A Cross-Workload Power Prediction Method Based on Transfer Gaussian Process Regression in Cloud Data Centers

Ruichao Mo, Weiwei Lin, *Senior Member, IEEE*, Haocheng Zhong, Minxian Xu, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Nowadays, machine learning (ML)-based power prediction models for servers have shown remarkable performance, leveraging large volumes of labeled data for training. However, collecting extensive labeled power data from servers in cloud data centers incurs substantial costs. Additionally, varying resource demands across different workloads (e.g., CPU-intensive, memory-intensive, and I/O-intensive) lead to significant differences in power consumption behaviors, known as domain shift. Consequently, power data collected from one type of workload cannot effectively train power prediction models for other workloads, limiting the exploration of the collected power data. To tackle these challenges, we propose *TGCP*, a cross-workload power prediction method based on multi-source transfer Gaussian process regression. *TGCP* transfers knowledge from abundant power data across multiple source workloads to a target workload with limited power data. Furthermore, Continuous normalizing flows adjust the posterior prediction distribution of Gaussian process, making it locally non-Gaussian, enhancing *TGCP*'s ability to handle real-world power data distribution. This method enhances prediction accuracy for the target workload while reducing the expense of acquiring power data for real cloud data centers. Experimental results on a realistic power consumption dataset demonstrate that *TGCP* surpasses four traditional ML methods and three transfer learning methods in cross-workload power prediction.**

***Index Terms*—Cloud computing, large data center, server power prediction, transfer Gaussian process regression.**

Received 28 June 2024; revised 5 May 2025; accepted 28 May 2025. Date of publication 2 June 2025; date of current version 8 September 2025. This work was supported in part by Guangdong Provincial Natural Science Foundation under Grant 2025A1515010113, in part by the Innovative Development Joint Fund of Natural Science Foundation in Shandong Province under Grant ZR2024LZH012, in part by Guangxi Key Research and Development Project under Grant 2024AB02018, and in part by the Major Key Project of PCL, China under Grant PCL2023A09. Recommended for acceptance by J. Aghaei. *(Corresponding author: Weiwei Lin.)*

Ruichao Mo and Haocheng Zhong are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: cs_moruichao@mail.scut.edu.cn; cshczhong@mail.scut.edu.cn).

Weiwei Lin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with Pengcheng Laboratory, Shenzhen 518066, China (e-mail: linww@scut.edu.cn).

Minxian Xu is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: mx.xu@siat.ac.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TCC.2025.3575790, provided by the authors.

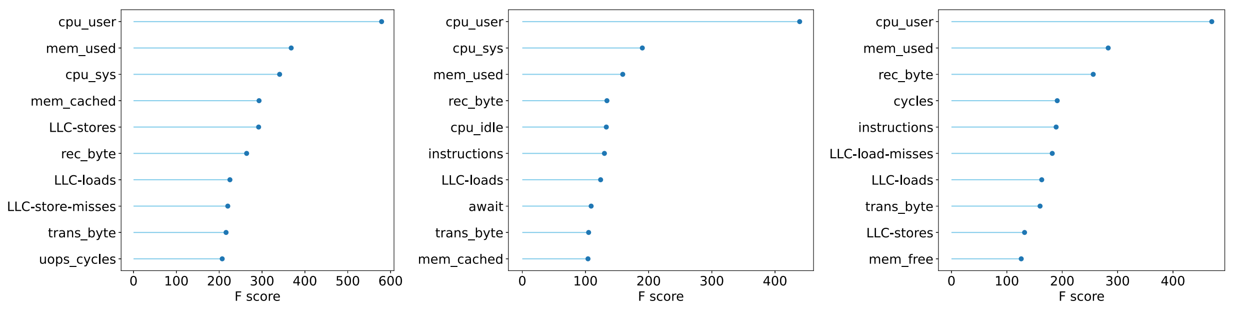
Digital Object Identifier 10.1109/TCC.2025.3575790

I. Introduction

As THE size and number of cloud data centers (CDCs) continue to expand, their energy consumption is increasing at an alarming rate. According to Data Centers Magazine, CDCs currently account for 3% of global energy consumption, a figure projected to rise to 4% by 2025 [1]. This high energy consumption not only leads to elevated operational costs but also has significant negative environmental impacts, making energy efficiency in CDCs a critical issue. Servers, the primary computing infrastructure within CDCs, exhibit a utilization rate between 10% and 20%, yet their energy consumption constitutes 42% of a CDC's total energy consumption [2], [3], [4]. This discrepancy indicates substantial potential for energy savings within cloud data centers, underscoring the importance of optimizing server resource utilization to achieve these savings. Consequently, recent research has focused on optimizing energy consumption through improved server resource utilization strategies, such as resource scheduling [5], [6], [7] and virtual machine consolidation [8], [9].

To pinpoint key areas for enhancing resource utilization and thereby optimize server energy consumption, power prediction models have received considerable attention [10], [11]. These models aim to establish a correlation between resource utilization and energy consumption. Currently, server power consumption prediction models primarily rely on various regression techniques [12]. Notably, several power prediction models have been developed by harnessing the powerful nonlinear fitting capabilities of machine learning, significantly improving prediction accuracy [13], [14]. However, these models typically depend on extensive amounts of independent and identically distributed power consumption data from servers running diverse workloads to achieve adequate training. This reliance presents two primary challenges for the current ML-based power prediction models when deployed on real servers:

1) Server power prediction with limited data from real workload runs: Traditional ML-based power prediction models rely on extensive power consumption data for training to achieve superior performance. However, in real CDCs, collecting power consumption data from actual workloads is often restricted by privacy constraints and logistical challenges, making the process both expensive and difficult. This limitation not only hampers the ability to gather sufficient data but also affects the diversity of



(a) CPU-intensive workload. (b) I/O-intensive workload. (c) Memory-intensive workload.

Fig. 1. Top 10 related features of different workloads for power consumption. These three sub-figures show the resource consumption features during the execution of each type of workload on the server. It can be observed that, while there are differences in the resource consumption features of different types of workloads, similar power consumption patterns also exist. *This observation motivates us to use transfer learning for cross-workload power prediction.*

the data collected, which is crucial for training robust models. As a result, achieving accurate server power prediction with only a limited amount of power consumption data remains a significant challenge. Moreover, the high costs associated with data collection further exacerbate this issue, increasing the need for more effective exploitation of the already collected power consumption data.

2) Cross-workload power prediction: The differences in the requirements for server hardware resources while running different types of workloads lead to variations in the power consumption behavior of the servers, known as domain shift, as shown in Fig. 1. However, traditional ML models, which are usually trained on specific types of workloads, may struggle to predict power consumption accurately when applied to new, unseen workloads, making cross-workload power prediction a significant challenge. For example, the models trained on the power data from CPU-intensive workloads and subsequently failed to make effective prediction when the models are being applied to memory-intensive or I/O-intensive workloads, which greatly limited the exploitation of the useful knowledge embedded in collected power consumption data.

To address these challenges, we propose a cross-workload power prediction model based on multi-source transfer GPR, which is dedicated to transferring useful knowledge from the substantial quantity of power data possessed by the source workloads to improve the performance of power prediction for the target workload while reducing the cost of data collection required for server power prediction. The primary contributions are stated as follows:

- We systematically analyze the power consumption features and behavior of servers running different types of workloads. This analysis reveals why existing ML-based power prediction models fail to achieve cross-workload power prediction and underscores the necessity of adopting transfer learning for this purpose.
- The multi-source transfer GPR method is employed to build a cross-workload power prediction model, called **TGCP**, allowing the exploitation of the usable knowledge from the substantial amount of power data held by multiple source workloads to enhance the power prediction performance for the target workload.
- Continuous Normalizing Flows (CNF) is adopted to adjust the posterior prediction distribution of the Gaussian process so that the transfer GPR posterior is locally non-Gaussian, further improving the performance of *TGCP* to cope with real power data.
- Extensive experiments are conducted on realistic power consumption datasets, and the results indicate that **TGCP** provides better performance of cross-workload power prediction than four traditional ML methods and three methods based on transfer learning.

The remaining parts of this paper are organized as follows: Section II provides the motivation for cross-workload power prediction. Section III details the proposed method. Section IV elaborates on the experimental results. Section V discusses the related work. Finally, Section VI presents the conclusion and future work.

II. Motivation

A. The Performance of Traditional ML-Based Power Prediction Models Trained With Limited Power Data

Most current power prediction models that rely on regression are trained on substantial quantities of labeled data to achieve superior performance in predicting power consumption. When there is insufficient power data available for specific types of workloads (e.g., memory-intensive and I/O-intensive workloads), it becomes challenging for power prediction models to be adequately trained, thus limiting their effectiveness. Table I presents the implementation of four widely used power prediction models: Linear Regression (LR), Multi-Layer Perceptron (MLP), Back Propagation Neural Network (BPNN), and Gaussian Process Regression (GPR). These models are trained on two workloads, specifically Cache (memory-intensive) and Random (I/O-intensive), each with a limited dataset of only 50 power data samples (**Target only**). The results demonstrate that the power prediction performance, when trained on such a limited amount of power data, remains unsatisfactory, indicating overfitting due to insufficient training data (See Section IV for experimental setup details).

