WIET: Harmonizing Group-aware Model Weighting and Worker Allocation for Ensemble Temporal Prediction MaaS

Binbin Feng[1,2], Shikun He[2,3], Yingxin Wang[2,3], Pengwei Wang[4], Xiang Gao[5], Zhijun Ding[2,3,*]

[1]School of Economics and Management, Tongji University, China
[2]Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, China
[3]School of Computer Science and Technology, Tongji University, China
[4]School of Computer Science and Technology, Donghua University, China
[5]High Performance Computing Facility Research Center, Zhejiang Lab, China
bining@tongji.edu.cn, tyson_he@tongji.edu.cn, 2432135@tongji.edu.cn, wangpengwei@dhu.edu.cn, gaoxiang@zhejianglab.com, dingzj@tongji.edu.cn

Abstract

Ensemble Temporal Prediction Model-as-a-Service (ETP-MaaS) has become crucial in fields like financial modeling and cloud monitoring. Existing solutions fail to co-optimally address a two-fold challenge of dynamic collaboration and heterogeneity, treating models as independent entities and employing simplistic worker allocation rules. However, at the model level, data volatility means that optimal performance requires identifying and weighting constantly shifting subgroups of base models, not just individual ones; at the system level, these model groups must be efficiently mapped to a pool of heterogeneous and dynamically available workers. To this end, we introduce WIET, an efficient ETP-MaaS system that co-optimizes model weighting and worker allocation. For adaptive weighting, WIET identifies evolving group behaviors among base models and propose a novel group temporal locality-enhanced weighting method. Additionally, WIET develops an efficient, multi-dimensional worker allocation method powered by hybrid heuristic optimization, effectively reducing bottlenecks and resource waste. Experiments show WIET consistently outperforms state-of-the-art methods in terms of accuracy, latency, and resource usage across various workloads and tasks.

1 Introduction

Ensemble Temporal Prediction Model-as-a-Service (ETP-MaaS) is widely applied in various scenarios, including financial modeling (Sun et al. 2023), climate prediction (Bailie et al. 2024), traffic management (Yang et al. 2022b), and workload prediction (Saxena et al. 2023). It combines predictions from multiple base models to capture complex patterns in data streams, enhancing accuracy and robustness while reducing bias and variance (Sakib, Mustajab, and Alam 2025). Therefore, ETP-MaaS is an effective solution for predicting dynamic changes in complex systems.

Some MaaS studies (Zhang et al. 2025a; Gan, Wan, and Philip 2023; Sun et al. 2022) focus on optimizing single models, either enhancing accuracy by increasing model depth or parameter size to capture complex data dependencies, or reducing computational cost via quantizing network parameters or pruning model architectures. However, these approaches overlook the complexities of ETP-MaaS.

While ensemble models usually outperform single models (de Carvalho, de Oliveira, and Roberta de Andrade 2025; Li et al. 2023; Ding, Feng, and Jiang 2022), they add operational complexities. Real-world ETP-MaaS systems face a key challenge ignored by current methods: holistic adaptation to dynamics at model and worker levels. On the one hand, evolving data patterns shift optimal model groups constantly; for instance, in cloud workload prediction, different model combinations are needed during weekdays, holidays, and sudden events (Feng, Ding, and Jiang 2023; Kim et al. 2022). On the other hand, these models run on heterogeneous, dynamically available workers; for instance, to balance performance and cost, compute-intensive models and lightweight models need to be distributed simultaneously across workers (He, Feng, and Ding 2024; Wang et al. 2023).

However, existing solutions fail to co-optimize for these intertwined dynamics. Specifically,

(1) Suboptimal weight distribution due to ignored model group dynamics. Existing methods follow two main paradigms: First, heuristic-rule-based static or semi-static strategies (e.g., sliding window averaging, error-inverse weighting) are simple but struggle to capture the non-linear evolution of base model errors, leading to bias accumulation. Second, although machine learning-based weighting offers some advancements, they are typically trained offline on historical data, where treating all error samples equally. This "one-size-fits-all" approach ignores differences between recent and past samples, overlooking constant shifts in optimal base model combinations and limiting ensemble potential.

(2) Inefficient worker allocation due to ignored heterogeneity and collaboration across workers. Existing methods primarily suffer from two limitations: First, homogeneous or experience-based worker allocation disregards fundamental differences in model requirements and the long-term impact of initial configurations, reducing service cost-effectiveness and exacerbating cold starts. Second, the isolated optimization of individual model-worker pairs creates

*Corresponding author.

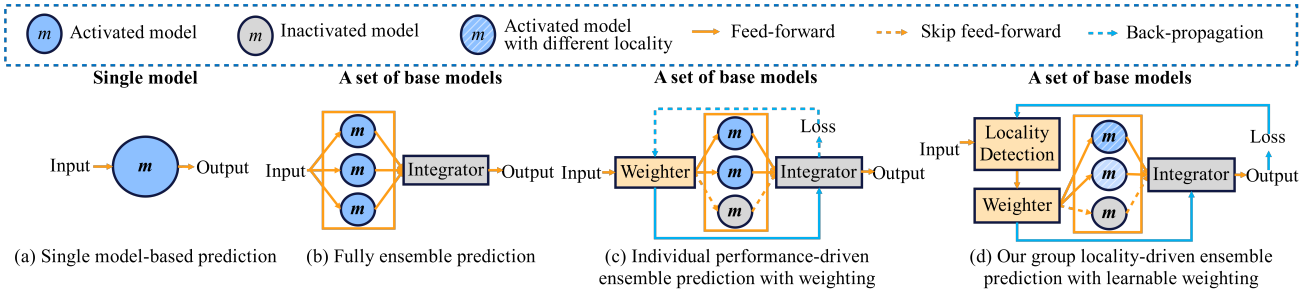


Figure 1: The development process of ETP-MaaS

systemic inefficiencies, leading to frequent performance bottlenecks and resource fragmentation in ETP-MaaS.

