CUFF: A Configurable Uncertainty-driven Forecasting Framework for Green AI Clusters

Priyanka Mary Mammen
University of Massachusetts Amherst
pmammen@cs.umass.edu

Noman Bashir
University of Massachusetts Amherst
nbashir@engin.umass.edu

Ramachandra Rao Kolluri
IBM Australia Ltd
rkolluri@au1.ibm.com

Eun Kyung Lee
IBM Research
eunkyung.lee@us.ibm.com

Prashant Shenoy
University of Massachusetts Amherst
shenoy@cs.umass.edu

ABSTRACT

AI applications are driving the need for large dedicated GPU clusters, which are highly energy- and carbon-intensive. To efficiently operate these clusters, operators leverage workload forecasts that inform resource allocation decisions to save energy without sacrificing performance. The traditional forecasting methods provide a single-point forecast and do not expose the uncertainty about their predictions, which can lead to an unexpected loss in performance. In this paper, we present an uncertainty-driven GPU demand forecasting framework that exposes the uncertainty in its predictions and provides a mechanism to configure the trade-off between energy savings and performance. We evaluate our approach using multiple GPU workload traces and demonstrate that the forecasting framework, called CUFF, outperforms state-of-the-art point predictions. CUFF predictor meets performance goals 83% of the time compared to 7.6% for the point predictions under high GPU demand. Furthermore, CUFF knob enables users to configure up to 98% performance target while providing 26% energy savings, comparable value to point forecasts that only ensure 68% performance target.

ACM Reference Format:
Priyanka Mary Mammen, Noman Bashir, Ramachandra Rao Kolluri, Eun Kyung Lee, and Prashant Shenoy. 2023. CUFF: A Configurable Uncertainty-driven Forecasting Framework for Green AI Clusters. In *The 14th ACM International Conference on Future Energy Systems (e-Energy '23), June 20–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3575813.3595203

1 INTRODUCTION

Advances in artificial intelligence (AI) and deep learning have enabled novel applications ranging from autonomous driving to chat-based question-answering tools like ChatGPT. These advances are driven by AI workloads that have been doubling roughly every 3.5 months, a rate much faster than Moore's law [15]. At the same time, the size of deep learning models is growing exponentially, with OpenAI's GPT-3 model comprising of 175 billion parameters and a memory footprint of 350GB [2]. Not surprisingly, researchers have begun to consider the environmental impact of such AI workloads, with one study estimating that the emissions from training a single AI model can exceed many common activities [4, 14].

Consequently, green AI where AI workloads are run in a sustainable manner is quickly emerging as an important research area. Since AI-based training workloads run on large dedicated GPU clusters, which are significantly more power-hungry than CPU clusters, reducing the energy and carbon footprint of GPU clusters is an important challenge in this area. Researchers have recently proposed many promising techniques to optimize energy and carbon footprint of large cloud workloads [18]. This includes time-shifting batch workloads such as ML training to time periods with plentiful low carbon electricity, intelligent power management of GPUs to eliminate idle power consumption, and even spatial shifting the training workloads to "green" cloud regions [13].

An important prerequisite for higher-level sustainability optimizations is accurate forecasting of loads from cloud and AI applications; since such forecasts are key for decision-making. Load forecasts (aka demand forecasts) is a well-studied problem both in cloud computing and energy grids [8, 11]. Conventional load forecasting techniques have ranged from time series forecasting [8], regression-based techniques [7, 9], hidden markov model [1], and more recently deep learning models. In general, these approaches work well when the demand patterns vary in a smooth manner or have temporal correlations. Load forecasting becomes more challenging when the workload exhibits burstiness or load spikes, such as flash crowds in web workloads. Current techniques produce point forecasts of future workload in both cases even though there may be a greater uncertainty in the predictions in the later case.

Our work argues that the load forecasting should incorporate uncertainty quantification, where confidence in the forecasts is provided alongside the forecasts themselves. Such uncertainty quantification based forecasting can significantly improve higher level methods such as optimizing energy or carbon used in AI clusters while maintaining its performance since optimizations can become more conservative when uncertainty is high (i.e. confidence is low) or more aggressive when uncertainty is low.

Motivated by these observations, in this paper, we present a Configurable Uncertainty-driven Forecasting Framework (CUFF) for GPU clusters used for AI workloads such as ML training. In designing and evaluating our GPU demand forecasting framework, we make the following contributions.

