ..., u_{t+s} | \{R_1\}, ..., \{R_w\})\,, \tag{19}$$

where $\{R_i\}$ is the set of all terms in the truncated frequency-domain representation of batch $i$.

Our architecture is inspired by the methodology for cGRUs using frequency-domain input proposed in [28]. The model accepts variable length input sequences and is applied to the truncated frequency-domain representation of each batch in the input window. The model capacity is kept constant between the time and frequency-domain models as parameters are shared for each batch in the frequency-domain window. The hidden states of the GRU for each of the input batches are concatenated and passed to a linear layer to arrive at the frequency-domain forecasts. An inverse Fourier transform is then applied to these to arrive at the final time-domain predictions, which are used to calculate the prediction error for backpropagation. Our proposed architecture can therefore be written as

$$x_{j_t} = R_{j_t}\,, 1 \leq j \leq w, 0 \leq t < |R_j|\,, \tag{20}$$

$$z_{j_t} = \sigma(W_z x_{j_t} + V_z h_{t-1} + b_z)\,, \tag{21}$$

$$r_{j_t} = \sigma(W_r x_{j_t} + V_r h_{j_{t-1}} + b_r)\,, \tag{22}$$

$$h_{j_t} = \sigma(z_{j_t} \circ h_{j_{t-1}} + (1 - z_{j_t}) \circ \phi_h(W_h x_{j_t} + V_h(r_{j_t} \circ h_{j_{t-1}}) + b_h)\,, \tag{23}$$

$$\hat{u}_{t+1}, ..., \hat{u}_{t+s} = \mathcal{F}^{-1}(f(h_{1_t}, ..., h_{w_t}))\,, \tag{24}$$

where $\circ$ denotes the Hadamard product, $W$, $V$ and $b$ are parameter matrices and biases, $R_{j_t}$ denotes the $t^{th}$ term of the truncated Fourier transform of the $j^{th}$ batch in the window, $f(\cdot)$ is the linear layer for final prediction, $\phi(\cdot)$ denotes the hyperbolic tangent, and $z_{j_t}$ and $r_{j_t}$ are the update and reset gates respectively. Since the inputs $x_{j_t} \in \mathbb{C}$ are in the complex domain, all subsequent weight matrices and bias vectors are also complex. However, in the spirit of [28], it is sufficient to simply concatenate the real and imaginary parts into a vector of reals and recombine them for the inverse Fourier transform at the end of the network. Note that contrary to the time-domain model, where the batches in the input window are concatenated along the temporal axis, the frequency-domain representations are passed as separate inputs. Explicitly combining the frequency-domain representations would

entail giving up the benefits of truncation and reduce the gains in computational overhead. Hence, we choose to let the forecasting model implicitly learn the relationship between the different batches.

Model training is implemented using mini-batch gradient descent with a RMSprop optimiser [33] and bucketisation. Bucketisation refers to grouping series of similar lengths into one batch and zero-padding the shorter ones. This is required to be able to use mini-bath gradient descent, which has several advantages over stochastic gradient descent, both in terms of convergence and training speed. The prediction error is calculated using the model's time-domain predictions and the original measured data for the predicted period. Specifically, this means that we only apply our proposed data collection mechanism to the batches in the input window, but not to the predicted time steps.

*Definition 4:* We define the time averaged RMSE between the predicted values and the original data during the forecasting period for all machines, as

$$\overline{RMSE}(t,s) = \sqrt{\frac{1}{s \times p} \sum_{j=1}^{p} \sum_{i=1}^{s} |\hat{u}_{j_{t+i}} - u_{j_{t+i}}|^2}\,, \qquad (25)$$

where $s$ denotes the number of predicted time steps and $p$ is the number of machines. We refer to this error as the prediction error.

While this allows the model to learn to forecast the true time series rather than the processed one, it also means that the entire unprocessed time series has to be collected for the interval used for training. Since our methodology does not encompass model retraining, all reported communication savings etc. refer to the period after model training, where data is collected according to the methodology proposed in Sections III-A and III-B.