How can we improve power prediction performance for servers with limited data from real workload runs? A feasible

TABLE I
THE PERFORMANCE OF TRADITIONAL ML METHODS TRAINED WITH LIMITED POWER DATA AND CROSS-WORKLOAD POWER PREDICTION USING THREE SOURCE WORKLOADS

Target workload	Method	Train dataset	RMSE (Watts) ↓	MAE (Watts) ↓	MAPE ↓
Cache	LR	Target only	–	–	–
		Target + 3 Source	133.84123	11.53303	0.48505
	MLP	Target only	28.09777	2.79282	0.11787
		Target + 3 Source	133.66093	11.52755	0.48486
	BPNN	Target only	31.25223	8.87250	0.01575
		Target + 3 Source	35.76102	23.75477	0.04207
	GPR	Target only	101.90660	43.80588	0.07752
		Target + 3 Source	215.53939	11.37090	0.73849
Random	LR	Target only	79.47137	3.47637	0.19260
		Target + 3 Source	100.64924	9.97255	0.55336
	MLP	Target only	22.09550	2.38786	0.13231
		Target + 3 Source	100.22408	9.96394	0.55295
	BPNN	Target only	29.73412	8.50425	0.02613
		Target + 3 Source	69.54733	33.44436	0.10280
	GPR	Target only	70.69727	25.53355	0.07839
		Target + 3 Source	210.15166	13.31667	0.73849

'↓' represents the lower the value the better.

approach is to expand the training dataset of power data for the specific workload, but this incurs high data acquisition costs in real CDCs. In fact, as shown in Fig. 1, despite the differences in hardware resource requirements for different types of workloads, there are also similarities in the power consumption patterns of servers during these operations. These similarities suggest that there is potential for transferring knowledge between different types of workloads to improve power prediction performance. Therefore, we propose a straightforward intuition: leveraging the large amounts of power data collected from servers running other types of workloads (referred to as source workloads) to improve power prediction for workloads with limited power data (referred to as target workloads), which we define as cross-workload power prediction for servers in this paper.

B. The Performance of Traditional ML for Cross-Workload Power Prediction

To validate our intuition, we construct cross-workload power prediction tasks by using Cache and Random as the target workloads and introducing three CPU-intensive workloads as source workloads (**Target + 3 Source**). Then, a large amount of power consumption data from multiple source workloads and a small amount of power consumption data from the target workloads are merged to train and evaluate four power consumption prediction models developed using traditional ML. The experimental results are also shown in Table I.

Compared to the performance of power prediction models trained solely on limited power data of the target workload, traditional ML methods fail to leverage the useful knowledge from the power data of source workloads to enhance the prediction performance for the target workload. This results in negative transfer, further deteriorating the power prediction performance for the target workload. Clearly, such adverse outcomes are undesirable. In the following subsection, we will analyze the power performance of servers running different types of workloads to understand why traditional ML methods face significant challenges in cross-workload power prediction.

C. Heterogeneous Power Performance Results From Differences in the Types of Workloads Running on the Servers

We analyze the features of resource requirements and the distributions of power data for different types of workloads. This analysis elucidates why traditional ML methods perform poorly in cross-workload power prediction and explains why transfer learning is leveraged to achieve cross-workload power prediction.

First, we employ XGBoost [15] to highlight the importance of various features for server power consumption. Fig. 1 displays the 10 most relevant features for power consumption when the server executes CPU-intensive, I/O-intensive, and memory-intensive workloads. Significant differences in resource requirements are evident when executing different types of workloads, primarily reflected in the varying impacts of different server resources on power consumption. Despite these differences, we also observe similarities in server resource requirements across different workloads, which we refer to as *domain shift*.

Traditional ML methods achieve good prediction performance only when both the training and test sets are from the same distribution. Consequently, they fail to deliver satisfactory performance in cross-workload power prediction. This is why we employ multi-source transfer GPR, a transfer learning method, to achieve cross-workload power prediction in this paper.

Additionally, we analyze the probability density functions (PDF) of power consumption for various workloads running on the server, a method commonly used to profile the intensity of continuous random variables [16]. The power consumption probability distributions for different workloads, fitted with Gaussian kernel density estimation (GKDE), are shown in Fig. 2. The noticeable differences in power distributions among workloads explain the poor performance of traditional ML methods in cross-workload power prediction.

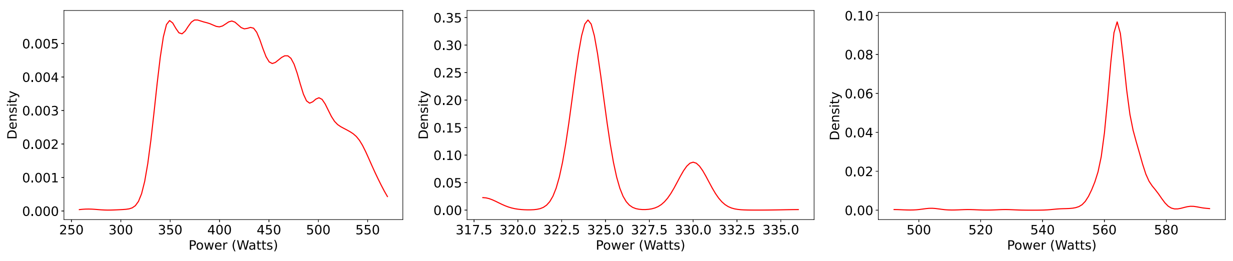
Furthermore, Fig. 2 indicates that the power consumption distribution for different workloads on real servers does not follow a normal distribution. However, multi-source transfer GPR, which relies on Gaussian processes, assumes a normal distribution to facilitate the closed-form computation of the posterior probability function. This assumption limits the flexibility of multi-source transfer GPR in capturing the complex distributions of real-world data. Therefore, we adjust the Gaussian process posterior prediction distribution using a conditional normalizing flow (CNF) to make it locally non-Gaussian, thereby improving its ability to handle real power data distributions.

III. **TGCP** : MULTI-SOURCE TRANSFER GPR FOR CROSS-WORKLOAD POWER PREDICTION

A. Preliminary Knowledge

In this subsection, we provide some preliminary knowledge to facilitate the understanding of our proposed method, **TGCP**.

1) Gaussian Process Regression: Gaussian process regression (GPR) is a regression method that is predicated upon the Gaussian process (GP), a stochastic process that entails a random variables set. The multivariate Gaussian distribution characterizes the joint distribution of any finite subset of these



(a) PDF of CPU-intensive workload. (b) PDF of I/O-intensive workload. (c) PDF of Memory-intensive workload.

Fig. 2. PDFs of different workloads running in the server.

random variables.

$$p(\mathcal{G}|\mathbf{x}) = \mathcal{N}(\mathcal{G}|\mu, \kappa), \tag{1}$$

where $\mathcal{G} = \{g(x_1), g(x_2), \ldots, g(x_N)\}$ denotes the mean function of the GP, which is often initialized to 0. $\kappa(x, x')$ represents the kernel function employed to evaluate the similarity between x and x'. An effective kernel function has to be positive semi-definite (PSD) according to Mercer's theorem. The mean and kernel functions can determine a GP, so that for any process $g()$, which follows the GP denoted by

$$g(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x')). \tag{2}$$

As a nonparametric Bayesian inference method, GP defines a prior for $g()$, which in turn allows direct inference of the distribution, rather than inferring the parametric distribution. Consider a regression problem is denoted as

$$\mathbf{y}_i = g(\mathbf{x}_i) + \gamma_i, i \in \{1, 2, 3, \ldots, N\}, \tag{3}$$

where $\gamma_i \sim \mathcal{N}(0, \sigma_i^2 I_i)$ is the noise variable. Suppose that $g()$ with training data $\{\mathbf{x}, \mathbf{y}\}$ and testing data $\{\mathbf{x}_*, \mathbf{y}_*\}$, and thus the joint distribution of $\mathbf{y}$ and $\mathbf{y}_*$ is expressed as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \kappa(\mathbf{x},\mathbf{x}) + \sigma_n^2 \mathbf{I} & \kappa(\mathbf{x},\mathbf{x}_*) \\ \kappa(\mathbf{x}_*, \mathbf{x}) & \kappa(\mathbf{x}_*,\mathbf{x}_*) \end{bmatrix}\right), \tag{4}$$

where $\mathbf{I}$ represents the identity matrix. Furthermore, the predictive distribution of the $g()$ is denoted as follow

$$p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{x}, \mathbf{y}) \sim \mathcal{N}(\mu_*, \kappa_*), \tag{5}$$

where

$$\mu_* = \kappa(\mathbf{x}_*, \mathbf{x})(\kappa(\mathbf{x}, \mathbf{x}) + \sigma^2 I)^{-1}\mathbf{y}, \tag{6}$$

$$\kappa_* = \kappa(\mathbf{x}_*, \mathbf{x}_*) + \sigma^2 \mathbf{I} - \kappa(x_*, x)(\kappa(\mathbf{x}, \mathbf{x}) + \sigma^2 \mathbf{I})^{-1}\kappa(\mathbf{x}, \mathbf{x}_*). \tag{7}$$

2) Multi-Source Transfer GPR: The multi-source transfer GPR has gained significant attention in recent years as an important yet relatively rare branch of transfer learning for regression problems. This method greatly enhances prediction performance by concurrently capturing useful knowledge from several different source domains, thereby effectively dealing with unseen data from the target domain. Generally, the multi-source transfer GPR establishes a unified Gaussian distribution across a set of N source domains $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ and a target domain $\mathcal{T}$. The effective transfer kernel is then formulated to implement relevance evaluation among the source domains and also between $\mathcal{S}$ and $\mathcal{T}$, thereby realizing that usable knowledge is learned from the source domains to augment the predictive performance of the model for unseen data from $\mathcal{T}$.