- We present an uncertainty-driven GPU demand forecasting model that explicitly incorporates uncertainty information in forecasts.
- We expose a higher-level knob that leverages uncertainty-driven forecasting and allows users to configure the trade-off between energy savings and performance.
- Using multiple GPU workload traces, we demonstrate that CUFF predictor meets performance goals 83% of the time compared to 7.6% for the point predictions under high GPU demand. Furthermore, CUFF knob enables users to configure up to 98% performance target while providing 26% energy savings, comparable value to point forecasts that only ensure 68% performance target.

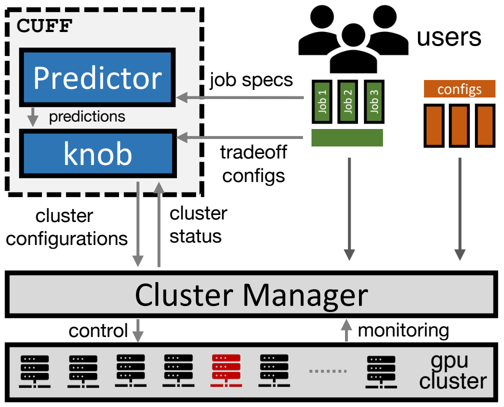


Figure 1: *System architecture of Configurable Uncertainty-driven Forecasting Framework (CUFF).*

2 BACKGROUND AND RELATED WORK

Cloud and GPU workloads. Cloud GPU workloads are mainly of two types: Deep Learning (DL) training jobs and DL inference tasks. DL training workloads train models using existing data and are very resource intensive. DL inference jobs make predictions using inputs from users and often have strict latency requirements (as less as 100ms) [3]. Most of the existing deep learning workloads in the public infrastructure are of inference category [16].

Existing GPU cluster traces and analysis. Recently, companies like Microsoft, SenseTime and Alibaba have published their GPU cluster traces. It is observed from these traces that GPU nodes are underutilized most of the hours. To overcome the issue of underutilization, in [5] developed ANDREAS, a job scheduling algorithm to keep the GPU nodes busy all the time. Similarly in [6] using helios clusters data, proposed to turn on/off the servers leveraging prediction values from Gradient Boosted Decision Trees.

GPU power management and their methodologies. With the improved chip technologies, DVFS is an effective methodology to reduce energy consumption in idle GPUs. When we scale the frequency of the idle components to minimum, power consumption also will be reduced to minimum. Additionally, dynamic power management (DPM) is also utilized in GPU nodes to save energy during idle times. This technique involves transitioning the node into different sleep states, which can result in significant energy savings compared to DVFS. However, DPM has a higher setup time and not all machines support different sleep states, limiting its applicability. Furthermore, power gating is another popular technique where the power to idle functional units is turned off. This results in a low power state and minimal energy consumption, but also has a higher recovery time compared to DPM and DVFS.

Load forecasting techniques. Initially, people used statistics-based load forecasting approaches such as ARIMA, Moving Average for load prediction in data centers. With the advent of deep learning models, people started using sophisticated models such as LSTM, GRU, ensemble models, etc for better prediction accuracy. All of the above methods can be categorized as point-based workload prediction techniques, which provide no measure of prediction uncertainty. Recently Rossi et. al [17] proposed a workload prediction technique using a Bayesian layer in a neural network model producing a gaussian distribution with a predicted mean and variance.

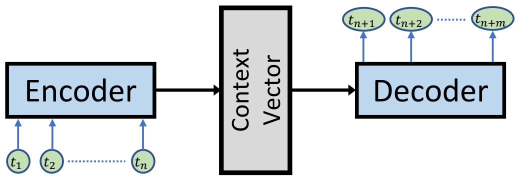


Figure 2: *Encoder-decoder based neural network architecture used for the CUFF predictor.*

The parametric distribution is prone to noise and less robust when compared to a specific quantile of the forecast distribution and thus, not the most suitable value to use for optimal decision making [19].

3 CUFF DESIGN

In this section, we present an high-level overview of our Configurable Uncertainty-driven Forecasting Framework (CUFF). We also describe the design and implementation of its different modules.

3.1 CUFF Architecture

Figure 1 shows the high-level architecture of CUFF. In addition to CUFF module, architecture diagram shows two additional entities that CUFF interacts with for its operation: *users* and *cluster manager*.