## IV. Illustrative Experiment

To illustrate the functionality of our proposed method and to motivate further experiments, we evaluate our approach on a synthetic dataset, consisting of a single time series. The time series is generated according to

$$y_t = sin(2\pi \times freq. \times \frac{t}{len}) + \mathcal{N}(0, \sigma)\,, \qquad (26)$$

where $freq.$ is the frequency of the sine wave occurring in the sample, $len$ is the total sample length and $\mathcal{N}(\cdot,\cdot)$ is a one-dimensional Gaussian probability distribution. Hereafter, we will refer to the Gaussian distribution as the noise component. We choose a sine wave to introduce seasonality and combine this with Gaussian noise to examine the impact of seasonality and random fluctuations on the effectiveness of our approach. Both components tend to be present in real-world data, as confirmed by our analysis of real-world datasets in Section V-A. We investigate both truncation mechanisms proposed in Section III-B, the effect of the standard deviation of the noise component on the communication savings that can be achieved as well as the inference speed attainable at different standard deviations of the noise component. We use the standard

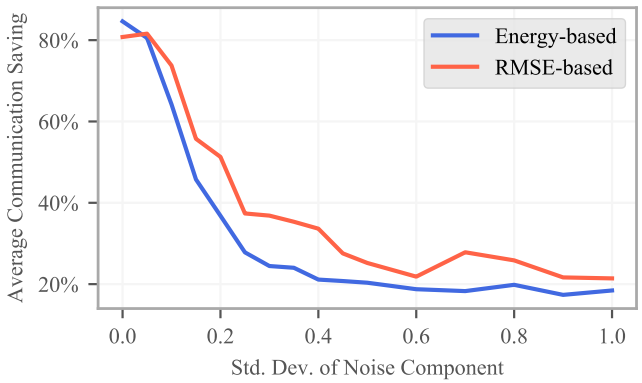


Fig. 3. Communication saving at different standard deviations used for the noise component in the synthetic dataset, using a batch length of 50 time steps and a fixed energy threshold value and RMSE threshold of $e = 0.9$ and $\epsilon_{RMSE} = 0.045$ respectively.

deviation of the noise component as a proxy for the degree of randomness in the synthetic dataset. For this illustrative experiment, we sample 3,000 time steps ($len = 3,000$) with a frequency of 15 ($freq. = 15$), using different standard deviations in the range between 0 and 1. We use a transmission batch size of 50 and use 4 batches as the input window, forecasting 50 time steps, i.e. one batch, into the future.

Fig. 3 demonstrates the relationship between the standard deviation of the noise component in the generated datasets and the achieved communication savings. The higher the standard deviation of the noise component, the higher the required number of terms, leading to lower communication savings. Both truncation methodologies perform similarly on the synthetic datasets.

These results are as expected, since very few terms of the Fourier transforms are sufficient to capture the sine wave component, but more terms are required to capture random noise fluctuations.

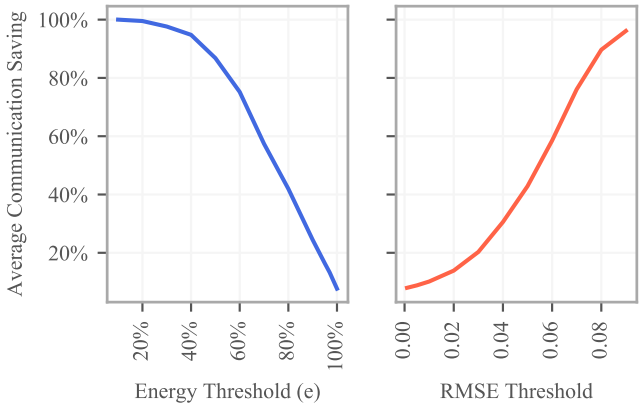


Fig. 4. Communication saving at different energy and RMSE threshold values achieved on a synthetic time series with $\sigma = 0.3$ for the noise component. The left-hand side shows the evaluation using the energy-based truncation method and the right-hand side using the RMSE threshold based truncation.