In [17], Wei et al. provided the definition of the transfer kernel systematically with the basic principle that it can explicitly capture the relevance of the domain, allowing the transfer GPR to adaptively regulate the transfer intensity of knowledge. This distinguishes the transfer kernel from the standard GPR kernel, which handles all instance pairs equally. In other words, the lesser the difference between the source and target domains, the greater its transferability of knowledge. Conversely, the greater the difference between the source and target domains, the lower the transferability of knowledge. In [18], Wei et al. developed a multi-source transfer kernel $k_{ms}()$ and provide detailed theoretical proof that this kernel is PSD, shown in the following equation

$$\kappa_{ms}(x, x') = \begin{cases} \lambda_{(\mathcal{D}_i, \mathcal{D}_j)}\kappa_{(\mathcal{D}_i, \mathcal{D}_j)}(x, x'), x \in \mathcal{D}_i \& x' \in \mathcal{D}_j, \\ \qquad \mathcal{D}_i, \mathcal{D}_j, \in \mathcal{D}, i \neq j, \\ \kappa_{(\mathcal{D}_i, \mathcal{D}_j)}(x, x'), x \& x' \in \mathcal{D}_i, \mathcal{D}_i \in \mathcal{D}. \end{cases} \tag{8}$$

where $\mathcal{D}$ represents a domain set that contain $\mathcal{S}$ and $\mathcal{T}$. And $\lambda_{(\mathcal{D}_i, \mathcal{D}_j)} \in [-1, 1]$ denotes a learnable parameter to regulate the degree for knowledge transfer.

3) Continuous Normalizing Flows: Normalizing flows (NF) transform an arbitrary data distribution into a specific distribution, such as a Gaussian distribution, by constructing a sequence of reversible transformations $\mathbf{y} = \mathbf{f}_N \circ \cdots \circ \mathbf{f}_1(\mathbf{z})$. NF is gaining popularity in generative models due to their flexibility and the convenience of being able to optimize training directly using negative log-likelihood (NLL). The potential variable $\mathbf{z}$ with a specifically known prior $p(\mathbf{z})$ can be mapped by the change of variable theorem to $\mathbf{y}$ in some observation space with an unknown distribution, which is defined as

$$p(\mathbf{y}) = p(\mathbf{z})\left|\det \frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right| = p(\mathbf{z})\left|\det \frac{\partial \mathbf{f}(\mathbf{z})}{\partial \mathbf{y}}\right|, \tag{9}$$

where $\mathbf{z} = \mathbf{f}_1^{-1} \circ \cdots \circ \mathbf{f}_N^{-1}(\mathbf{y})$ is a result of the invertible mapping. Furthermore, the likelihood function for $\mathbf{y}$ is denoted as

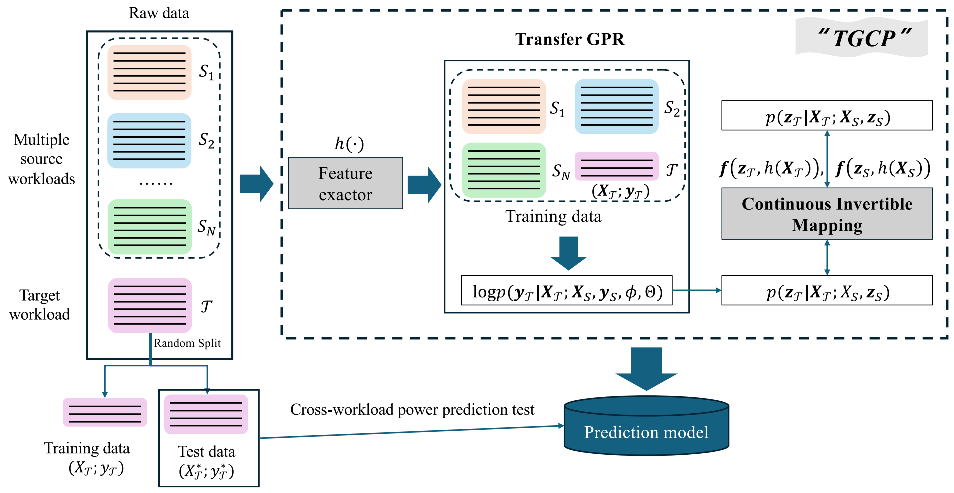

Raw data
Multiple source workloads
S_1
S_2
……
S_N
Target workload
$\mathcal{T}$
Random Split
Training data $(X_\mathcal{T}; y_\mathcal{T})$
Test data $(X_\mathcal{T}^*; y_\mathcal{T}^*)$
$h(\cdot)$
Feature exactor
Transfer GPR
$(X_\mathcal{T}; y_\mathcal{T})$
Training data
$\log p(y_\mathcal{T}|X_\mathcal{T}; X_S, y_S, \phi, \Theta)$
"TGCP"
$p(z_\mathcal{T}|X_\mathcal{T}; X_S, z_S)$
$f(z_\mathcal{T}, h(X_\mathcal{T}))$, $f(z_S, h(X_S))$
Continuous Invertible Mapping
Cross-workload power prediction test
Prediction model

Fig. 3. The workflow of **TGCP**.

$$\log p(\mathbf{y}) = \log p(\mathbf{z}) - \sum_{i=1}^{N} \log \left| \det \frac{\partial \mathbf{f}_i}{\partial \mathbf{z}_{i-1}} \right|. \tag{10}$$

Finding such an invertible transformation $\mathbf{y} = \mathbf{f}_N \circ \cdots \circ \mathbf{f}_1(\mathbf{z})$ that allows the Jacobian matrix $\det \frac{\partial \mathbf{f}(\mathbf{z})}{\partial \mathbf{y}}$ to be computed efficiently while maintaining the expressiveness of the transformation poses a major challenge. CNF [19] can effectively address this challenge by replacing a series of discrete functions $\mathbf{y} = \mathbf{f}_N \circ \cdots \circ \mathbf{f}_1(\mathbf{z})$ with continuous and time-independent transformations. Furthermore, a function $u_\beta(z(t), t) = \frac{\partial z(t)}{\partial t}$ with a parameter β is leverage to model the dynamics of $z(t)$. And assuming a known a priori initial state $\mathbf{z} := z(t_0)$, the difference equation is used to find a strategy $\mathbf{y} := z(t_1)$. Thus, the transformation function is expressed as

$$\mathbf{y} = \mathbf{f}_\beta(\mathbf{z}) = \mathbf{z} + \int_{t_0}^{t_1} u_\beta(z(t), t) dt. \tag{11}$$

Consequently, the inverse transformation is denoted as

$$\mathbf{f}_\beta^{-1}(\mathbf{y}) = \mathbf{y} - \int_{t_0}^{t_1} u_\beta(z(t), t) dt. \tag{12}$$

Then, the log-likelihood function of $\mathbf{y}$ is rewritten as

$$\log p(\mathbf{y}) = \log p(\mathbf{z}) - \int_{t_0}^{t_1} Tr \left(\frac{\partial u_\beta(t)}{\partial z(t)} \right) dt, \tag{13}$$

where $f_\beta^{-1}(\mathbf{y}) = \mathbf{z}$.

B. Overview of TGCP

To achieve cross-workload power prediction, we build **TGCP** based on a multi-source transfer GPR in this paper. To acquire a power prediction model that performs excellently on unseen power data from the target workload, **TGCP** employs a substantial quantity of power consumption data from N source workloads, which we denote as $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$, and a smaller amount of data from the target workload, which we denote as $\mathcal{T}$.

The workflow of **TGCP** is shown in Fig. 3. First, the source workload power features $\mathbf{X}_\mathcal{S}$ and target workload power features $\mathbf{X}_\mathcal{T}$ are passed through the feature extractor $h()$ to create their respective latent embeddings. These embeddings are then modeled using transfer GPR to capture the distribution of the latent variables $\mathbf{z}$. Then, **TGCP** applies an ordinary differential equation (ODE)-based invertible transformation, denoted as $\mathbf{f}()$, to the latent variables $\mathbf{z}$ sampled from the posterior distribution of transfer GPR. This transformation is carried out through a series of invertible steps, each step of the transformation maps the components of the latent variables while being conditioned on the output of $h()$. Furthermore, the invertibility of the CNF guarantees that this transformation can be reversed, ensuring that the exact likelihood of the real power data can be computed. This is achieved using the change of variables theorem, shown in (9), in combination with the Jacobian of the transformation, enabling efficient density estimation as formulated in (13). By solving the ODE, the CNF generates a transformed distribution $z(t)$ at time t that approximates the complex distributions of the real power data. As a result, **TGCP** can adjust the posterior output $\mathbf{z}$, enabling it to model the complex distributions that real power data $\mathbf{y} = [\mathbf{y}_\mathcal{S}; \mathbf{y}_\mathcal{T}]$ belong to.

