RESEARCH-ARTICLE

Predictive Resource Management in the Computing Continuum: Transfer Learning from Virtual Machines to Containers using Transformers

DANNY DE NOVI, University of Messina, Messina, ME, Italy

LORENZO CARNEVALE, University of Messina, Messina, ME, Italy

DANIEL BALOUEK, INRIA Institut National de Recherche en Informatique et en Automatique, Le Chesnay, Ile-de-France, France

MANISH PARASHAR, INRIA Institut National de Recherche en Informatique et en Automatique, Le Chesnay, Ile-de-France, France

MASSIMO VILLARI, University of Messina, Messina, ME, Italy

Open Access Support provided by:

University of Messina

INRIA Institut National de Recherche en Informatique et en Automatique

Published: 01 December 2025

Citation in BibTeX format

UCC '25: 2025 IEEE/ACM 18th International Conference on Utility and Cloud Computing
December 1 - 4, 2025
France, France

Conference Sponsors:
SIGARCH

Predictive Resource Management in the Computing Continuum: Transfer Learning from Virtual Machines to Containers using Transformers

Danny De Novi*
University of Messina
Messina, Italy
dadenovi@unime.it

Lorenzo Carnevale*
University of Messina
Messina, Italy
lcarnevale@unime.it

Daniel Balouek*
INRIA
Nantes, France
daniel.balouek@inria.fr

Manish Parashar*
INRIA
Salt Lake City, Utah, USA
manish.parashar@utah.edu

Massimo Villari*
University of Messina
Messina, Italy
mvillari@unime.it

Abstract

Efficient workload forecasting is a key enabler of modern AIOps (Artificial Intelligence for IT Operations), supporting proactive and autonomous resource management across the computing continuum, from edge environments to large-scale cloud infrastructures. In this paper, we propose a Temporal Transformer architecture for CPU utilization prediction, designed to capture both short-term fluctuations and long-range temporal dependencies in workload dynamics. The model is first pretrained on a large-scale Microsoft Azure VM dataset and subsequently fine-tuned on the Alibaba container dataset, enabling effective transfer learning across heterogeneous virtualization environments. Experimental results demonstrate that the proposed approach achieves high predictive accuracy while maintaining a compact model size and inference times compatible with real-time operation. Qualitative analyses further highlight the model's ability to reproduce workload patterns with high fidelity. These findings indicate that the proposed Temporal Transformer constitutes a lightweight and accurate forecasting component for next-generation AIOps pipelines, suitable for deployment across both cloud and edge intelligence scenarios.

CCS Concepts

• **Computing methodologies** → *Neural networks*; **Supervised learning**; • **Computer systems organization** → **Cloud computing**; *Neural networks.*

Keywords

Cloud-Edge Continuum, Time-Series Forecasting, Artificial Intelligence Operations, Temporal Transformer

*Both authors contributed equally to this research.

UCC '25, Nantes, France

ACM ISBN 979-8-4007-2285-1/25/12
https://doi.org/10.1145/3773274.3774856

ACM Reference Format:
Danny De Novi, Lorenzo Carnevale, Daniel Balouek, Manish Parashar, and Massimo Villari. 2025. Predictive Resource Management in the Computing Continuum: Transfer Learning from Virtual Machines to Containers using Transformers. In *2025 IEEE/ACM 18th International Conference on Utility and Cloud Computing (UCC '25), December 01–04, 2025, Nantes, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3773274.3774856

1 Introduction

The continuous growth and accelerated deployment of cloud, fog, and edge infrastructures have brought the computing continuum paradigm into the spotlight. This paradigm proposes to unify diverse resources, from large cloud data centers and intermediate fog nodes to geo-distributed edge nodes and billions of IoT devices, into a cohesive, end-to-end infrastructure. By leveraging heterogeneous and dynamically shared resources, the continuum enables latency-sensitive applications, real-time analytics, and pervasive artificial intelligence [6].

Efficient resource management across heterogeneous and dynamic computing environments remains a major technical challenge. The variability of workloads, especially in mission-critical contexts such as disaster response or real-time industrial control, demands intelligent and adaptive operational strategies. In this regard, the AIOps (Artificial Intelligence for IT Operations) paradigm has emerged as a key enabler, leveraging data-driven methods to automate decision-making, support self-adaptation, and improve predictive control in distributed infrastructures. Accurate workload forecasting is central to this vision, as it underpins the allocation, provisioning, and dynamic reconfiguration of resources based on neighboring capacities and connectivity states. Robust prediction mechanisms thus form the backbone of AIOps systems, enabling proactive offloading, optimized scheduling, and sustained performance and energy efficiency at scale [9, 12, 18].

Recent advances in machine learning have improved workload forecasting, particularly in cloud environments where Virtual Machines (VMs) are predominant. Yet, with the growing adoption of containerization, offering greater elasticity but introducing higher volatility, resource allocation must adapt to finer-grained and more dynamic execution environments. Bridging predictive models for

CPU utilization developed for VMs with the requirements of container-based systems is a key enabler for AIOps-driven resource orchestration within the computing continuum. Traditional time-series methods often struggle to generalize across these heterogeneous environments and to capture both short-term variability and long-range dependencies, limiting their applicability in real-world operational pipelines.

To address these limitations, we investigate the use of Temporal Transformer architectures. Transformers, originally proposed in the natural language processing domain, have recently demonstrated strong performance in time-series forecasting due to their ability to model complex temporal dependencies without the constraints of recurrent architectures [19].