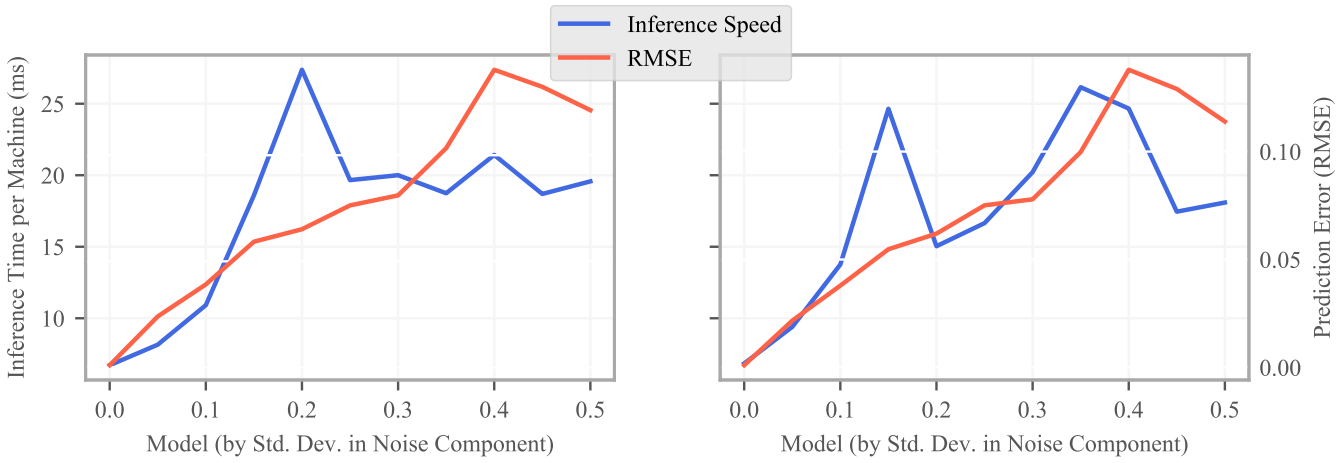


Fig. 5. Average inference speed for predicting a single batch for a single machine and the average prediction error on the entire test set for different standard deviation values for the noise component, using a batch length of 50 time steps. The left-hand side shows the results using energy-based truncation with $e = 0.9$ and the right-hand side using RMSE threshold based truncation with $\epsilon_{RMSE} = 0.045$.

Fig. 4 shows the communication savings achieved for different values for the energy and RMSE truncation threshold. Again, both methods perform similarly. However, Fig. 4 highlights a problem of using RMSE based truncation for a larger set of time series: For each time series, the magnitude needs to be known in order to set a reasonable value for the error tolerance. For instance, a time series that takes on values between 100 and 1000 will require a different RMSE threshold than one that varies between 0 and 1. While RMSE based truncation is still useful when dealing with a single or very few time series, energy based truncation is more easily applied to a large set of time series without specifying an individual error tolerance for each series. Consequently, all experiments on real-world data in the following Section are carried out using the energy based truncation mechanism.

Lastly, we examine the effect of the standard deviation of the noise component on the inference time and error of our proposed forecasting model. Using a fixed energy and RMSE threshold of 0.9 and 0.045 respectively, we can see from Fig. 5 that the prediction error as well as the inference time of the models trained on the synthetic dataset increase together. This is due to the growing inability to predict random fluctuations and the higher number of terms required to capture the series as the standard deviation of the noise component increases. The more terms of the Fourier transforms are required to approximate the time series at hand within the given error bounds, the more computations need to be performed by the forecasting model, resulting in higher inference times. The preliminary results on synthetic data confirm that the proposed methodology can be used to achieve significant communication savings as well as improvements in the inference time of recurrent neural networks. Furthermore, they allude to the limitations of using a RMSE based truncation mechanism and demonstrate some of the limitations of our approach that will be further discussed in Section VI.

## V. Experiment Results

### A. Preliminary Data Exploration

We evaluate the proposed methodology and forecasting models using two traces from large multi-purpose computing clusters, operated by Google and Alibaba respectively [34], [35]. The datasets are publicly available and have been frequently used in previous research (see for instance [1], [7], [36], [37]), enabling comparison with our approach. Both datasets are pre-processed to contain memory and CPU utilisation on a per machine basis for the entire sampling period. The Google trace contains measurements for 12,480 machines over a period of 29 days, while the Alibaba trace has a sampling period of eight days and contains measurements for 4,022 machines. Both clusters run a mixture of long-running services as well as batch workloads, co-hosted on the same set of machines. Raw samples are collected in 5 minute and 1 minute intervals in the Google and Alibaba traces, respectively. We have pre-processed the raw data to contain CPU and memory utilisation values in percent on a per machine basis, according to the methodology used in [7]. We also resample both traces to a sampling frequency of 5 minutes, i.e., a measurement is collected every 5 minutes, which results in a total of 8,351 and 2,302 observations per machine in the Google and