C. Learning of the TGCP

In this section, we introduce the training procedure of **TGCP**, in which the main objective is to exploit the knowledge available from $\mathcal{S}$ to improve the prediction performance of the model coping with unseen power data of $\mathcal{T}$. As a transfer GPR, **TGCP** is consistent with the common GPR training optimization objective, which is to optimize the conditional probability distribution $p(\mathbf{y}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S})$. For subsequent adjustment of the power data distribution, we consider the feature extractor $h()$ with parameter ϕ for feature embedding from the multi-source workloads feature $\mathbf{X}_\mathcal{S}$ and the target workload feature $\mathbf{X}_\mathcal{T}$. Furthermore, for the purpose of efficient knowledge transfer, we embrace the transfer kernel $k_{ms}()$ proposed in [18] for implementation of **TGCP**, whose input is the output of the extractor $h()$. Then, the transfer kernel of **TGCP** is represented as $k_{ms}(h_\phi(\cdot), h_\phi(\cdot))$. The Gram matrix $\mathcal{K}$ obtained from the observation of $k_{ms}()$ on $\mathbf{X}_\mathcal{S}$ and $\mathbf{X}_\mathcal{T}$ can be expressed as

$$\mathcal{K} = \begin{bmatrix} \mathbf{K}_{S_1,S_1} & \cdots & \lambda_{(S_1,\mathcal{T})}\mathbf{K}_{S_1,\mathcal{T}} \\ \lambda_{(S_2,S_1)}\mathbf{K}_{S_2,S_1} & \cdots & \lambda_{(S_2,\mathcal{T})}\mathbf{K}_{S_2,\mathcal{T}} \\ \cdots & \cdots & \cdots \\ \lambda_{(\mathcal{T},S_1)}\mathbf{K}_{S_\mathcal{T},S_1} & \cdots & \mathbf{K}_{\mathcal{T},\mathcal{T}} \end{bmatrix}. \tag{14}$$

For easy comprehension, the Gram matrix $\mathcal{K}$ is denoted as the block form

$$\mathcal{K} = \begin{bmatrix} \hat{\mathcal{K}}_{\mathcal{S},\mathcal{S}} & \hat{\mathcal{K}}_{\mathcal{T},\mathcal{S}} \\ \hat{\mathcal{K}}_{\mathcal{S},\mathcal{T}} & \hat{\mathcal{K}}_{\mathcal{T},\mathcal{T}} \end{bmatrix}, \tag{15}$$

where $\hat{\mathcal{K}}_{\mathcal{S},\mathcal{S}}$ and $\hat{\mathcal{K}}_{\mathcal{T},\mathcal{T}}$ denote the block matrix for the source workloads and target workload, respectively. And $\hat{\mathcal{K}}_{\mathcal{S},\mathcal{T}} = \hat{\mathcal{K}}_{\mathcal{T},\mathcal{S}}$ is the block matrix that combines the source workloads and the target workload. Thus, $p(\mathbf{y}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S})$ can be expressed as

$$p(\mathbf{y}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S}) = \mathcal{N}(\mu^*, \mathcal{K}^*), \tag{16}$$

where

$$\mu^* = \hat{\mathcal{K}}_{\mathcal{T},\mathcal{S}}\left(\hat{\mathcal{K}}_{\mathcal{S},\mathcal{S}} + \sigma^2_{\mathcal{S},\mathcal{S}}\mathbf{I}_{\mathcal{S},\mathcal{S}}\right)^{-1}\mathbf{y}_\mathcal{S}, \tag{17}$$

$$\mathcal{K}^* = \left(\hat{\mathcal{K}}_{\mathcal{T},\mathcal{T}} + \sigma^2_{\mathcal{T},\mathcal{T}}\mathbf{I}_{\mathcal{T},\mathcal{T}}\right) - \hat{\mathcal{K}}_{\mathcal{T},\mathcal{S}}\left(\hat{\mathcal{K}}_{\mathcal{S},\mathcal{S}} + \sigma^2_{\mathcal{S},\mathcal{S}}\mathbf{I}_{\mathcal{S},\mathcal{S}}\right)^{-1}\hat{\mathcal{K}}_{\mathcal{S},\mathcal{T}}. \tag{18}$$

Moreover, the log-likelihood of **TGCP** can be represented by

$$\begin{aligned} &\log p(\mathbf{y}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S}; \phi, \Theta) \\ &= -\frac{1}{2}\log|\mathcal{K}_*| - \frac{1}{2}(\mathbf{y}_\mathcal{T} - \mu_*)^T\mathcal{K}_*^{-1}(\mathbf{y}_\mathcal{T} - \mu_*) - \frac{N_\mathcal{T}}{2}\log(2\pi), \end{aligned} \tag{19}$$

where ϕ denotes the parameters of the feature extractor $h()$, Θ denotes all learnable parameters in the transfer GPR, and $N_\mathcal{T}$ represents the amount of power consumption data involved in training for the target workload.

To enable the adjustment of the power data distribution using the CNF, the log-likelihood of the *TGCP* needs to be reformulated in combination with (13), reformulated as

$$\begin{aligned} &\log p(\mathbf{z}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{z}_\mathcal{S}; \phi, \Theta, \beta) \\ &= \log p(\mathbf{z}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{z}_\mathcal{S}; \phi, \Theta) - \int_{t_0}^{t_1} Tr\left(\frac{\partial\mu_\beta}{\partial z(t)}\right)dt, \end{aligned} \tag{20}$$

where $\mathbf{f}^{-1}(\mathbf{y}_\mathcal{S}) = \mathbf{z}_\mathcal{S}$ and $\mathbf{f}^{-1}(\mathbf{y}_\mathcal{T}) = \mathbf{z}_\mathcal{T}$. The flow transformation is applied independently to the marginal elements of the power values $\mathbf{y} = [\mathbf{y}_\mathcal{S}; \mathbf{y}_\mathcal{T}]$ used for training from the multi-source workloads $\mathbf{y}_\mathcal{S}$ and target workloads $\mathbf{y}_\mathcal{T}$, that is, $\mathbf{f}^{-1}(\mathbf{y}) = [\mathbf{f}^{-1}(\mathbf{y}_1), \mathbf{f}^{-1}(\mathbf{y}_2), \ldots, \mathbf{f}^{-1}(\mathbf{y}_d), \ldots, \mathbf{f}^{-1}(\mathbf{y}_D)]^T$, where D represents the aggregate the amount of training data coming from both $\mathbf{X}_\mathcal{S}$ and $\mathbf{X}_\mathcal{T}$. And the parameters of $\mathbf{f}^{-1}()$ are shared over all $\mathbf{y}$.

Meanwhile, $\mathbf{f}^{-1}()$ is conditional on the information encoded by the $h()$, which enables the context information $h(\mathbf{x}_d)$ related to the corresponding input value $\mathbf{x}_d$ to be considered, where $\mathbf{x}_d \in [\mathbf{X}_\mathcal{S}; \mathbf{X}_\mathcal{T}]$.

$$\mathbf{y}_d = \mathbf{f}_\beta(\mathbf{z}_d, h_\phi(\mathbf{x}_d)) = \mathbf{z}_d + \int_{t_0}^{t_1} \mu_\beta(z_d(t), t, h_\phi(\mathbf{x}_d))dt. \tag{21}$$

The inverse transformation can be easily calculated as follow

$$\mathbf{f}_\beta^{-1}(\mathbf{y}_d) = \mathbf{y}_d - \int_{t_0}^{t_1} \mu(z_d(t), t, h_\phi(\mathbf{x}_d))dt. \tag{22}$$

Finally, the log-likelihood of **TGCP** can be rewritten by

$$\begin{aligned} &\log p(\mathbf{y}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S}; \phi, \Theta, \beta) \\ &= \log p(\mathbf{z}_\mathcal{T}|\mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{z}_\mathcal{S}; \phi, \Theta) - \sum_{d=1}^{D}\int_{t_0}^{t_1} Tr\left(\frac{\partial\mu_\beta}{\partial z_d(t)}\right)dt. \end{aligned} \tag{23}$$

To efficiently implement the above transformations, we employ in this paper the CNF model implemented by Ffjord [19], which performs better on low-dimensional data. Furthermore, since the CNF is applied independently to each component of the output of **TGCP** and shared among them, thus, we do not have any problem with the computation of the Jacobian matrix, which corresponds to the first-order derivative of the output [20].