In this paper, we propose a Temporal Transformer architecture for CPU forecasting designed specifically for the computing continuum and aligned with the objectives of AIOps. We investigate the critical aspect of knowledge transfer by adapting predictive models trained on VM workload traces to containerized environments. Evaluation is performed on real-world datasets, utilizing a pre-training phase on Microsoft Azure VM traces and fine-tuning on Alibaba container traces. Our core design goals are to achieve: accurate workload prediction for short-term forecasts, robustness under novel workload conditions, and low-latency, lightweight inference suitable for both cloud and resource-constrained edge/fog scenarios.

The remainder of the paper is organized as follows. Section 2 introduces related work on resource management, workload forecasting, and AIOps frameworks. Section 3 describes the selected datasets, formulates the problem, and presents the proposed model. The experimental setup, metrics, and results are discussed in Section 4. Finally, Section 5 concludes the paper and outlines directions for future work.

2 Related Work

Resource management plays an important role in optimizing large and micro-data centers [7, 8]. In this regard, we are interested in the role of CPU utilization for the prediction of the workload. The state of the art is an exploration of the literature mixing the keywords *CPU*, *usage*, *utilization*, *prediction* and *forecasting*. Generally speaking, the most popular models adopted by the scientific communities are the ARIMA and LSTM [14, 16] models. Not many solutions adopt Transformers to solve the task. In the following, we reference the most relevant ones.

Duggan et al. [5] proposed a study which focused on the comparison between prediction models that were one and many steps ahead. The model was a RNN with a Back-Propagation-Through-Time (BPTT) algorithm. PlanetLab was the dataset, out of 800 cloud hosts machines simulated with the CloudSim simulator. CPU utilization values were measured every 5 minutes. The model converges to a good solution in a short time, outperforming the compared models (e.g., Random Walk, Moving Avg, Backpropagation). The analysis on multi-steps highlights a slow degradation of the model while the steps increase. This means that the model may be used to train more steps ahead (e.g., 20, 30 minutes), but this is inconvenient to extend the time.

Janardhanan et al. [11] investigated CPU workload forecasting by comparing the LSTM and ARIMA models. Their study used Google Cluster Data Trace, which includes resource monitoring for 29 consecutive days. The dataset is over 300 GB and consists of scheduling information of over 12000 data center machines. Data for a single machine were extracted, creating a new dataset of about 8300 readings. The ARIMA model correctly predicts near-term predictions, but failed to predict long-term values. The LSTM forecasted both near-term and long-term values better. The paper lacked reproducibility, such as input shape and size, accurate details on the LSTM architecture, or any source link for codebase and dataset.

Mason et al. [13] investigated the problem of CPU prediction for VMs by applying bioinspired AI methods to optimize a Recurrent Neural Network (RNN). The Particle Swarm Optimization (PSO), the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) and the Differential Evolution (DE) were used as optimization algorithms for fine tuning the RNN model. The dataset was the PlanetLab. It includes CPU utilization samples measured every 5 minutes in a 24h day.

Bauer et al. [1] introduced UtilML, a resource prediction system based on LSTM for the computing continuum. The Rectified Linear Unit (ReLU) activation function was replaced with the LeakyReLU, an extension that is beneficial when there is a large amount of negative values. Batch normalization was proposed to mitigate internal covariate shifts. The dataset was the Alibaba Cloud GPU trace. The model was compared with baseline variants considering RMSE, MAPE and sMAPE.

Daraghmeh et al. [4] proposed a multilevel learning model for CPU usage prediction. The architecture involves three main phases: Anomaly Detection using an Isolation Forest to filter outliers, Data Clustering via k-means to identify recurrent CPU usage patterns, and Ensemble Learning which combines the best four regression models using stacking or voting regression methods. The input time window was fixed at 60 minutes with 5-minute samples. The Gradient Boosting Regressor demonstrated the highest performance across R^2, MAE, and MSE metrics.

Wang et al. [18] proposed an AIOps-oriented forecasting framework combining statistical, machine learning, and deep learning models. After outlier removal using the 4-sigma rule and signal smoothing with a Butterworth filter, predictions from methods such as SARIMAX, Prophet, Holt-Winters, XGBoost, and LSTM were integrated through a metamodel. Among all, XGBoost achieved the best accuracy across MAE, MSE, and MAPE metrics. I'm

Wang et al. [17] introduced ExtremoNet, a model for predicting extreme CPU load conditions using time series data that include CPU, memory, and network metrics. Outliers were detected with an isolation forest, and a one-step-ahead regression was performed. Features were selected via Pearson correlation, and the approach was tested on the AliCloud container trace dataset. Evaluations using MAE, MSE, and R^2 showed competitive results compared to three baseline models.

Carnevale et al. [2] discussed a distributed solution based on the federated learning paradigm. A bidirectional LSTM was developed and distributed over a number of local zone clients. The weights of the neural network were, therefore, aggregated on a regional server, using the Federated Average aggregation algorithm. The

methodology preserves data privacy and reduces latency by moving the computation to the edge of the network, where data are generated. Performance analysis, in terms of R^2, MSE, RMSE and MAE, prove the quality of the methodology.

The discussed works cover a broad range of knowledge about the prediction of CPU utilization. LSTM [15] was often used as a reference model to predict one step ahead (e.g., 5 minutes). Only the work [1] mentions the possibility of using such a model for the computing continuum, but the dataset was trained on a GPU trace dataset, which is typically, not the hardware architecture of edge devices. We aim to build a compute continuum model, training in the phases. During the first phase, the model is trained on the Microsoft Azure Trace dataset to build a pretrained based on VMs. During the second phase, the model is re-trained on the Alibaba Trace dataset to build a finetuning based on containers.

