软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007424] [CSTR: 32375.14.jos.007424]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

云计算中基于 SAC 的多视角工作负载预测集成框架*

曾文瑄[1], 应 时[1], 李田港[1], 田相波[1], 姜宇虹[2], 刘虎杰[3], 郝诗魁[3]

[1](武汉大学 计算机学院, 湖北 武汉 430072)

[2](中国地质大学 (武汉) 计算机学院, 湖北 武汉 430078)

[3](湖北省楚天云有限公司, 湖北 武汉 430076)

通信作者: 应时, E-mail: yingshi@whu.edu.cn

摘 要: 工作负载的准确预测对于云资源管理至关重要. 然而, 现有预测模型通常使用固化结构从不同视角提取序列特征, 导致不同模型结构之间难以灵活组合以进一步提升预测性能. 提出一种基于软演员-评论家算法 (soft actor-critic, SAC) 的多视角工作负载预测集成框架 SAC-MWF. 首先, 设计一组特征序列构建方法来生成多视角特征序列, 该方法能够以低成本从历史窗口生成特征序列, 从而引导模型关注不同视角下的云工作负载序列模式. 其次, 在历史窗口和特征序列上分别训练基础预测模型和若干特征预测模型, 以捕获不同视角下的云工作负载模式. 最后, 利用 SAC 算法集成基础预测模型和特征预测模型, 生成最终的云工作负载预测. 在 3 个数据集上的实验结果表明, SAC-MWF 方法在有效性和计算效率方面表现优秀.

关键词: 云计算; 工作负载预测; 强化学习; 多视角工作负载

中图法分类号: TP311

中文引用格式: 曾文瑄, 应时, 李田港, 田相波, 姜宇虹, 刘虎杰, 郝诗魁. 云计算中基于SAC的多视角工作负载预测集成框架. 软件学报. http://www.jos.org.cn/1000-9825/7424.htm

英文引用格式: Zeng WX, Ying S, Li TG, Tian XB, Jiang YH, Liu HJ, Hao SK. SAC-based Ensemble Framework for Multi-view Workload Forecasting in Cloud Computing. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7424.htm

SAC-based Ensemble Framework for Multi-view Workload Forecasting in Cloud Computing

ZENG Wen-Xuan[1], YING Shi[1], LI Tian-Gang[1], TIAN Xiang-Bo[1], JIANG Yu-Hong[2], LIU Hu-Jie[3], HAO Shi-Kui[3]

[1](School of Computer Science, Wuhan University, Wuhan 430072, China)

[2](School of Computer Science, China University of Geosciences, Wuhan 430078, China)

[3](Hubei ChuTianYun Co. Ltd., Wuhan 430076, China)

Abstract: Accurate workload forecasting is essential for effective cloud resource management. However, existing models typically employ fixed architectures to extract sequential features from different perspectives, which limits the flexibility of combining various model structures to further improve forecasting performance. To address this limitation, a novel ensemble framework SAC-MWF is proposed based on the soft actor-critic (SAC) algorithm for multi-view workload forecasting. A set of feature sequence construction methods is developed to generate multi-view feature sequences at low computational cost from historical windows, enabling the model to focus on workload patterns from different perspectives. Subsequently, a base prediction model and several feature prediction models are trained on historical windows and their corresponding feature sequences, respectively, to capture workload dynamics from different views. Finally, the SAC algorithm is employed to integrate these models to generate the final forecast. Experimental results on three datasets demonstrate that SAC-MWF performs excellently in terms of effectiveness and computational efficiency.

Key words: cloud computing; workload forecasting; reinforcement learning; multi-view workload

---

* 基金项目: 国家自然科学基金 (62472329, 62072342); 国家重点研发计划 (2022YFB3304300)
收稿时间: 2024-08-22; 修改时间: 2024-10-24, 2025-01-21; 采用时间: 2025-03-03; jos 在线出版时间: 2025-08-20

云资源管理是云计算领域的核心挑战之一[1,2], 其对于优化运营成本、提高资源利用率以及保障系统可靠性具有关键作用[3]. 云资源管理的一个有效策略是预测未来的工作负载并提前采取措施, 通过准确的工作负载预测, 云服务提供商可以主动识别潜在资源瓶颈, 并基于实时需求调整资源配给策略[2], 从而实现高效的资源调度与节能优化[4,5]. 若缺乏有效的预测机制, 资源管理将被动进行, 低负载期间容易导致资源闲置浪费, 高负载时则可能引发节点过载, 并最终影响服务质量协议 (service level agreement, SLA)[6].

云工作负载序列模式具有动态变化的特点[7,8]. 传统统计学习和机器学习模型在预测任务中存在误差累积和泛化能力不足的问题[9]. 近年来, 深度学习模型通过一些途径来提升预测性能, 例如: 1) 改进现有网络架构, 如长短期记忆网络 (long short-term memory, LSTM), 以提高其预测准确度[10,11]; 2) 构建混合模型, 如结合卷积神经网络 (convolutional neural networks, CNN) 和 LSTM, 利用 CNN 提取特征后通过 LSTM 进行特征学习和预测[12,13]; 3) 进行序列分解, 将工作负载分解为多个子分量分别预测并融合各分量上的结果[14]. 然而, 上述方法多从单一视角提取特征, 未能充分挖掘多视角信息的互补性. 近年来, 一些新的时序预测模型从不同视角着手捕获序列信息, 如 SCINet[15]通过将原始输入分解为奇数索引序列和偶数索引序列, 并应用递归下采样来捕获时序特征. TimeMixer[16]通过趋势分解和多尺度信息融合来构建预测模型. DLinear[17]通过简单的趋势-季节性分解和线性模型来构建简单有效的预测模型. 上述模型的成功证明了从多视角提取序列特征能够进一步提高模型的预测性能, 然而固化的特征提取结构难以灵活组合以构建预测性能更强的模型.

针对上述问题, 本文提出了一个基于软演员-评论家算法 (soft actor-critic, SAC)[18]的多视角工作负载预测集成框架 (soft actor-critic-based multi-view workload forecasting, SAC-MWF). 首先, 设计了一组特征序列构建方法, 通过对历史窗口应用数学变换构建多视角特征序列, 突出不同视角下的负载模式特征. 其次, 基于历史窗口与特征序列分别训练基础预测模型与特征预测模型, 捕获多视角特征, 生成具有强互补性的预测结果. 最后, 利用 SAC 算法自适应分配权重, 集成多视角模型预测结果并生成最终预测结果. 在 3 个数据集上的实验结果表明, SAC-MWF 能够有效提升多种模型的预测准确度, 且不会显著增加时间开销.

本文的主要贡献如下.

(1) 提出一个基于强化学习的多视角工作负载预测集成框架 SAC-MWF, 通过特征序列构建与动态权重分配提升现有模型的预测准确度.

(2) 提出了一个基于 SAC 的模型集成方法, 其状态空间涵盖历史窗口与预测误差, 奖励函数基于集成预测的均方误差设计, 能够灵活分配组合权重.

(3) 设计了一组轻量级特征序列构建方法, 增强模型对趋势变化、峰值波动、滞后等关键模式的敏感度.

(4) 在 3 个数据集上进行了广泛实验, 结果证明了 SAC-MWF 的有效性和效率.

本文第 1 节综述云计算负载预测方法与新型时序模型. 第 2 节形式化定义问题并详述 SAC-MWF 框架设计. 第 3 节通过多组实验全面评估方法性能. 第 4 节总结研究成果并展望未来方向.

1 相关工作

工作负载预测作为云计算资源管理的核心环节, 其方法可归纳为传统模型方法、深度学习模型方法与混合模型方法这 3 大类.

1.1 传统模型方法

传统模型方法以统计学习模型和经典机器学习模型为主, 如差分自回归滑动平均模型 (autoregressive integrated moving average model, ARIMA)、支持向量机 (support vector machine, SVM) 和随机森林 (random forest, RF) 等. 研究人员在早期提出了许多基于传统模型的方法以解决云计算领域的预测问题. 例如, Yang 等人[19]提出了一种动态线性回归模型用于短期近线性负载预测. Fang 等人[20]通过 ARIMA 预测服务器资源需求以驱动调度策略生成. Chen 等人[21]应用递归神经网络 (recurrent neural network, RNN) 预测云服务器的工作负载, 验证了 RNN 在小样本场景下相较于传统回归方法的快速适应能力. Yang 等人[22]提出了一个基于进化算法的预测模型, 模型使用相空间

重构处理负载数据, 并将重建后的多维时间序列输入基于进化算法的数据分组处理方法以预测未来的主机负载.

统计学习模型依赖工作负载都是线性的假设, 该假设过于简单且此类模型无法应对模式变化, 因而并不适合长期预测任务[23]. 经典机器学习模型适合处理低维数据, 而在处理高维数据时易受维度灾难限制[24]. 传统模型现阶段多作为多模型方法的组件.

1.2 深度学习模型方法

深度学习模型方法通过非线性建模能力打破传统模型方法的性能瓶颈. 在各类深度学习模型方法中, 基于 LSTM 的预测模型因其优异的时序依赖捕获能力和计算开销优势而得到广泛应用[25]. 例如, Gao 等人[26]使用 LSTM 预测边缘数据中心的响应时间等指标. Ruan 等人[27]采用改进的 LSTM 模型预测 CPU 利用率. 近年来, 研究者进一步从多视角特征提取角度创新模型结构. 例如, Zeng 等人[17]提出了简单有效的 DLinear 模型, 通过趋势-季节性分解与线性层构建轻量预测模型. Liu 等人[15]提出的 SCINet 模型采用奇偶索引分解与多分辨率卷积捕获动态时序模式. Wang 等人[16]提出一种基于多尺度融合架构的预测模型 TimeMixer, 通过趋势分解与多尺度融合生成互补性预测结果并整合为最终预测结果.

尽管深度学习模型在复杂模式建模上表现优异, 但其网络复杂度和计算成本会随着数据维度增大而快速增加. 这导致模型的运行时开销激增, 并限制了其在高计算成本场景中的应用[24]. 新型时序预测模型大多有着独特的结构设计, 对于序列特征的提取能力较传统模型有明显增强[28−30]. 但其特征提取结构高度固化, 难以灵活组合以构建预测性能更强的模型.

1.3 混合模型方法

混合模型方法通过融合传统模型与深度学习模型的优势以进一步提升模型预测性能. 串联式混合架构是主流设计方式之一, 如, Devi 等人[14]提出了一种 ARIMA 加人工神经网络 (artificial neural network, ANN) 的混合模型, 其中 ARIMA 处理数据的线性成分, 而 ANN 处理非线性成分. Patel 等人[13,31]构建 CNN-LSTM 混合模型, 利用 CNN 提取局部空间特征后通过 LSTM 进行时序预测. Chen 等人[12]引入 Savitzky-Golay 滤波器进行数据平滑预处理, 之后使用 CNN-BiLSTM 模型进行预测.

另一类研究聚焦集成式混合模型, 即通过对多个模型的预测结果进行选择、平均、加权来生成最终预测结果. 在实际研究中, Saadallah 等人[32]使用演员-评论家算法来学习组合模型的最优权重. 在其研究中, 强化学习的动作被设计为组合基础模型的权重向量, 状态则对应之前若干步集成模型的预测输出, 奖励函数由集成模型在全部模型中的预测误差排名决定. 强化学习可以通过与环境的交互自主学习模型集成的最优权重, 并根据环境变化及时优化策略, 其强适应性和捕捉长期回报的能力有助于生成更稳健的集成模型.

现有混合模型方法存在模型冗余问题. 串联架构中多个子模型的叠加会显著增加计算开销, 且其相对固定的结构难以保证模型的泛化能力. 而大型集成模型所使用的模型池包含大量不同种类的模型, 这种方式虽能确保集成模型的预测准确度优于对比方法, 但难以保证模型的通用性, 且存在资源开销过大的问题.

2 研究动机

如图 1 所示, 在对同一负载序列施加不同的特征变换后, 基于各特征序列的预测结果 (蓝色) 在不同时间区间呈现较大差异, 在不同的预测步长上有不同的最优预测模型. 这表明多视角特征序列预测值之间存在互补性, 若合理融合多视角预测结果, 能够进一步提升模型预测性能.