TABLE I
Data Summary

| Statistic | Google | Alibaba |
|---|---|---|
| Sampling Period | 29 days | 8 days |
| Number of Machines | 12,480 | 4,022 |
| Observations per Machine | 8,351 | 11,519 |
| Sampling Frequency | 5 min | 1 min |
| Size | 41GB | 48GB |
| Std. Dev. (CPU) | 0.125 | 0.156 |
| Average Utilisation (CPU) | 22.3% | 37.4% |

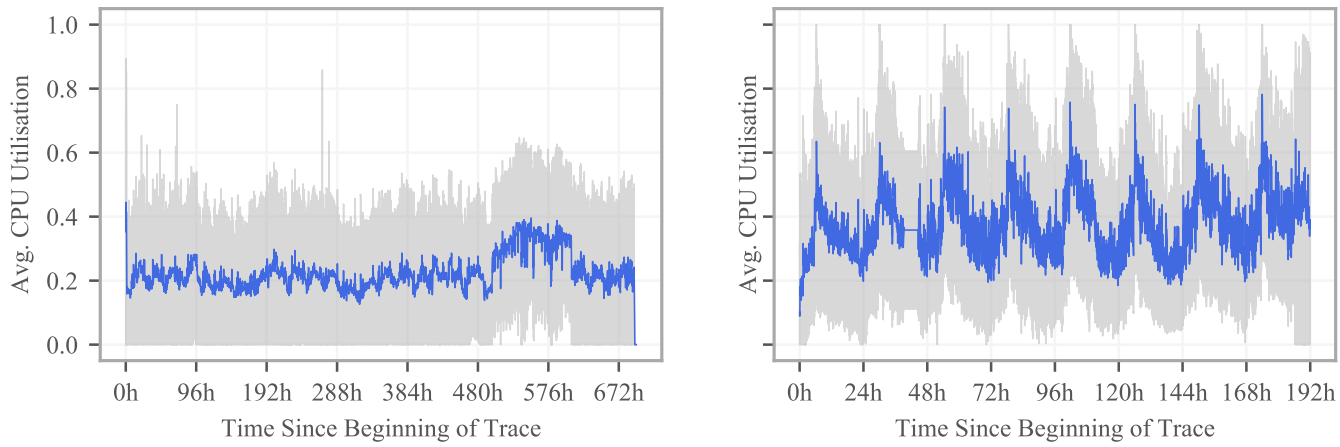


Fig. 6. Average CPU utilisation across all machine in the Google (left-hand side) and Alibaba (right-hand side) cluster traces with 95% confidence intervals.

Alibaba traces, respectively. For ease of exposition, we focus our evaluation on CPU utilisation, but the same approach can be easily applied to forecast memory utilisation. Fig. 6 shows the average CPU utilisation over the entire sampling period in the Google and Alibaba traces. While both traces exhibit daily seasonality, this seasonal component is much more pronounced in the Alibaba trace than in the Google one. This observation is also confirmed by Fig. 7, which displays the average autocorrelation at different lags for both traces. While both datasets exhibit fairly high autocorrelation, the Alibaba trace has a stronger seasonal (i.e., daily) component.

### B. Setup

Given the daily seasonality in the data, we choose a period of 24 hours, i.e., 288 time steps as the input to our forecasting models. As the batch length for the conducted experiments, we use a period of 6 hours, i.e., 72 data points, which results in a total of four batches per input window. The models are trained to predict one complete batch of utilisation data, i.e., 6 hours into the future. This represents the minimum prediction length required to fully interpolate between the batch arrival times $\theta$. Due to the large number of evaluation runs, we use a random sub-sample of 20 machines from each of the two datasets for both training and testing. All models are trained on a personal computer with 32GB of RAM, an $8^{th}$ generation Intel Core i7-8700 with 3.20GHz and 6 cores and a 256GB SATA hard drive. The truncation mechanisms from Section III-B as well as the models from Section III-C are implemented in Python. We use the PyTorch [38] library to implement the proposed machine learning models. The hyper parameters for each model – one model for each of the evaluated energy thresholds and datasets – are tuned using Bayesian Optimisation with an Expected Improvement acquisition function [39]. Since different energy thresholds results in different data characteristics, we choose to tune the hyper parameters of each model individually to ensure optimal performance. We split the dataset into three parts, using the first 50% of the time steps for training, the next 25% for hyper parameter tuning (validation) and the last 25% for testing (i.e., prediction comparison). We only report the results on the test set after hyper parameter tuning and complete retraining on the training and validation set. The reported error is calculated using the model's predictions and the original dataset, i.e., without applying our proposed truncation methodology to the predicted period, but only to the model's input window. This way of calculating the prediction error makes it more reliable as we test how well the model predicts true resource utilisation.