To this end, we propose WIET, an ETP-MaaS system that co-optimizes model weighting and worker allocation to handle time-varying dynamics. Our main contributions are:

1. We propose a novel group-aware weighting method that shifts the focus from individual models to dynamic model coalitions. Instead of just reacting to past errors, it learns from the local collaborative performance of model groups. It then builds a multi-label classification weighting model to proactively assign weights, anticipating which coalition will perform best for the near future.
2. We present an efficient multi-dimensional worker allocation method that achieves near-optimal matching across heterogeneous workers. It builds a resource configuration tree with a min-heap, enabling rapid adjustments across workers to meet performance needs. It then employs a hybrid heuristic algorithm for worker-server mappings to jointly optimize end-to-end latency and resources.
3. Extensive experiments across multiple datasets show that WIET improves the average absolute error by over 15.6%, while decreasing resource consumption by 0.94-3.29 cores and 0.38-1.90 GB, and response time by 2.47-195.62 ms, compared to state-of-the-art methods.

2 Related Work

2.1 Weight Distribution

Figure 1 shows the evolution of ETP-MaaS. Monolithic single-model architecture (Fig. 1(a)) is lightweight but limited in performance and generalization (Zhang et al. 2025a; Gan, Wan, and Philip 2023; Sun et al. 2022). Fully-ensemble architecture (Fig. 1(b)) enhances robustness but incurring high latency and variance (Zhang et al. 2025b; Xie et al. 2023; Dogan et al. 2022). Further, individual performance-driven ensemble architecture (Fig. 1(c)) is explored (Sarkar et al. 2024; Li et al. 2023; Kim et al. 2022; Ding, Feng, and Jiang 2022). Hard selection activates a subset based on short-term performance, overlooking detailed model collaborations. Soft weighting employs error-inverse weighting or weighting functions that are built offline, ignoring time-varying group correlations and resulting in suboptimal results. Therefore, we propose an ensemble architecture with learnable weighting driven by group temporal locality (Fig. 1(d)), dynamically adjusting model combinations via online feedback to address time-varying ensemble effects.

2.2 Worker Allocation

Serverless architecture, a neoteric cloud service model, is gaining popularity for MaaS deployment due to its elastic scaling and granular cost models. It dynamically allocates workers based on traffic and supports the "scale-to-zero". However, existing worker orchestration platforms for MaaS, such as Kubernetes (The Kubernetes Authors 2014), Knative (The Knative Authors 2025), and OpenFaaS (Ellis and Authors 2025), employ simple, homogeneous worker allocation strategies that assign uniform specifications to all workers or manual limits, thereby overlooking workload heterogeneity and reducing cost-effectiveness. While studies (Predoaia and García-López 2024; Yu et al. 2024; Mampage, Karunasekera, and Buyya 2021) believe auto-scaling reduces the initial resource allocation impact, suboptimal configurations remain costly and exacerbate cold starts. Other approaches (He, Feng, and Ding 2024; Feng, Ding, and Jiang 2024; Rahimi Jafari et al. 2024; Yang et al. 2022a) optimize allocation by modeling worker resource-performance functions and using heuristics, but neglect collaborative relationships and bottlenecks in base models and integrators, leading to suboptimal latency and resource costs.

3 Proposed Approach

3.1 Problem Definition

Formally, the input to WIET consists of (1) a set of base models $M = \{m_1, m_2, \ldots, m_N\}$ and an integrator m_0, (2) a data request stream $D_t = \{d_1, d_2, \ldots, d_i, \ldots\}$ arriving at timestamp t, and (3) a cluster with n servers $S = \{s_1, s_2, s_3, \ldots, s_n\}$. WIET processes real-time data requests, with N base models and n servers. Its core is to dynamically determine two key solutions:

Weight distribution scheme A: It defines the contribution ratio w_t^i of each base model m_i to the final prediction at timestamp t, adjusting weights based on data stream and model performance. A weight distribution model is generated at each short interval T_1 (typically 5 minute-level (Kim et al. 2022)) governing the weight distribution for all $t \in T_1$:

$$A_{T_1} = \{w_t^1, w_t^2, \ldots, w_t^N\} \quad \text{s.t.} \sum_{i=1}^{N} w_t^i = 1, \forall t \in T_1 \quad (1)$$