图 2 展示了原始序列预测 (绿色实线)、特征序列预测 (虚线)、最优集成预测 (蓝色实线) 和真实值 (红色实线) 共 9 条序列的对比. 从图中可以看出, 在起伏较大的波峰和波谷片段, 原始序列预测会出现增减趋势相反、峰值幅度不足、滞后问题, 与真实序列偏差较大. 而部分特征序列在这些时刻能够提供更优预测值, 在一定程度上弥补原始序列预测的偏差.

基于上述分析可以认为, 通过构建特征序列来帮助模型提升信息捕获能力是有效的. 基于不同特征序列生成的预测结果之间存在互补性, 而这种互补性可用于构建更加强大的预测模型. 然而, 现有方法难以有效利用此类互

补性: 固化的特征提取结构难以灵活组合, 而传统集成方法 (如加权平均、固定规则选择) 缺乏对动态负载模式变化的适应能力, 导致集成效果受限.

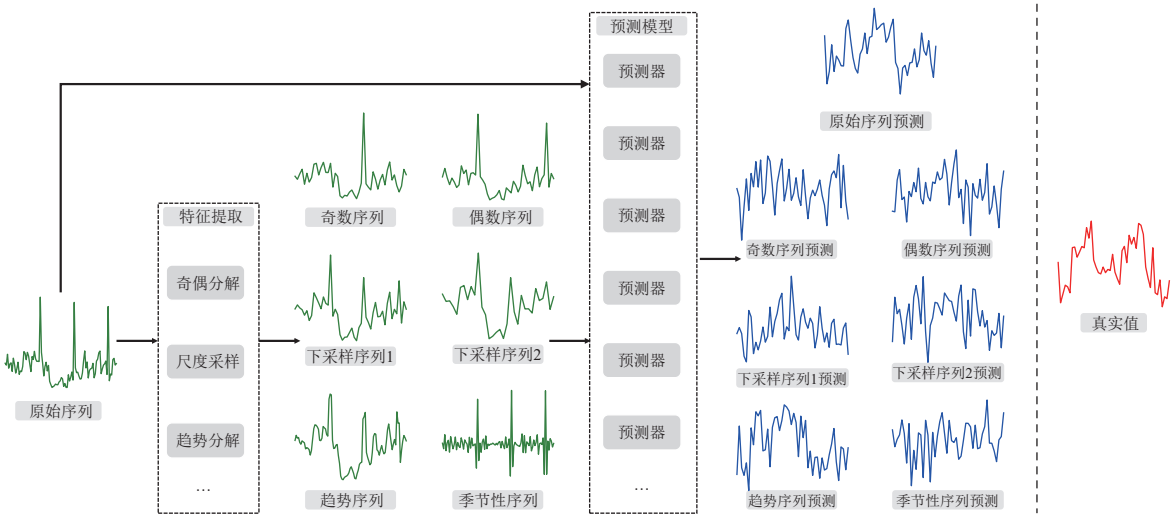


图 1 特征序列预测值对比

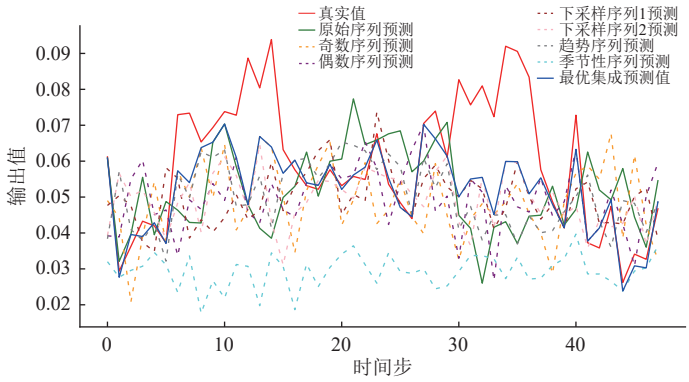


图 2 特征序列预测值集成可视化

因此, 我们需要解决两个问题: 1) 如何以低成本生成具有互补信息的特征序列; 2) 如何动态融合多模型预测结果以有效集成. 针对上述问题, 本文提出 SAC-MWF 框架, 通过设计轻量级特征变换操作构建多视角特征序列, 并利用强化学习驱动的动态权重分配机制实现预测结果的动态集成.

3 多视角工作负载预测集成框架 SAC-MWF

3.1 问题定义

给定单变量时间序列 X, 设历史窗口大小 T, 序列 $X_t = \{x_{t-T}, \ldots, x_i, \ldots, x_{t-1}\}$ 对应 t 时刻的历史窗口, 其中 $x_i \in \mathbb{R}$ 表示序列 X 在 i 时刻的负载值. 预测任务的目标是训练模型 f_θ, 使其能够基于历史窗口 X_t 生成未来 H 步的预测值:

$$\hat{Y}_t = f_\theta(X_t) = \{\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+j-1}, \ldots, \hat{y}_{t+H-1}\} \tag{1}$$

其中, $\hat{y}_{t+j-1}$ 表示第 j 个预测步长上的预测值, H 为预测窗口大小.

对于多模型集成任务, 设基础预测模型集合为 $F = \{f_1, f_2, \ldots, f_M\}$, 其中 f_m 表示第 m 个模型的预测函数, M 为基础预测模型总数. 多模型集成的目标是学习一组动态权重 ω, 以对 M 个模型进行加权运算以生成集成预测值

$\hat{Y}_t^{\text{ensemble}}$. 记第 m 个基础模型在第 j 个预测步长上的权重系数为 ω_m^j, ω_m^j 为非负数且同一预测步长上的权重系数和为 1:

$$\hat{Y}_t^{\text{ensemble}} = \sum_{m=1}^{M} \omega_m^j \cdot \hat{Y}_t^m, \quad \text{s.t.} \sum_{m=1}^{M} \omega_m^j = 1, \omega_m^j \geqslant 0 \tag{2}$$

3.2 SAC-MWF 结构设计

SAC-MWF 的目标是通过构建集成模型来实现准确的云工作负载预测. 框架以过去的云工作负载数据作为输入, 使用多种特征序列构建方法生成特征序列, 随后训练一个基础预测模型、多个特征预测模型和一个 SAC 模型用于集成预测. 当进行预测时, 它将使用基础预测模型和特征预测模型生成一组多视角预测值, 随后用 SAC 算法对多视角预测值进行加权以生成最终预测.

SAC-MWF 的整体架构如图 3 所示, 它由 3 个部分组成: ① 特征序列构建, 通过数学变换基于历史窗口生成多条特征序列. ② 多视角预测模型构建, 基于历史窗口和多条特征序列训练一组预测模型生成多视角预测值. ③ 多视角预测模型集成, 利用 SAC 算法对多视角预测值进行集成以生成最终负载预测值.

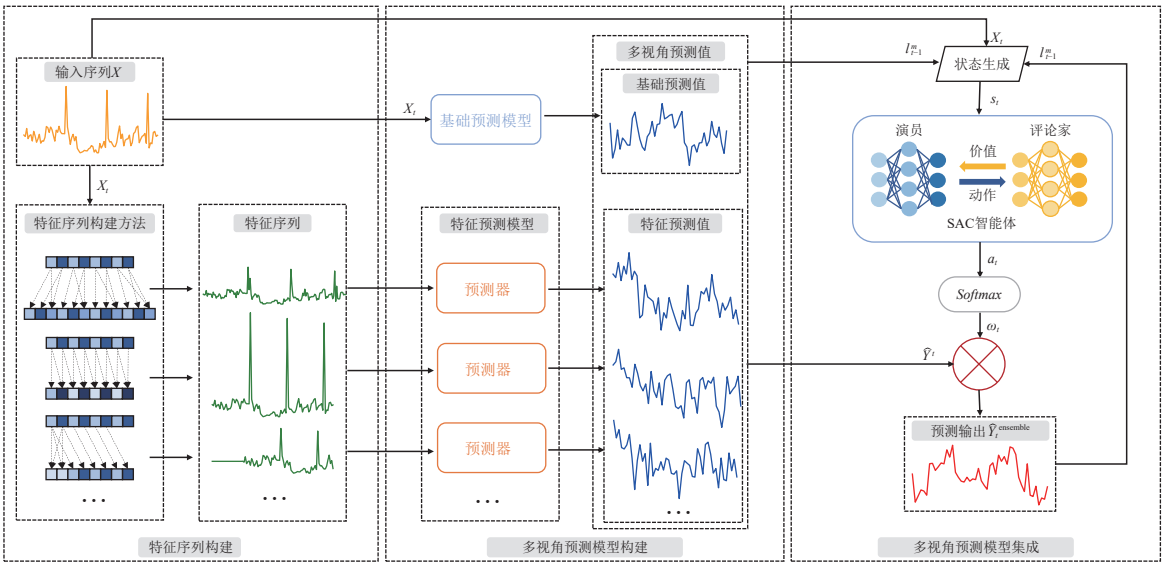


图 3 SAC-MWF 框架设计

3.2.1 特征序列构建

集成模型的性能依赖于多视角预测值的互补性, 而互补性源于特征序列对原始负载中不同模式的显式增强. 通过对比图 4 中的真实值和原始序列预测值可以观察到, 原始序列预测值相较于真实值存在预测峰值幅度不足 (紫色虚线框)、增减趋势相反 (橙色虚线框) 和预测值偏移 (蓝色虚线框) 的问题. 针对上述现象, SAC-MWF 采用一组特征序列构建方法来生成若干特征序列, 以生成和原始序列预测值具有强互补性的特征序列预测值. 本文通过 5 种数学变换构建特征序列, 有针对性地引导模型关注特定维度的模式.

(1) 横向扩展

原始序列存在局部波动频繁的问题, 导致模型在预测时难以准确捕捉部分趋势变化的发生时刻和幅度, 从而出现增减趋势相反现象. 针对该问题, 本文对原始序列进行扩展处理, 通过在原始序列中插入均值点的方式增强原始序列的局部变化信息, 帮助模型更准确地预测趋势变化的发生时刻与幅度. 图 5 展示了原始序列预测值与 MLP 模型在应用插值方法后的扩展序列上的预测值对比. 可以看出, 处理后序列的预测值在增减趋势上的预测优于原始序列预测值.

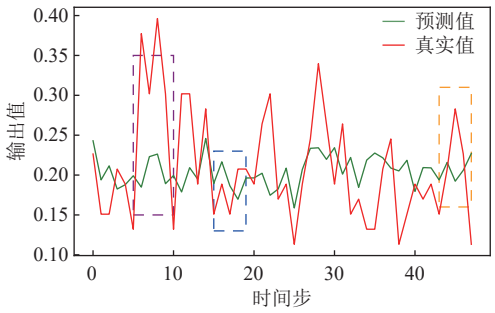


图 4 预测峰值幅度不足、增减趋势相反和预测值偏移

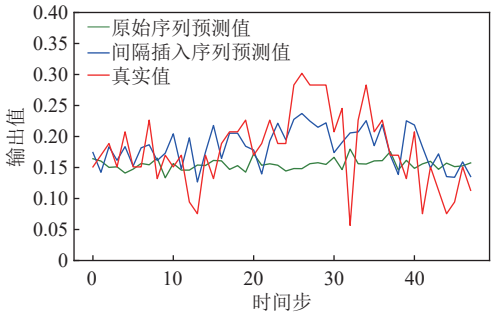


图 5 间隔插入扩展序列与原始序列预测结果对比

将该序列构建方法称为横向扩展, 其核心目的是通过延展输入序列的方式, 增强原始序列中的局部依赖信息. 图 6 展示了横向拓展序列构建的过程. 设输入历史窗口为 $X=\{x_1,x_2,\ldots,x_T\}$, 插入间隔为 k, 则扩展后的序列长度 T_{new} 为:

$$T_{\text{new}}=T+\left\lfloor\frac{T-1}{k}\right\rfloor \tag{3}$$

定义扩展后的序列为 $X_{\text{HE}}=\{y_1,y_2,\ldots,y_{T_{\text{new}}}\}$, 其中 y_i 的值为:

$$y_i=\begin{cases}x_j, & i=j+\left\lfloor\dfrac{j-1}{k}\right\rfloor, j=1,2,\ldots,T\\ \dfrac{1}{k}\displaystyle\sum_{m=(j-1)k+1}^{jk}x_m, & i=j+\dfrac{j-1}{k}, j=1,2,\ldots,\left\lfloor\dfrac{T-1}{k}\right\rfloor\end{cases} \tag{4}$$

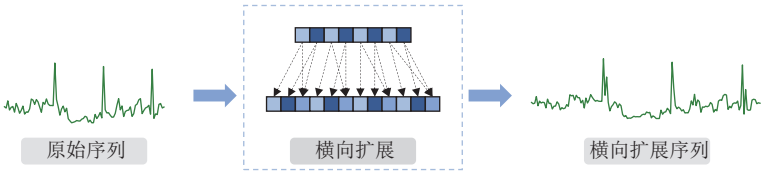


图 6 横向扩展序列构建

(2) 横向收缩

原始序列中包含大量的细粒度波动信息, 这些信息可能源于正常的序列波动, 也可能由噪声引起, 而后者会干扰模型对整体趋势的判断. 为此, 本文设计方法引导模型捕捉序列的整体变化趋势, 避免过度拟合局部波动.