3 Material and Methods

3.1 Dataset

Considering the target of this paper is building a model for the Computing Continuum paradigm, that is, including cloud and edge resources, we selected two datasets, one for each kind of resources. Indeed, the cloud dataset is the Microsoft Azure Trace, while the edge dataset is the Alibaba Cloud Trace.

Microsoft Azure Trace Dataset: The Microsoft Azure trace dataset[1] was released in 2019 [3]. Dataset includes metrics from more than 2,5 million VMs. The uncompressed dataset has a size of 235 GiB divided into 198 files. VM CPU utilization was sampled every 5 minutes for 30 consecutive days. The total of samples recorded more than 100 million hours and almost 2 billion readings. Dataset schema contains many attributes divided into multiple files (e.g., deployment size, VM virtual core count bucket VM memory bucket). However, we focus on the attributes reported in the table 1. Services deployed in the VMs are first-parties (internal services, i.e., research, development, testing) and third-parties (costumers services, i.e., web service, e-commerce). Services are balanced between Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). Most of the VMs have a few virtual cores and no more than 4 GB.

metric	description
Timestamp	The time when the sample was collected
VM ID	The unique identifier of the virtual machine
Minimum CPU	The minimum CPU sampled at that time
Maximum CPU	The maximum CPU sampled at that time
Average CPU	The average CPU sampled at that time

Table 1: Features description of the Microsoft Azure Trace Dataset

Alibaba Cloud Trace Dataset: The Alibaba Cloud Trace dataset[2] was released in 2022 [20]. Dataset includes metrics from more than 40 thousand bare-metal nodes during 13 days, including more than 470 thousand containers for more than 28 thousand microservices. The uncompressed dataset has a size of more than 2TB divided into multiple files. The CPU utilization of the containers was sampled every minute for 13 consecutive days. Data schema contains many attributes divided into multiple files (e.g., bare-metal node resources, container resources, microservice call rate and response time). However, we focus on the attributes reported in the table 2. There is no information about the type of services deployed in the containers.

metric	description
Timestamp	The time when the sample was collected
Microservice ID	The unique identifier of the microservice
CPU Utilization	Normalized CPU sampled at that time

Table 2: Features description of the Alibaba Cloud Trace Dataset

3.2 Problem Definition

We consider a k-dimensional time-series with:

$$X_t = [x_t^i, x_t^{ii}, x_t^{iii}, ..., x_t^k] \quad (1)$$

In the proposed approach, focused on CPU utilization, it includes the CPU utilization value (x_t^i). The size k of the input is 1.

The model uses historical observations $(X_{t-n+1}, ..., X_t)$ to estimate CPU utilization X_{t+1} at one or more next points in time $t+1$, $t+2$ and $t+3$. Therefore,

$$X_{t+1}, X_{t+2}, X_{t+3} = f(X_{t-n+1}, ..., X_t) \quad (2)$$

where f is the relationship between the historical data and the predicted CPU utilization. It represents the models that we propose in the following. The time t represents the timestamp recorded when the sample is read. This can vary according to the need of the application. According to equation 1 and considering $k = 1$, the equation 2 becomes:

$$\left[x_{t+1}^i\right], \left[x_{t+2}^i\right], \left[x_{t+3}^i\right] = f(\left[x_{t-n+1}^i\right], ..., \left[x_t^i\right]) \quad (3)$$

The choice of predicting a few step in the future is two-fold. On the one hand, this is explained in [17]. Indeed, this avoids accumulation of errors in the long-term forecast if it is present. Our choice minimizes errors and guarantees the reliability of the forecasted results. On the other hand, considering a $t = 300s$, it is enough time to be informed about the future behavior of the infrastructure resource, plan and act the migration from the source to a target destination [10]. The choice is therefore confirmed by the literature.

3.3 Preprocessing

Both datasets contain cloud and edge CPU readings sampled, respectively, every 5 and 1 minutes from different VMs and containers. It is therefore required to extract the readings from one single VM or container to create coherent sequences of samples.

We downloaded all 195 files from the Microsoft Azure Trace and 47 files from the Alibaba Cloud Trace. In both cases, we selected the first file and randomly collected a set of 100 IDs. These are the candidates for analysis. At this point, the remaining files were opened to extract and add the readings of VMs and containers to

[1] https://github.com/Azure/AzurePublicDataset
[2] https://github.com/alibaba/clusterdata

Pandas DataFrames. The final dataset for a single VM has a size of approximately 1 MB and 8600 readings, while for a single container it has a size of approximately 110 KB and 1440 readings. The features are the same as explained in Tables 1 and 2.

An additional challenge was represented by the different sampling frequencies of the two datasets. Azure traces are natively sampled every 300 s, while Alibaba traces are sampled every 60 s. To align the two datasets and obtain comparable sequences, the Alibaba container readings were aggregated by computing the mean value over non-overlapping 300 s windows. This resampling step produced a time grid coherent with Azure, but inevitably reduced the number of available points by a factor of five, leading to information loss in the Alibaba traces.

The values of the curves before and after the preprocessing phase change frequently over time, with relevant spike–peak phenomena. This indicates that the system was experiencing sudden increases in resource demand and the running applications had irregular behavior. The dataset shows different trends, pointing out the variability of user behavior with the deployed services. The dataset now represents a good fit for testing the proposed approach, covering the case when the services stress the VMs.

