Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 3, No. 8, 2024

A Meta-Learning Framework for Cross-Service Elastic Scaling in Cloud Environments

Tengda Tang
University of Michigan, Ann Arbor, USA
ttengda@umich.edu

Abstract: This paper addresses the challenge of limited generalization and adaptability in elastic scaling strategies for service instances under cloud computing environments. It proposes a meta-learning-based framework for cross-service scaling strategy modeling. The framework integrates a service-aware task construction mechanism with a dual-stage strategy prediction model. By extracting transferable knowledge through multi-task learning, the method enables fast adaptation and high-accuracy prediction for new service scenarios. In the task construction phase, the framework introduces service context representation and structural information. This leads to meta-task partitioning with stronger semantic consistency, improving both learning stability and model generalization. In the strategy prediction phase, a dual-stage model architecture is designed. It combines meta-initialized parameters with local fine-tuning. By fusing global coarse prediction with local refinement, the model generates scaling strategies that balance global knowledge transfer with service-specific modeling. Experiments are conducted on a real-world cloud service dataset. The model is systematically evaluated across multiple dimensions, including accuracy, robustness, and adaptability. Results show that the proposed method outperforms mainstream approaches across key metrics and demonstrates strong transferability. In addition, ablation studies and sensitivity analyses confirm the individual contributions of each module to strategy performance. These findings highlight the effectiveness and practicality of the method in complex service scheduling scenarios.

Keywords: Elastic scaling, meta-learning, service modeling, and policy prediction

1. Introduction

With the rapid development of cloud computing and microservice architecture, dynamic and elastic scaling of service instances has become a key approach to ensuring high availability and efficient resource utilization[1,2,3]. Modern distributed systems are often composed of a large number of microservices. These services need to scale in or out dynamically based on changing workloads to handle traffic peaks and resource pressure at different times. Traditional scaling strategies mostly rely on static thresholds or rule-based engines. While effective in simple scenarios, they often show poor adaptability and delayed responses in complex production environments. Especially in the presence of highly heterogeneous resource profiles and interaction patterns among services, manual tuning and rule design become increasingly impractical. This calls for smarter and more generalizable modeling methods.

In recent years, scaling decisions based on supervised learning have attracted growing interest. These methods train models on historical metrics to predict future workload trends or determine whether scaling is needed. Although they can perform well in specific services, they heavily rely on training data and show limited generalization. When deployment conditions change or a new service is introduced, the model usually needs retraining. This leads to substantial overhead in data collection and manual adjustment, reducing system maintainability and scalability. Therefore, enabling fast model generalization across different services has become a critical challenge in intelligent scaling research[4,5].

Against this backdrop, meta-learning offers a promising solution. Meta-learning focuses on "learning to learn" by extracting shared knowledge structures from multiple tasks. This enables models to adapt to new tasks with very limited data[6]. Applying meta-learning to service scaling strategy modeling can significantly enhance cross-service adaptability. By training on multiple existing services, the model can learn key features that influence scaling behavior. It can then generalize to new services and produce effective scaling decisions with minimal data, reducing tuning costs and improving overall efficiency[7].

Moreover, elastic scaling is not only a matter of resource management. It is closely tied to system performance and user experience. In high-concurrency scenarios, failure to scale out in time may cause delays or request loss. On the other hand, excessive scaling leads to resource waste and increased costs. A generalizable scaling strategy framework must balance responsiveness and precision[8]. It should support diverse service patterns while ensuring robust resource control. This is particularly important in complex environments involving multi-tenancy, cross-cluster deployments, and geographically distributed systems. In such settings, a generalizable modeling framework can greatly enhance automation and intelligent management. Building a cross-service scaling strategy framework based on meta-learning holds significant theoretical

and practical value[9,10]. It addresses the need for transferable learning in intelligent systems and promotes deeper integration between AI and system engineering. At the same time, it introduces a new paradigm for intelligent resource scheduling in large-scale microservice environments. It provides strong support for improving elasticity and resource efficiency in cloud-native architectures. As cloud platforms continue to scale, intelligent control methods with self-adaptive and self-learning capabilities will play an increasingly central role.

2. Related work

2.1 Supervised Learning

Supervised learning methods have been widely used in the study of elastic scaling for service instances due to their strong predictive performance[11]. Typical approaches collect time series data such as resource usage, request volume, and response latency. These data are used to train classifiers or regression models to determine whether to trigger scaling operations. Such methods can effectively extract patterns from historical data[12,13]. Compared with static threshold strategies, they offer better adaptability and generalization. Common models include decision trees, support vector machines, and multilayer perceptrons. These models have shown good prediction accuracy in real systems and are suitable for scenarios with regular load fluctuations[14].