将该序列构建方法称为横向收缩, 方法通过最大值下采样压缩序列长度, 突出原始序列中与趋势变化相关的关键信息, 减少模型对细粒度波动的关注. 横向收缩序列构建的过程如图 7 所示. 设输入历史窗口为 $X=\{x_1,x_2,\ldots,x_T\}$, 分组大小为 g, 完整分组数量 $N=\lfloor T/g\rfloor$, 则每个完整分组序列为:

$$G_i=\{x_{(i-1)g+1},x_{(i-1)g+2},\ldots,x_{ig}\},\ i=1,2,\ldots,N \tag{5}$$

对于每个完整分组, 取组内最大值:

$$M_i=\max(G_i) \tag{6}$$

对于剩余序列 G_{rem}, 同样取序列内最大值:

$$M_{\text{rem}}=\max(G_{\text{rem}}) \tag{7}$$

最终得到横向收缩序列 X_{HC} 为:

$$X_{\text{HC}}=\{M_1,M_2,\ldots,M_N,M_{\text{rem}}\} \tag{8}$$

(3) 纵向扩展

原始序列中的趋势变化可能幅度较小, 导致模型在预测时难以准确识别和响应这些变化, 从而出现预测峰值幅度不足问题. 从图 5 中可以看出, 虽然扩展序列预测值在预测序列增减变化趋势上表现良好, 但对于增减幅度的

预测仍存在不足. 为进一步帮助模型学习原始序列中的增长和下降模式, 本文对原始序列中的各点进行缩放以增强趋势变化的显著性. 图 8 展示了原始序列预测值与 MLP 模型在峰值增强序列上的预测结果对比. 可以看出, 峰值增强序列预测值在峰值部分的预测表现优于原始序列预测值.

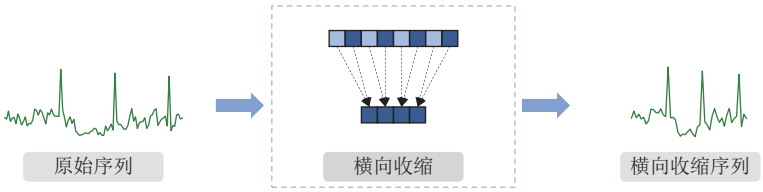


图 7　横向收缩序列构建

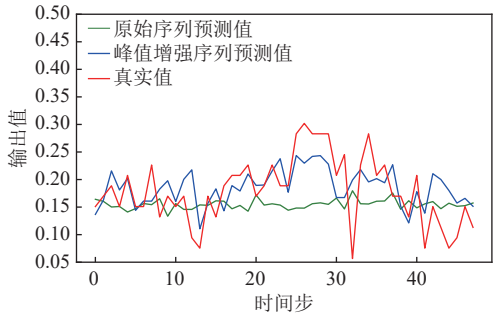


图 8　峰值增强序列与原始序列预测结果对比

将该序列构建方法称为纵向扩展, 方法依据序列极差和单步变化幅度逐点扩大序列趋势变化幅度. 具体而言, 通过计算相邻点的差值绝对值与历史窗口极差的比值确定每个点的缩放因子. 随后, 对于增长趋势点, 对其真实值进行加倍以增大其值, 而对于下降趋势点, 对其真实值应用除法以缩小其值, 从而实现扩大趋势变化幅度的目的. 图 9 展示了纵向扩展序列构建的过程.

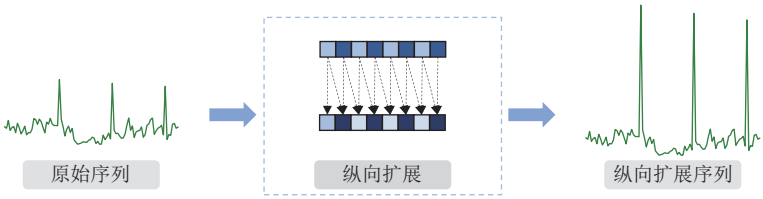


图 9　纵向扩展序列构建

设输入历史窗口为 $X=\{x_1,x_2,\ldots,x_T\}$, 纵向扩缩因子为 m, 计算输入序列极差 R:

$$R=\begin{cases}\max(X)-\min(X), & \max(X)\neq\min(X)\\ \varepsilon, & \max(X)=\min(X)\end{cases} \tag{9}$$

其中, ε 表示极小常数 (如 10^{-6}), 用于处理极差为 0 的序列. 定义相邻点之间的差值序列 D:

$$D=\{d_1,d_2,\ldots,d_{T-1}\},\ d_i=x_{i+1}-x_i,\ i=1,2,\ldots,T-1 \tag{10}$$

计算每个差值与极差 R 的比值得到缩放序列 N, 其中每个缩放比例 n_i 为:

$$n_i=\frac{|d_i|}{R},\ i=1,2,\ldots,T-1 \tag{11}$$

将缩放比例 n_i 转化为不同的放大因子等级 b_i:

$$b_i=\begin{cases}1, & n_i<0.3\\ 2, & 0.3\leqslant n_i<0.6\\ 3, & n_i\geqslant 0.6\end{cases} \tag{12}$$

根据 b_i 计算扩张因子序列 S, 其中每个扩张因子 s_i 为:

$$s_i = (1+m)^{b_i},\ i = 1,2,\ldots,T-1 \tag{13}$$

依据差值的正负确定纵向扩展序列 X_{VE} 中的元素:

$$X_{\mathrm{VE}} = \{x'_1, x'_2, \ldots, x'_T\} \tag{14}$$

$$x'_i = \begin{cases} x_{i+1} \cdot s_i, & d_i > 0 \\ x_{i+1} / s_i, & d_i \leqslant 0 \end{cases},\ i = 1,2,\ldots,T-1 \tag{15}$$

(4) 纵向收缩

纵向扩展能通过增强趋势变化幅度以帮助模型学习峰值部分的序列模式, 但对于学习图 8 中 t=10 附近区间的低峰值部分效果有限. 为提升模型捕获低峰值序列模式的能力, 本文设计方法对原始序列中的高峰值部分进行削弱而保留变化幅度较低的波动信息, 以促使模型关注低峰值模式. 图 10 展示了原始序列预测值与 MLP 模型在削减峰值信息后的序列上的预测结果对比. 可以看出, 峰值削减序列预测值在较低峰值部分的预测准确度优于原始序列预测值.

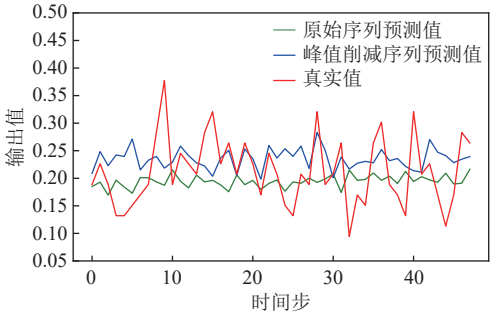


图 10 峰值削减序列与原始序列预测结果对比

将该序列构建方法称为纵向收缩, 方法依据序列的极差和单步变化幅度逐点削减序列的趋势变化幅度. 其整体设计思路和纵向扩展一致, 不同点在于计算扩张因子 s_i 时采取相反操作. 图 11 展示了纵向收缩序列构建的过程. 设输入历史窗口为 $X = \{x_1, x_2, \ldots, x_T\}$, 纵向扩缩因子为 m, 采用公式 (9)–公式 (11) 计算变量极差 R、差值序列 D、缩放序列 N 和放大因子等级 b_i. 随后计算收缩因子序列 S, 其中每个收缩因子 s_i 为:

$$s_i = (1-m)^{n_i},\ i = 1,2,\ldots,T-1 \tag{16}$$

最终以差值正负确定纵向收缩序列 X_{VC}, 序列中元素计算方式与公式 (15) 一致.

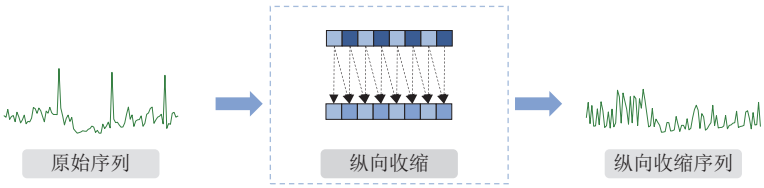


图 11 纵向收缩序列构建

(5) 横向平移

除峰值预测幅度不足、增减趋势相反外, 预测值偏移也同样影响预测准确度. 从图 8 中 t 取值为 [20, 25] 的区间可以观察到, 虽然峰值增强序列预测值在增减趋势上与真实值接近, 但存在一定滞后偏移, 即预测值的增长出现在真实值序列增长的若干步之后. 针对这一现象, 本文对原始序列进行平移操作, 通过头部填充与尾部截断构建横向平移序列, 使模型更早感知趋势变化的初始信号. 图 12 展示了原始序列预测值与 MLP 模型在平移序列上的预测结果对比. 可以看出, 在 t 取值为 [6, 12]、[15, 18]、[35, 40] 区间内, 平移序列预测值相较于图 8 中峰值增强序列预测值的滞后情况有所改善, 平移序列预测值与真实值中峰值所处时间步更加一致.