### C. Results

We evaluate the performance of our proposed methodology in terms of communication savings, prediction errors and inference times for different energy thresholds $e$ on both the Google and Alibaba traces. The Fourier processing mechanism leads to higher communication savings, the higher the error tolerance as demonstrated in Fig. 8, where error tolerance is expressed via the energy threshold. This relationship is better than linear and communication savings in excess of 60% can already be achieved at a small error tolerance. From Fig. 8, it is apparent that there is a clear trade-off between the communication savings achieved via our proposed methodology and the error it introduces in the data, i.e., the truncation error. The truncation error introduced in the data refers to the error between the original data and the truncated representation according to Equation (5), but not the prediction error.

While there is a trade-off between communication savings and error in the data, we do not find such a relationship for the prediction error. Fig. 9 shows the effect of different truncation thresholds on the prediction accuracy of the forecasting models. While the Fourier processing mechanism introduces some error in the data (see Fig. 8) it also has a smoothing effect that may help to filter out some random fluctuations in the data, which could otherwise have a detrimental effect on the forecasting performance. For the Alibaba trace, fore-

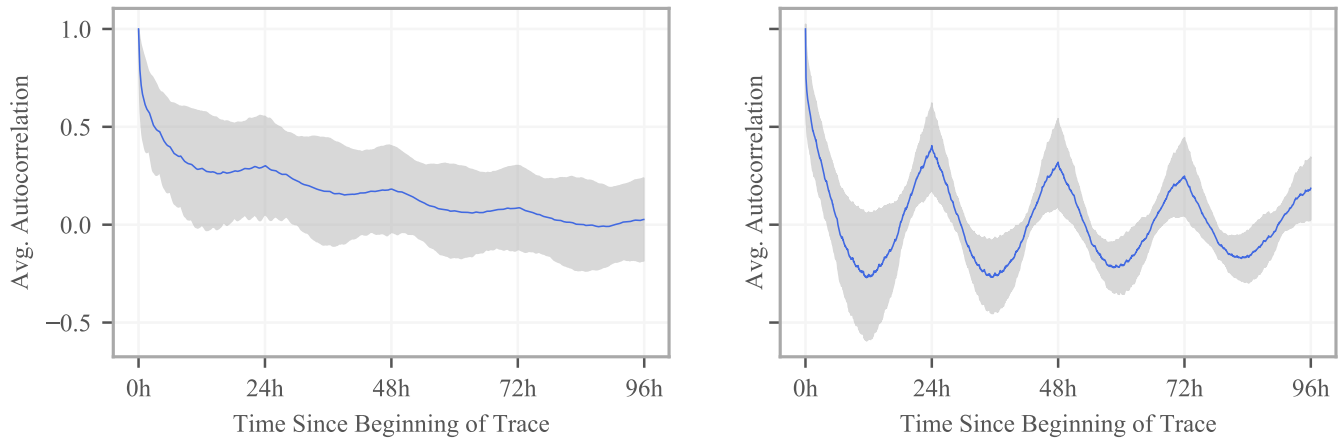


Fig. 7. Average autocorrelation across all machines in the Google (left-hand side) and Alibaba (right-hand side) cluster traces with 95% confidence intervals.