With the progress of deep learning, researchers have introduced more complex model architectures to capture nonlinear patterns in resource usage. For example, long short-term memory (LSTM) networks are widely used to model temporal dependencies in system metrics. They can predict future workloads in advance and enable proactive scaling actions. Convolutional neural networks (CNNs) are also used to capture short-term fluctuation features. In some microservice architectures, hybrid models combining CNN and LSTM have shown promising results. These deep learning-based supervised methods not only improve prediction accuracy but also expand the range of model choices for elastic scheduling[15].

Nevertheless, supervised learning still faces limitations in practical deployment, especially when there are large differences between services. Each service has its own workload pattern, resource consumption profile, and invocation behavior. As a result, traditional supervised models often perform well only in specific services. Their performance drops significantly when applied to new ones. Enhancing the adaptability of models across services and reducing the cost of data collection and retraining for each deployment has become a key research issue. This also raises new challenges for designing more general and efficient strategy modeling methods.

2.2 Meta-Learning

In modeling elastic scaling strategies for service instances, traditional supervised learning methods often show limited generalization when facing service heterogeneity and dynamic environments[16,17,18]. To address this challenge, meta-learning has emerged as a promising paradigm to enhance model transferability. Its core idea is to learn a set of general initialization parameters or meta-knowledge across multiple tasks. This enables the model to adapt quickly to new tasks with only a few samples, significantly reducing training cost and time[19,20].

In elastic scaling tasks, different services may vary significantly in resource usage, workload fluctuations, and dependency structures[21,22,23]. This makes direct model transfer difficult. Meta-learning methods can extract shared features that influence scaling decisions across different services. These features help the model infer effective strategies for new services, improving generality and robustness. For example, with task-level training mechanisms, the model can capture both the behavioral patterns of individual services and the structural similarities among them. This provides a theoretical and methodological foundation for cross-service modeling[24].

In addition, integrating meta-learning with deep learning models offers a new path toward building scalable and transferable elastic management systems[25]. Whether through model-agnostic meta-learning approaches that optimize parameter spaces, or context-based embedding networks that capture dynamic features of service environments, meta-learning demonstrates strong adaptability and generalization. As cloud computing systems grow in complexity, modeling mechanisms with fast generalization capabilities will play an increasingly important role in advancing intelligent elastic scheduling.

This integration not only enhances the learning efficiency of resource management strategies but also enables rapid adaptation to unseen service behaviors with minimal overhead. By decoupling system-specific logic from core learning mechanisms, meta-learning empowers models to generalize across diverse workloads and heterogeneous deployment contexts. Moreover, embedding-based methods allow these systems to continuously incorporate temporal patterns, workload dynamics, and structural dependencies of services, leading to more refined and context-aware decision-making processes[26].

In practical terms, this hybrid approach ensures that elastic management frameworks can maintain high responsiveness under workload fluctuations, while reducing the need for frequent manual tuning or retraining. It also creates a foundation for automating elastic scaling decisions in large-scale, multi-tenant cloud environments where service-level objectives must be maintained with high precision. The resulting systems are more robust, more flexible, and more intelligent in responding to the complex demands of modern distributed infrastructures[27,28].

Ultimately, as the scale and diversity of cloud-native applications continue to expand, meta-learning-infused architectures are expected to become a fundamental component of next-generation elastic management solutions. Their ability to leverage prior knowledge, adapt rapidly, and generalize effectively positions them as a promising direction for both academic research and industry adoption in cloud automation and intelligent system optimization.

3. Method

This study addresses the challenge of generalizable modeling in elastic scaling decisions for service instances. It proposes a meta-learning-based framework for cross-service scaling strategy modeling. The method shows clear advantages in improving the adaptability to heterogeneous services and enhancing the intelligence of scaling decisions. Specifically, the first contribution is a Service-Aware Task Construction Mechanism(SATCM). It builds tasks by incorporating runtime contextual features of services, such as resource configurations and call chain topologies. This makes the meta-learning process more aligned with the dynamic behavior patterns of real services. The second contribution is a Dual-Stage Strategy Prediction Model(DSSPM) for fast adaptation. It designs a policy generation network that combines global initialization with local fine-tuning. This enables quick transfer and high-accuracy prediction with few samples when facing new services. Together, these two innovations form a generalizable, low-cost, and highly responsive modeling system for intelligent scaling strategies. The model architecture is shown in Figure 1.