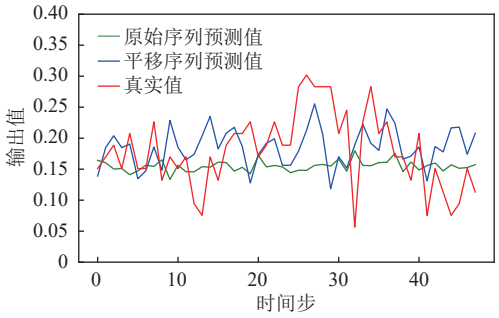


图 12　平移序列与原始序列预测结果对比

将该序列构建方法称为横向平移, 方法通过删除原始序列尾部的部分元素, 并将序列头部元素的均值添加到头部来构建与原始序列等长的新序列. 图 13 展示了横向平移序列构建的过程. 设输入历史窗口为 $X=\{x_1,x_2,\ldots,x_T\}$, 平移比例为 p, 按比例取出头序列 X_h 和尾序列 X_t:

$$X_h=\{x_1,x_2,\ldots,x_N\} \tag{17}$$

$$X_t=\{x_{T-N+1},x_{T-N+2},\ldots,x_T\} \tag{18}$$

对头序列, 计算其均值并构造一个长度为 N 的均值序列:

$$X_{\text{repeat}}=\{\underbrace{\mu,\mu,\ldots,\mu}_{\text{长度为}N}\} \tag{19}$$

$$\mu=\frac{1}{N}\sum_{i=1}^{N}x_i \tag{20}$$

将均值序列与去除尾部序列的历史窗口拼接, 得到横向平移序列 X_{CR}:

$$X_{\text{CR}}=X_{\text{repeat}}\cup\{x_1,x_2,\ldots,x_{T-N}\} \tag{21}$$

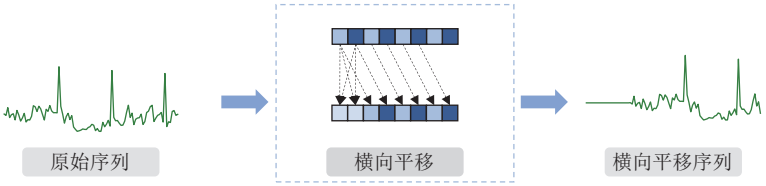


图 13　横向平移序列构建

3.2.2　多视角预测模型构建

特征序列构建完成后, 需要训练一组预测模型以学习原始序列和特征序列模式并作出预测, 为后续集成任务提供基础. 基于原始输入序列训练一个预测模型, 称为基础预测模型. 对于若干特征序列, 在每条序列上分别训练一个预测模型, 称为特征预测模型. 设特征序列总数为 N_{mv}, 多视角预测模型集合表示为 $P=\{predictor_1,predictor_2,\ldots,predictor_{N_{mv}+1}\}$. 对于第 i 个特征预测模型, 其预测输出为:

$$\hat{Y}_t^i=predictor_i(X_i),\ X_i\in\{X_{\text{HE}},X_{\text{HC}},X_{\text{VE}},X_{\text{VC}},X_{\text{CR}}\} \tag{22}$$

之后计算多视角预测模型在各预测步长上的损失:

$$l_t^m=\{l_1^1,l_2^1,\ldots,l_j^i,\ldots,l_H^{N_{mv}+1}\} \tag{23}$$

其中, H 为预测步长, l_j^i 表示第 i 个模型在第 j 个预测步长上的损失. $i=1$ 对应基础预测模型, $i=2,3,\ldots,N_{mv}+1$ 对应特征预测模型. 各模型的预测值将用于计算最终预测值, 而各模型的损失将作为模型集成任务中强化学习智能体状态的一部分.

考虑到特征序列已通过数学规则增强了特定维度的信息, 该处理使得序列中趋势方向、峰值强度等关键特征

被突出, 降低了对预测模型的要求. 因此, 本文选择双层 MLP 作为特征预测模型. MLP 在此类特征明显的序列上可以取得较高预测准确度, 同时其结构轻量, 能有效减少集成模型的资源开销. 这种设计不仅保证了预测的准确性, 还兼顾了模型的计算效率, 适合目标场景.

3.2.3 多视角预测模型集成

不同的多视角预测模型侧重于不同维度的特征信息, 因而各模型能准确预测的区间存在差异. 为使集成模型取得预期的效果, 需要正确融合多视角预测值. 本文采用加权的方式融合多视角预测值, 基于该设计, 可选择平均、线性层、自注意力和强化学习进行集成.

平均是最简单的集成方法, 该方法不需要训练, 且在各模型性能差异较小的情况下能取得一定效果. 但当特征预测值在部分步长上预测误差过大, 或者需要动态调整模型权重时, 该方法会明显欠拟合, 集成效果不稳定. 线性层结构简单, 训练快速, 可以在一定程度上学习多视角预测值到最终预测值之间的非线性关系. 但线性层输出的权重系数难以灵活变化, 导致在动态权重分配任务中表现欠佳.

自注意力能够捕捉不同特征信息之间的关联, 在建模长距离依赖关系时表现出色. 但在目标场景下, 自注意力会试图从多条预测值序列中找出关键信息, 并将其用于生成最终预测结果. 该操作类似于基于多视角预测值进行二次预测而非集成, 因此易受多视角预测值中误差较大项的干扰, 集成效果较差. 后续的消融实验将进一步证明自注意力并不适合目标场景, 且相较于其他集成方法, 自注意力的训练时间和推理时间更长.

相比之下, 强化学习在集成效果和时间开销上能取得更好的平衡. 与平均和线性层相比, 强化学习并不假设所有场景下各模型的权重是固定的, 而是能通过观察环境中的真实工作负载、预测模型误差和预测值等信息来灵活输出组合权重. 与自注意力相比, 强化学习的时间开销更低, 且会严格遵照任务设定, 在每一个预测步长上选择组合多视角预测值的最优权重, 而非进行二次预测.

强化学习可通过在线交互动态优化权重输出策略, 然而其在云负载预测场景中面临两大问题: 1) 高维连续动作空间的探索效率低下, 多步预测需为每个时间步独立分配权重, 容易因维度爆炸导致难以收敛. 2) 历史数据规模较小, 而策略梯度方法 (如 PPO) 需大量交互数据才能稳定训练. 在此背景下, SAC 算法凭借其最大熵优化框架与离线策略训练机制成为理想选择. 首先, SAC 通过引入熵正则化项在策略优化中平衡探索与利用, 避免模型过早收敛至局部最优, 这对负载模式多变场景至关重要. 其次, 其离线策略特性允许从历史交互数据中随机采样进行训练, 能有效提升数据利用效率, 适合目标场景下的小规模数据集. 此外, SAC 的平滑动作分布使其易于处理具有细微差异的动作选择问题, 能减少集成结果突变引起的不稳定. 相较于确定性策略 (如 DDPG), 这一设计在高维动作空间中展现出更强的稳定性.

与一般的强化学习任务类似, 本文通过马尔可夫决策过程来描述多视角预测模型集成任务.

3.2.3.1 动 作

动作空间的设计需兼顾两个需求: 1) 为不同预测步长分配独立权重以适配负载序列的动态特性; 2) 支持动态增减特征预测模型数量. 在 SAC-MWF 中, 动作被定义为每个预测步长上各模型的权重分数, 即未经过 *Softmax* 处理的权重系数. 设特征预测模型数量为 N_{mv}, 预测窗口大小为 H, 则 t 时刻的动作 a_t 对应一个 $(N_{mv}+1)\times H$ 维的连续向量:

$$a_t = \{a_t^{1,1}, a_t^{1,2}, \ldots, a_t^{i,j}, \ldots, a_t^{(N_{mv}+1),H}\} \tag{24}$$

其中, $a_t^{i,j} \in \mathbb{R}$ 表示 t 时刻第 i 个模型在第 j 个预测步长上的权重分数. 通过 *Softmax* 层对每个预测步长的权重分数进行归一化, 生成权重系数 ω:

$$\omega_t^{i,j} = \frac{\exp(a_t^{i,j})}{\sum_{k=1}^{N_{mv}+1} \exp(a_t^{k,j})} \tag{25}$$

该操作确保同一预测步长的所有权重系数满足 $\sum_{i=1}^{N_{mv}+1} \omega_t^{i,j} = 1$ 且 $\omega_t^{i,j} \geqslant 0$.

3.2.3.2 状　态

状态空间需表征实时负载与模型预测行为, 在 SAC-MWF 中, 状态由当前历史窗口与上一时刻各模型预测误差构成:

$$s_t = \{X_t, L_{t-1}\} \tag{26}$$

其中, 历史窗口 $X_t = \{x_{t-T}, x_{t-T+1}, \ldots, x_{t-1}\}$ 提供负载的时序上下文信息, 各模型误差向量 $L_{t-1} = \{l^m_{t-1}, l^e_{t-1}\}$ 编码上一时刻各模型的预测表现, l^m_{t-1} 为多视角预测模型的误差集合, l^e_{t-1} 为集成模型的整体误差.

3.2.3.3 奖励函数

奖励函数的设计需准确量化集成预测性能, 同时平衡短期误差控制与长期稳定性. 基于上述目标, SAC-MWF 采用均方误差构建奖励函数:

$$r_t = 1 - \frac{1}{H}\sum_{j=1}^{H}\left(y_{t+j-1} - \hat{y}^{\text{ensemble}}_{t+j-1}\right)^2 \tag{27}$$

其中, y_{t+j-1} 和 $\hat{y}^{\text{ensemble}}_{t+j-1}$ 分别表示第 j 个预测步长上的真实值与集成预测值, H 为预测窗口大小. 为缓解不同负载量级场景下的奖励尺度差异, SAC-MWF 实验所用数据集均经过归一化处理.

3.2.3.4 状态转移函数

状态转移函数定义了环境在动作 a_t 作用下的变化规律. 在 SAC-MWF 中, 状态由当前历史窗口和上一时刻的预测误差组成. 历史窗口是真实负载序列中的连续值, 因此在任意时刻, 其中的数据都是确定的. 对于预测误差, 当动作 a_t 被确定时, t 时刻的集成模型预测值和预测误差也随之确定. 而多视角预测模型的预测误差在生成多视角预测值时也已经确定. 因此, 对于确定的状态 s_t 和动作 a_t, 只会导致唯一的下一时刻状态 s_{t+1}:

$$s_{t+1} = \{X_{t+1}, L_t\} \tag{28}$$

其中, 下一时刻历史窗口 $X_{t+1} = \{x_{t-T+1}, x_{t-T+2}, \ldots, x_t\}$ 从原始负载序列中获取, 误差向量 $L_t = \{l^m_t, l^e_t\}$ 由 t 时刻各模型的预测结果计算得到. 对于状态转移函数 P, 当且仅当下一时刻状态与公式 (28) 一致时输出概率为 1, 否则为 0:

$$P(s_{t+1} \mid s_t, a_t) = \begin{cases} 1, & s_{t+1} = \{X_{t+1}, L_t\} \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

3.3 总体预测算法设计

在强化学习任务中, 经验回放池存入的内容包括本轮的状态、动作、奖励、下一状态以及是否结束. 但在任务场景下, 由于无法预知下一时刻的历史窗口数据, 也因此无法确定下一时刻的状态. 为解决该问题, 本文将上一时刻的状态、动作、奖励和当前时刻的状态组成经验存入经验回放池. 在训练时, 当每一回合的状态被确定后, 先将经验存入回放池 (算法 1 第 12 行) 再执行后续流程. SAC-MWF 的整体训练流程如算法 1 所示.

算法 1. SAC-MWF 训练流程.

输入: 最大训练轮数: M, 最大序列长度: *Time*, 历史窗口大小: T, 预测窗口大小: H, 归一化负载序列: $X' = \{x'_1, x'_2, \ldots, x'_{Time}\}$, SAC 网络更新频率: *iteration*;

1. 初始化 SAC 策略网络 π_θ、 经验回放池 D、价值网络与目标网络
2. **for** $epoch \leftarrow 1$ to M **do**
3. $L_0 \leftarrow 0$; //初始化上一轮预测误差
4. $s_{t+1} \leftarrow \varnothing$; //初始化上一时刻的状态
5. $a_t \leftarrow \varnothing$; //初始化动作向量
6. $r_t \leftarrow 0$; //初始化奖励
7. **for** $t \leftarrow T$ to $Time - H$ **do**
8. $X'_t \leftarrow \{x'_{t-T}, x'_{t-T+1}, \ldots, x'_{t-1}\}$; //生成历史窗口

$Y'_t \leftarrow \{x'_t, x'_{t+1}, \ldots, x'_{t+H-1}\}$; //生成预测窗口
$l_t^m, \hat{Y}_t^{\text{base}} \leftarrow MultiViewPredict(X'_t, Y'_t)$; //计算多视角模型预测值和损失
$s_t \leftarrow \{X'_t, L_0\}$; //构造当前状态
$D \leftarrow D \cup \{(s_{t-1}, a_t, r_t, s_t, 0)\}$; //存入经验
$a_t \leftarrow \pi_\theta(s_t)$; //选取动作
for $j \leftarrow 0$ to $H-1$ **do**
    $\omega_t^j = Softmax(a_t^j)$;
    $\hat{Y}_t^{\text{ensemble}}[j] \leftarrow \sum_{i=1}^{N_{mv}+1} \omega_t^j[i] \cdot \hat{Y}_t^{\text{base}}[i][j]$; //计算集成预测值
end for
$l_t^e \leftarrow CalcMSELoss(Y_t', \{\hat{Y}_t^{\text{base}}, \hat{Y}_t^{\text{ensemble}}\})$; //计算集成损失
$r_t \leftarrow 1 - l_t^e$;
if t mod $iteration = 0$ **then**
    $UpdateSAC(\cdot)$;
end if
$L_0 \leftarrow \{l_t^m, l_t^e\}$;
$s_{t-1} \leftarrow s_t$;
end for
end for

4 实验评估

本节将介绍实验数据集、对比方法、模型参数设置以及评估指标, 随后将对 SAC-MWF 进行全面评估.