Min-Max normalized data were used for both training and evaluation in order to assess the quality of the learning process. The dataset was partitioned into 70% for training, 15% for validation, and the remaining 15% for testing. The splitting procedure is defined as follows:

$$t = \lfloor r_{\text{train}} N \rfloor, \quad v = \lfloor r_{\text{val}} N \rfloor, \quad g = \max(1,\ s + p - 1),$$

$$\text{Train} = [\ 0,\ t\ ),$$

$$\text{Validation} = [\ \min(t+g,\ N),\ \min(t+g+v,\ N)\ ),$$

$$\text{Test} = [\ \min(t+g+v,\ N),\ N\ ).$$

where N is the total sequence length, r_{train} and r_{val} are the training and validation ratios, s is the input sequence length, p is the prediction length, g is the applied gap, $\lfloor x \rfloor$ denotes the floor operator, t and v are the computed training and validation lengths. A gap g was introduced in order to avoid information leakage from future windows that could bias the training process.

3.4 Predictive Model

To address the problem of multi-step forecasting of CPU utilization, we designed a Temporal Transformer architecture, illustrated in Figure 1. The model is specifically tailored to capture both short-term fluctuations and long-range dependencies in workload dynamics, such as variations in average CPU consumption.

The model input is a four-dimensional tensor:

$$\text{Input} \in \mathbb{R}^{B \times T \times N \times F} \tag{4}$$

where B denotes the batch size, T the length of the input sequence (e.g., the lookback window), N the number of VMs, and F the number of features, that is the average CPU utilization. In this work, we set $N = 100$. Each univariate CPU usage value is first projected into a $d_{\text{model}} = 192$-dimensional latent space through a linear transformation, producing richer internal representations:

$$h \in \mathbb{R}^{B \times T \times N \times d_{\text{model}}}$$

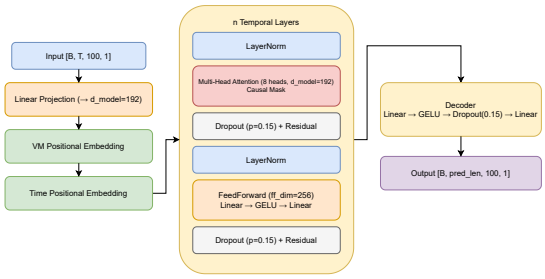


Figure 1: Architecture of the proposed temporal transformer for CPU utilization forecasting.

This projection is initialized with Xavier uniform initialization to stabilize training and facilitate convergence.

Two complementary embeddings are incorporated into the model: i) the **VM Embedding**, which assigns a unique vector to each VM, enabling the model to disambiguate between different machines and capture VM-specific behavior; and ii) the **Time Embedding**, which encodes the temporal position of each timestep, thereby compensating for the permutation invariance of the attention mechanism.

The core of the model consists of three stacked temporal transformer layers. Each layer contains:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{head}}}} + M\right)V$$

where M is a causal mask that ensures that predictions at time t cannot access future information. Each attention block employs eight heads, with $d_{\text{head}} = d_{\text{model}}/8 = 24$. The attention mechanism is followed by a position-wise feed-forward network with hidden dimension $d_{\text{ff}} = 256$ and a Gaussian Error Linear Unit (GELU) activation, which expands and contracts the feature space to capture complex non-linear patterns. Residual connections, dropout ($p = 0.15$), and layer normalization are applied throughout the network to stabilize optimization and reduce overfitting.

Finally, a decoder maps the transformer outputs back to the original feature space, yielding predictions with the following shape:

$$\text{Output} \in \mathbb{R}^{B \times \texttt{pred\_len} \times N \times 1} \tag{5}$$

which corresponds to the forecasted CPU utilization for each VM at each future timestep.

Overall, the model comprises approximately 8.0×10^5 trainable parameters, corresponding to a memory footprint of about 3.2 MB in single-precision representation. This balance between expressiveness and efficiency makes the architecture suitable not only for cloud applications but also for edge intelligence scenarios.

3.5 Training Phase

The training phase was conducted separately for each dataset. After splitting the timeseries into training, validation, and test sets with appropriate gaps, as previously discussed, the training set was normalized using the min–max scaler, fitted exclusively on the training data to avoid information leakage. The same transformation was then applied to the validation and test sets.

The model was trained on the Microsoft Azure dataset using the PyTorch framework with early stopping based on the validation loss to prevent overfitting. The optimization process relied on the Adam optimizer with a learning rate tuned to 0.001, and the loss function considered was the Mean Squared Error (MSE). Each training run involved multiple epochs until convergence, which often happened around 256 epochs, with the validation set continuously monitored to ensure generalization.

After training on the Microsoft Azure dataset, the model weights were saved and subsequently reused in the transfer learning phase, where fine-tuning was performed on the Alibaba dataset. This two-step process allowed the model to take advantage of the knowledge acquired from Azure while adapting to the characteristics of Alibaba.

4 Results and Discussion

4.1 Environmental Setup

The experiments were executed on a dedicated VM provisioned with 8 GB of RAM and powered by an AMD Ryzen 9 7950X processor with 16 cores, a high-performance CPU designed for parallel workloads. For accelerated training and inference, the system was equipped with an NVIDIA RTX 4090 Super GPU, featuring 24 GB of dedicated VRAM.

The software stack was centered on Python 3 as programming language, with the PyTorch deep learning framework serving as the backbone for model implementation. PyTorch v2.6.0 was chosen for its efficient memory management, flexibility for custom architecture design, and strong integration with the CUDA framework. Other modules used are Numpy v2.3.1 for the array management, Pandas v2.3.2 for the `.csv` files processing, Matplotlib v3.10.0 for data visualization and Scikit-learn v1.7.2 and Scipy v1.16.1 for metrics calculations, such as R^2, MSE, MAE, RMSE and confidence intervals in inference time calculations.