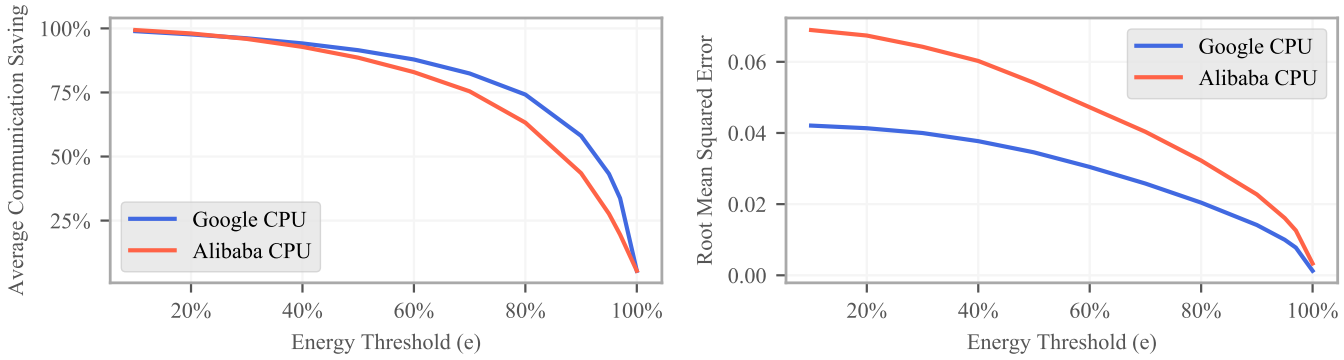


Fig. 8. Communication savings and truncation error (RMSE) obtained from the Fourier processing mechanism at different energy threshold values $e$ for the Google and Alibaba cluster trace sub-samples.

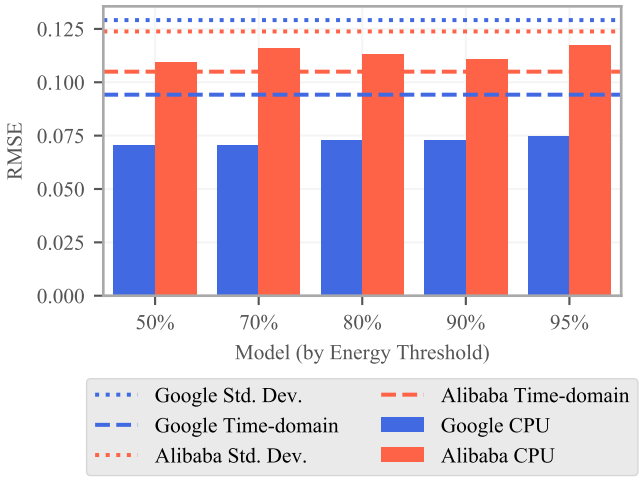


Fig. 9. Prediction error on the test set for models trained at different values of the energy threshold $e$ on the Google and Alibaba cluster traces. The time-domain benchmark models are included for both datasets.

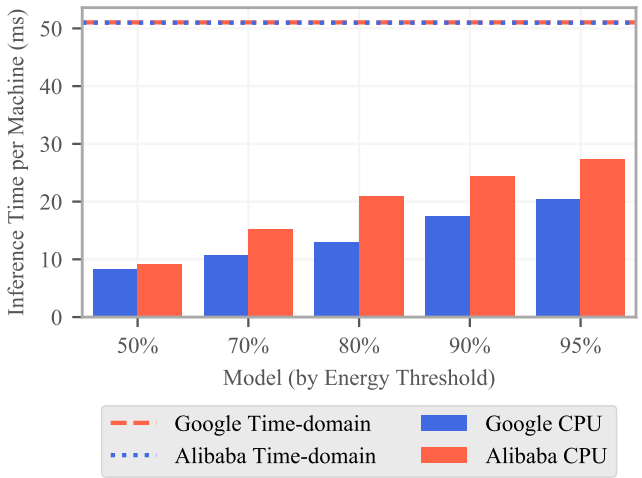


Fig. 10. Inference speed for a single batch for a single machine using the proposed model architecture, evaluated on the Google and Alibaba datasets at different energy threshold values $e$. The time-domain benchmark models are included for both datasets.

casting model performance in terms of the prediction error on the test set does not deteriorate significantly as the energy threshold level is reduced. The models trained on the Google cluster trace even exhibit an improvement in performance as the energy threshold is decreased. This improvement can be attributed to the smoothing effect of our proposed Fourier processing methodology, which reduces the noise present in the Google dataset and allows the forecasting model to learn more effectively. Generally, the models trained on the Alibaba trace perform worse than those trained on the Google trace. This may be due to a variety of factors, such as different hyper parameters and the shorter sampling period, leading to fewer data points in the sub-sample, and higher variance in the dataset.