D. Inference of the *TGCP*

The inference process of *TGCP* incorporating CNF is also similar to the common GPR, as we presented in Section II. For unseen data $\{\mathbf{X}^*_\mathcal{T}, \mathbf{y}^*_\mathcal{T}\}$ of the target workload, the posterior prediction distribution can be expressed as $p(\mathbf{y}^*_\mathcal{T}|\mathbf{X}^*_\mathcal{T}, \mathbf{X}_\mathcal{T}, \mathbf{y}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S}; \phi, \Theta, \beta)$. Since the transformation given by Eq. (21) is independent of the output $\mathbf{y}$. Therefore, we are still able to estimate the posterior in closed form, denoted as

$$\begin{aligned} &\log p(\mathbf{y}^*_\mathcal{T}|\mathbf{X}^*_\mathcal{T}, \mathbf{X}_\mathcal{T}, \mathbf{y}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{y}_\mathcal{S}; \phi, \Theta, \beta) \\ &= \log p(\mathbf{z}^*_\mathcal{T}|\mathbf{X}^*_\mathcal{T}, \mathbf{X}_\mathcal{T}, \mathbf{z}_\mathcal{T}, \mathbf{X}_\mathcal{S}, \mathbf{z}_\mathcal{S}; \phi, \Theta) \\ &\quad - \sum_{n=1}^{N^*_\mathcal{T}}\int_{t_0}^{t_1} Tr\left(\frac{\partial\mu_\beta}{\partial z_n(t)}\right)dt, \end{aligned} \tag{24}$$

where $\mathbf{z}^*_\mathcal{T} = \mathbf{f}_\beta^{-1}(\mathbf{y}^*_\mathcal{T}, h_\phi(\mathbf{X}^*_\mathcal{T}))$, $\mathbf{z}_\mathcal{T} = \mathbf{f}_\beta^{-1}(\mathbf{y}_\mathcal{T}, h_\phi(\mathbf{X}_\mathcal{T}))$, and $\mathbf{z}_\mathcal{S} = \mathbf{f}_\beta^{-1}(\mathbf{y}_\mathcal{S}, h_\phi(\mathbf{X}_\mathcal{S}))$ are the inverted transformations for testing and training data from $\mathcal{S}$ and $\mathcal{T}$, respectively. $N^*_\mathcal{T}$ denotes the amount of test data from $\mathcal{T}$.

E. Theoretical Analysis

1) Generalization Error Bound Analysis: To analyze how to ensure transfer performance in the presence of distributional shift among different workload, we quantify the relatedness between the source and target workloads by $\lambda_{(\mathcal{D}_i,\mathcal{D}_j)}$, and derive the generalized error bounds for *TGCP*.

*Theorem 1 (Generalization error bound of **TGCP**):* Based on Proposition 1 in [17] for bound of posterior variance, we can define the error bound for **TGCP** on the $\mathbf{X}^*_\mathcal{T}$, the upper

TABLE II
DESCRIPTIONS OF THE WORKLOADS FOR POWER CONSUMPTION DATA COLLECTION

Workload type	Workload	Descriptions
CPU-intensive	AES	Encryption and decryption of data using the AES algorithm.
	Compress	Compression and decompression of data.
	Lu	LU decomposition of the matrix.
	OLTP	Online Transactional Processing.
	Sha256	Hash the data using the Sha256. algorithm.
	Sor	Approximate solutions to a system of linear equations using the Successive Over Relaxation iteration algorithm.
	Sort	Sorting data.
memory-intensive	Cache	Cache performance test.
I/O-intensive	Random	Random read and write.

generalization error bound is formulated as:

$$\epsilon_t \leq \int \delta^2(\mathbf{1}, \max_i \left(\bar{\epsilon}^2_{\mathcal{S}_i}\right), \sigma^2_{\mathcal{T}}) p(\mathbf{x}_t) d\mathbf{x}_t, \tag{25}$$

where $\bar{\epsilon}^2_{\mathcal{S}_i} = \lambda^{-2}_{(\mathcal{S}_i,\mathcal{T})}(\sigma^2_{\mathcal{S}_i} + \overline{\eta}) - \overline{\eta}$. And $\overline{\eta}$ represents the maximum eigenvalue of $\hat{\mathcal{K}}_{\mathcal{S},\mathcal{S}}$. $\sigma_{\mathcal{S}_i}$ and $\sigma_{\mathcal{T}}$ represent the noise variance in source and target workloads, respectively. From Therorem 1, we can observe that the error bound of **TGCP** is directly related to $\lambda_{(\mathcal{S}_i,\mathcal{T})}$, while the source-source relatedness, represented by the eigenvalues of $\hat{\mathcal{K}}_{\mathcal{S},\mathcal{S}}$, only affects the performance implicitly. This indicates that the source-target relatedness has a greater impact on transfer performance than the source-source relatedness. Moreover, the bound shows that it is a monotonically decreasing function of $\lambda_{(\mathcal{S}_i,\mathcal{T})}$. In general, the higher the relatedness between the $\mathcal{S}_i$ and $\mathcal{T}$, the better the transfer performance of **TGCP**. The proof of Theorem 1 is provided in the appendix.

2) Complexity Analysis: The computational complexity of **TGCP** mainly comes from two parts: feature embedding using $h()$ for reversible transformations with CNF, and posterior inference based on transfer GPR. Since $h()$ is simple and the number of transformation steps of CNF is limited, the complexity of these parts is negligible. Suppose that the data from N source workloads is $N_{\mathcal{S}}$. Therefore, the main complexity of **TGCP** is dominated by the posterior inference, which is $O((N_{\mathcal{S}} + N_{\mathcal{T}})^3)$. Additionally, since the scale of $N_{\mathcal{T}}$ is sparse, the complexity can be further reduced to $O((N_{\mathcal{S}})^3)$. The inference complexity of trained **TGCP** is $O((N^*_{\mathcal{T}})^2)$, which is very short. It is important to emphasize that, compared to the high cost of power data collection in real data centers, *TGCP* leverages collected data to improve prediction performance for unseen workloads, making the training process both tractable and cost-effective.

IV. EXPERIMENTAL EVALUATIONS

A. Experimental Setup

The proposed **TGCP** is implemented on a server with i5-12400 CPU, 32 G RAM, NVIDIA RTX 3060Ti GPU for training and evaluation.

1) Description of Power Consumption Dataset: To derive the power consumption data during the implementation of different types of workloads, we ran three types of workloads supported by the server energy efficiency benchmark tool (BenchSEE[1]) on a blade server. The workload descriptions and server configuration are provided in Tables II and III, respectively.

TABLE III
CONFIGURATION OF THE BLADE SERVER FOR POWER CONSUMPTION DATA COLLECTION

OS version	CentOS Linux release 7.9.2009
Kernel version	3.10.0-1160.el7.x86_64
CPU	2*Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
Memory	8*Samsung DDR4 2933MT/s 32G
BenchSEE version	2.0.1

We also launched a performance monitor counter to capture the hardware parameters of the server during the workload runs, collecting 51 system parameters as features for the power prediction model. However, several collected feature values did not show significant variation, indicating they are not directly related to the server's power consumption. Therefore, we performed pre-processing on the collected power consumption datasets using principal component analysis (PCA), ensuring that only parameters directly related to power are retained while irrelevant features are discarded. For each source workload, 450 power data samples were randomly selected for training. Meanwhile, for each target workload, 50 and 400 power data samples were used for training and testing, respectively.

2) Evaluation Metrics: To demonstrate the performance of *TGCP*, Root Mean Squared Error (*RMSE*) in watts, Mean Absolute Error (*MAE*) in watts, and Mean Absolute Percentage Error (*MAPE*) have been chosen as the evaluation metrics, which are defined as

$$RMSE = \sqrt{\frac{1}{M}\sum\nolimits_{i=1}^{M}(\tilde{p}_i - p_i)^2}, \tag{26}$$

$$MAE = \frac{1}{M}\sum_{i=1}^{M}|(\tilde{p}_i - p_i)|, \tag{27}$$

$$MAPE = \frac{1}{M}\sum_{i=1}^{M}\frac{|(\tilde{p}_i - p_i)|}{p_i}, \tag{28}$$

where $\tilde{p}_i$ and p_i are prediction value and ground truth, respectively, and M is number of test data.

[1]https://www.energylabel.com.cn/benchsee/benchSEE_en.html

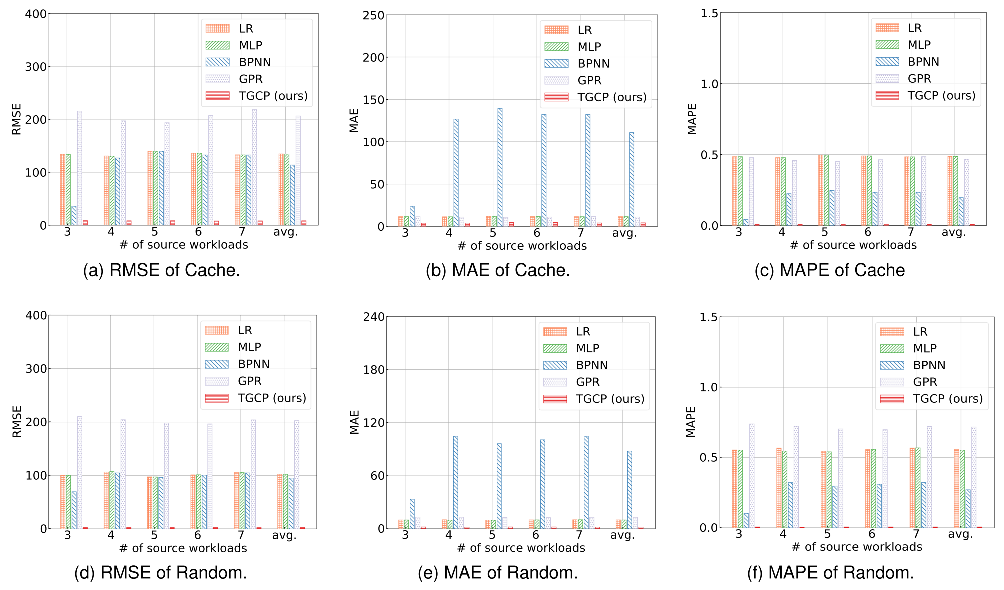


Fig. 4. Comparison of cross-workload power prediction with different traditional ML methods (Lower values for *RMSE*, *MAE*, and *MAPE* indicate better performance).