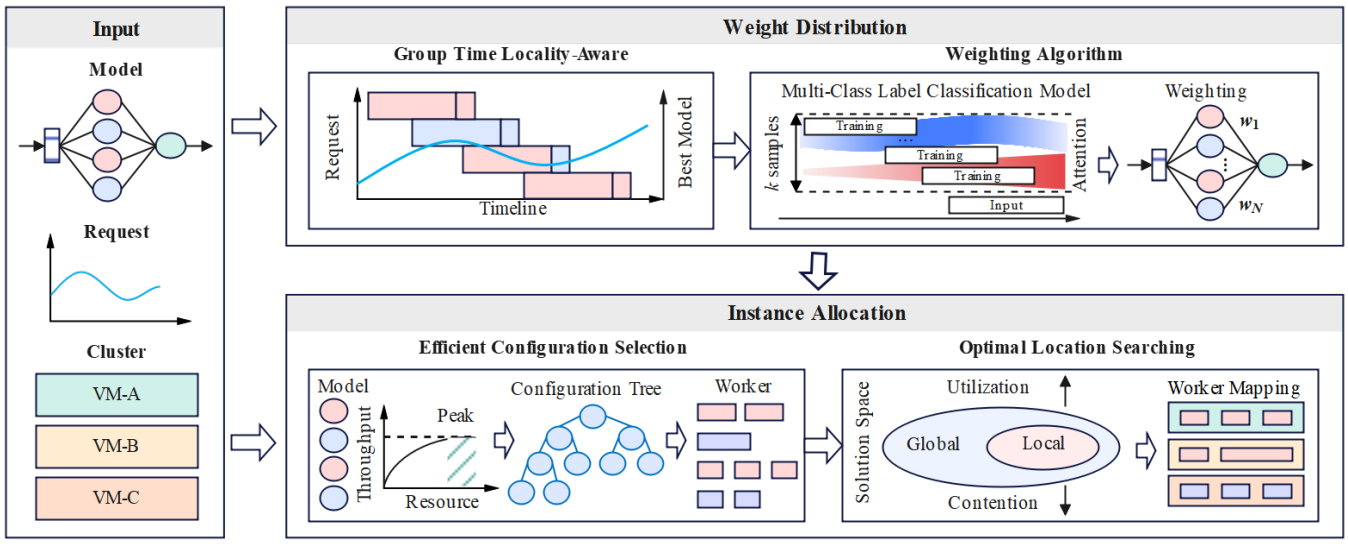


Figure 2: Overall algorithm framework

Worker allocation scheme B: It defines the worker configuration $c^i_{T_2}$ and deployed location $l^i_{T_2}$ of each base model m_i and integrator m_0 in cluster S. Worker allocation decision updates at a relatively long interval T_2 due to physical resource scheduling and deployment (typically 10 minute-level (Feng, Ding, and Jiang 2024)):

$$B_{T_2} = \{(c^0_{T_2}, l^0_{T_2}), (c^1_{T_2}, l^1_{T_2}), \ldots, (c^N_{T_2}, l^N_{T_2})\} \quad (2)$$

WIET aims to improve prediction accuracy by optimizing weight distribution at T_1 while minimizing resource cost and response time by optimizing worker allocation at T_2.

3.2 Framework Overview

Figure 2 shows WIET's overall structure. It begins by initializing model components, data requests, and computing clusters. At time interval T_1, for weight distribution, WIET analyzes the time-varying performance heterogeneity and collaboration characteristics among base models, identifying the key group temporal locality patterns through absolute error. It then constructs a classification weighting model enhanced by group temporal locality, enabling the online adjustment of all base models' weights. At time interval T_2, WIET monitors worker metrics (such as CPU, memory, throughput) to characterize resource heterogeneity and builds a component-level resource configuration tree with a min-heap. It then efficiently determines worker deployment locations using a hybrid heuristic strategy. The detailed optimization process will be discussed in subsequent chapters.

3.3 Weight Distribution for ETP-MaaS

Multi-class label Classification Weighting We design a weighting model wf that periodically outputs the contribution weights of all base models. It treats the problem of "evaluating the performance of N base models" as a multi-class label classification problem, where the output probabilities represent the likelihood of each model contributing effectively to the current observation.

PM_{t-1} is the performance matrix of model set M at timestamp $t-1$, with size $N \times o$, used to train and test wf. It contains the absolute errors from N models across all timestamps (default $o = 10$) in the time window $[t-o, t-1]$, one basis model per row.

$$PM_{t-1} = \begin{bmatrix} p^1_{t-o} & \cdots & p^1_{t-2} & p^1_{t-1} \\ p^2_{t-o} & \cdots & p^2_{t-2} & p^2_{t-1} \\ \vdots & \ddots & \vdots & \vdots \\ p^N_{t-o} & \cdots & p^N_{t-2} & p^N_{t-1} \end{bmatrix} \quad (3)$$

Figure 3 shows the sample distribution with shared input features for training and testing wf. At timestamp t, the training inputs PS_{train}, training labels Y_{train}, and prediction input PS_{pred} are defined as:

$$\begin{aligned} PS_{train} &= \{PM_{t-k-1}, PM_{t-k}, \ldots, PM_{t-2}\} \\ Y_{train} &= \{Y_{t-k-1}, Y_{t-k}, \ldots, Y_{t-2}\} \\ PS_{pred} &= \{PM_{t-1}\} \end{aligned} \quad (4)$$

where training samples are from k consecutive timestamps (defaulting to 30), label $Y_{t-1} = [y^1_{t-1} y^2_{t-1} \ldots y^N_{t-1}]$ is an N-dimensional binary vector (each element y^i_{t-1} is set to 1 for inclusion, 0 for exclusion in the optimal subset), defaulting to selecting the top $N/2$ performing base models.

The trained wf outputs probabilities $w^i_t \in [0, 1]$, indicating the likelihood that each base model m_i is better for the observation PS_{pred}. A higher w^i_t suggests better performance for the near future. By default, the random forest (RF) classification (Breiman 2001) is used as the base function for wf; as an ensemble learning method, it combines multiple decision trees to improve classification accuracy, especially in multiple class label scenarios.

$$\hat{Y}_t = \begin{bmatrix} w^1_t & w^2_t & \ldots & w^N_t \end{bmatrix} \quad (5)$$

Group Temporal Locality Enhancement Training wf uniformly across all samples in PS_{train} is suboptimal, as it neglects temporal effects. The temporal prediction task faces data drift (You et al. 2021), evolving patterns (Feng, Ding, and Jiang 2023), and environmental noise (Yan et al. 2024), resulting in varying base model performance, which reflects their dynamic preferences. Therefore, we analyze sample importance for wf training by examining the performance heterogeneity and collaboration among base models over time.

(1) Performance characterization and ranking: For k samples at k timestamps, we extract rankings for each base model m_i from performance matrices PS_{train}, quantifying temporal variations in performance to identify recent group behaviors. To prioritize data relevant for future predictions, we focus on the $\frac{k}{2}$ timestamps closest to prediction timestamp t, resulting in a $N \times \frac{k}{2}$ ranking matrix RM. Using RM, we calculate an average ranking r_i for each m_i.

$$r_i = \frac{2}{k} \sum_{j=1}^{k/2} RM[i, j] \tag{6}$$

(2) Performance heterogeneity and collaboration analysis: Building on the above comprehensive rankings r_i, we now divide the base model set M into better (M_{A}) and worse (M_{B}) subsets with m defaulting to $N/2$:

$$\begin{aligned} M_{\text{A}} &= \{i \mid r_i \text{ ranks among the top } m\} \\ M_{\text{B}} &= \{i \mid r_i \text{ does not rank among the top } m\} \end{aligned} \tag{7}$$