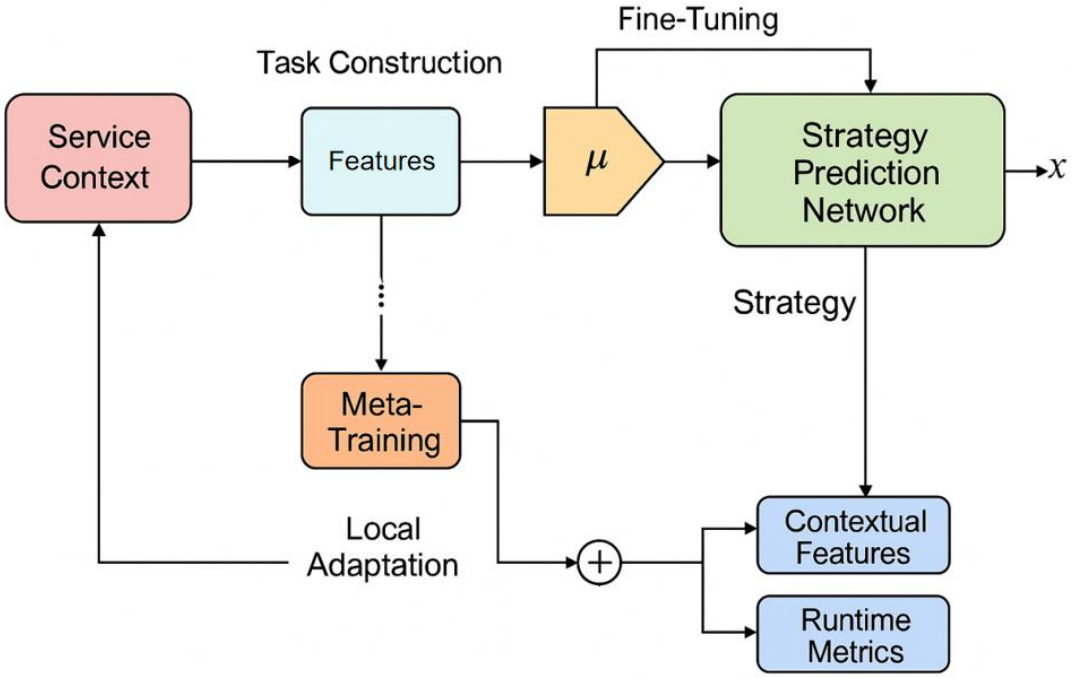

Figure 1. Overall model architecture diagram

3.1 SATCM

In elastic scaling strategy modeling, different service instances have significant heterogeneity in terms of resource usage characteristics, load fluctuation patterns, call chain structures, etc., which makes it difficult for a unified modeling task design to cover all service types, thus affecting the effectiveness and generalization ability of the meta-learning process. To solve this problem, this study proposes a service-aware task construction mechanism (SATCM), which introduces service context semantics for task division and feature extraction, so that each meta-task has stronger service adaptability and policy consistency. Its module architecture is shown in Figure 2.

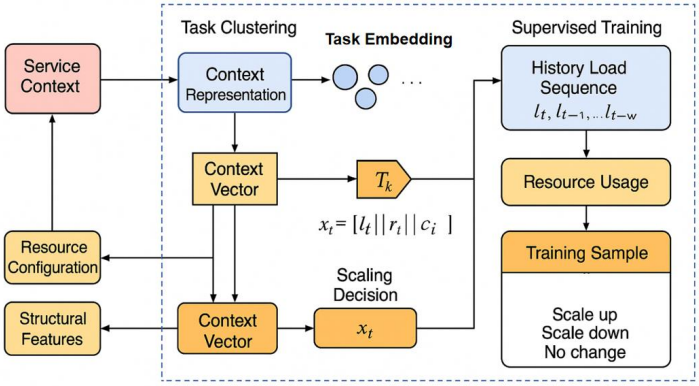

Figure 2. SATCM module architecture

The core idea of SATCM is to abstract the operating environment of each service instance into a context vector c_i, which is encoded by static configuration parameters (such as the number of service replicas and resource limits) and structural features (such as dependency depth and call fan-out).