4.1 实验设置

4.1.1 数据集

实验采用两个公开数据集和一个由我们构建的数据集对方法进行评估, 图 14 展示了 3 个数据集的部分工作负载数据. 各数据集详细介绍如下.

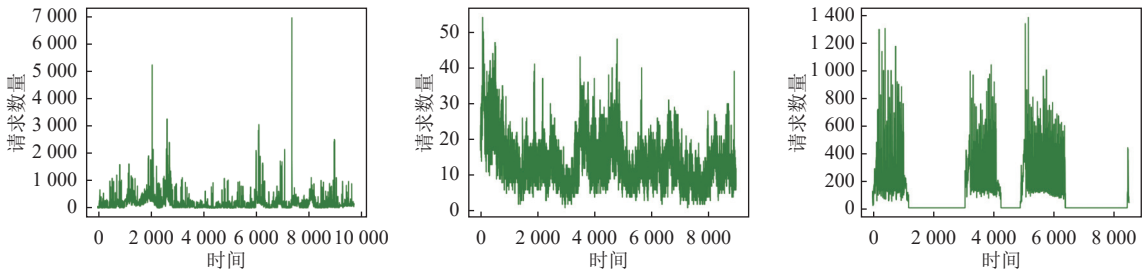


图 14 实验数据集工作负载示例

● AzureFunctionsInvocationTrace2021[33] (以下简称为 Azure2021): 该数据集记录了 Microsoft Azure 中 Azure Functions 的调用数据, 包含自 2021 年 1 月 31 日开始两周的函数调用追踪. 本文以 0.125 s 为间隔聚合数据, 统计出每个时间间隔内应用程序被调用的次数.

● GoogleClusterData2019[34] (以下简称为 Google2019): 该数据集记录了 Google Borg 集群上的工作负载数据,

包含了 2019 年 5 月中 8 个 Borg 单元若干小时的运行时信息. 本文从中筛选出作业相关信息, 去除其中 time 标签为 0 的数据, 以 1 s 为间隔统计用户提交作业请求的数量.

● SockShop: 我们在 Kubernetes 集群中部署开源微服务系统 SockShop, 并使用 Prometheus 收集监控数据. 该数据集包含微服务系统在集群中真实运行约 28 h 的每秒请求数 (requests per second, RPS) 数据. 本文将 RPS 作为负载值用于负载预测任务.

数据集按照 6:2:2 比例划分训练集、验证集与测试集, 并通过归一化将数值映射至 [0, 1] 区间.

4.1.2 对比模型

为全面评估 SAC-MWF 的有效性, 本文选取 9 类代表性时序预测模型作为基线方法, 包含 3 类传统时序预测模型, 3 类 Transformer 架构模型和 3 类新式模型.

● BiLSTM: 一种改进的 LSTM 架构, 通过双向门控结构同时捕获序列的前向与反向依赖关系.

● GRU: 简化 LSTM 结构, 通过更新门与重置门平衡长短期记忆.

● TCN[35]: 基于因果卷积与扩张卷积提取长期时序特征, 通过残差连接增强深层网络稳定性.

● Crossformer[36]: 设计跨尺度注意力机制, 融合不同时间粒度的特征表示, 强化多分辨率时序建模能力.

● Informer[37]: 采用稀疏注意力与概率采样策略优化 Transformer.

● iTransformer[38]: Informer 的改进版本, 通过通道独立性假设与多层特征交互优化注意力机制.

● SCINet[15]: 基于递归奇偶分解与动态卷积操作, 实现多尺度时序特征解耦与重构.

● TimeMixer[16]: 结合趋势分解与多尺度 MLP 架构, 通过通道-时间混合操作捕获非线性时序关联.

● DLinear[17]: 将序列分解为趋势与残差分量, 分别通过线性层建模.

4.1.3 参数设置

我们在 PyTorch 2.1.2 平台上使用 Python 3.10 实现了 SAC-MWF 模型框架与实验环境. 具体参数设置如表 1 所示. 实验共设置 4 组“历史窗口-预测窗口”组合, 分别为 12-4、24-8、48-16 和 96-48.

表 1 模型参数设置

模型	参数
全部模型	learning_rate=0.003, weight_decay=0, 0001, dropout=0.1
SAC	learning_rate=0.001, soft_update_param=0.005, entropy_coefficient=0.1
BiLSTM	hidden_size=256, num_layers=2
GRU	hidden_size=256, num_layers=2
TCN	levels=2, kernel_size=3, hidden_units_per_layer=1
Crossformer	enc_in=1, seg_len=12, win_size=2, nheads=2, d_model=128, d_ff=64, factors=5
Informer	enc_in=1, nheads=2, d_model=128, d_ff=512, factors=5, e_layers=2
iTransformer	nheads=2, d_model=128, d_ff=512, e_layers=2
SCINet	hidden_size=64, num_stacks=2, num_levels=2, kernel_size=3
TimeMixer	d_model=128, d_ff=64, down_sampling_window=3, moving_avg=4
DLinear	moving_avg=5, enc_in=1

4.1.4 评估指标

本文选择均方误差 (*MSE*) 作为评估指标, 其计算公式为:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{30}$$

其中, N 为样本数量, $\hat{y}_i$ 和 y_i 分别对应 i 时刻的预测值与真实值.

4.2 有效性实验

将 SAC-MWF 框架与 9 个基线模型分别组合, 在 3 个工作负载数据集上进行验证. 表 2–表 5 依次展示了 4 个历史窗口-预测窗口组合下, 各模型在应用 SAC-MWF 前后的 *MSE* 对比 (表 2–表 5 中用粗体表示最优结果).

表 2 历史窗口=12, 预测窗口=4 时, 预测损失 *MSE* 对比 (%)

模型	项目	数据集		
		Azure2021	Google2019	SockShop
BiLSTM	对比方法	0.238	0.497	0.325
	SAC-MWF	**0.105**	**0.491**	**0.314**
GRU	对比方法	0.129	0.501	0.364
	SAC-MWF	**0.105**	**0.492**	**0.314**
TCN	对比方法	0.126	0.510	0.361
	SAC-MWF	**0.105**	**0.497**	**0.314**
Crossformer	对比方法	0.189	0.559	0.397
	SAC-MWF	**0.116**	**0.517**	**0.313**
Informer	对比方法	0.155	0.632	0.332
	SAC-MWF	**0.094**	**0.500**	**0.319**
iTransformer	对比方法	0.131	0.516	0.305
	SAC-MWF	**0.104**	**0.509**	**0.304**
SCINet	对比方法	0.139	0.563	0.339
	SAC-MWF	**0.105**	**0.519**	**0.312**
TimeMixer	对比方法	0.130	0.508	0.357
	SAC-MWF	**0.105**	**0.499**	**0.310**
DLinear	对比方法	0.132	0.541	0.357
	SAC-MWF	**0.105**	**0.522**	**0.314**

表 3 历史窗口=24, 预测窗口=8 时, 预测损失 *MSE* 对比 (%)

模型	项目	数据集		
		Azure2021	Google2019	SockShop
BiLSTM	对比方法	0.239	0.522	0.336
	SAC-MWF	**0.170**	**0.475**	**0.313**
GRU	对比方法	0.166	0.524	0.481
	SAC-MWF	**0.152**	**0.478**	**0.336**
TCN	对比方法	0.167	0.556	0.416
	SAC-MWF	**0.147**	**0.536**	**0.319**
Crossformer	对比方法	0.265	0.535	0.414
	SAC-MWF	**0.172**	**0.476**	**0.326**
Informer	对比方法	0.174	0.550	0.322
	SAC-MWF	**0.156**	**0.539**	**0.266**
iTransformer	对比方法	0.177	0.573	0.311
	SAC-MWF	**0.170**	**0.535**	**0.302**
SCINet	对比方法	0.159	0.699	0.368
	SAC-MWF	**0.142**	**0.558**	**0.303**
TimeMixer	对比方法	0.162	0.534	0.370
	SAC-MWF	**0.144**	**0.526**	**0.340**
DLinear	对比方法	0.161	0.534	0.413
	SAC-MWF	**0.146**	**0.479**	**0.218**

表 4 历史窗口=48, 预测窗口=16 时, 预测损失 *MSE* 对比 (%)

模型	项目	数据集		
		Azure2021	Google2019	SockShop
BiLSTM	对比方法	0.238	0.569	0.335
	SAC-MWF	**0.180**	**0.552**	**0.219**
GRU	对比方法	0.183	0.568	0.429
	SAC-MWF	**0.176**	**0.560**	**0.244**
TCN	对比方法	0.192	0.591	0.194
	SAC-MWF	**0.176**	**0.584**	**0.186**
Crossformer	对比方法	0.242	0.662	0.370
	SAC-MWF	**0.181**	**0.593**	**0.220**
Informer	对比方法	0.211	0.591	0.322
	SAC-MWF	**0.185**	**0.562**	**0.212**
iTransformer	对比方法	0.228	0.653	**0.144**
	SAC-MWF	**0.178**	**0.595**	0.185
SCINet	对比方法	0.204	0.666	0.231
	SAC-MWF	**0.176**	**0.597**	**0.184**
TimeMixer	对比方法	0.195	0.579	**0.165**
	SAC-MWF	**0.177**	**0.559**	0.185
DLinear	对比方法	0.193	0.580	0.187
	SAC-MWF	**0.176**	**0.559**	**0.185**

表 5 历史窗口=96, 预测窗口=48 时, 预测损失 *MSE* 对比 (%)

模型	项目	数据集		
		Azure2021	Google2019	SockShop
BiLSTM	对比方法	0.241	0.678	0.299
	SAC-MWF	**0.196**	**0.657**	**0.210**
GRU	对比方法	0.229	0.673	0.328
	SAC-MWF	**0.196**	**0.660**	**0.221**
TCN	对比方法	0.229	0.731	**0.164**
	SAC-MWF	**0.202**	**0.678**	0.181
Crossformer	对比方法	0.231	0.760	0.273
	SAC-MWF	**0.213**	**0.680**	**0.212**
Informer	对比方法	0.228	0.787	0.574
	SAC-MWF	**0.208**	**0.678**	**0.205**
iTransformer	对比方法	0.286	0.771	**0.150**
	SAC-MWF	**0.206**	**0.681**	0.161
SCINet	对比方法	0.223	0.774	0.200
	SAC-MWF	**0.202**	**0.680**	**0.198**
TimeMixer	对比方法	0.247	0.699	**0.156**
	SAC-MWF	**0.197**	**0.685**	0.188
DLinear	对比方法	0.223	0.697	**0.167**
	SAC-MWF	**0.196**	**0.678**	0.194

总体来看, SAC-MWF 能够有效降低基线方法的 *MSE*. 从模型视角来看, SAC-MWF 能为 9 个对比模型带来平均 4.6%–21.7% 的 *MSE* 提升. 提升较为明显的模型包括 Informer、Crossformer 和 BiLSTM, 平均提升幅度均超过 18.6%. 提升中等的模型包括 GRU、SCINet 和 DLinear, 平均提升幅度为 10.5%–14.5%. SAC-MWF 对 TimeMixer、

TCN 和 iTransformer 的提升幅度相对较小, 但同样有 4.6%–7.8%. 从数据集的角度看, SAC-MWF 在 Azure2021 数据集上的效果最明显. 相较于其他数据集, 该数据集的波动较小, 较为稳定. 在 Google2019 数据集上, SAC-MWF 同样显示出一定的提升, 但整体幅度低于 Azure2021. 而在 SockShop 数据集上, SAC-MWF 的 *MSE* 提升幅度波动较大, 在部分实验组中出现了负提升. 当预测窗口较大且数据集波动较剧烈时, 简单的 MLP 模型难以充分捕获特征序列信息, 这会导致部分特征序列预测值误差较大. 且较大预测窗口对应的状态空间与动作空间更大, 会导致学习难度升高, 因而部分场景下应用 SAC-MWF 后误差增加.