Users submit jobs to the cluster manager and job specs are also provided to CUFF module. Users also specify the configurations for the performance vs. energy efficiency tradeoff. For example, a user may want to maximize the energy efficiency while satisfying the resource requirements for 90% of the time.

Cluster manager, in our context, takes cluster configurations from CUFF and controls the state of GPUs within the cluster. It also monitors the physical GPU cluster to get energy consumption, which is provided to CUFF alongside other job related statistics such as percentage of jobs rejected and job resource requirements.

We describe the different sub-modules of CUFF module next.

3.2 CUFF Predictor

Goal. The goal of this module is to take a time series data of GPU resource allocation with n observations from the past and predict the next m values with a predefined quantile confidence value q.

Prediction model. Our forecasting approach uses a multi-horizon strategy instead of a recursive forecasting strategy. To do so, we directly predict next m values (y_{t+1}, y_{t+2}, ..., y_{t+m}) based on the past n values (y_{t-n}, ..., y_{t-1},y_t) to avoid error accumulation that can happen in a recursive approach. As we want to estimate an upper bound of the predicted resources with a desired confidence level, we use quantile regression to forecast the conditional quantiles,

$$P(y_{t+1}, y_{t+2}...y_{t+m}/y_{t-n}, ...y_{t-1}, y_t) = q. \tag{1}$$

In quantile-based forecasting, we train the models by jointly minimizing the quantile loss calculated across all quantiles.

$$L(y, y^*) = max(q(y - y^*), (1 - q)(y^* - y)) \tag{2}$$

here, y is the true value, y^* is the predicted, value and q is the given quantile. At $q = 0.5$, we get the mean prediction values.

Neural network architecture. To train our uncertainty-driven forecasting model, we use the encoder-decoder architecture for neural networks. Figure 2 shows the high-level encoder-decoder architecture. Input to the encoder is the observed time series values and input to the decoder is the context vector/output from

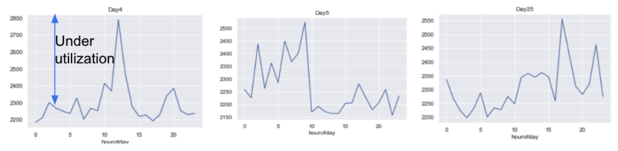


Figure 3: ***GPU cluster daily trends- 3 representative traces from each of the dataset.***

Dataset	GPU-specific Attributes	Duration	GPUs
Alibaba	allocation, utilization	2 months	6000
Venus (Helios)	allocation, utilization	2.5 months	2490
Philly (Microsoft)	allocations, utilization	6 months	1064

Table 1: ***Summary of datasets used in evaluation.***

Dataset	Delay (at 1 minute)	Delay (at 2 minutes)
Alibaba	17 kWh	26.3 kWh
Venus (Helios)	11.85 kWh	20.55 kWh
Philly (Microsoft)	26 kWh	39 kWh

Table 2: ***24hrs energy overhead from state-transition delays.***

the encoder. The choice of this architecture is motivated by their suitability for sequence-to-sequence (seq2seq) predictions and ability to handle variable input and output sequences, which may be needed if an insufficient historic data is available or forecasts on a longer/shorter horizon are needed to handle variable workload.

3.3 CUFF Knob

This module models the *energy savings* vs. *performance* relationship and allows users to configure the desired performance level while maximizing the energy savings. Here, we define *performance* as the percentage of jobs that are allocated resources and are executed. For example, if a user submits 100 jobs and only 80 get to run since GPUs are not available, its *performance* is 80%. *Energy savings* are defined as the ratio between the amount of reduced energy and the total energy when all GPUs are ON. For example, if there are 100 homogeneous GPUs, an *energy savings* of 25% would mean turning off 25 GPUs in anticipation of under-utilization. As the *performance* requirements increase, the *energy savings* decrease and vice versa.

Given these definitions, the task of the module is to learn the *energy savings* vs. *performance* relationship on per user, workload, and cluster basis. *Knob* module is trained using historical workload traces, cluster configurations, and energy usage information from similar clusters. Any simple or a machine learning model can be used for this task. At run-time, the trained *knob* model receives the full distribution of GPU demand predictions, cluster status from cluster manager, and the *performance* goals from the users. Given these variables, it decides on the minimum number of GPUs required to satisfy the *performance* objectives.