Suppose there is a service set $S = \{S_1, S_2, ..., S_n\}$, and each service S_i corresponds to a context representation:

$$c_i = f_{enc}(S_i)$$

Where f_{enc} is the context encoding function, which can be represented by MLP or embedding layer. Based on the context representation, SATCM clusters similar services into meta-task clusters, using K-Means or cosine similarity-based similarity measurement:

$$sim(c_i, c_j) = \frac{c_i \cdot c_j}{\| c_i \| \| c_j \|}$$

This similarity is used as a metric for task division to construct a more consistent meta-training task set $T = \{T_1, T_2, ..., T_k\}$.

In each meta-task T_k, we further construct supervised training sample pairs (x_t, y_t), where x_t represents the service operation status within a certain time window, including the historical load sequence $I_t = [I_{t-w}, ..., I_t]$ and the resource usage vector r_t, which is in the following form:

$$x_t = [I_t \| r_t \| c_i]$$

y_t is the actual scaling behavior (expansion, reduction, or unchanged) at that time point, and participates in the meta-training phase as a supervisory signal. By introducing context c_i into the input space, SATCM achieves a strong coupling between task input and service structure, thereby ensuring semantic consistency between tasks.

In addition, to improve the distinguishability of task representation, SATCM introduces a context-based task embedding mechanism to map each task T_k to the embedding space:

$$e_k = g_{task}(\{c_i\}_{S_i \in T_k})$$

g_{task} can be constructed based on attention aggregation or graph neural encoding modules, which facilitates the dynamic perception of semantic differences between tasks in the subsequent meta-learning stage. Through the above mechanism, SATCM not only improves the structural rationality of task construction, but also lays a more stable training foundation for the learning of global initialization parameters.

In summary, SATCM, based on the heterogeneity of original service instances, effectively alleviates the performance degradation problem caused by "excessive heterogeneity between tasks" in meta-task construction by introducing contextual information modeling and semantic consistency optimization. It is the key technical support for this study in intelligent modeling of service scheduling.

3.2 DSSPM

To further improve the accuracy and adaptability of scaling strategy prediction in cross-service scenarios, this study designs a Dual-Stage Strategy Prediction Model (DSSPM). The model is built on meta-learned initialization parameters and generates strategies through a two-stage structure of coarse adaptation followed by fine adaptation. It achieves both generalization over known task distributions and rapid adaptation to local features of new services. This enables precise prediction of scaling behavior. DSSPM consists of two sub-modules: the Global Init Predictor and the Context-Aware Refiner. These modules work together to perform dynamic inference of service scaling strategies. The model architecture is shown in Figure 3.

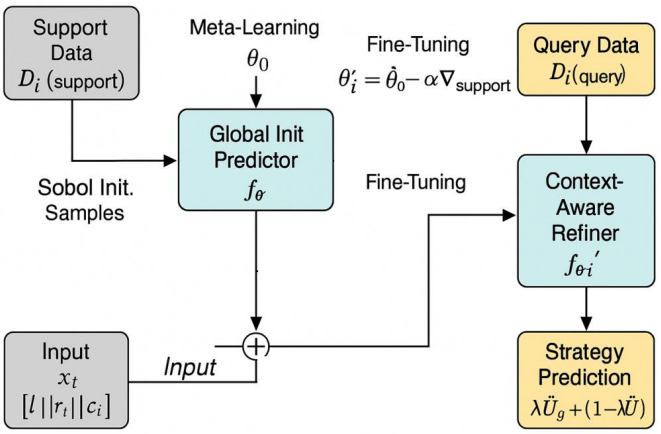

Figure 3. DSSPM module architecture

In the first stage, the model receives the initialization parameters θ_0 obtained by Meta-Learning training, builds a basic network $f_\theta(x_t)$ for scaling classification tasks, inputs the service state representation C at time step t, and outputs the probability distribution of policy decisions. The state vector is defined as follows:

$$x_t = [I_t \| r_t \| c_i]$$

I_t represents the historical load sequence, r_t is the current resource utilization vector, and c_i is the service context embedding. The global prediction stage outputs a three-category decision probability $y_t'^{(g)} = f_\theta(x_t) \in R^3$, corresponding to "scale up", "scale down" and "no change".