3) Comparison Methods: Four traditional ML methods, LR, MLP, BPNN and GPR are implemented by the Scikit-learn library.[2] Meanwhile, three transfer learning methods are implemented for comparison of transfer performance, listed as follows

- Two-stage TrAdaBoost.R2 (TrAdaBoost.R2) [21][3]
- Deep Kernel Transfer (DKT) [22][4]
- Multi-source Transfer GPR (TGP) [18]

TrAdaBoost.R2 is a classical instance-based transfer regression method that effectively handles domain shifts by reweighting source instances, making it suitable for scenarios where workload characteristics differ across domains. DKT and TGP are recent advanced models based on Gaussian processes, capable of capturing complex nonlinear relationships and modeling uncertainty in transfer learning settings. Given that cross-workload power prediction involves distributional shifts and limited target data, these methods align well with the challenges of our problem setting. Although they have not been specifically applied to power prediction tasks, their general-purpose design and strong performance in other regression-based transfer learning applications make them appropriate choices for comparison in this context. For fair comparison, we implemented those methods with default hyper-parameter settings. Meanwhile, we followed the TGP for setting up the radial basis function (RBF) kernel to implement $k_{ms}()$ for *TGCP*.

B. Evaluation Results

Note that since **TGCP** can simultaneously assess the relatedness between source-target workloads and between source-source workloads, there is no need to pre-select source workloads based on their similarity to the target workloads. To verify the performance of **TGCP**, we construct source domains with different numbers of CPU-intensive workloads in the order shown in Table II to enhance the power prediction performance for target workloads, which in our experiments, consist of memory-intensive workload (Cache) and I/O-intensive workload (Random).

1) Evaluations of Cross-Workload Power Prediction Compared to Traditional ML Methods: Fig. 4 shows the performance comparison between *TGCP* and four traditional ML methods in achieving cross-workload power consumption prediction. As the number of introduced source workloads increases, the performance of traditional ML methods for cross-workload power consumption prediction deteriorates. This decline occurs because the introduction of a large amount of source workload data causes the model to become biased towards learning the power consumption patterns of the source workloads, thereby neglecting the power consumption characteristics of the target workloads. Consequently, the model's performance worsens when applied to the target workloads. In contrast, *TGCP* can extract useful knowledge from different numbers of source workloads, thereby improving the power prediction performance for the target workloads. This effectively avoids negative transfer and demonstrates the superior cross-workload power prediction capability of *TGCP*.

2) Evaluations of Cross-Workload Power Prediction Compared to Transfer Learning Methods: We conducted an analysis of the performance based on different numbers of source workloads for cross-workload power prediction to better illustrate the transfer performance of **TGCP**.

As depicted in Table IV, which presents a performance comparison with three evaluation metrics for cross-workload power prediction under varying numbers of source workloads, TrAdaboost.R2 exhibits the poorest performance. Notably, the accuracy of TrAdaBoost.R2 noticeably deteriorates as the

[2]https://scikit-learn.org/stable/modules/classes.html
[3]https://github.com/jay15summer/Two-stage-TrAdaboost.R2
[4]https://github.com/BayesWatch/deep-kernel-transfer

TABLE IV
RESULTS FOR CROSS-WORKLOAD POWER PREDICTION WITH DIFFERENT TRANSFER LEARNING METHODS

		TrAdaBoost.R2			**DKT**			**TGP**			***TGCP (ours)***		
Target workload	**# of source workload**	**RMSE (Watts)**	**MAE (Watts)**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**
Cache	3	27.32095867	10.37681159	0.018438531	14.92796	8.622513	0.015329	9.085782	**2.828403**	**0.005187**	**8.644735**	3.746977	0.006725
	4	25.20006901	9.376811594	0.016708712	14.67136	9.130924	0.01617	9.023333	4.609683	0.008292	**8.396909**	**3.868876**	**0.006927**
	5	31.86656964	14.43478261	0.02558102	12.75695	7.912362	0.014038	9.008656	4.560887	0.008205	**8.471752**	**4.53261**	**0.008099**
	6	32.1977585	16.11594203	0.028564057	16.95812	10.74339	0.019043	9.046659	**4.634497**	**0.008337**	**8.06862**	4.727466	0.008389
	7	37.31883777	17.97101449	0.031901994	21.41678	10.87942	0.019308	8.760383	4.278553	0.007665	**8.300558**	**3.990739**	**0.007128**
	avg.	30.78083872	13.65507246	0.024238863	16.14623	9.457722	0.016778	8.984963	4.182405	0.007537	**8.376515**	**4.173334**	**0.007454**
Random	3	18.91674702	5.723502304	0.01760081	16.87775	9.418804	0.028907	2.735083	2.073052	0.006367	**2.66934**	**2.04287**	**0.006285**
	4	48.32722563	21.40092166	0.065906807	13.37542	8.162148	0.025038	3.138915	2.14501	0.006578	**2.702827**	**1.871907**	**0.005744**
	5	50.9369358	30.48387097	0.093698359	20.65187	10.39993	0.031934	3.037145	2.21297	0.006792	**2.648105**	**2.000362**	**0.006152**
	6	42.5025887	17.0875576	0.052493876	14.31744	8.4446	0.025904	2.869306	**2.002854**	**0.006141**	**2.706798**	2.06685	0.006357
	7	46.56495389	25.60368664	0.078857479	16.4204	9.237282	0.028362	3.136298	2.087572	0.006394	**2.566299**	**1.923918**	**0.005911**
	avg.	41.44969021	20.05990783	0.061711466	16.32858	9.132552	0.028029	2.983349	2.104292	0.006454	**2.658674**	**1.981181**	**0.00609**

The best results are bolded and the second best results are underlined.

number of source workloads increases, indicating a negative transfer phenomenon, as expected. The multi-source transfer regression implemented by TrAdaBoost.R2 only considers the relatedness of samples between the source and target workloads to tune the learning degree in the source domain, neglecting the relatedness between source workloads and the target workload, leading to the inclusion of similar samples that do not necessarily contain useful knowledge. Similarly, DKT faces similar concerns in implementing cross-workload power prediction, albeit its performance surpasses that of TrAdaBoost.R2 owing to its few-shot learning setting and deep transfer kernel.

Conversely, the cross-workload power prediction implemented based on TGP demonstrates robust performance superiority as a multi-source transfer regression method. Regardless of the workload (Random or Cache), TGP consistently leverages useful knowledge captured from source domains with varying numbers of source workloads to enhance predictive accuracy for the target workload. Unlike TrAdaBoost.R2 and DKT, the transfer kernel $k_{ms}()$ of TGP accounts for the relatedness between source and target workloads, as well as between source workloads themselves. This ability enables TGP to adjust the learning degree based on workload similarity, thereby avoiding negative transfer. Consequently, the cross-workload power prediction performance of TGP surpasses that of TrAdaBoost.R2 and DKT.

However, as TGP relies on GPR for transfer regression, its efficacy is limited by the normal distribution assumption and the closed-form computation of the posterior probability function. This limitation restricts TGP's flexibility in capturing complex distributions of real-world data, as discussed in Section II regarding power consumption distribution, where real power consumption data often deviates from a normal distribution. To address this, we propose **TGCP** to further enhance the cross-workload power prediction performance of TGP for real power consumption data scenarios by using CNF to locally non-Gaussianize the posterior prediction distribution of TGP. Table IV demonstrates that **TGCP** surpasses TGP in most cases across the three metrics, illustrating its cross-workload power prediction performance. However, we also observe that **TGCP** does not consistently achieve optimal performance, particularly when the number of source workloads is small, where *TGCP* performs slightly worse than TGP. Nevertheless, with an increasing number of source workloads, *TGCP* outperforms TGP, indicating room for further improvement in *TGCP*'s performance.

3) Analysis of the Influence of Target Workload Data Quantity on Transfer Learning Training: Additionally, we investigated the impact of varying amounts of target workload data involved in the training of transfer learning methods on the performance of cross-workload power prediction with three source workloads. We examined the performance of cross-workload power prediction by setting the target workload power data amount from 10 to 50, increasing in intervals of 5. Fig. 5 illustrates that as the amount of target workload power data increases, the prediction errors for all four methods exhibit a decreasing trend. This result is expected because the growth in target workload data provides more information about the target workload, thereby improving prediction performance. However, it can also be concluded that the performance advantage of transfer learning will eventually diminish as the target workload's training data continues to grow.