To quantify the performance gap among base models at each timestamp t, we calculate the average ranking difference $\Delta(t)$ between M_{A} and M_{B}:

$$\Delta(t) = \frac{1}{N-m} \sum_{i \in M_{\text{B}}} RM[i, t] - \frac{1}{m} \sum_{i \in M_{\text{A}}} RM[i, t] \tag{8}$$

The sequence $\{\Delta(t)\}$ corresponding to k timestamps provides insights into how the performance gap evolves, which is key to understanding collaboration and heterogeneity. On the one hand, we calculate the slope of $\{\Delta(t)\}$; on the other hand, we calculate the stability of $\{\Delta(t)\}$:

$$sl = \frac{\sum_{t=1}^{k/2} (t - \bar{t})\big(\Delta(t) - \bar{\Delta}\big)}{\sum_{t=1}^{k/2} (t - \bar{t})^2}, st = 1 - \frac{\text{std}(\Delta(t))}{\text{mean}(\Delta(t))} \tag{9}$$

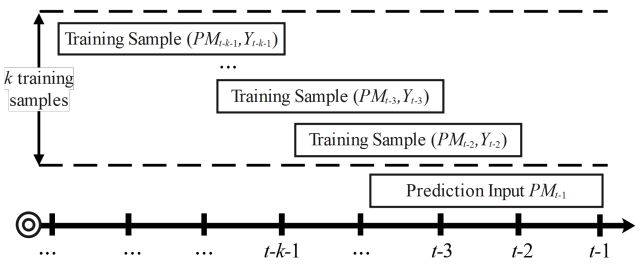


Figure 3: Sample distribution of wf

where std() denotes standard deviation and mean() denotes the mean. A positive sl suggests an increasing performance gap, and a higher st value (closer to 1) indicates more stable and consistent performance differences.

(3) Group temporal locality-enhanced weighting:

Definition 1 (Group Temporal Locality (GTL) in ETP–MaaS)**.** In temporal prediction, if a dominant subset M^* of base model set M consistently shows better performance compared to others in the recent past, M^* is likely to maintain better performance in the near future.

When the GTL condition holds, training samples closer to prediction timestamp t are more likely to repeat data patterns in the near future, making them more important for building wf. GTL may arise in two cases: (1) a significant increasing trend ($sl \gg 0$), showing the gap between M_{A} and M_{B} is widening; (2) a stable advantage ($|sl| \approx 0$ and $st \approx 1$), indicating a stable, significant recent gap. Parameters are user-configurable (default (1) $sl > 0.5$; (2) $|sl| < 0.1, st > 0.9$).

Finally, wf integrates group temporal locality. If GTL is absent (no recent subset dominance), all PV_{train} samples are trained uniformly; if GTL is present (recent subset dominance), samples in PV_{train} are weighted by the Sigmoid function (Ren et al. 2007), which is monotonically bounded in [0,1], smoothly differentiable to avoid discontinuities, and S-shaped to prioritize samples closer to t.

3.4 Worker Allocation for ETP-MaaS

Figure 4 shows the process of worker allocation at T_2.

Efficient Configuration Tree with Min-Heap

Definition 2 (Throughput Bottleneck Effect)**.** Two scenarios limit ETP-MaaS's overall throughput: (1) Base Model Bottleneck: If base model $m_i \in M$ has much lower throughput than others, and integrator m_0 synchronously depends on M'outputs, overall performance is limited by m_i:

$$P(E) \leq \min(P_{m_1}, P_{m_2}, ..., P_{m_N}) \tag{10}$$

(2) Integrator Bottleneck: If m_0 has much lower throughput than M'output speed, overall performance is limited by m_0:

$$P(E) \leq P_{m_0} \tag{11}$$

The bottleneck effect exists if either condition is met.

To address it, steps ① to ⑤ in Fig. 4 determine resource configuration for M and m_0. Supported by the Serverless Knative framework, resource configuration for each component involves worker count and resource capacity. Options include: (1) worker count $C_{m_i}[0]$ can be adjusted in the integer range, defaulting to $0 - 10$ with a step of 1; (2) worker resource capacity $C_{m_i}[1]$ can be adjusted from $(0.1 \text{ core}, 0.2 \text{ GB})$ to $(1 \text{ core}, 2 \text{ GB})$, with a step of $(0.1 \text{ core}, 0.2 \text{ GB})$. To efficiently determine resource configuration, we use a min-heap to build a resource configuration tree for ETP-MaaS, leveraging the heap property's alignment with the bottleneck effect. Algorithm 1 shows the detailed process. As a complete binary tree, the min-heap guarantees the root is the global minimum (Chen, Sheu, and Kuo 2023), enabling: (1) locating the throughput bottleneck component in $O(1)$ time; and (2) restoring the heap order in $O(\log N)$ time after adjustments. Therefore, it reduces complexity and enables efficient resource optimization.