In the second stage, the fine-tuning module is introduced to quickly update the global model parameters through a small number of local samples $D_i^{support}$ to obtain the context-adapted parameters θ'_i. The fine-tuning process is implemented using gradient descent:

$$\theta'_i = \theta_0 - \alpha \nabla \theta L_{support}(f_\theta(x), y)$$

The updated model f_θ receives the same input x_t and outputs a refined policy prediction $y'^{(l)}_t = f_\theta(x_t)$. The final prediction result is a weighted fusion of the outputs of the two stages:

$$y'_t = \lambda y'^{(g)}_t + (1-\lambda) y'^{(l)}_t$$

$\lambda \in [0,1]$ controls the contribution weight of coarse prediction and fine-tuning prediction. The system can adaptively learn this parameter according to the training stage to enhance the robustness under different service characteristics.

The training goal of DSSPM is to minimize the joint classification loss of the global and local stages. The comprehensive loss function is as follows:

$$L_{total} = \beta \cdot L_{global} + (1-\beta) \cdot L_{local}$$

$$L_{global} = \frac{1}{|D_{query}|} \cdot \sum_{(x,y)\in D_{query}} CE(f_{\theta_0}(x), y)$$

$$L_{local} = \frac{1}{|D_{query}|} \cdot \sum_{(x,y)\in D_{query}} CE(f_{\theta'_i}(x), y)$$

$CE(\cdot)$ represents the cross entropy loss, and β is the weight adjustment factor of the two-stage loss. Through the two-stage optimization structure, DSSPM can quickly perceive service-specific behaviors based on global learning of general policy knowledge, and achieve effective policy migration and local optimization for new tasks.

In general, DSSPM provides a new policy prediction paradigm that combines initialization generalization with local customization, solving the common contradiction between "overfitting a single service" and "lack of generalization ability" in elastic policy learning. It not only improves the accuracy of scaling decisions, but also significantly shortens the adaptation time when the model is deployed to a new service, reflecting the practical value and engineering feasibility of AI algorithm design in system intelligent scheduling tasks.

4. Experimental Results

4.1 Dataset

This study uses the Azure Functions Traces Dataset as the primary source of experimental data. The dataset consists of execution logs from real cloud-based function services. It covers scheduling, resource usage, and state transitions of millions of function instances across different time periods. The data include key metrics such as start time, completion time, memory usage, CPU utilization, and concurrent request count. These metrics reflect the dynamic resource usage and workload patterns of microservices during real operations.

The dataset has two main advantages. It offers high-frequency sampling and wide coverage of service types. It includes function instances from various business scenarios, enabling the construction of complex service context features and multi-dimensional time series inputs. During preprocessing, we group services based on function names and deployment environments. We extract key fields to build service context representations and generate sliding-window input sequences with corresponding scaling labels.

This dataset allows us to simulate the scaling decision process in real-world cloud service environments. It provides diverse service task samples for training the meta-learning model. Its broad coverage and rich service diversity offer strong support for studying cross-service generalization. These features also enhance the engineering relevance and practical value of the experimental results.

4.2 Experimental setup

In the experimental setting, we regard the service scaling modeling task as a multi-task supervised classification problem, build independent tasks for each service, and use time series windows as the basic input unit. The input of the model includes the load sequence of several past time steps, resource utilization, and service context vector, and the output is a three-category label (expanded, reduced, unchanged). All models are built under a unified data preprocessing process. The dataset is divided into training set, validation set, and test set according to the service to ensure that the test task does not overlap with the training task to truly simulate the generalization ability of cross-service migration.

The experiment was conducted in a server environment with 4 V100 GPUs. PyTorch was used to implement all model frameworks. Adam was selected as the optimizer, and the initial learning rate was set to 0.001. The meta-learning part uses the 5-way 5-shot setting to build the support set and query set, the number of gradient update steps is set to 3 steps, and the λ parameter in the fusion stage is selected by grid search on the validation set. All experimental results are based on the stable results of averaging multiple runs to ensure the reliability and consistency of the evaluation.

4.3 Experimental Results

1) Loss function changes with epoch

This paper first gives a graph of the change of the loss function with epoch during the training process, as shown in Figure 4.