4.2 Metrics

The quality of the methodology is expressed in terms of several key metrics that evaluate accuracy, error magnitude, and operational suitability. R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables, providing an indication of the model's overall accuracy. Error magnitudes are quantified by the Mean Squared Error (MSE), which measures the average of squared errors, emphasizing the impact of larger prediction mistakes, and the Root Mean Squared Error (RMSE), which provides the standard deviation of the prediction errors in the same units as the target variable. The Mean Absolute Error (MAE) offers a complementary measure by calculating the average of the absolute errors, providing a linear assessment of the model's prediction accuracy. Finally, the operational characteristics essential for deployment in constrained environments are assessed using the Average inference time, which measures the time in seconds required for making a prediction, and the Model size, which quantifies the model's memory footprint in MB.

4.3 Results Discussion

The obtained results underline specific common trends across the different experimental scenarios as follows: i) Azure, ii) Alibaba and iii) Azure after Transfer Learning.

Experiments were setup with 6, 12 and 24 steps as lookback window size. The accuracy metrics in Table 3 show that for all the lookback window sizes considered, the R^2 score remains high (above 0.76), with few slight decreases as the forecasting horizon increases. This pattern is expected in multi-step forecasting, where uncertainty naturally grows with longer horizons. Performance on the Alibaba dataset is generally higher than on Azure, suggesting either more regular behavior in the data or better generalization of the model in that setting. Transfer learning from Azure to Alibaba yields results that are comparable, and in some cases slightly better, than direct training on Azure, showing the benefit of reusing learned representations.

The MSE, MAE, and RMSE scores indicate stable behavior in different scenarios. Performance does not deteriorate dramatically when increasing the lookback from 6 to 24 steps, which suggests that the architecture can effectively exploit longer input windows without clear signs of overfitting.

As reported in Table 5, inference times remain low and consistent. Using the Microsoft Azure dataset, predictions take about 0.6–0.8 seconds, with narrow confidence intervals, while using the Alibaba dataset they are much faster than 0.04 seconds, likely due to differences in dataset size or complexity. Transfer learning does not add any noticeable overhead, keeping inference times essentially unchanged. This is an important property for real-time or near real-time applications, where fast response is essential.

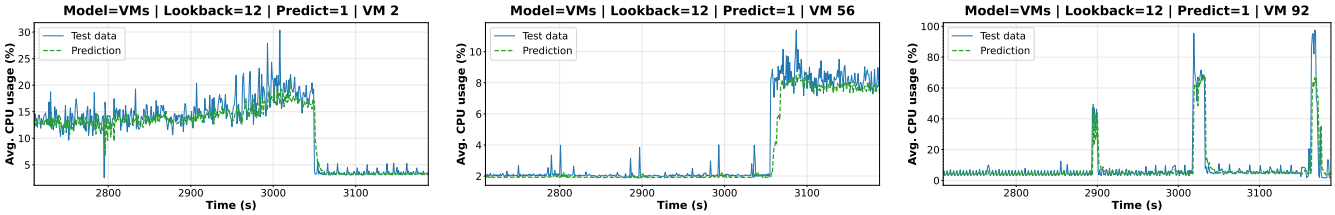
The size of the model remains constant at 3.2MB in all lookback windows and forecasting steps as shown in Table 4. This behavior is expected because the underlying architecture does not change with these settings. As a result, the observed differences in performance are entirely due to the data dynamics rather than to variations in the model complexity. The compact and fixed size of the model also makes it suitable for deployment in environments with limited memory resources, such as edge devices.

When it comes to qualitative analysis, referring to the Azure dataset, Figure 2 clearly shows how the predicted curve closely follows the testing values, often matching the shape of the actual timeseries. This qualitative evidence is consistent with the quantitative metrics previously discussed, which confirm that the transfer learning process improved the ability of the model to capture relevant patterns. Such improvements can also be observed in Figures 2a, 2b, and 2c, and are further highlighted in the corresponding transfer learning results reported in Figures 2d, 2e, and 2f.

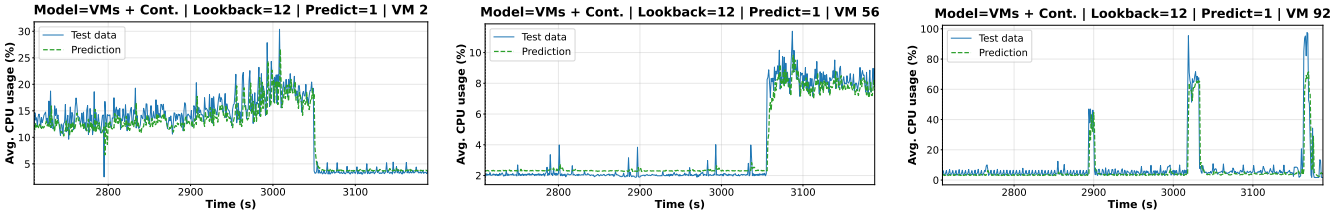
Regarding the predictions on the Alibaba dataset as shown in Figure 3, where the data has been denormalized, the test curves exhibit a good alignment with the actual values, showing only minimal deviations. This result highlights the ability of the Temporal Transformer, once fine-tuned through transfer learning, to generalize effectively across different environments. The predicted series not only follows the overall trend of the ground truth, but also captures local fluctuations and short-term variations, which are typically harder to reproduce. This behavior further confirms the robustness of the approach and supports the conclusions drawn

Table 3: Forecasting metrics by lookback window (Window) and steps ahead. The acronyms PT and FT respectively indicate the pre-trained and the fine-tuned model.