A key benefit of the approach we propose is a large improvement in inference speed, as previously demonstrated on the synthetic dataset in Section IV. The experiments on real-world data confirm that a significant improvement in inference speed can be achieved using our Fourier truncation methodology, as Fig. 10 demonstrates. At an energy threshold level of $e = 0.9$, the inference time of forecasting one batch of 72 time steps for a single machine is reduced by more than 50% compared to the time-domain benchmark model. This further decreases to less than one fifth of the time-domain model's inference time at $e = 0.5$. This improvement in inference speed is a direct result of the reduced length of the data in the input window, which entails a reduction in the amount of computations required to forecast a single batch of data. The improvement in inference speed has two beneficial implications. On the one hand, quick inference is often required for mission critical systems. On the other hand, the reduction in computational overhead for forecasting resource utilisation could make deploying pre-trained models on less powerful machines in a distributed system a viable option.

## VI. Conclusion and Future Work

We have proposed an approach for the efficient transmission and forecasting of time series data in distributed systems. The approach combines a flexible data-reduction mechanism, integrated with a forecasting architecture that can achieve substantial improvements in communication overhead and inference speed. We demonstrate the effectiveness of our approach using real-world and synthetic datasets and provide a comprehensive evaluation of the proposed methodology. Our experiments show that communication savings of approximately 60% can already be achieved at a small error tolerance and that inference speed can be improved by more than 50% without compromising the forecasting accuracy of our proposed model. There are, however, some limitations to the approach that could be the subject of future research.

We have imposed error bounds to guide the data reduction rather than imposing explicit communication constraints. While it is possible to impose explicit communication constraints, for example, by introducing an upper bound on the number of terms that can be transmitted, this would entail loosing guarantees on the error introduced by the truncation algorithm. Extending our approach to explicitly include communication constraints is left for future work.

Our proposed truncation methodology can be described as an adaptive low-pass filter without attenuation. Specifically, this means that we use the low-frequency terms of the Fourier transforms and eliminate some of the higher frequency terms according to the captured energy compared to the energy threshold used. While this approach works well for the data at hand, different frequency bands may be desirable for other problems or datasets. For instance, if the mid-frequency range captures the signal of interest, our methodology could be changed to resemble a bandpass filter. An investigation into such an adaptation may also be a fruitful avenue for further research. Future research can also study the integration of our data collection and forecasting methodology with scheduling and system management frameworks.

## References


[1] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *Proceedings of the International Symposium on Quality of Service*, ser. IWQoS '19. New York, NY, USA: ACM, 2019, pp. 39:1–39:10. [Online]. Available: http://doi.acm.org/10.1145/3326285.3329074

[2] C. Vázquez, R. Krishnan, and E. John, "Time series forecasting of cloud data center workloads for dynamic resource provisioning," *JoWUA*, vol. 6, pp. 87–110, 2015.

[3] X. Ren, R. Lin, and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, Sep. 2011, pp. 220–224.

[4] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 300–301, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/885651.781067

[5] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*, July 2011, pp. 500–507.

[6] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155 – 162, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11001129

[7] T. Tuor, S. Wang, K. K. Leung, and B. Ko, "Online collection and forecasting of resource utilization in large-scale distributed systems," *CoRR*, vol. abs/1905.09219, 2019. [Online]. Available: http://arxiv.org/abs/1905.09219

[8] R. Hu, J. Jiang, G. Liu, and L. Wang, "Kswsvr: A new load forecasting method for efficient resources provisioning in cloud," in *2013 IEEE International Conference on Services Computing*, June 2013, pp. 120–127.

[9] W. Zhong, Y. Zhuang, J. Sun, and J. Gu, "A load prediction model for cloud computing using pso-based weighted wavelet support vector machine," *Applied Intelligence*, vol. 48, no. 11, pp. 4072–4083, Nov 2018. [Online]. Available: https://doi.org/10.1007/s10489-018-1194-2

[10] Zhenhuan Gong, Xiaohui Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management*, Oct 2010, pp. 9–16.

[11] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, April 2012, pp. 2056–2060.

[12] M. Leinonen, M. Codreanu, and M. Juntti, "Compressed acquisition and progressive reconstruction of multi-dimensional correlated data in wireless sensor networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6449–6453.

[13] J. Enric Barcelo-Llado, A. Morell, and G. Seco-Granados, "Enhanced correlation estimators for distributed source coding in large wireless sensor networks," *Sensors Journal, IEEE*, vol. 12, pp. 2799–2806, 09 2012.