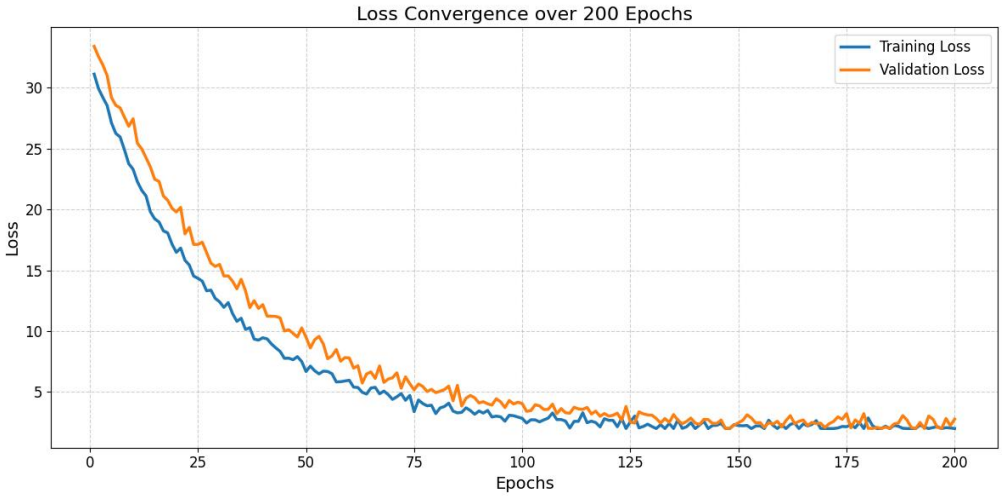

Figure 4. Loss function changes with epoch

As shown in the figure, both the training loss and validation loss exhibit a steady downward trend throughout the training process. This indicates that the model effectively fits the data distribution as learning progresses. During the first 50 epochs, both curves decrease rapidly. The training loss drops from an initial value of around 30 to below 10. This suggests that the model quickly captures key feature information and achieves rough parameter convergence at an early stage.

After around 75 epochs, the training enters the mid-to-late phase. Both training and validation losses begin to stabilize and gradually converge to around 2. This shows that the model has essentially completed the learning task, with later training focusing on fine-tuning of detailed features. During this stage, the training loss is slightly lower than the validation loss. This reflects a typical generalization gap, but the difference remains small, and there is no clear sign of overfitting.

Overall, these results confirm that the proposed model demonstrates good stability and generalization during training. The consistent decrease of validation and training loss indicates that the model performs well not only on the training data but also on unseen service tasks. This provides a solid foundation for subsequent strategy prediction.

2) Comparative experimental results

In this section, this paper gives the experimental results of the algorithm under different models and different settings. The experimental results are shown in Table 1.

Table 1: Comparative experimental results

Method	Acc(%)	F1-Score	Precision	Recall	AUC	Macro-F1
Ours (SATCM + DSSPM)	93.7	0.916	0.927	0.908	0.952	0.914
ProtoNet + LSTM[29]	88.9	0.864	0.871	0.856	0.912	0.859
MAML + MLP[30]	89.5	0.873	0.888	0.860	0.918	0.870
Meta-STN[31]	91.1	0.889	0.902	0.877	0.930	0.883
TAML (Task-Agnostic Meta-Learn)[32]	87.8	0.851	0.862	0.840	0.906	0.847
Reptile + TemporalConvNet[33]	90.2	0.875	0.881	0.869	0.921	0.872
FedMeta (centralized version)[34]	89.3	0.866	0.874	0.858	0.915	0.864

The experimental results in the table show that the proposed method (SATCM + DSSPM) achieves the best performance across all evaluation metrics. This demonstrates its strong adaptability and generalization ability in elastic scaling strategy modeling. In particular, it reaches 93.7% in Accuracy, 0.916 in F1-Score, and 0.952 in AUC. These results are significantly better than those of mainstream meta-learning and time series prediction models. This indicates that the proposed approach delivers more stable and accurate decisions across different scaling scenarios.

Compared with baseline methods, models such as MAML + MLP and ProtoNet + LSTM show some generalization ability through meta-learning. However, they lack precision in modeling service context and fine-grained temporal dynamics. As a result, their Precision and Recall scores are relatively low. Reptile + TemporalConvNet performs better in capturing time-based features. Yet, it lacks structured modeling of contextual information during task transfer, which limits its overall generalization.

Meta-STN, as a structure-aware model, achieves results close to our method in Recall and AUC. It shows good intra-task generalization but fails to incorporate service context semantics. This leads to accuracy degradation when switching between tasks. Federated approaches like FedMeta have distributed advantages in decentralized learning. However, in this study's centralized setting, their modeling accuracy and fine control remain insufficient.

Overall, the proposed method combines service-aware task construction with a dual-stage prediction model. It maintains strong generalization while capturing service-specific scaling behavior in detail. This makes it particularly suitable for real production environments with a large number of heterogeneous services. It offers a practical and effective solution for intelligent resource management in cloud-native systems.

3) Ablation experiment

This paper also conducted ablation experiments, and the experimental results are shown in Table 2.