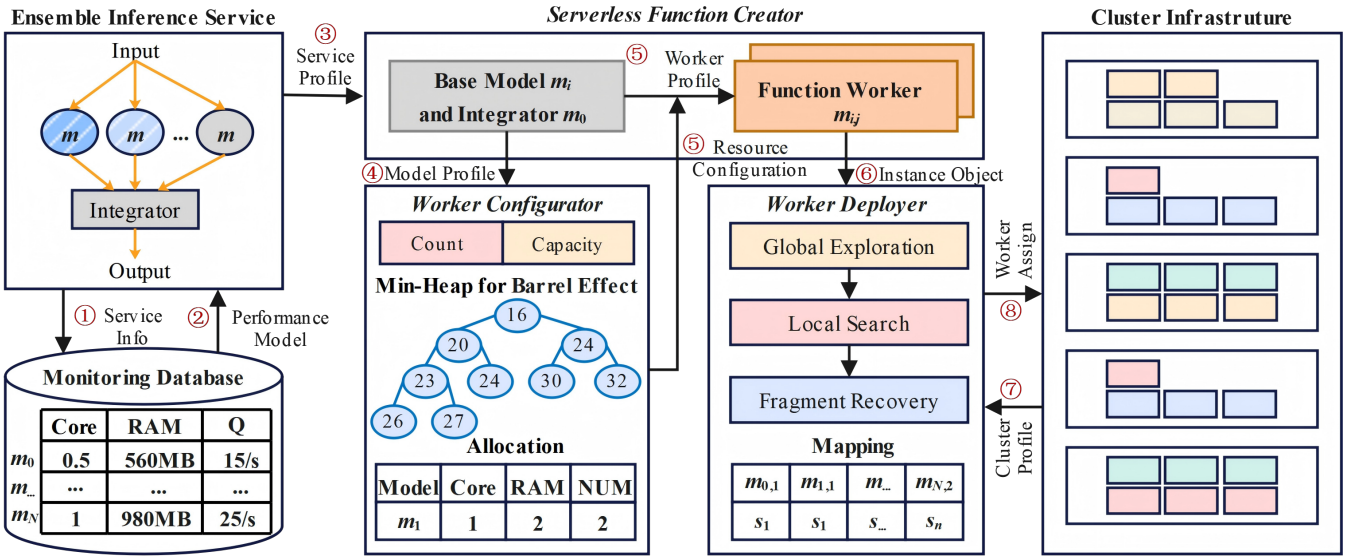


Figure 4: Efficient worker allocation process for ETP-MaaS

Worker Mapping with Hybrid Heuristic Optimization To avoid competition and waste of resources, we map model workers with multi-dimensional resources to servers. We define binary variables: $x_{i,j,l}$ (worker $m_{i,j}$ on server s_l), and y_l (server s_l activation). Key parameters include server resource capacity C_{s_l}, server utilization u_l, and worker count $|m_i|$. The objective function is to maximize cluster resource utilization while minimizing worker contention.

$$\begin{aligned}&\max\{\text{Util}(X) + \text{NoActive}(X) - \text{Penalty}(X)\} = \\ &\sum_{l=1}^{n} y_l \cdot \frac{\sum_{i=0}^{N}\sum_{j=1}^{|m_i|} x_{i,j,l} \cdot C_{m_i}[1]}{C_{s_l}} + \sum_{l=1}^{n}(1-y_l) \\ &- \sum_{l=1}^{n} y_l \cdot [\max(0, 0.2 - u_l) + \max(0, u_l - 0.9)]\end{aligned} \tag{12}$$

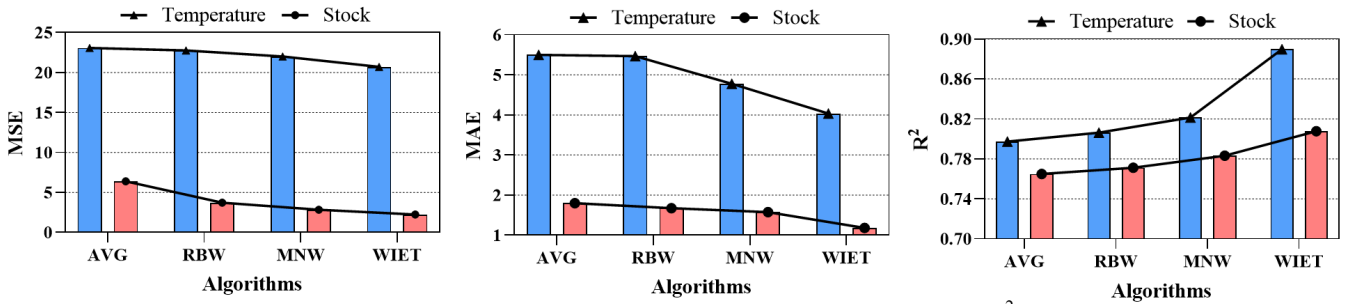
The three normalized terms indicate: active servers' utilization, inactive server count, and load balancing penalty. Hard constraints include: (1) each worker is allocated to one server, $\sum_{l=1}^{n} x_{i,j,l} = 1$; (2) workers only assigned to active servers, $x_{i,j,l} \leq y_l, \forall i, j, l$; (3) server capacity limits, $\sum_{i,j} x_{i,j,l} \cdot C_{m_i}[1] \leq C(s_l), \forall l$.

To address this, we designed a quantum-inspired simulated annealing hybrid optimization framework (steps ⑥–⑧ in Fig. 4), combining global exploration and local search. The quantum-inspired part initially generates a population covering deployment schemes, with decision variables $x_{i,j,l}$ totaling $\sum_{i=0}^{N} |m_i| \times n$. Quantum rotation guides the population toward high-fitness regions, while interference enhances solutions, enabling broad exploration. The annealing layer then refines elite solutions via neighborhood searches using the Metropolis criterion, preventing entrapment in local optima. Neighborhood operations include: (1) migrating worker $m_{i,j}$ between servers s_l and $s_{l'}$ for load balancing, within resource constraints; (2) merging workers on low-load servers to improve NoActive(X). Heuristic stopping criteria is fitness convergence or max iterations$= 100$.

After mapping, the fragmentation-recycling step computes each server's remaining resources $R(s_l) = C(s_l) - \sum_{i,j} x_{i,j,l} \cdot C(m_i)[1]$. It then incrementally allocates resource units to the lowest-throughput worker up to the available boundary, thus approaching Pareto optimality.