[14] G. Coluccia, E. Magli, A. Roumy, and V. Toto-Zarasoa, "Lossy compression of distributed sparse sources: A practical scheme," in *2011 19th European Signal Processing Conference*, Aug 2011, pp. 422–426.

[15] C. Anagnostopoulos and S. Hadjiefthymiades, "Advanced principal component-based compression schemes for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 11, no. 1, pp. 7:1–7:34, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2629330

[16] S. Silvestri, R. Urgaonkar, M. Zafer, and B. J. Ko, "An online method for minimizing network monitoring overhead," in *2015 IEEE 35th International Conference on Distributed Computing Systems*, June 2015, pp. 268–277.

[17] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, Jun. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390681.1390689

[18] Chong Liu, Kui Wu, and Min Tsao, "Energy efficient information collection with the arima model in wireless sensor networks," in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, vol. 5, Nov 2005, pp. 5 pp.–2474.

[19] Y. W. Law, S. Chatterjea, J. Jin, T. Hanselmann, and M. Palaniswami, "Energy-efficient data acquisition by adaptive sampling for wireless sensor networks," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, ser. IWCMC '09. New York, NY, USA: ACM, 2009, pp. 1146–1151. [Online]. Available: http://doi.acm.org/10.1145/1582379.1582631

[20] H. Harb, A. Makhoul, A. Jaber, R. Tawil, and O. Bazzi, "Adaptive data collection approach based on sets similarity function for saving energy in periodic sensor networks," *Int. J. Inf. Technol. Manage.*, vol. 15, no. 4, pp. 346–363, Jan. 2016. [Online]. Available: https://doi.org/10.1504/IJITM.2016.079603

[21] S. Chatterjea and P. Havinga, "An adaptive and autonomous sensor sampling frequency control scheme for energy-efficient data acquisition in wireless sensor networks," in *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems*, ser. DCOSS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 60–78. [Online]. Available: https://doi.org/10.1007/978-3-540-69170-9_5

[22] N. Korprasertsak and T. Leephakpreeda, "Nyquist-based adaptive sampling rate for wind measurement under varying wind conditions," *Renewable Energy*, vol. 119, pp. 290 – 298, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148117312144

[23] Y. Zhu, "High performance data mining in time series: Techniques and case studies," Ph.D. dissertation, New York, NY, USA, 2004, aAI3114238.

[24] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Foundations of Data Organization and Algorithms*, D. B. Lomet, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 69–84.

[25] A. Mueen, S. Nath, and J. Liu, "Fast approximate correlation for massive time-series data," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 171–182. [Online]. Available: http://doi.acm.org/10.1145/1807167.1807188

[26] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "Fcnn: Fourier convolutional neural networks," in *Machine Learning and Knowledge Discovery in Databases*, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds. Cham: Springer International Publishing, 2017, pp. 786–798.

[27] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," 2013.

[28] M. Wolter and A. Yao, "Fourier rnns for sequence analysis and prediction," *CoRR*, vol. abs/1812.05645, 2018. [Online]. Available: http://arxiv.org/abs/1812.05645

[29] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965. [Online]. Available: http://www.jstor.org/stable/2003354

[30] M. Wolter and A. Yao, "Complex gated recurrent neural networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 10 536–10 546. [Online]. Available: http://papers.nips.cc/paper/8253-complex-gated-recurrent-neural-networks.pdf

[31] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

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

[33] G. Hinton, N. Srivastava, , and K. Swersky, "Lecture 6a overview of mini–batch gradient descent," 2012. [Online]. Available: https://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf

[34] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[35] H. Ding, "Alibaba cluster data," https://github.com/alibaba/clusterdata, 2018, accessed: 2019-05-31.

[36] F. Li and B. Hu, "Deepjs: Job scheduling based on deep reinforcement learning in cloud data center," in *Proceedings of the 2019 4th International Conference on Big Data and Computing*, ser. ICBDC 2019. New York, NY, USA: ACM, 2019, pp. 48–53. [Online]. Available: http://doi.acm.org/10.1145/3335484.3335513

[37] S. Ismaeel and A. Miri, "Using ELM techniques to predict data centre VM requests," in *IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*. New York, NY, USA: IEEE, Nov. 2015.

[38] PyTorch Contributors. PyTorch online documentation. https://pytorch.org/docs. Accessed: 2019-08-13.

[39] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, Dec 1998. [Online]. Available: https://doi.org/10.1023/A:1008306431147