		R^2			MSE			MAE			RMSE		
Window	Steps	PT (Azure Test)	FT (Alibaba Test)	FT (Azure Test)	PT (Azure Test)	FT (Alibaba Test)	FT (Azure Test)	PT (Azure Test)	FT (Alibaba Test)	FT (Azure Test)	PT (Azure Test)	FT (Alibaba Test)	FT (Azure Test)
6	1	0.8425	0.8658	0.8525	0.0047	0.0063	0.0044	0.0304	0.0523	0.0354	0.0686	0.0794	0.0664
	2	0.8203	0.8405	0.8194	0.0054	0.0075	0.0054	0.0326	0.0589	0.0428	0.0733	0.0867	0.0735
	3	0.7644	0.8302	0.7980	0.0070	0.0080	0.0060	0.0363	0.0619	0.0416	0.0839	0.0894	0.0777
12	1	0.8468	0.8674	0.8504	0.0046	0.0062	0.0045	0.0292	0.0516	0.0367	0.0677	0.0788	0.0669
	2	0.8163	0.8404	0.8196	0.0055	0.0075	0.0056	0.0313	0.0590	0.0425	0.0741	0.0866	0.0749
	3	0.7826	0.8262	0.8034	0.0064	0.0081	0.0061	0.0342	0.0618	0.0432	0.0801	0.0898	0.0781
24	1	0.8489	0.8701	0.8513	0.0045	0.0061	0.0046	0.0288	0.0508	0.0375	0.0670	0.0781	0.0677
	2	0.8231	0.8428	0.8220	0.0053	0.0074	0.0055	0.0309	0.0579	0.0437	0.0729	0.0860	0.0745
	3	0.7915	0.8295	0.8078	0.0062	0.0080	0.0063	0.0337	0.0609	0.0446	0.0787	0.0894	0.0793



(a) Model=VMs (Pre-Trained) | Dataset=Azure | VM 2 (b) Model=VMs (Pre-Trained) | Dataset=Azure | VM 56 (c) Model=VMs (Pre-Trained) | Dataset=Azure | VM 92



(d) Model=VMs+Cont. (Fine-Tuned) | Dataset=Azure | VM 2 (e) Model=VMs+Cont. (Fine-Tuned) | Dataset=Azure | VM 56 (f) Model=VMs+Cont. (Fine-Tuned) | Dataset=Azure | VM 92

Figure 2: Comparison of prediction results on the Microsoft Azure Trace dataset. Subfigures 2a-2b-2c show model trained only on the Azure dataset, while Subfigures 2d-2e-2f show the corresponding model fine-tuned with transfer learning.

Table 4: Model size (MB) by lookback window and steps ahead.

Window	Steps	Pre-Trained (Azure)	Fine-Tuned (Alibaba)
6, 12, 24	1, 2, 3	3.2	3.2

from the quantitative metrics. Qualitative evidence can be clearly observed in Figures 3a, 3b, and 3c, where the predicted trajectories are nearly indistinguishable from the corresponding actual curves.

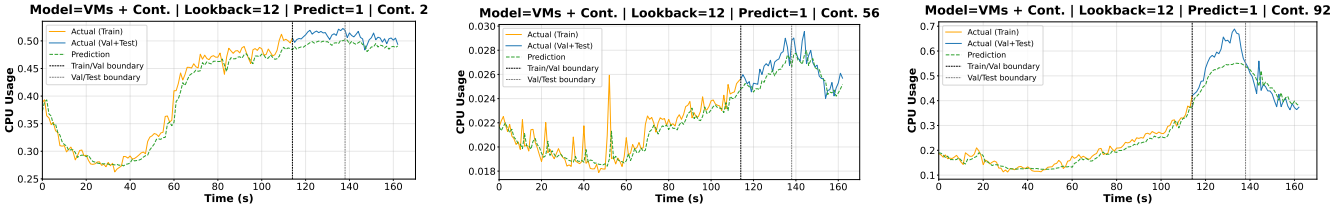
Table 6 compares the Temporal Transformer trained with transfer learning with other models available in the literature considered, in this work, as baselines. The first difference is that the other models are trained with one dataset. Unfortunately, this is often neither the Microsoft Azure Trace or the Alibaba Trace dataset. PlanetLab is a old dataset, while the custom dataset is not accessible. The proposed model is comparable in terms of MSE, RMSE and MAE when the lookback window size is 6 and 12 steps long. In the first case, all variants of the proposed model have better results than the baselines. The experiment with a window 12 steps long is comparable with the Gradient Boosting, but better results are obtained by the model that predicts 3 steps ahead.

5 Conclusion and Future Works

In this paper, we proposed a Temporal Transformer architecture to forecast CPU workloads in the computing continuum, with a focus on leveraging transfer learning from VMs to containers. By pretraining the model on the Microsoft Azure dataset and subsequently fine-tuning it on the Alibaba dataset, we demonstrated that knowledge can be effectively transferred across different virtualization environments. The model achieved consistently high R^2 values (above 0.80) with low MSE, MAE, and RMSE, while preserving a compact size of 3.2 MB and inference times suitable for real-time operation. These results highlight both the accuracy and efficiency of the approach, making it viable not only for cloud platforms but also for edge intelligence deployments. From a qualitative perspective, the predicted CPU utilization curves closely follow the actual values, capturing both global trends and local fluctuations. This

Table 5: Average inference time (seconds) with 95% confidence interval by lookback window (Window) and steps ahead.