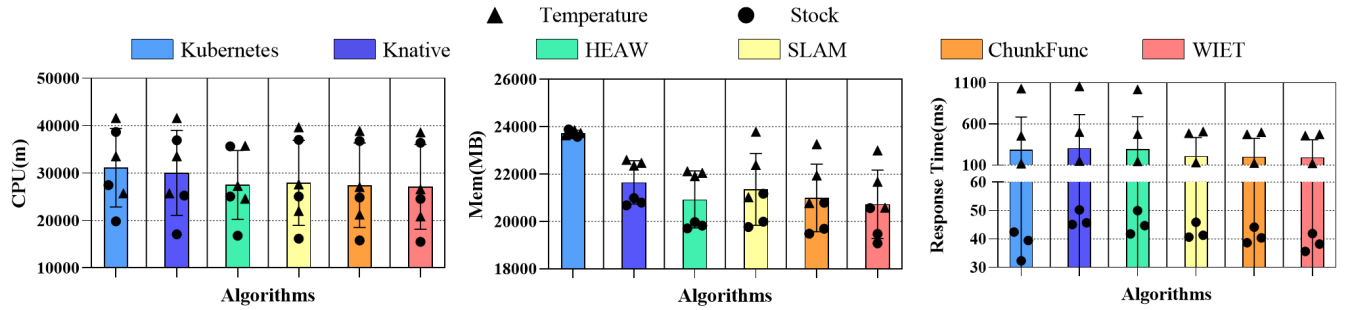
Algorithm 1: Efficient configuration based on min-heap

Require: Number of base models N, target throughput Q, performance functions $P_{m_{1,\ldots,N}}$ and P_{m_0}
Ensure: Configurations $C_{m_{1,\ldots,N}}$ and C_{m_0}
for $i = 1$ to N **do**
  Initialize $C_{m_i} \leftarrow (1, (0.1\text{ core}, 0.2\text{ GB}))$
  Insert (P_{m_i}, i) into min-heap H
end for
Initialize $C_{m_0} \leftarrow (1, (0.1\text{ core}, 0.2\text{ GB}))$
while H is not empty **do**
  Get root throughput and index $(P_{min}, i^*) \leftarrow H.extract\_min()$
  if $P_{min} \geq Q$ **and** $P_{m_0} \geq Q$ **then**
    break
  end if
  Attempt to increase resource capacity or worker count, update $C_{m_{i^*}}$ and $P_{m_{i^*}}$ (prioritize capacity adjustment for utilization)
  Insert $(P_{m_{i^*}}, i^*)$ into min-heap H
end while
Adjust C_{m_0} such that $P_{m_0} \geq \max(P_{m_1}, \ldots, P_{m_N})$
if $\min(P_{m_1}, \ldots, P_{m_N}) \geq Q$ **then**
  return $C_{m_{1,\ldots,N}}, C_{m_0}$
else
  return "Cannot achieve target throughput Q"
end if



(a) MSE metric for performance evaluation (b) MAE metric for performance evaluation (c) R^2 metric for performance evaluation

Figure 5: Performance evaluation of weight distribution methods



(a) CPU usage for overhead evaluation (b) Memory usage for overhead evaluation (c) Response time for overhead evaluation

Figure 6: Resource usage and latency evaluation of worker allocation methods with different request arrivals

4 Experiments

4.1 Experiment Setup

Datasets and Environments We utilized widely adopted ensemble temporal models and datasets for temperature prediction (DevCHyderabad 2020) and stock prediction (Akhtar and Khursheed 2019). We set up a cluster with three servers (each with 16-core CPUs and 32GB RAM; Python 3.10, PyTorch 2.5.0, Kubernetes 1.27, Knative 1.12). We tested three load scales (Q=15, 25, 35 RPS) through three rounds of stress tests (lasting 30 minutes and containing 45,000 requests) to evaluate different methods under low, medium, and high loads.

Compared Methods We compared the WIET method with other strategies in two experimental groups: (1) Weight distribution strategies: average weighting (AVG) (Li et al. 2023), error-based linear weighting (RBW) (Savargiv, Masoumi, and Keyvanpour 2022), and multi-class nonlinear weighting (MNW) (Kim et al. 2022), evaluated using Mean Square Error (MSE), Mean Absolute Error (MAE), and coefficient of determination (R^2). (2) Worker allocation strategies: Kubernetes (The Kubernetes Authors 2014), Knative (The Knative Authors 2025), heterogeneous-aware serverless worker balancing (HEAW) (Heidari and Azizi 2023), SLO-aware serverless worker allocation (SLAM) (Safaryan et al. 2022), and Bayesian-based serverless worker allocation (ChunkFunc) (Pusztai and Nastic 2025), evaluated using CPU and memory consumption and response time.

4.2 Evaluation Results

Performance improvement of WIET for ETP-MaaS. Figure 5 shows WIET performing the highest among four weighting methods for both prediction tasks. It is due to random forest-based nonlinear weighting and its ability to detect the group temporal locality in the base models' performance. In addition, models perform better in stock prediction with lower errors, while temperature prediction shows larger errors due to fundamental differences in data characteristics. Nonetheless, the WIET enhanced temporal locality mechanism still delivers clear relative improvements by adaptively adjusting weights to align with explicit dynamic characteristics of temperature sequences.

Concerning other methods, AVG performs weakest as its fully integrated approach ignores base model dynamics and complex temporal patterns. RBW offers minor improvement, relying on linear weights from short-term errors, but still neglects long-term performance patterns and nonlinear events. MNW achieves better results by weighting with RF; however, it's suboptimal due to training that fails to account for the time-varying performance pattern of base models and differences in the importance of samples. Compared to MNW, WIET achieved superior results on two datasets, reducing MSE by 6.0% and 21.1%, MAE by 15.6% and 25.2%, and increasing R^2 by 8.3% and 3.1%.

Resource usage and latency improvement of WIET for ETP-MaaS. Figure 6 presents CPU and memory usage, and response time for temperature and stock prediction tasks, with points indicating different load scales Q; larger Q re-

Method	Classifiers	Temperature			Stock		
		MSE	MAE	R^2	MSE	MAE	R^2
MNW(w/o GTL)/WIET	RF	1.064	1.185	0.923	1.268	1.336	0.970
	SVM	1.536	1.363	0.946	1.353	1.400	0.972
	XGBoost	1.076	1.190	0.940	1.236	1.335	0.966

Table 1: Relative accuracy evaluation of MNW and WIET with different base classifiers

Methods	Temperature			Stock		
	Q=15	Q=25	Q=35	Q=15	Q=25	Q=35
WIET(w/o Quantum)	124.73	468.97	487.63	36.53	43.28	39.39
WIET	122.61	460.23	476.33	35.68	41.96	38.22

Table 2: Absolute latency evaluation of WIET(w/o Quantum) and WIET with different request arrivals

sults in higher resource usage and response time. Meanwhile, the temperature prediction task consumes greater resources because it relies on more complex models.