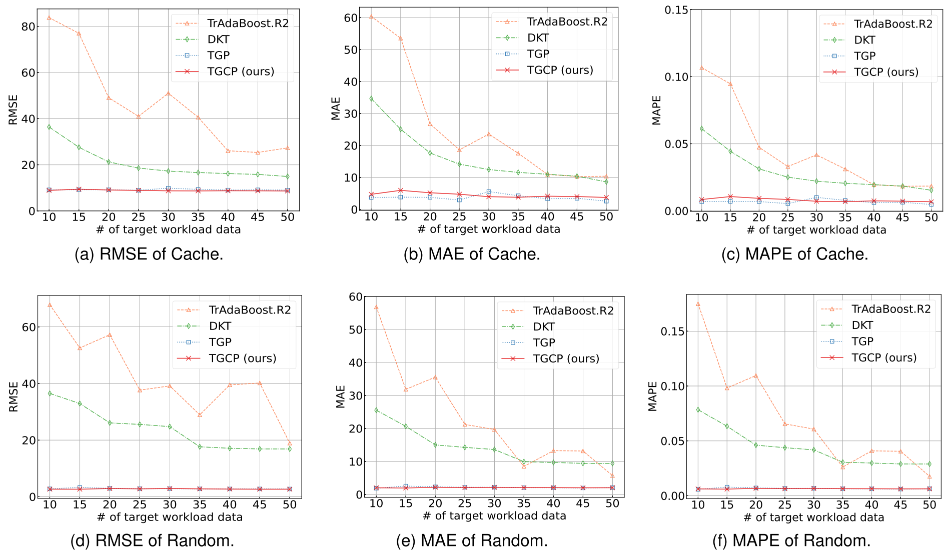
Overall, the cross-workload power prediction performance achieved by *TGCP* and TGP outperforms that of TrAdaBoost.R2 and DKT across different amounts of target workload power data. Specifically, *TGCP* outperforms TGP in terms of *RMSE*, *MAE*, and *MAPE* for Random workloads. For Cache workloads, *TGCP* only outperforms TGP in terms of *RMSE*, while the two methods perform variably on *MAE* and *MAPE*, indicating the need for further tuning of *TGCP*'s performance.

The primary reason for these results is that *TGCP* allows for similarity estimation between source and target workloads while controlling the degree of knowledge transfer to different source workloads. This approach effectively avoids negative transfer and improves power prediction when only a small amount of target workload power data is available for training. Furthermore, the adoption of CNF enables *TGCP* to better handle the complex distribution of real power data from different types of workloads compared to TGP, thereby further enhancing the performance of cross-workload power prediction.

V. RELATED WORK

A. Power Prediction for Servers in Cloud Data Centers

In recent years, power prediction of cloud servers has become an attractive field of research, critical for developing energy-efficient technologies in cloud computing [23], [24]. Recent reviews have analyzed and discussed power prediction models from different perspectives [3], [25], [10], [11].



(a) RMSE of Cache. (b) MAE of Cache. (c) MAPE of Cache.

(d) RMSE of Random. (e) MAE of Random. (f) MAPE of Random.

Fig. 5. Comparison of cross-workload power prediction with different number of target data for training (Lower values for *RMSE*, *MAE*, and *MAPE* indicate better performance).

Machine learning (ML), in particular, has demonstrated powerful non-linear fitting capabilities, driving the implementation of server power prediction towards ML methods. Lin et al. [26] analyzed parameters of servers running different types of workloads using data collected by performance monitor counters. They trained power models using BPNN, ENN, and LSTM on different workloads, and these well-trained models showed superior power prediction performance. Liang et al. [27] designed a power feature selection method based on information entropy theory to identify high-impact features during various workloads. They then constructed a power prediction model using DNN, achieving high prediction performance through training with sufficient power data. Jing et al. [28] developed a power prediction model based on CNN-BiLSTM, using a one-dimensional CNN for local feature extraction and dimensionality reduction of power consumption features, and a BiLSTM network for high-level feature extraction, incorporating an attention mechanism to filter irrelevant features. This approach enabled power prediction for heterogeneous servers and workloads. Zhou et al. [29] presented a power model built on SVM, utilizing random forests for parameter tuning and a grid search method for hyperparameter optimization, achieving accurate power prediction across different workloads. Moolchandani et al. [30] focused on power prediction for servers executing multiple GPU-intensive workloads concurrently, proposing a method for power prediction during the concurrent execution of multiple GPU-intensive applications. Ferroni et al. [31] identified server working regimes based on hardware events and constructed power models for each state, addressing the accuracy degradation problem caused by workload changes. Given the heterogeneity of server software and hardware, Bernard et al. [32] proposed a hybrid power prediction model, Hydra, which dynamically selects the best power model for given server conditions. Lin et al. [33] proposed an adaptive workload-aware power prediction method. This method first classifies workloads using K-means clustering and then selects an appropriate power model based on the server's current workload to achieve accurate power prediction.

While these traditional ML-based power prediction models exhibit good performance, they still require a substantial amount of runtime data, imposing high data acquisition costs. Additionally, the assumption of independent and identically distributed training and test data, which traditional machine learning models rely on, poses a challenge in handling the domain shift problem in cross-workload power prediction scenarios. This limitation makes it difficult for current power prediction models to adapt effectively when power consumption behavior varies significantly between different workloads.

B. Transfer Learning for Regression

The principle of transfer learning entails utilising knowledge gained from a source domain that possesses plenty of labelled data to a target domain with constrained data. It promotes the model's accuracy in predicting unseen data of the target domain and has exhibited remarkable achievements in classification tasks, including natural language processing [34], [35] and object detection [36]. However, the main research interest in transfer learning focuses on classification problems, while research on transfer learning for regression problems is comparatively limited. Typically, AdaBoost-based transfer learning methods for classification problems [37] have been extended to regression problems, leading to the proposal of TrAdaBoost [21]. Liu et al. [38] proposed a deep adaptation method for conditional distribution in regression problems, reducing the difference in conditional distribution between source and target domains. Patacchiola et al. [22] introduced a Bayesian-based deep kernel transfer method.

Furthermore, transfer learning methods based on GPR have gained significant attention in recent years. In [39], Cao et al. designed an effective transfer kernel $k_\lambda()$, which was also theoretically proven to be PSD. $k_\lambda()$ enables a learnable parameter λ to control the learning degree of the single-source data, effectively avoiding negative transfer. Recent studies on single-source domain transfer GPR [40], [41] have enabled the implementation of a more powerful transfer kernel to deal with heterogeneous data. Single-source domain transfer GPR, however, has major limitations since it is only capable of enhancing performance for unseen data from the target domain using information from the single source domain. Then, Wei et al. [42] proposed a transfer covariance function based on multi-kernel learning to model the heterogeneous sub-similarity of domains, achieving regression transfer learning with complete theoretical proof of effectiveness. To facilitate the retrieval of valuable knowledge from multiple source domains simultaneously, multi-source transfer GPR methods have been explored. Yang et al. [43] proposed a source-target pairwise segmentation method, segmenting different source and target domains into similar parts and extracting the most similar parts from different source domains for knowledge transfer. Wei et al. [44] implemented similarity estimation between multiple source and target domains by introducing a transfer covariance function, thus preventing negative transfer. To address the challenge of time complexity in Gaussian models dealing with large-scale datasets, Yang et al. [45] proposed a sparse GP-based regression transfer learning method, reducing time complexity for large datasets. Da et al. [46] proposed an aggregation model for multi-source domain transfer GPR, enabling parallel training and testing of models with multi-source data for efficient computation.

To the best of our knowledge, this is the first work to implement cross-workload power prediction for servers in CDCs via transfer learning. Our method leverages abundant power consumption data from source workloads to improve the performance of target workloads with limited power data, unlocking the valuable knowledge embedded in collected power data while reducing the cost of power data collection.

VI. Conclusion and Future Work

In this paper, we systematically answer why traditional ML methods cannot enable cross-workload power prediction by analyzing the features and distribution of the collected power data. To implement cross-workload power prediction, we propose **TGCP** based on multi-source transfer GPR, which is dedicated to exploiting the practical knowledge contained in the massive power data of source workloads towards improving the power prediction performance of target workload that have limited power data, thus significantly reducing the cost for power data collection required to train power models of cloud servers. In addition, to enhance the flexibility of the multi-source transfer GPR to dealing with real power data, CNF is used to adjust the posterior prediction distribution of the transfer GPR to make it locally non-Gaussian, further enhancing the performance of the cross-workload power prediction. The results of experiments revealed that **TGCP** delivers a better performance in cross-workload power prediction than four traditional ML methods as well as three transfer learning methods.

While **TGCP** enables efficient knowledge transfer across workloads and reduces data collection costs, it incurs higher training-time complexity due to the cubic scaling of Gaussian processes. To address this, future work will explore sparse Gaussian process techniques to improve scalability without compromising accuracy.

References

[1] M. Law, “Energy efficiency predictions for data centres in 2023,” 2022. [Online]. Available: https://datacentremagazine.com/articles/efficiency-to-loom-large-for-data-centre-industry-in-2023

[2] H. Cheung, S. Wang, C. Zhuang, and J. Gu, “A simplified power consumption model of information technology (IT) equipment in data centers for energy system real-time dynamic simulation,” *Appl. Energy*, vol. 222, pp. 329–342, 2018.

[3] W. Lin, F. Shi, W. Wu, K. Li, G. Wu, and A.-A. Mohammed, “A taxonomy and survey of power models and power modeling for cloud servers,” *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–41, 2020.

[4] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, “Adaptive energy-aware algorithms for minimizing energy consumption and sla violation in cloud computing,” *IEEE Access*, vol. 6, pp. 55923–55936, 2018.