Window	Steps	Pre-Trained (Azure Test)	Fine-Tuned (Alibaba Test)	Fine-Tuned (Azure Test)
6	1	0.5993 [0.5961–0.6025]	0.0323 [0.0322–0.0323]	0.5967 [0.5947–0.5988]
	2	0.6760 [0.6741–0.6780]	0.0379 [0.0378–0.0379]	0.6835 [0.6779–0.6891]
	3	0.7866 [0.7861–0.7870]	0.0428 [0.0427–0.0428]	0.7772 [0.7723–0.7822]
12	1	0.6175 [0.6170–0.6180]	0.0320 [0.0319–0.0320]	0.6187 [0.6149–0.6226]
	2	0.7065 [0.7002–0.7127]	0.0371 [0.0368–0.0373]	0.7121 [0.7088–0.7153]
	3	0.7871 [0.7813–0.7928]	0.0405 [0.0401–0.0408]	0.7963 [0.7957–0.7969]
24	1	0.6235 [0.6232–0.6238]	0.0317 [0.0317–0.0318]	0.6128 [0.6125–0.6131]
	2	0.7240 [0.7231–0.7250]	0.0344 [0.0344–0.0344]	0.7114 [0.7078–0.7149]
	3	0.7923 [0.7916–0.7930]	0.0383 [0.0383–0.0384]	0.7926 [0.7920–0.7931]



(a) Model=VMs+Cont. (Fine-Tuned) | Dataset=Alibaba | Cont. 2

(b) Model=VMs+Cont. (Fine-Tuned) | Dataset=Alibaba | Cont. 56

(c) Model=VMs+Cont. (Fine-Tuned) | Dataset=Alibaba | Cont. 92

Figure 3: Comparison of prediction results on the Alibaba dataset on Training-Validation-Test data. Subfigures 3a-3b-3c show the corresponding model trained with transfer learning.

confirms the ability of the Temporal Transformer to model workload dynamics and adapt to heterogeneous environments through transfer learning.

Future work will focus on extending the methodology to multivariate forecasting by incorporating additional metrics such as memory and network utilization. Another promising direction is the integration of federated and distributed learning strategies, enabling workload prediction directly at the edge while preserving data privacy. Furthermore, we envision the development of a predictive scheduling framework to manage the deployment of microservices, specifically containers and VMs, in order to assess efficiency in real-world scenarios. Finally, deploying and benchmarking the proposed approach in operational continuum infrastructures will provide further validation of its robustness and practical impact.

Acknowledgments

This work is partially supported by the Horizon Europe projects *"Open source deep learning platform dedicated to Embedded hardware and Europe"* (Grant Agreement No. 101112268 – **NEUROKIT2E**) and *"Trusted Extremely Precise Mapping and Prediction for Emergency Management"* (Grant Agreement No. 101093003 – **TEMA**). It also received support from the Italian Ministry of University and Research (MUR) under the *"Research projects of National Interest (PRIN-PNRR)"* program through the project *"Cloud Continuum aimed at On-Demand Services in Smart Sustainable Environments"* (CUP: J53D23015080001 – IC: P2022YNBHP) and the *"SEcurity and RIghts in the CyberSpace (SERICS)"* partnership (PE00000014), within the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU. In particular, the research has been supported within the SERICS partnership through the projects **FF4ALL** (CUP: D43C22003050001) and **SOP** (CUP: H73C22000890001). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] Christian Bauer, Narges Mehran, Radu Prodan, and Dragi Kimovski. 2023. Machine Learning Based Resource Utilization Prediction in the Computing Continuum. In *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, Edinburgh, United Kingdom, 219–224. doi:10.1109/CAMAD59638.2023.10478387

[2] Lorenzo Carnevale, Daniel Balouek, Serena Sebbio, Manish Parashar, and Massimo Villari. 2025. Private Distributed Resource Management Data: Predicting CPU Utilization with Bi-LSTM and Federated Learning. In *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 266–275. doi:10.1109/CCGRID64434.2025.00048

[3] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, Shanghai China, 153–167. doi:10.1145/3132747.3132772

[4] Mustafa Daraghmeh, Anjali Agarwal, and Yaser Jararweh. 2023. A Multilevel Learning Model for Predicting CPU Utilization in Cloud Data Centers. In *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, Abu Dhabi, United Arab Emirates, 1016–1023. doi:10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361305

[5] Martin Duggan, Karl Mason, Jim Duggan, Enda Howley, and Enda Barrett. 2017. Predicting host CPU utilization in cloud computing using recurrent neural networks. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, Cambridge, 67–72. doi:10.23919/ICITST.2017.8356348

model	metric	dataset	Window=6	Window=12	Window=24
PSE-RNN [13]	MSE	PlanetLab	0.0483	-	-
	MAE	PlanetLab	0.1564	-	-
DE-RNN [13]	MSE	PlanetLab	0.0465	-	-
	MAE	PlanetLab	0.1495	-	-
CMA-ES-RNN [13]	MSE	PlanetLab	0.0468	-	-
	MAE	PlanetLab	0.1498	-	-
BPTT [5]	MSE	PlanetLab	0.038	-	-
	MAE	PlanetLab	0.153	-	-
Temporal Transformer	R^2 (1 step)	Fine-Tuned (Azure Test)	0.8525	0.8504	0.8513
	MSE (1 step)	Fine-Tuned (Azure Test)	0.0044	0.0045	0.0046
	RMSE (1 step)	Fine-Tuned (Azure Test)	0.0664	0.0669	0.0677
	MAE (1 step)	Fine-Tuned (Azure Test)	0.0354	0.0367	0.0375
	R^2 (2 steps)	Fine-Tuned (Azure Test)	0.8194	0.8196	0.8220
	MSE (2 steps)	Fine-Tuned (Azure Test)	0.0054	0.0056	0.0055
	RMSE (2 steps)	Fine-Tuned (Azure Test)	0.0735	0.0749	0.0745
	MAE (2 steps)	Fine-Tuned (Azure Test)	0.0428	0.0425	0.0437
	R^2 (3 steps)	Fine-Tuned (Azure Test)	0.7980	0.8034	0.8078
	MSE (3 steps)	Fine-Tuned (Azure Test)	0.0060	0.0061	0.0063
	RMSE (3 steps)	Fine-Tuned (Azure Test)	0.0777	0.0781	0.0793
	MAE (3 steps)	Fine-Tuned (Azure Test)	0.0416	0.0432	0.0446
Gradient Boosting [4]	MSE	Custom	-	0.0058	-
	RMSE	Custom	-	0.0758	-
	MAE	Custom	-	0.0509	-