4 IMPLEMENTATION AND EVALUATION

In this section, we evaluate the accuracy of CUFF predictor in forecasting GPU demand with the desired confidence level. We also establish a relationship between *performance* and *energy savings*.

4.1 Experimental Setup

Datasets. We use three publicly available GPU traces: Alibaba trace [20], Microsoft Philly trace [10], and Helios trace [6]. Table 1 shows a high level summary of all the datasets. Figure 3 shows the average demand on a typical day from each dataset. It demonstrates that there is a significant variation in GPU demand and the GPU clusters are not fully utilized most of the time.

Model	Alibaba		Venus		Philly	
	under (%)	MAE	under (%)	MAE	under (%)	MAE
GRU	76.72	10.09	76.72	**10.09**	28.18	**8.66**
Bi-LSTM	**59.32**	**7.54**	**58.27**	14.56	**14.21**	9.95
CNN	68.90	12.41	82.72	19.71	33.12	11.75

Table 3: ***5-step prediction performance of point predictor across all three traces using mean squared error loss.***

Model	Alibaba		Venus		Philly	
	under (%)	MAE	under (%)	MAE	under (%)	MAE
GRU	69.68	**8.68**	69.04	**12.21**	54.83	**7.06**
Bi-LSTM	**27.63**	10.92	**32.32**	16.20	77.95	8.91
CNN	77.18	13.79	52.99	16.03	**51.63**	10.77

Table 4: ***5-step prediction performance of* CUFF *predictor across all three traces using median quantile settings ($q = 0.5$).***

Models for* CUFF *predictor and baseline. We use different deep learning models, including Bi-LSTM, GRU and CNN, to implement the encoder-decoder architecture of CUFF predictor. In training these models for uncertainty-driven forecasting, we train with a quantile loss function with the desired q values. As a fair baseline, we use these same models trained with mean squared error (mse) as a loss function to generate point-predictions for GPU demand.

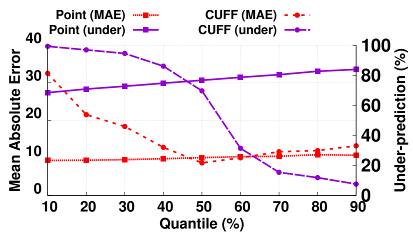
Training and testing procedure. We first compute the overall demand for GPUs at every minute by aggregating resources allocated to all running jobs. We use a look-up horizon of 2 hours ($n = 120$) with a prediction length of 5 minutes ($m = 5$). While forecasts for shorter horizon of up to 1 minute are possible, we make predictions for the next five minutes to avoid fluctuations in number of GPUs required and provide cluster manager time to enforce the configuration decisions. For training, we use quantiles with 0.1 interval up to 0.9 and 0.01 from 0.9 to 0.99. The hyper-parameters for all models for each dataset are selected using grid search.

Metrics. We use three metrics while evaluating CUFF. First metric is the mean absolute error between the actual GPU demand and the forecasted demand, which quantifies the accuracy of forecasting model. Second metric is the under-prediction (%), which is computed as the percentage of time slots when the forecasted demand was less than the actual demand and jobs could not be served. It quantifies the performance and lower value is better. Finally, the third metric is the percentage energy savings, which are computed as the amount of energy consumption reduced as compared to always on status for all GPUs. Higher value is better.

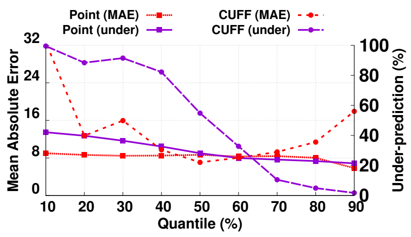
4.2 Benchmarking Experiments

Different GPU power saving methods provide a trade-off between energy savings and state transition delay (time needed to bring the GPU online). First, our empirical experiments on in-house GPUs suggest that power consumption of a single GPU can be reduced by up to 25W by transitioning to the lowest possible frequency with less transition delay. As an alternative, we can save more than 50% of the power by putting a GPU node into a deep sleep state, but bringing it online will incur a significant delay.