4.3 消融实验

4.3.1 特征序列构建方法

为评估各特征序列构建方法的有效性, 本文在 3 个数据集上进行消融实验, 选择 BiLSTM、Informer 和 DLinear 作为基础预测模型, 设置历史窗口为 96, 预测窗口为 48. 实验设计以下实验组合.

- SAC-MWF: 完整的负载预测集成框架, 保留原始输入序列和全部特征序列.
- w/o Origin: 去除原始输入序列, 保留全部特征序列.
- w/o VC: 去除纵向收缩特征序列, 保留原始输入序列和其他特征序列.
- w/o VE: 去除纵向扩展特征序列, 保留原始输入序列和其他特征序列.
- w/o HC: 去除横向收缩特征序列, 保留原始输入序列和其他特征序列.
- w/o HE: 去除横向扩展特征序列, 保留原始输入序列和其他特征序列.
- w/o CR: 去除横向平移特征序列, 保留原始输入序列和其他特征序列.

特征序列构建策略消融实验结果如后文图 15 所示. 可以看出, 完整的 SAC-MWF 在全部实验组合下取得了最低的 *MSE* 误差. 在 Azure2021 数据集上, 各变体的 *MSE* 误差有所增加但相对接近. 在 Google2019 数据集上, 以 Informer 和 DLinear 为基础预测模型的变体的 *MSE* 误差较为接近. 但在以 BiLSTM 为基础预测模型时, w/o HE 变体的 *MSE* 误差增加幅度较大. 在 SockShop 数据集上, 各变体的 *MSE* 相差较大, 且在以 BiLSTM 和 DLinear 为基础预测模型时, w/o HE 变体的 *MSE* 误差明显高于其他变体. 由此可见, 原始序列和各特征序列均有助于提升集成模型预测准确度, 而横向扩展特征序列 (HE) 对于提升模型在 Google2019 和 SockShop 等波动相对剧烈的数据上预测准确度尤其有效.

4.3.2 集成算法

为验证 SAC 算法更适合作为 SAC-MWF 的集成算法, 本文进一步选择了 PPO 算法、自注意力、双层线性层和平均作为对比集成算法. 实验以 BiLSTM、Crossformer、SCINet 作为基础模型, 在全部 3 个数据集上开展实验. 选择 96-48 和 24-8 两组历史窗口-预测窗口设置.

表 6 展示了预测窗口为 48 时不同集成算法的预测误差, 在所有实验组合中, 以 SAC 为集成算法时, 集成模型的 *MSE* 最小. Azure2021 数据集整体波动较小, 集成不同模型的难度相对较低, 因而不同集成算法的预测误差较为接近. 而在 Google2019 数据集和 SockShop 数据集上, 数据波动幅度变大, 集成算法的学习难度也相应增加, 因此不同集成算法之间的预测误差差距增大. 整体来看, 自注意力的集成效果最差, 尤其是在 Google2019-BiLSTM、SockShop-BiLSTM 和 SockShop-Crossformer 这 3 个实验组合上显著高于其他集成算法. 双层线性层在 Azure2021 和 SockShop 数据集上表现优秀, 特别是在 SockShop 数据集上, 其预测误差仅次于 SAC. 平均作为最简单的集成方式表现中等, 既不会因参数学习而出现预测误差过大的问题, 但也无法辨识有用信息与误差以进一步优化集成效果. SAC 和 PPO 作为两种强化学习集成算法在 Azure2021 上的预测误差十分接近且较低. 但是在复杂数据集上, PPO 算法的集成效果有所下降, 仅比表现最差的自注意力机制略好一些, 但明显差于其他集成算法. 因而在目标任务场景下, SAC 算法无论是集成性能还是稳定性均优于其他方法.

表 7 展示了预测窗口大小为 8 时不同集成算法的预测误差. 与预测窗口为 48 时相比, 不同集成算法的预测误差差距有所减小. 虽然 SAC 仍然在所有实验场景中取得了最优预测误差, 但是和次优算法相比差距较小. 整体来看, 自注意力仍是表现最差的集成算法. 线性层和平均在短时预测中的表现中等, 在 Azure2021 数据集上的表现略

有下滑, 但在 Google2019 上的表现有所提升. PPO 算法在该场景下同样有所提升, 特别是在 Google2019 数据集上, 与 SAC 算法的预测误差更加接近. 当预测窗口较小时, 强化学习的动作空间和状态空间维度相对更小, 学习难度更低, 因而 SAC 作为集成算法的优势并不突出.

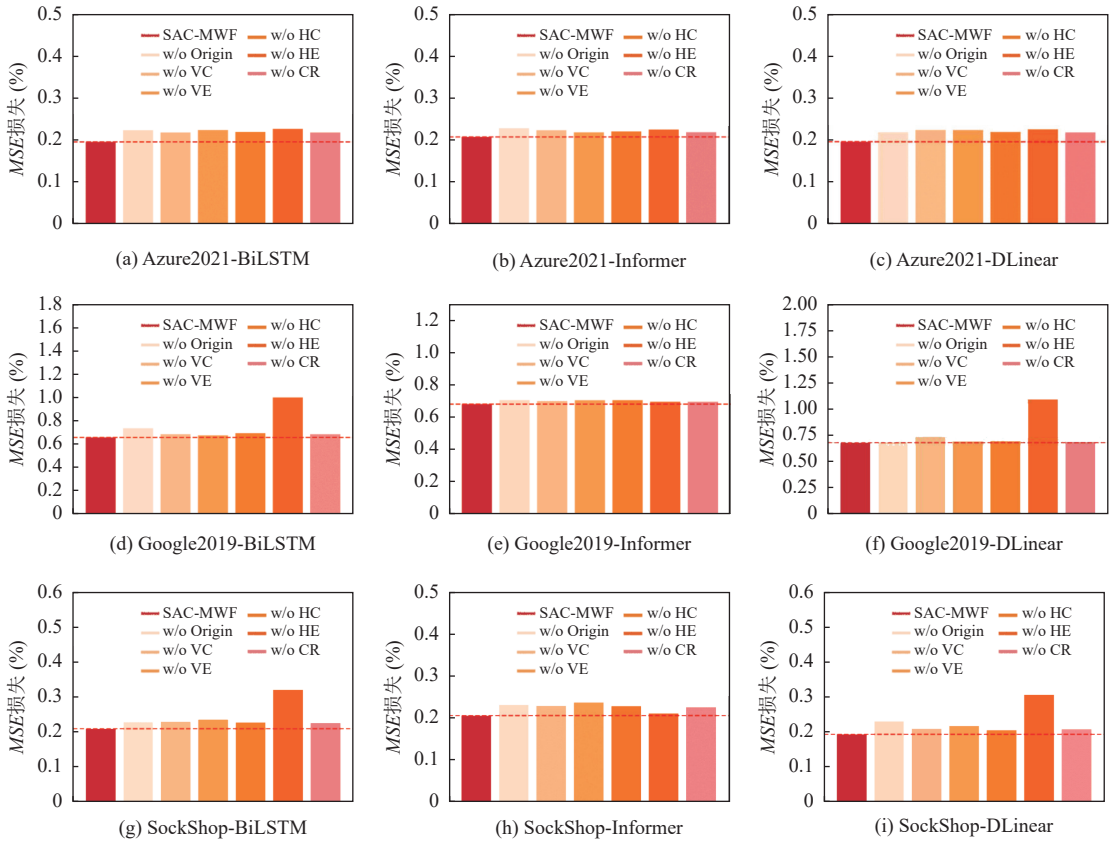


图 15　特征序列构建策略消融实验

表 6　历史窗口=96, 预测窗口=48 时, 不同集成方法下模型损失 *MSE* 对比 (%)

方法	Azure2021			Google2019			SockShop		
	BiLSTM	Crossformer	SCINet	BiLSTM	Crossformer	SCINet	BiLSTM	Crossformer	SCINet
SAC	**0.179**	**0.216**	**0.217**	**0.568**	**0.686**	**0.678**	**0.218**	**0.209**	**0.185**
PPO	0.183	0.234	0.227	0.709	0.705	0.712	0.234	0.228	0.224
自注意力	0.231	0.233	0.263	0.694	0.790	0.726	0.439	0.582	0.342
线性层	0.229	0.225	0.228	0.691	0.698	0.695	0.224	0.215	0.194
平均	0.216	0.222	0.238	0.698	0.699	0.694	0.231	0.227	0.218

表 7　历史窗口=24, 预测窗口=8 时, 不同集成方法下模型损失 *MSE* 对比 (%)

方法	Azure2021			Google2019			SockShop		
	BiLSTM	Crossformer	SCINet	BiLSTM	Crossformer	SCINet	BiLSTM	Crossformer	SCINet
SAC	**0.156**	**0.154**	**0.151**	**0.541**	**0.540**	**0.559**	**0.294**	**0.290**	**0.289**
PPO	0.160	0.162	0.153	0.562	0.547	0.576	0.322	0.321	0.321
自注意力	0.167	0.177	0.178	0.625	0.632	0.645	0.361	0.364	0.362
线性层	0.179	0.179	0.173	0.542	0.539	0.561	0.298	0.312	0.294
平均	0.189	0.183	0.180	0.553	0.559	0.564	0.313	0.308	0.316

4.3.3 *Softmax* 层

为进一步探究应用 *Softmax* 层的影响, 本文选择全部 9 个对比模型在 3 个数据集上开展实验. 设置历史窗口为 96, 预测窗口为 48, 每组实验分别设置使用 *Softmax* 和不使用 *Softmax* 两组并进行对比. 实验结果如图 16 所示. 可以看出, 移除 *Softmax* 层的 SAC-MWF 的预测误差显著增加. 在集成过程中, 应用 *Softmax* 层能将每一预测步长上的最终预测结果严格限制在多视角预测模型的预测值极值之间. 且因为有 *Softmax* 层进行处理, SAC 算法在选择动作时并不需要太过精细, 即便为单一模型分配极高权重, 也不会造成集成结果的剧烈抖动. 而在去除 *Softmax* 层后, SAC 算法需要准确地为每一个模型分配权重系数, 这近乎是使用强化学习完成预测任务, 因此其集成效果明显更差.

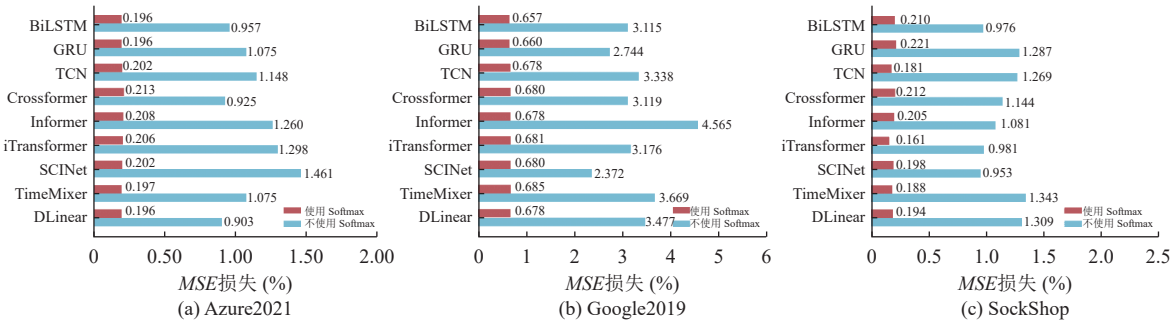


图 16 *Softmax* 层消融实验

4.4 特征序列构建方法超参数实验

为进一步研究特征序列构建中方法超参数设置的影响, 本文以 iTransformer 为基础预测模型在 Google2019 数据集上进行实验. 实验研究的超参数包括: 横向平移比例 p、横向扩展插入间隔 k 和纵向扩缩因子 m.

(1) 横向平移比例 p: 横向平移比例影响横向平移序列中保留的原始序列长度. 随着参数增大, 原始序列周期平移的幅度增大, 原始信息保留减少. 参数 p 最大为 1, 对应将原始序列替换为等长度的均值序列. 实验中设置横向平移比例参数 p 从 [0.1, 1] 区间内以 0.1 为间隔选取.