Table 2: Ablation Experiment Results

Method	Acc(%)	F1-Score	Precision	Recall	AUC	Macro-F1
Ours (SATCM + DSSPM)	93.7	0.916	0.927	0.908	0.952	0.914
w/o SATCM (No Task Clustering)	90.5	0.879	0.887	0.870	0.923	0.874
w/o DSSPM (No Dual-Stage)	89.1	0.865	0.876	0.853	0.916	0.861
w/o Context Vector in Input	88.3	0.853	0.867	0.840	0.911	0.849
w/o Fine-Tuning Stage	87.6	0.844	0.858	0.831	0.903	0.840
Only Single-Stage Prediction (No Fusion)	88.9	0.861	0.873	0.850	0.908	0.857

The ablation study results in the table clearly show that the complete model (SATCM + DSSPM) achieves the best performance across all metrics. This confirms the synergistic effect of the two core modules within the overall framework. When the task construction mechanism SATCM is removed, both accuracy and F1 score drop significantly. This indicates that service-context-driven task partitioning plays a critical role in improving task consistency and training stability. Simple random task division fails to effectively capture structural similarities across services.

When the dual-stage prediction module DSSPM is removed, the model can only make decisions based on a single-stage output. This leads to reduced generalization and lower prediction accuracy. The results suggest that the dual-stage structure provides more flexible adaptation to new service tasks. It maintains generalization from global knowledge while using fine-tuning to capture local service features. This combination significantly enhances the accuracy and robustness of strategy prediction.

Furthermore, removing the service context vector from the input and relying only on resource and workload data leads to a noticeable performance drop. This shows that structural and contextual information provides important decision support during prediction. It helps the model distinguish different service behaviors under similar load conditions, allowing for more targeted scaling decisions.

Finally, when the fine-tuning stage is removed or the fusion strategy is not applied, and the prediction is made using only global initialization or a single output, the model still maintains some accuracy. However, generalization and recall performance both decline. This shows that local adaptation and the fusion mechanism play key roles in maintaining diversity and consistency in strategy decisions. Overall, the results validate the individual value of each module and the effectiveness of the complete system, providing strong support for the model design.

4) Supports sensitivity experiments on adaptation performance due to changes in set size

Furthermore, this paper also presents a sensitivity experiment on the adaptation performance to changes in the support set size, and the experimental results are shown in Figure 5.

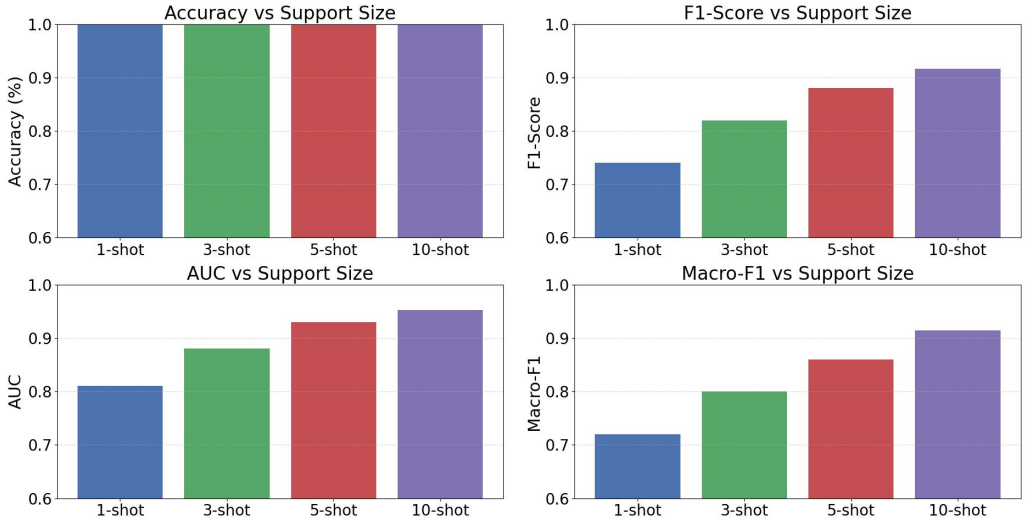

Figure 5. Supports sensitivity experiments on adaptation performance due to changes in set size

As shown in Figure 5, the size of the support set has a significant impact on the model's adaptability. As the number of support samples increases from 1-shot to 10-shot, the model shows consistent improvement across all metrics, including Accuracy, F1-Score, AUC, and Macro-F1. This result indicates that in a meta-learning framework, more support samples provide richer task priors. This helps the model better capture local characteristics of services and improves the accuracy of strategy prediction.