Overall, WIET significantly reduces resource usage while maintaining low response time and consistently demonstrates advantages across various prediction tasks and load scales, showcasing its adaptability to diverse computing demands. It is because WIET efficiently identifies and reduces resource redundancy and bottlenecks using a min-heap-based configuration tree, while achieving balanced worker allocation through hybrid heuristic optimization.

Regarding other methods, Kubernetes offers stable but inflexible scheduling, which makes it susceptible to resource waste and competition under high loads. Knative enhances efficiency with on-demand scaling, but improper worker configuration can result in cold starts and increased response time. Despite optimizations, HEAW, SLAM, and ChunkFunc have limitations: HEAW suffers from resource fragmentation during high concurrency, SLAM often overallocates to meet the efficiency goals, and ChunkFunc's Bayesian optimization is constrained in high-dimensional spaces. Compared to these three methods, WIET reduced CPU usage by 0.94-3.29 cores and memory by 0.38-1.90 GB, with average response time decrease of 2.47-195.62 ms.

4.3 Ablation Study

Effectiveness of WIET with different classifiers. Table 1 shows the performance advantage of WIET with GTL, over MNW across different classifiers. The values represent the ratio of MNW to WIET results, enabling a direct comparison of their relative performance. For both prediction tasks, MSE and MAE ratios exceeded 1, confirming WIET's accuracy improvements across datasets. Additionally, R^2 values for all combinations were close to or higher than those of MNW, indicating that incorporating GTL minimizes errors without sacrificing interpretability. Furthermore, WIET's effects with various base classifiers are not uniform, allowing for tailored selections in specific applications.

Effectiveness of WIET with hybrid heuristics and single heuristics. Figure 7 and Table 2 compare resource usage and response time between hybrid and single heuristic optimizations. Across all scenarios (Q=15, 25, 35 RPS) for both prediction tasks, WIET showed lower resource overhead and

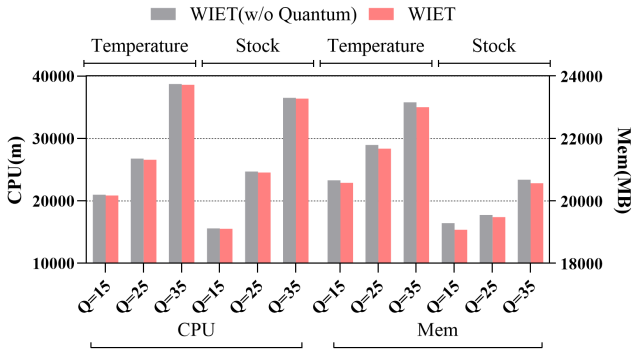


Figure 7: The resource usage evaluation of WIET(w/o Quantum) and WIET with different request arrivals

latency in most cases, as the quantum-heuristic-simulated annealing framework more effectively explores the solution space, optimizing worker allocation to balance utilization and reduce contention. Simultaneously, the experimental results reveal that the algorithm's advantages vary with the load queue length. At low loads (Q = 15 RPS), abundant resources constrain the optimization potential, whereas at high loads (Q = 35 RPS), saturation limits further improvements despite potential gains being available. The phenomenon observed across both datasets highlights the effectiveness of the hybrid heuristic algorithm across different load scales.

5 Conclusion

This paper introduces WIET, an efficient system for ETP-MaaS. It addresses time-varying group performance changes of base models by utilizing temporal locality-based weight distribution for enhanced accuracy. Additionally, it employs hybrid heuristics-based worker allocation to mitigate performance bottlenecks and resource fragmentation. Experiments have shown that WIET improves accuracy, reduces response time, and lowers resource usage compared to state-of-the-art solutions. Future work will expand WIET to support a wider variety of base model types, including both large and small models, and accelerate heterogeneous resources in scenarios involving the mixed use of GPUs and CPUs.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4501700, in part by the National Natural Science Foundation of China under Grant 62372330, and in part by the Shenzhen Science and Technology Plan Project of China under Grant CJGJZD20240729113801003.

References

Akhtar, Z.; and Khursheed, M. O. 2019. Stock market prediction using a hybrid model. In *Symposium on Machine Learning and Metaheuristics Algorithms, and Applications*, 75–89. Springer.

Bailie, T.; Koh, Y. S.; Rampal, N.; and Gibson, P. B. 2024. Quantile-regression-ensemble: A deep learning algorithm for downscaling extreme precipitation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 21914–21922.

Breiman, L. 2001. Random forests. *Machine learning*, 45(1): 5–32.

Chen, C.-M.; Sheu, J.-P.; and Kuo, Y.-C. 2023. Resource allocation with multi-connectivity in 5G heterogeneous networks. In *GLOBECOM 2023-2023 IEEE Global Communications Conference*, 1197–1203. IEEE.

de Carvalho, H. D. P.; de Oliveira, J. F. L.; and Roberta de Andrade, A. F. 2025. Dynamic Selection of Ensemble-Based Regression Models: Systematic Literature Review. *Expert Systems with Applications*, 128429.

DevCHyderabad. 2020. Data Science Repository. https://github.com/DevCHyderabad/Data-Science. Accessed: 2025-04-01.

Ding, Z.; Feng, B.; and Jiang, C. 2022. Coin: a container workload prediction model focusing on common and individual changes in workloads. *IEEE Transactions on Parallel and Distributed Systems*, 33(12): 4738–4751.

Dogan, S.; Barua, P. D.; Kutlu, H.; Baygin, M.; Fujita, H.; Tuncer, T.; and Acharya, U. R. 2022. Automated accurate fire detection system using ensemble pretrained residual network. *Expert Systems with Applications*, 203: 117407.

Ellis, A.; and Authors, O. 2025. OpenFaaS: Serverless Functions Made Simple. https://www.openfaas.com. Accessed: 2025-04-01.

Feng, B.; Ding, Z.; and Jiang, C. 2023. FAST: A forecasting model with adaptive sliding window and time locality integration for dynamic cloud workloads. *IEEE Transactions on Services Computing*, 16(2): 1184–1197.