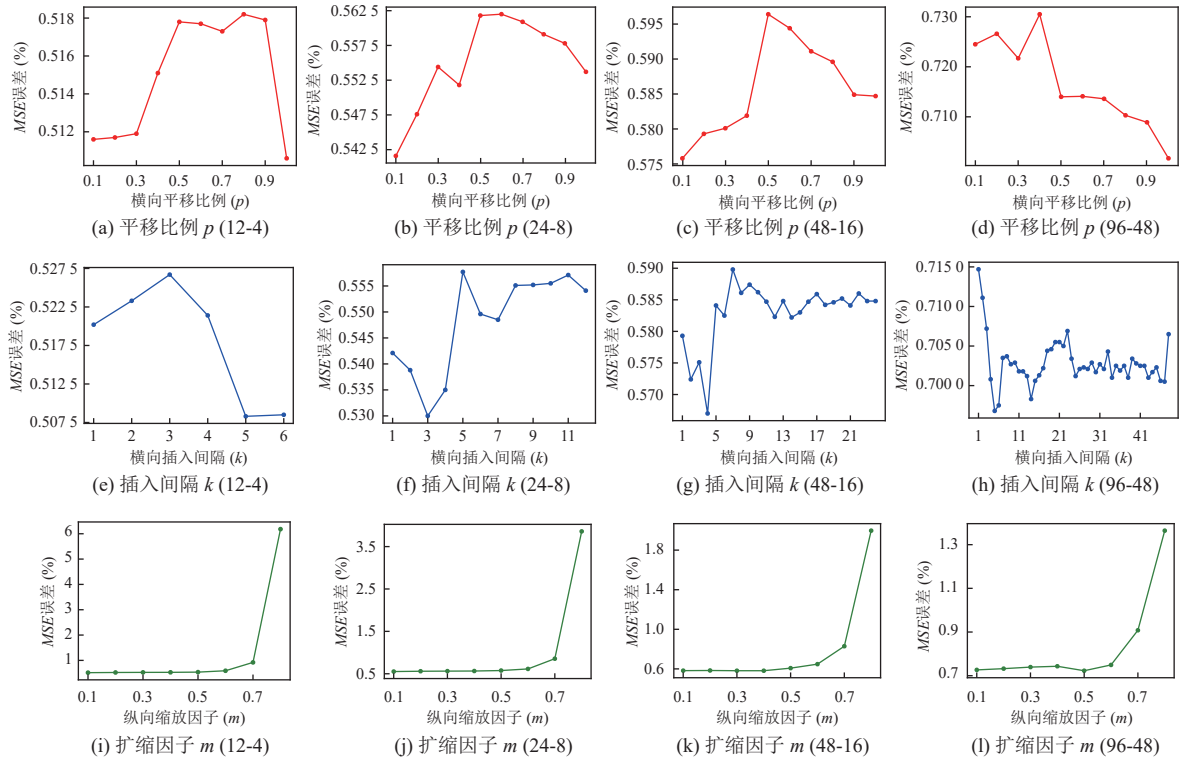
(2) 横向扩展插入间隔 k: 横向扩展插入间隔影响横向扩展序列的最终长度. 随着参数增大, 插入点的数量逐渐减少, 特征序列与原始序列逐渐接近. 为充分验证插入点数量和位置对模型预测准确度的影响, 实验中设置横向扩展插入间隔参数 k 最小为 1, 最大为序列长度的一半, 取值间隔为 1. 即插入点数量最小为 2, 最大为原始序列长度.

(3) 纵向扩缩因子 m: 纵向扩缩因子影响纵向扩展序列和纵向收缩序列中趋势变化幅度的处理力度. 随着参数增大, 纵向扩展序列中的峰值部分逐渐突出, 纵向收缩序列中的峰值削减更加明显. 为避免增强后的峰值数值超出序列上限而影响特征序列预测准确度, 实验中设置纵向扩缩因子 m 从 [0.1, 0.8] 区间内以 0.1 为间隔选择.

实验结果如图 17 所示. 可以看出, 在不同的历史窗口-预测窗口组合下, 各参数对 SAC-MWF 的影响有所不同. 对于横向平移比例 p, 随着参数增大, SAC-MWF 的误差整体呈现先增加后减少的变化趋势. 在 12-4 和 96-48 两组实验中, p 为 1.0 时, SAC-MWF 的预测误差最低, 此时对应使用和原始序列等长的均值序列作为横向平移特征序列. 而在 24-8 和 48-16 两组实验中, SAC-MWF 在 p 为 0.1 时取得最低预测误差. 横向平移的目的是让模型关注预测序列增减趋势变化滞后问题, 当 p 较小时, 输入特征预测模型的原始序列信息较多, 此时特征预测值可以在不显著影响大多预测结果的前提下, 提升部分预测步长上的预测准确度. 而当 p 较大时, 输入的特征序列接近均值序列, 此时的特征预测值能够有效修正基础预测值中增减趋势错误的部分. 上述两种情况下, 横向平移策略均能有效提升模型预测准确度.

对于横向扩展插入间隔 k, 各实验组均有不同的最优参数设置. 在 12-4 和 96-48 实验组中, SAC-MWF 在 k 为 5 时取得最低误差. 在 24-8 实验组中, SAC-MWF 在 k 为 3 时取得最低误差. 而在 48-16 实验组中, k 为 4 时 SAC-MWF 取得最低误差. 整体来看, 除 12-4 实验组外, 随着插入间隔 k 的增大, SAC-MWF 的预测误差呈现先减少后

增加的变化趋势, 且在转折点处 SAC-MWF 的预测误差最低. 当插入间隔 k 较小时, 特征序列相较于原始序列扩展明显, 特征预测模型会在某些预测步长上给出更优预测结果, 但同时也会为其他预测步长引入更大误差. 而当插入间隔 k 较大时, 特征序列与原始序列趋同, 信息增强不明显, 对模型的预测准确度提升有限.



(a) 平移比例 p (12-4) (b) 平移比例 p (24-8) (c) 平移比例 p (48-16) (d) 平移比例 p (96-48)

(e) 插入间隔 k (12-4) (f) 插入间隔 k (24-8) (g) 插入间隔 k (48-16) (h) 插入间隔 k (96-48)

(i) 扩缩因子 m (12-4) (j) 扩缩因子 m (24-8) (k) 扩缩因子 m (48-16) (l) 扩缩因子 m (96-48)

图 17 特征序列构建超参数实验

对于纵向扩缩因子 m, 整体来看, SAC-MWF 的预测误差随着参数增大而增加. 在 12-4、24-8 和 48-16 这 3 个实验组中, 当扩缩因子 m 为 0.1 时, SAC-MWF 的预测误差最低. 而在 96-48 实验组中, 当 m 为 0.5 时, SAC-MWF 的预测误差最低. 纵向扩缩采用指数运算的方式增加或削减序列的增减趋势变化幅度. 当参数 m 较大时, 纵向扩缩方法对原始序列的调节显著, 纵向扩展序列波动更加剧烈, 而纵向收缩序列的波动幅度信息则几乎被抹平, 特别是当纵向扩缩因子 m 的取值在 0.7 及以上时, SAC-MWF 的预测误差增加显著. 而在选择合适的参数 m 时, 纵向扩缩方法可以有效引导模型关注预测序列趋势增减幅度, 从而增强模型的预测性能.

4.5 时间开销实验

为充分展现 SAC-MWF 的时间开销, 本文对比了 9 个基线方法在各数据集上应用 SAC-MWF 前后, 每个迭代的平均训练时间和单次推理时间. 设置历史窗口为 96, 预测窗口为 48.

表 8 展示了训练过程中各部分的平均时间开销, 其中基础表示基础预测模型的训练时间, 特征表示全部特征预测模型的训练时间, 集成表示 SAC 智能体的训练时间. 从结果来看, 应用集成学习导致的训练时间增加处在可接受范围内. 对于基础预测模型部分, DLinear、TCN 等轻量模型的训练时间较短, 而基于 Transformer 结构的模型 Crossformer 和 Informer 的平均训练时间较长, SCINet 基于堆叠结构捕获信息, 其训练时间同样偏长. 对于特征序列预测模型, 由于 SAC-MWF 的特征提取采用简单的数学计算, 而预测模型为双层 MLP, 因而其时间开销较低, 全部特征预测模型单次迭代时间总和仅为约 0.13 s. 对于 SAC 集成部分, 其时间开销主要包括基础模型预测、SAC 集成和网络更新等. 结合表 9 中的基础列可以看出, 对于平均推理时间较长的基础模型如 Crossformer、SCINet 等, SAC 算法的训练速度较慢, 而对于 TCN 等推理速度快的模型, SAC 的训练速度也更快. 整体来看, SAC 的单次

迭代时间大部分在 1.2–2.0 s, 虽高于 DLinear 等轻量模型, 但与 Crossformer、SCINet 等高开销模型相比训练时间较短, 因而应用 SAC-MWF 并不会造成训练时间的大幅增加.

表 8　模型每轮的平均训练时间对比 (s)

模型	Azure2021			Google2019			SockShop		
	基础	特征	集成	基础	特征	集成	基础	特征	集成
BiLSTM	2.51		2.02	2.23		1.81	2.10		1.73
GRU	0.81		1.22	0.71		1.16	0.67		1.14
TCN	0.49		1.33	0.41		1.14	0.38		1.03
Informer	3.01		2.08	2.67		1.88	2.49		1.83
Crossformer	5.75	0.15	3.47	5.12	0.13	2.83	4.73	0.13	2.93
iTransformer	0.98		1.63	0.87		1.33	0.82		1.26
DLinear	0.19		1.57	0.17		1.08	0.16		1.33
TimeMixer	1.54		1.65	1.37		1.44	1.27		1.42
SCINet	3.05		2.30	2.70		1.84	2.51		1.82

表 9　各模型平均推理时间对比 (ms)

模型	Azure2021			Google2019			SockShop		
	基础	特征	集成	基础	特征	集成	基础	特征	集成
BiLSTM	13.96		1.68	9.24		2.98	13.48		1.22
GRU	2.99		2.61	2.99		2.99	2.36		2.30
TCN	1.00		3.60	1.00		3.07	1.00		3.09
Informer	11.81		1.56	10.76		2.28	11.27		2.36
Crossformer	17.58	1.38	5.85	19.43	1.64	4.83	17.38	1.32	6.06
iTransformer	2.99		2.61	2.99		2.48	2.99		3.66
DLinear	1.01		1.60	1.00		3.34	1.00		2.47
TimeMixer	6.97		1.62	4.99		2.34	6.31		1.72
SCINet	6.30		2.81	5.99		3.34	7.28		3.37

表 9 展示了 SAC-MWF 中各部分的平均单次推理时间, 其中基础表示各基础预测模型单次推理的时间, 特征表示全部特征预测模型单次推理的累计时间, 集成表示 SAC 算法从接收多视角预测模型预测结果到生成最终预测结果的单次计算耗时. 从结果来看, TCN、DLinear 等轻量模型的推理速度同样较快, 单次推理时间仅为 1 ms. 基于 Transformer 的模型 Crossformer 和 Informer 的单次推理耗时较长, 均超过了 10 ms. SAC-MWF 中全部特征预测模型的单次推理时间总和约为 1.4 ms, 其时间开销与 TCN 等轻量级模型相当. 集成推理部分的单次推理时间为 1.2–6.1 ms, 与 9 个对比模型的推理时间相比处于中等水平, 最坏时长与 SCINet 等中等开销模型相当. 因此, SAC-MWF 在实现预测准确度提升的同时, 其计算效率也处于中高水平.