[5] Y. Xie, X.-Y. Wang, Z.-J. Shen, Y.-H. Sheng, and G.-X. Wu, “A two-stage estimation of distribution algorithm with heuristics for energy-aware cloud workflow scheduling,” *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4183–4197, Nov./Dec. 2023.

[6] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, “Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing,” *IEEE Trans. Veh. Technol*, vol. 69, no. 12, pp. 14 198–14 211, Dec. 2020.

[7] W. Zhang, R. Yadav, Y.-C. Tian, S. K. S. Tyagi, I. A. Elgendy, and O. Kaiwartya, “Two-phase industrial manufacturing service management for energy efficiency of data centers,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 7525–7536, Nov. 2022.

[8] R. Yadav and W. Zhang, “MeReg: Managing energy-SLA tradeoff for green mobile cloud computing,” *Wireless Commun. Mobile Comput.*, vol. 2017, no. 1, 2017, Art. no. 6741972.

[9] R. Yadav, W. Zhang, K. Li, C. Liu, and A. A. Laghari, “Managing overloaded hosts for energy-efficiency in cloud data centers,” *Cluster Comput.*, vol. 24, pp. 2001–2015, 2021.

[10] L. Ismail and H. Materwala, “Computing server power modeling in a data center: Survey, taxonomy, and performance evaluation,” *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, 2020.

[11] Q. Zhang et al., “A survey on data center cooling systems: Technology, power consumption modeling and control strategy optimization,” *J. Syst. Architecture*, vol. 119, 2021, Art. no. 102253.

[12] W. Dargie, “A stochastic model for estimating the power consumption of a processor,” *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1311–1322, May 2015.

[13] T. Khan, W. Tian, S. Ilager, and R. Buyya, “Workload forecasting and energy state estimation in cloud data centres: ML-centric approach,” *Future Gener. Comput. Syst.*, vol. 128, pp. 320–332, 2022.

[14] Z. Shen, X. Zhang, B. Xia, Z. Liu, and Y. Li, “Multi-granularity power prediction for data center operations via long short-term memory network,” in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw.*, 2019, pp. 194–201.

[15] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[16] J. Zhang et al., “Minority disk failure prediction based on transfer learning in large data centers of heterogeneous disk systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2155–2169, Sep. 2020.

[17] P. Wei, Y. Ke, Y. S. Ong, and Z. Ma, “Adaptive transfer kernel learning for transfer gaussian process regression,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7142–7156, Jun. 2023.

[18] P. Wei, T. V. Vo, X. Qu, Y. S. Ong, and Z. Ma, “Transfer kernel learning for multi-source transfer Gaussian process regression,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3862–3876, Mar. 2023.

[19] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, “FFJORD: Free-form continuous dynamics for scalable reversible generative models,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[20] M. Sendera et al., "Non-Gaussian processes for few-shot regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 10285–10298.
[21] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 863–870.
[22] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. J. Storkey, "Bayesian meta-learning for the few-shot setting via deep kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 16108–16118.
[23] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N. K. Karn, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Netw.*, vol. 26, pp. 1905–1919, 2020.
[24] N. Mc Donnell, E. Howley, and J. Duggan, "Dynamic virtual machine consolidation using a multi-agent system to optimise energy efficiency in cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 288–301, 2020.
[25] R. Yadav, W. Zhang, H. Chen, and T. Guo, "MuMs: Energy-aware VM selection scheme for cloud data center," in *Proc. IEEE 28th Int. Workshop Database Expert Syst. Appl.*, 2017, pp. 132–136.
[26] W. Lin, G. Wu, X. Wang, and K. Li, "An artificial neural network approach to power consumption model construction for servers in cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 3, pp. 329–340, Jul.-Sep. 2020.
[27] Y. Liang, Z. Hu, and K. Li, "Power consumption model based on feature selection and deep learning in cloud computing scenarios," *IET Commun.*, vol. 14, no. 10, pp. 1610–1618, 2020.
[28] C. Jing and J. Li, "CBLA_PM: An improved ANN-based power consumption prediction algorithm for multi-type jobs on heterogeneous computing server," *Cluster Comput.*, vol. 27, pp. 377–394, 2023.
[29] Z. Zhou, M. Shojafar, M. Alazab, and F. Li, "IECL: An intelligent energy consumption model for cloud manufacturing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8967–8976, Dec. 2022.
[30] D. Moolchandani, A. Kumar, and S. R. Sarangi, "Performance and power prediction for concurrent execution on GPUs," *ACM Trans. Architecture Code Optim.*, vol. 19, no. 3, pp. 1–27, 2022.
[31] M. Ferroni et al., "Power consumption models for multi-tenant server infrastructures," *ACM Trans. Architecture Code Optim.*, vol. 14, no. 4, pp. 1–22, 2017.
[32] N. Bernard et al., "Hydra: Hybrid server power model," 2022, *arXiv:2207.10217*.
[33] W. Lin, Y. Zhang, W. Wu, S. Fong, L. He, and J. Chang, "An adaptive workload-aware power consumption measuring method for servers in cloud data centers," *Computing*, vol. 105, pp. 515–538, 2023.
[34] J. Tu et al., "Domain adaptation for deep entity resolution," in *Proc. Int. Conf. Manage. Data*, 2022, pp. 443–457.
[35] Y. Dai, J. Liu, X. Ren, and Z. Xu, "Adversarial training based multi-source unsupervised domain adaptation for sentiment analysis," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 7618–7625.
[36] P. Oza, V. A. Sindagi, V. V. Sharmini, and V. M. Patel, "Unsupervised domain adaptation of object detectors: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 6, pp. 4018–4040, Jun. 2024.
[37] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 193–200.
[38] X. Liu, Y. Li, Q. Meng, and G. Chen, "Deep transfer learning for conditional shift in regression," *Knowl.-Based Syst.*, vol. 227, 2021, Art. no. 107216.
[39] B. Cao, S. J. Pan, Y. Zhang, D.-Y. Yeung, and Q. Yang, "Adaptive transfer learning," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 407–412.
[40] P. Wei, R. Sagarna, Y. Ke, and Y. S. Ong, "Uncluttered domain subsimilarity modeling for transfer regression," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 1314–1319.
[41] M. Papež and A. Quinn, "Transferring model structure in Bayesian transfer learning for gaussian process regression," *Knowl.-Based Syst.*, vol. 251, 2022, Art. no. 108875.
[42] P. Wei, R. Sagarna, Y. Ke, and Y.-S. Ong, "Easy-but-effective domain sub-similarity learning for transfer regression," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4161–4171, Sep. 2020.
[43] K. Yang, J. Lu, W. Wan, and G. Zhang, "Multi-source transfer regression via source-target pairwise segment," *Inf. Sci.*, vol. 556, pp. 389–403, 2021.
[44] P. Wei, R. Sagarna, Y. Ke, and Y.-S. Ong, "Practical multisource transfer regression with source–target similarity captures," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3498–3509, Aug. 2021.
[45] K. Yang, J. Lu, W. Wan, G. Zhang, and L. Hou, "Transfer learning based on sparse gaussian process for regression," *Inf. Sci.*, vol. 605, pp. 286–300, 2022.
[46] B. Da, Y.-S. Ong, A. Gupta, L. Feng, and H. Liu, "Fast transfer gaussian process regression with large-scale sources," *Knowl.-Based Syst.*, vol. 165, pp. 208–218, 2019.

Ruichao Mo received the bachelor's and master's degrees from the School of Computer and Software, Nanjing University of Information Science and Technology, in 2019 and 2022, respectively. He is currently working toward the PhD degree with the School of Computer and Engineering, South China University of Technology. His research interests include cloud computing, edge computing, and transfer learning.

Weiwei Lin (Senior Member, IEEE) received the BS and MS degrees from Nanchang University, in 2001 and 2004, respectively, and the PhD degree in computer application from the South China University of Technology, in 2007. He has been a visiting scholar with Clemson University from 2016 to 2017. Currently, he is a professor with the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, and AI application technologies. He has published more than 150 papers in refereed journals and conference proceedings. He has been a reviewer for many international journals, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Computers*, *IEEE Transactions on Cybernetics*, etc. He is a distinguished member of CCF.

Haocheng Zhong received the bachelor's degree from the School of Computer Science and Technology, University of Science and Technology of China, in 2022. He is currently working toward the master's degree with the School of Computer Science and Engineering, South China University of Technology. His research areas include cloud computing, cluster computing, and cluster performance modeling.

Minxian Xu (Senior Member, IEEE) received the BSc and MSc degrees in software engineering from the University of Electronic Science and Technology of China, in 2012 and 2015, respectively, and the PhD degree from the University of Melbourne, in 2019. He is currently an associate professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests include resource scheduling and optimization in cloud computing. He has coauthored 50+ peer-reviewed papers published in prominent international journals and conferences, such as *ACM Computing Surveys*, *IEEE Transactions on Sustainable Computing*, *IEEE Transactions on Automation Science and Engineering*, *Journal of Parallel and Distributed Computing*, *Journal of Systems and Software* and International Conference on Service-Oriented Computing. His PhD Thesis was awarded the 2019 *IEEE TCSC Outstanding PhD Dissertation Award*. He is a member of CCF. More information can be found at: minxianxu.info.

Keqin Li (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York. He is also a distinguished professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, Big Data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published more than 740 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*.