Feng, B.; Ding, Z.; and Jiang, C. 2024. Heterogeneity-aware proactive elastic resource allocation for serverless applications. *IEEE Transactions on Services Computing*, 17(5): 2473–2487.

Gan, W.; Wan, S.; and Philip, S. Y. 2023. Model-as-a-service (MaaS): A survey. In *2023 IEEE International Conference on Big Data (BigData)*, 4636–4645. IEEE.

He, S.; Feng, B.; and Ding, Z. 2024. Proactive Elastic Scheduling for Serverless Ensemble Inference Services. In *2024 IEEE International Conference on Web Services (ICWS)*, 1025–1035. IEEE.

Heidari, S.; and Azizi, S. 2023. Heterogeneity-aware load balancing in serverless computing environments. In *2023 7th International Conference on Internet of Things and Applications (IoT)*, 1–7. IEEE.

Kim, I. K.; Wang, W.; Qi, Y.; and Humphrey, M. 2022. Forecasting Cloud Application Workloads With CloudInsight for Predictive Resource Management. *IEEE Transactions on Cloud Computing*, 10(3): 1848–1863.

Li, Z.; Ren, K.; Yang, Y.; Jiang, X.; Yang, Y.; and Li, D. 2023. Towards inference efficient deep ensemble learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8711–8719.

Mampage, A.; Karunasekera, S.; and Buyya, R. 2021. Deadline-aware dynamic resource management in serverless computing environments. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 483–492. IEEE.

Predoaia, I.; and García-López, P. 2024. A Cloud-Agnostic Serverless Architecture for Distributed Machine Learning. In *2024 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, 131–140. IEEE.

Pusztai, T.; and Nastic, S. 2025. ChunkFunc: Dynamic SLO-Aware Configuration of Serverless Functions. *IEEE Transactions on Parallel and Distributed Systems*.

Rahimi Jafari, M.; Su, J.; Zhang, Y.; Wang, O.; and Zhang, W. 2024. PISeL: Pipelining DNN Inference for Serverless Computing. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 1951–1960.

Ren, J.; McIsaac, K. A.; Patel, R. V.; and Peters, T. M. 2007. A potential field model using generalized sigmoid functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2): 477–484.

Safaryan, G.; Jindal, A.; Chadha, M.; and Gerndt, M. 2022. SLAM: SLO-aware memory optimization for serverless applications. In *2022 IEEE 15th international conference on cloud computing (CLOUD)*, 30–39. IEEE.

Sakib, M.; Mustajab, S.; and Alam, M. 2025. Ensemble deep learning techniques for time series analysis: a comprehensive review, applications, open issues, challenges, and future directions. *Cluster Computing*, 28(1): 73.

Sarkar, M. R.; Anavatti, S. G.; Dam, T.; Ferdaus, M. M.; Tahtali, M.; Ramasamy, S.; and Pratama, M. 2024. GATE: A guided approach for time series ensemble forecasting. *Expert Systems with Applications*, 235: 121177.

Savargiv, M.; Masoumi, B.; and Keyvanpour, M. R. 2022. A new ensemble learning method based on learning automata. *Journal of Ambient Intelligence and Humanized Computing*, 13(7): 3467–3482.

Saxena, D.; Kumar, J.; Singh, A. K.; and Schmid, S. 2023. Performance analysis of machine learning centered workload prediction models for cloud. *IEEE Transactions on Parallel and Distributed Systems*, 34(4): 1313–1330.

Sun, S.; Wang, X.; Xue, W.; Lou, X.; and An, B. 2023. Mastering stock markets with efficient mixture of diversified trading experts. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2109–2119.

Sun, T.; Shao, Y.; Qian, H.; Huang, X.; and Qiu, X. 2022. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, 20841–20855. PMLR.

The Knative Authors. 2025. Kubernetes-based, scale-to-zero, request-driven compute. https://github.com/knative/serving. Accessed: 2025-04-01.

The Kubernetes Authors. 2014. Kubernetes. https://kubernetes.io/. Accessed: 2025-04-01.

Wang, Y.; Chen, K.; Tan, H.; and Guo, K. 2023. Tabi: An efficient multi-level inference system for large language models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, 233–248.

Xie, Y.; Sun, W.; Ren, M.; Chen, S.; Huang, Z.; and Pan, X. 2023. Stacking ensemble learning models for daily runoff prediction using 1D and 2D CNNs. *Expert Systems with Applications*, 217: 119469.

Yan, K.; Long, C.; Wu, H.; and Wen, Z. 2024. Multi-resolution expansion of analysis in time-frequency domain for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11): 6667–6680.

Yang, Y.; Zhao, L.; Li, Y.; Zhang, H.; Li, J.; Zhao, M.; Chen, X.; and Li, K. 2022a. Infless: a native serverless system for low-latency, high-throughput inference. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 768–781.

Yang, Z.; Sun, H.; Huang, J.; He, L.; Jia, X.; Zhao, J.; and Qiao, S. 2022b. Robust traffic speed inference with Ensemble Learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(10): 17241–17257.

You, X.; Zhang, M.; Ding, D.; Feng, F.; and Huang, Y. 2021. Learning to learn the future: Modeling concept drifts in time series prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2434–2443.

Yu, H.; Li, J.; Hua, Y.; Yuan, X.; and Wang, H. 2024. Cheaper and faster: Distributed deep reinforcement learning with serverless computing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16539–16547.

Zhang, W.; Hu, Y.; Shi, J.; and Bai, X. 2025a. Poplar: Efficient Scaling of Distributed DNN Training on Heterogeneous GPU Clusters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 22587–22595.

Zhang, Z.; Pham, T. D.; An, Y.; Doan, N. P.; Alsharari, M.; Tran, V.-H.; Hoang, A.-T.; Vandierendonck, H.; and Mai, S. T. 2025b. WaveletMixer: a multi-resolution wavelets based MLP-mixer for multivariate long-term time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 22741–22749.