Table 6: Experimental results for the testset. Classical models are evaluated on 1-step forecasting with different input windows. The proposed Temporal Transformer is evaluated after transfer learning on the Alibaba Cloud Trace Dataset and tested on the Microsoft Azure Trace Dataset (Fine-Tuned (Azure Test)) on multiple prediction horizons (1, 2, and 3 steps ahead) with window sizes of 6, 12, and 24. All other models predict 1 step. All datasets are normalized with min-max scaling between 0 and 1.

[6] Schahram Dustdar, Victor Casamayor Pujol, and Praveen Kumar Donta. 2022. On distributed computing continuum systems. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2022), 4092–4105.

[7] Rebeca Estrada, Irving Valeriano, and Xavier Aizaga. 2023. CPU Usage Prediction Model: A Simplified VM Clustering Approach. In *Complex, Intelligent and Software Intensive Systems*, Leonard Barolli (Ed.). Vol. 176. Springer Nature Switzerland, Cham, 210–221. doi:10.1007/978-3-031-35734-3_21 Series Title: Lecture Notes on Data Engineering and Communications Technologies.

[8] Muhammad Fahimullah, Rohit Gupta, Shohreh Ahvar, and Maria Trocan. 2022. Explaining Predictive Scheduling in Cloud. In *Recent Challenges in Intelligent Information and Database Systems*, Edward Szczerbicki, Krystian Wojtkiewicz, Sinh Van Nguyen, Marcin Pietranik, and Marek Krótkiewicz (Eds.). Vol. 1716. Springer Nature Singapore, Singapore, 81–91. doi:10.1007/978-981-19-8234-7_7 Series Title: Communications in Computer and Information Science.

[9] Jiechao Gao, Haoyu Wang, and Haiying Shen. 2020. Machine learning based workload prediction in cloud computing. In *2020 29th international conference on computer communications and networks (ICCCN)*. IEEE, 1–9.

[10] Red Hat. 2024. How to create and scale 6,000 virtual machines in 7 hours with Red Hat OpenShift Virtualization. https://cloud.redhat.com/learning/learn:how-create-and-scale-6000-virtual-machines-7-hours-red-hat-openshift-virtualization/resource/resources:virtual-machine-migration-workflows.

[11] Deepak Janardhanan and Enda Barrett. 2017. CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, Cambridge, 55–60. doi:10.23919/ICITST.2017.8356346

[12] Mohammad Masdari and Afsane Khoshnevis. 2020. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing* 23, 4 (2020), 2399–2424.

[13] Karl Mason, Martin Duggan, Enda Barrett, Jim Duggan, and Enda Howley. 2018. Predicting host CPU utilization in the cloud using evolutionary neural networks. *Future Generation Computer Systems* 86 (Sept. 2018), 162–173. doi:10.1016/j.future.2018.03.040

[14] Seon-cheol Park and Young-han Kim. 2022. An Improvement of Multi-Cluster Stability of Private Cloud Systems through LSTM-Based CPU Usage Prediction. *The Journal of Korean Institute of Communications and Information Sciences* 47, 8 (Aug. 2022), 1081–1095. doi:10.7840/kics.2022.47.8.1081

[15] Eva Patel and Dharmender Singh Kushwaha. 2023. An Integrated Deep Learning Prediction Approach for Efficient Modelling of Host Load Patterns in Cloud Computing. *Journal of Grid Computing* 21, 1 (March 2023), 5. doi:10.1007/s10723-022-09639-6

[16] Khanh Nguyen Quoc, Van Tong, Cuong Dao, Tuyen Ngoc Le, and Duc Tran. 2024. Boosted regression for predicting CPU utilization in the cloud with periodicity. *The Journal of Supercomputing* (Aug. 2024). doi:10.1007/s11227-024-06451-9

[17] Chaoxue Wang and Zhenbang Wang. 2024. Isolated Forest-Based Prediction of Container Resource Load Extremes. *Applied Sciences* 14, 7 (March 2024), 2911. doi:10.3390/app14072911

[18] Zihao Wang, Huan Zhao, and Rong Dai. 2023. Research on Stacked Model for Cluster CPU Utilization Prediction Based on Butterworth Filter. In *2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. IEEE, Hangzhou, China, 191–197. doi:10.1109/ICBAIE59714.2023.10281293

[19] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2023. Transformers in Time Series: A Survey. arXiv:2202.07125 [cs.LG] https://arxiv.org/abs/2202.07125

[20] Xiantao Zhang, Xiao Zheng, Zhi Wang, Hang Yang, Yibin Shen, and Xin Long. 2020. High-density Multi-tenant Bare-metal Cloud. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 483–495. doi:10.1145/3373376.3378507