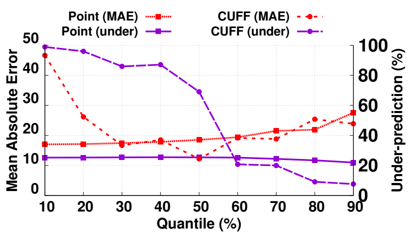
In addition, a large state transition delay can incur significant energy overhead. We conduct empirical experiments with all the dataset traces assuming we make decisions every one minute and save 25W power by transitioning to the lowest possible frequency. Table 2 shows the results of the experiment that higher state transition delay leads to high energy overhead and less energy savings.



(a) Alibaba

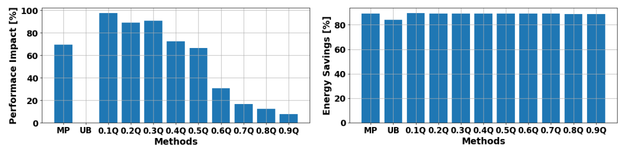


(b) Philly

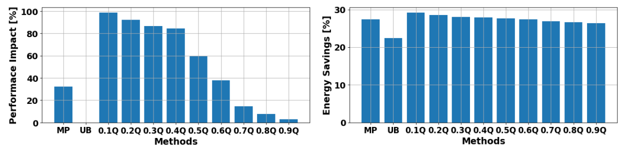


(c) Venus

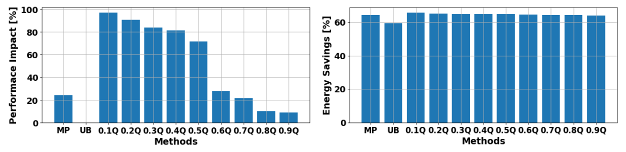
Figure 4: ***Mean absolute error (left y-axis) and under-prediction (right y-axis) results for both point predictions and* CUFF *predictor for all datasets. Look at the red curves for MAE comparison and look at the dark violet curves for under-prediction comparison. Circles (●) represent* CUFF *predictor and squares (■) represent point predictions.***



(a) Alibaba



(b) Philly



(c) Venus

Figure 5: ***Configuring performance versus energy saving knob.***

Key takeaways. *Cluster managers should choose power management approach with low transition delay when GPU demand is fluctuating and use deep sleep states when GPU demand is stable.*

4.3 Evaluating CUFF Predictor Accuracy

We evaluate the accuracy of our forecasting module in two steps. First, we compare the MAE and under-prediction percentage for various DL model options for all the datasets using a quantile setting of 0.5, as shown in Table 4. We compare it with the baseline of point forecasts, shown in Table 3. Our uncertainty-driven forecasting approach achieves comparable performance to point forecasts on MAE for all the traces (within 2 points), but significantly outperform point forecasts on under-prediction percentage for Alibaba and Venus traces. Point forecasts for Philly trace are better on under-prediction metric and we plan to investigate the reason in the future. Furthermore, as GRU fairs much better than other models across three traces, we pick it for all the future experiments.

Second, we compare the performance across all the quantiles, as shown in Figure 4. This will allow us to understand how CUFF predictor compares with the point forecasts as the GPU load changes. On MAE, at higher than median workload, CUFF predictor always have a comparable performance to the point forecasts. Also, as cluster sizes are in the range of hundreds to thousands, an MAE value of under 50 across all datasets is reasonable. A better metric is the under-prediction percentage as it tells how often these small errors occur, which can be problematic. Again, under high workloads, CUFF significantly outperforms point forecasts.

Key takeaways. CUFF *always performs better on under-prediction metric and has best performance for Alibaba trace, 7.6% versus 83% (lower is better). It has a comparable performance on MAE metric.*

4.4 Configuring CUFF Knob

As discussed in CUFF design, the purpose of the knob module is to establish a configurable mapping between performance and energy savings. To do that, we run both point forecasts, threshold-based forecasts (point prediction + 5% extra GPUs) [12], and CUFF predictor at different workload levels and monitor the performance (measured in under-prediction percentage) and energy savings. The results for all the traces are shown in Figures 5. The performance impact of CUFF is very low at high workloads, as establish in the previous section, but the energy savings almost remain constant. The threshold approach has absolutely no case of under-provisioning. This is because we added a larger value to the point predictions to hedge against under-predictions. However, this hedging costs additional energy and results in reduced energy savings across all approaches. Similarly, there is a decreasing trend for under-predictions for an increase in value of Q values.