5 总　结

通过多视角提取序列特征以增强模型信息捕获能力已被证明是一种有效的策略, 基于不同特征序列生成的预测结果之间存在互补性, 而这种互补性可用于构建更加大的预测模型. 然而, 现有方法大多采用固化结构进行特征提取, 很难将不同组件融合使用. 因此, 本文提出了一个基于 SAC 的多视角工作负载预测集成框架 SAC-MWF, 通过低成本特征序列构建与动态权重分配策略, 有效提升了多种基线模型的负载预测准确度. SAC-MWF 通过横向扩展等数学变换生成的多视角特征序列, 有效引导模型关注增减趋势、峰值幅度、滞后等序列信息. 而 SAC 算法的熵正则化策略则为高维数据场景下动态集成提供了良好解决方案. 此外, 框架在计算效率与预测准确度的均衡性上表现优秀, 其轻量化特征预测模型与离线训练机制使其能更好地适应目标场景. 通过在 3 个工作负载数据集上进行实验验证, SAC-MWF 可将不同基线模型的预测误差平均降低 4.6%–21.7%, 而时间开销并没有显著增加.

尽管 SAC-MWF 的有效性得到验证, 但其在目标场景中仍有一定改进空间. 其一, 虽然现有特征序列构建方

法在集成框架内能够实现预期目标, 但在长时预测任务上的效果有限. 因而需要寻找能够生成具备更强互补性的特征序列的构建方法, 以进一步提高 SAC-MWF 的通用性. 其二, 在本文的设计中, 特征预测模型统一使用 MLP. 此举的目的是保证良好预测准确度的同时尽可能降低资源开销. 然而, 在综合考虑预测性能和资源开销时, MLP 不一定是最优模型. 因而可以进一步平衡集成模型的性能提升和运行时开销增加, 以构建出更具性价比的预测模型.

References

[1] Yang ZH, Wang XG, Li RT, Liu YL. HMM-CPM: A cloud instance resource prediction method tracing the workload trends via hidden Markov model. Cluster Computing, 2024, 27(8): 11823–11838. [doi: 10.1007/s10586-024-04580-7]

[2] Gao JC, Wang HY, Shen HY. Task failure prediction in cloud data centers using deep learning. IEEE Trans. on Services Computing, 2022, 15(3): 1411–1422. [doi: 10.1109/TSC.2020.2993728]

[3] Rahmanian AA, Ghobaei-Arani M, Tofighy S. A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. Future Generation Computer Systems, 2018, 79: 54–71. [doi: 10.1016/j.future.2017.09.049]

[4] Hameed A, Khoshkbarforoushha A, Ranjan R, Jayaraman PP, Kolodziej J, Balaji P, Zeadally S, Malluhi QM, Tziritas N, Vishnu A, Khan SU, Zomaya A. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing, 2016, 98(7): 751–774. [doi: 10.1007/s00607-014-0407-8]

[5] Baig SUR, Iqbal W, Berral JL, Carrera D. Adaptive sliding windows for improved estimation of data center resource utilization. Future Generation Computer Systems, 2020, 104: 212–224. [doi: 10.1016/j.future.2019.10.026]

[6] Chen ZY, Hu J, Min GY. Learning-based resource allocation in cloud data center using advantage actor-critic. In: Proc. of the 2019 IEEE Int'l Conf. on Communications. Shanghai: IEEE, 2019. 1–6. [doi: 10.1109/ICC.2019.8761309]

[7] Guo J, Chang ZH, Wang S, Ding HY, Feng YH, Mao L, Bao YG. Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. In: Proc. of the 2019 Int'l Symp. on Quality of Service. Phoenix: ACM, 2019. 39. [doi: 10.1145/3326285.3329074]

[8] Zhang QC, Yang LT, Yan Z, Chen ZK, Li P. An efficient deep learning model to predict cloud workload for industry informatics. IEEE Trans. on Industrial Informatics, 2018, 14(7): 3170–3178. [doi: 10.1109/TII.2018.2808910]

[9] Hsu YF, Matsuda K, Matsuoka M. Self-aware workload forecasting in data center power prediction. In: Proc. of the 18th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGRID). Washington: IEEE, 2018. 321–330. [doi: 10.1109/CCGRID.2018.00047]

[10] Nguyen C, Klein C, Elmroth E. Multivariate LSTM-based location-aware workload prediction for edge data centers. In: Proc. of the 19th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGRID). Larnaca: IEEE, 2019. 341–350. [doi: 10.1109/CCGRID.2019.00048]

[11] Tang XH, Liu QY, Dong YC, Han JZ, Zhang ZY. Fisher: An efficient container load prediction model with deep neural network in clouds. In: Proc. of the 2018 IEEE Int'l Conf. on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). Melbourne: IEEE, 2018. 199–206. [doi: 10.1109/BDCloud.2018.00041]

[12] Chen L, Zhang WW, Ye HM. Accurate workload prediction for edge data centers: Savitzky-Golay filter, CNN and BiLSTM with attention mechanism. Applied Intelligence, 2022, 52(11): 13027–13042. [doi: 10.1007/s10489-021-03110-x]

[13] Patel E, Kushwaha DS. A hybrid CNN-LSTM model for predicting server load in cloud computing. The Journal of Supercomputing, 2022, 78(8): 1–30. [doi: 10.1007/s11227-021-04234-0]

[14] Devi KL, Valli S. Time series-based workload prediction using the statistical hybrid model for the cloud environment. Computing, 2023, 105(2): 353–374. [doi: 10.1007/s00607-022-01129-7]

[15] Liu MH, Zeng AL, Chen MX, Xu ZJ, Lai QX, Ma LN, Xu Q. SCINet: Time series modeling and forecasting with sample convolution and interaction. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 421.

[16] Wang SY, Wu HX, Shi XM, Hu TG, Luo HK, Ma LT, Zhang JY, Zhou J. TimeMixer: Decomposable multiscale mixing for time series forecasting. In: Proc. of the 12th Int'l Conf. on Learning Representations (ICLR 2024). Vienna: OpenReview.net, 2024.

[17] Zeng AL, Chen MX, Zhang L, Xu Q. Are Transformers effective for time series forecasting? In: Proc. of the 37th AAAI Conf. on Artificial Intelligence. Washington: AAAI, 2023. 11121–11128. [doi: 10.1609/aaai.v37i9.26317]

[18] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: OpenReview.net, 2018. 1856–1865.

[19] Yang JQ, Liu CC, Shang YL, Cheng B, Mao ZX, Liu CH, Niu LS, Chen JL. A cost-aware auto-scaling approach using the workload prediction in service clouds. Information Systems Frontiers, 2014, 16(1): 7–18. [doi: 10.1007/s10796-013-9459-0]

[20] Fang W, Lu ZH, Wu J, Cao ZY. RPPS: A novel resource prediction and provisioning scheme in cloud data center. In: Proc. of the 9th IEEE Int'l Conf. on Services Computing (SCC). Honolulu: IEEE, 2012. 609–616. [doi: 10.1109/SCC.2012.47]

[21] Chen ZJ, Zhu YC, Di YQ, Feng SC. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. Computational Intelligence and Neuroscience, 2015, 2015(1): 919805. [doi: 10.1155/2015/919805]

[22] Yang QP, Peng CL, Zhao H, Yu Y, Zhou Y, Wang ZQ, Du SD. A new method based on PSR and EA-GMDH for host load prediction in cloud computing system. The Journal of Supercomputing, 2014, 68(3): 1402–1417. [doi: 10.1007/s11227-014-1097-x]

[23] Kumar AS, Mazumdar S. Forecasting HPC workload using ARMA models and SSA. In: Proc. of the 2016 Int'l Conf. on Information Technology (ICIT). Bhubaneswar: IEEE, 2016. 294–297. [doi: 10.1109/ICIT.2016.065]

[24] Baig SUR, Iqbal W, Berral JL, Erradi A, Carrera D. Adaptive prediction models for data center resources utilization estimation. IEEE Trans. on Network and Service Management, 2019, 16(4): 1681–1693. [doi: 10.1109/TNSM.2019.2932840]

[25] Yazdanian P, Sharifian S. E2LG: A multiscale ensemble of LSTM/GAN deep learning architecture for multistep-ahead cloud workload prediction. The Journal of Supercomputing, 2021, 77(10): 11052–11082. [doi: 10.1007/s11227-021-03723-6]

[26] Gao HH, Huang WQ, Duan YC. The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments: A QoS prediction perspective. ACM Trans. on Internet Technology (TOIT), 2021, 21(1): 6. [doi: 10.1145/3391198]

[27] Ruan L, Bai Y, Li SN, Lv JX, Zhang TY, Xiao LM, Fang HG, Wang CH, Xue YZ. Cloud workload turning points prediction via cloud feature-enhanced deep learning. IEEE Trans. on Cloud Computing, 2023, 11(2): 1719–1732. [doi: 10.1109/TCC.2022.3160228]

[28] Yi K, Zhang Q, Fan W, Wang SJ, Wang PY, He H, Lian DF, An N, Cao LB, Niu ZD. Frequency-domain MLPs are more effective learners in time series forecasting. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2023. 3349.

[29] Zhang TP, Zhang YZ, Cao W, Bian J, Yi XH, Zheng S, Li J. Less is more: Fast multivariate time series forecasting with light sampling-oriented MLP structures. arXiv:2207.01186, 2022.

[30] Nie YQ, Nguyen NH, Sinthong P, Kalagnanam J. A time series is worth 64 words: Long-term forecasting with Transformers. In: Proc. of the 11th Int'l Conf. on Learning Representations (ICLR 2023). Kigali: OpenReview.net, 2023.

[31] Patel E, Kushwaha DS. An integrated deep learning prediction approach for efficient modelling of host load patterns in cloud computing. Journal of Grid Computing, 2023, 21(1): 5. [doi: 10.1007/s10723-022-09639-6]

[32] Saadallah A, Tavakol M, Morik K. An actor-critic ensemble aggregation model for time-series forecasting. In: Proc. of the 37th IEEE Int'l Conf. on Data Engineering (ICDE). Chania: IEEE, 2021. 2255–2260. [doi: 10.1109/ICDE51399.2021.00233]

[33] Zhang YQ, Goiri Í, Chaudhry GI, Fonseca R, Elnikety S, Delimitrou C, Bianchini R. Faster and cheaper serverless computing on harvested resources. In: Proc. of the 28th ACM SIGOPS Symp. on Operating Systems Principles. Koblenz: ACM, 2021. 724–739. [doi: 10.1145/3477132.3483580]

[34] Verma A, Pedrosa L, Korupolu M, Oppenheimer D, Tune E, Wilkes J. Large-scale cluster management at Google with Borg. In: Proc. of the 10th European Conf. on Computer Systems. Bordeaux: ACM, 2015. 18. [doi: 10.1145/2741948.2741964]

[35] Bai SJ, Kolter JZ, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271, 2018.

[36] Zhang YH, Yan JC. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: Proc. of the 11th Int'l Conf. on Learning Representations (ICLR 2023). Kigali: OpenReview.net, 2023.

[37] Zhou HY, Zhang SH, Peng JQ, Zhang S, Li JX, Xiong H, Zhang WC. Informer: Beyond efficient Transformer for long sequence time-series forecasting. In: Proc. of the 2021 AAAI Conf. on Artificial Intelligence. AAAI, 2021. 11106–11115. [doi: 10.1609/aaai.v35i12.17325]

[38] Liu Y, Hu TG, Zhang HR, Wu HX, Wang SY, Ma LT, Long MS. iTransformer: Inverted Transformers are effective for time series forecasting. In: Proc. of the 12th Int'l Conf. on Learning Representations (ICLR 2024). Vienna: OpenReview.net, 2024.

作者简介

曾文瑄, 硕士生, 主要研究领域为云计算, 资源管理.

应时, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为云计算与云服务软件, 机器学习, 人工智能, 复杂软件系统智能化运维管理, 软件工程的形式化理论与方法.

李田港, 博士生, CCF 学生会员, 主要研究领域为云计算, 软件工程, 资源管理, 强化学习.

田相波, 博士生, 主要研究领域为软件工程, 微服务, 智能化运维.

姜宇虹, 博士, 讲师, CCF 专业会员, 主要研究领域为智能运维, 图神经网络, 因果机器学习及应用.

刘虎杰, 主要研究领域为云计算解决方案的设计与实施, 智能运维, IT 服务管理.

郝诗魁, 硕士, 主要研究领域为云平台建设, 云运营管理, 自动化运维.