Key takeaways. CUFF *knob enables users to set precise performance targets and take over the task of maximizing energy savings. It rovides upto 98% performance guarantee while yielding 26% energy savings.*

5 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we present an uncertainty-driven GPU demand forecasting framework, called CUFF. We demonstrate that CUFF predictor meets performance goals 83% of the time compared to 7.6% for the point predictions under high GPU demand. Furthermore, CUFF knob enables users to configure up to 98% performance target while providing 26% energy savings, comparable value to point forecasts with only 68% performance measure. As a next step, we plan to extend our work by adding an optimization module to decide the energy state of individual GPUs considering various constraints such as predicted GPU usage, state-transition delays etc.

ACKNOWLEDGMENTS

This work is supported in parts by NSF grants 1836752, 1722792, 1763834, 1802523, 2230143, 2107133, 2105494, 2213636, 2211302, 1908536, 1925464, 2211888, US Army contract W911NF-17-2-0196, and CX-027429

REFERENCES

[1] Ahmed Adel and Amr El Mougy. 2022. Cloud Computing Predictive Resource Management Framework Using Hidden Markov Model. In *2022 5th Conference on Cloud and Internet of Things (CIoT)*. IEEE, 205–212.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[3] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System.. In *NSDI*, Vol. 17. 613–627.

[4] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the Carbon Intensity of AI in Cloud Instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency* (Seoul, Republic of Korea) *(FAccT '22)*. Association for Computing Machinery, New York, NY, USA, 1877–1894. https://doi.org/10.1145/3531146.3533234

[5] Federica Filippini, Danilo Ardagna, Marco Lattuada, Edoardo Amaldi, Maciek Riedl, Katarzyna Materka, Paweł Skrzypek, Michele Ciavotta, Fabrizio Magugliani, and Marco Cicala. 2021. ANDREAS: Artificial intelligence traiNing scheDuler foR accElerAted resource clusterS. In *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 388–393.

[6] Qinghao Hu, Peng Sun, Shengen Yan, Yonggang Wen, and Tianwei Zhang. 2021. Characterization and prediction of deep learning workloads in large-scale gpu datacenters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.

[7] Rongdong Hu, Jingfei Jiang, Guangming Liu, and Lixin Wang. 2013. CPU load prediction using support vector regression and Kalman smoother for cloud. In *2013 IEEE 33rd international conference on distributed computing systems workshops*. IEEE, 88–92.

[8] Rongdong Hu, Jingfei Jiang, Guangming Liu, and Lixin Wang. 2014. Efficient resources provisioning based on load forecasting in cloud. *The Scientific World Journal* 2014 (2014).

[9] Md Toukir Imam, Sheikh Faisal Miskhat, Rashedur M Rahman, and M Ashraful Amin. 2011. Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources. In *14th International Conference on Computer and Information Technology (ICCIT 2011)*. IEEE, 333–338.

[10] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Wencong Xiao, and Fan Yang. 2019. Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads.. In *USENIX Annual Technical Conference*. 947–960.

[11] Shahida Khatoon, Arunesh Kr Singh, et al. 2014. Effects of various factors on electric load forecasting: An overview. In *2014 6th IEEE Power India International Conference (PIICON)*. IEEE, 1–5.

[12] Dorian Minarolli, Artan Mazrekaj, and Bernd Freisleben. 2017. Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. *Journal of Cloud Computing* 6 (2017), 1–18.

[13] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2022. *The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink*. Technical Report. Google Inc.

[14] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training.

[15] Raymond Perrault, Yoav Shoham, Erik Brynjolfsson, Jack Clark, John Etchemendy, Barbara Grosz, Terah Lyons, James Manyika, Saurabh Mishra, and Juan Carlos Niebles. 2019. The AI Index 2019 Annual Report. *AI Index Steering Committee, Human-Centered AI Institute, Stanford University, Stanford, CA* (2019).

[16] Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021. INFaaS: Automated Model-less Inference Serving.. In *USENIX Annual Technical Conference*. 397–411.

[17] Andrea Rossi, Andrea Visentin, Steven Prestwich, and Kenneth N Brown. 2022. Bayesian uncertainty modelling for cloud workload prediction. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 19–29.

[18] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. Ecovisor: A Virtual Energy System for Carbon-Efficient Applications. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 252–265. https://doi.org/10.1145/3575693.3575709

[19] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053* (2017).

[20] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. 2022. MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, 945–960.