The performance gain is especially notable in the transition from 1-shot to 3-shot. Multiple metrics increase sharply, showing that even a few additional samples can significantly enhance the model's adaptability. During this phase, the model shifts from an initial general state to a more task-specific state. This reflects the high sensitivity of meta-learning algorithms to limited information and validates the effectiveness of the task construction mechanism and parameter initialization strategy.

When the support set increases further to 5-shot and 10-shot, performance continues to improve but begins to converge. This suggests that once sufficient contextual information is absorbed, the model can make stable predictions. The convergence trend is expected and shows that the model is both stable and robust. It does not suffer from overfitting as the data size grows.

Overall, this experiment confirms the proposed method's strong adaptability under limited sample conditions. It also highlights the support set size as a key tuning factor for optimizing model performance in real-world deployments. These findings further support the design choice of using meta-learning as the core framework in this study.

5) Experiment on the impact of service type heterogeneity on strategy prediction accuracy

Furthermore, this paper presents an experiment on the impact of service type heterogeneity on strategy prediction accuracy, and the experimental results are shown in Figure 6.

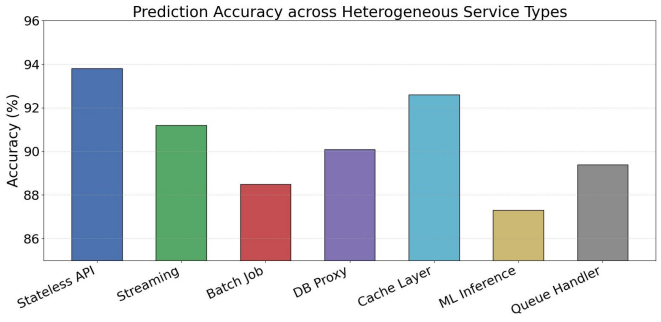

Figure 6. Experiment on the impact of service type heterogeneity on strategy prediction accuracy

As shown in Figure 6, there are clear differences in prediction accuracy across different service types. This confirms the significant impact of service heterogeneity on model performance. Overall, services of the Stateless API and Cache Layer types achieve the highest accuracy, both exceeding 93 percent. These services tend to have stable runtime behavior and relatively consistent workload patterns, which help the model learn highly consistent feature distributions during training.

In contrast, services like ML Inference and Batch Job show lower accuracy. This may be due to their more volatile resource usage patterns and rapid state changes. These characteristics make it difficult for the model to capture key triggers for scaling decisions. Such services often involve uneven loads and sudden computational peaks, which place higher demands on the model's temporal perception and contextual understanding.

Services like Streaming and Queue Handler exhibit moderate prediction performance. Although they have periodic or task-driven features, their behavioral patterns are more complex than traditional web services. This suggests that the current model can generalize to some extent but still requires optimization based on service semantics. DB Proxy services show slightly lower accuracy as well, possibly due to strong dependencies on external service interactions.

This analysis shows that the predictability of scaling strategies is directly influenced by the service's operational logic, call structure, and resource usage characteristics. Therefore, in real-world deployments, it is important to consider the impact of service heterogeneity on model performance. Specialized modeling or multi-model integration mechanisms should be introduced for critical service types to improve the stability and applicability of strategy prediction.

6) Test of the impact of time series feature truncation length on model performance

Finally, this paper also presents a test on the impact of the time series feature truncation length on the model performance, and the experimental results are shown in Figure 7.

The experimental results in Figure 7 show that the truncation length of temporal features has a significant impact on model performance. When the truncation length is short (e.g., 5 or 10), the model struggles to capture the full dynamic patterns of resource usage and workload fluctuations. As a result, performance on metrics such as Accuracy, F1-Score, and AUC is relatively low. This suggests that insufficient contextual information limits the accuracy of strategy prediction.

When the truncation length increases to 20, all metrics reach their peak values. The model achieves optimal performance at this length. This indicates that the window captures representative features of historical behavior. It provides enough contextual information while avoiding the interference caused by redundant inputs. This leads to more accurate strategy decisions. It also confirms the model's strong expressive capacity when handling limited historical information.

However, when the truncation length is further increased to 30 or 50, performance shows a slight decline. This may be due to overly long input sequences introducing noise or irrelevant information. Such inputs may distract the model from important state changes and reduce generalization. Longer sequences may also cause parameter redundancy and unstable training, affecting prediction stability.

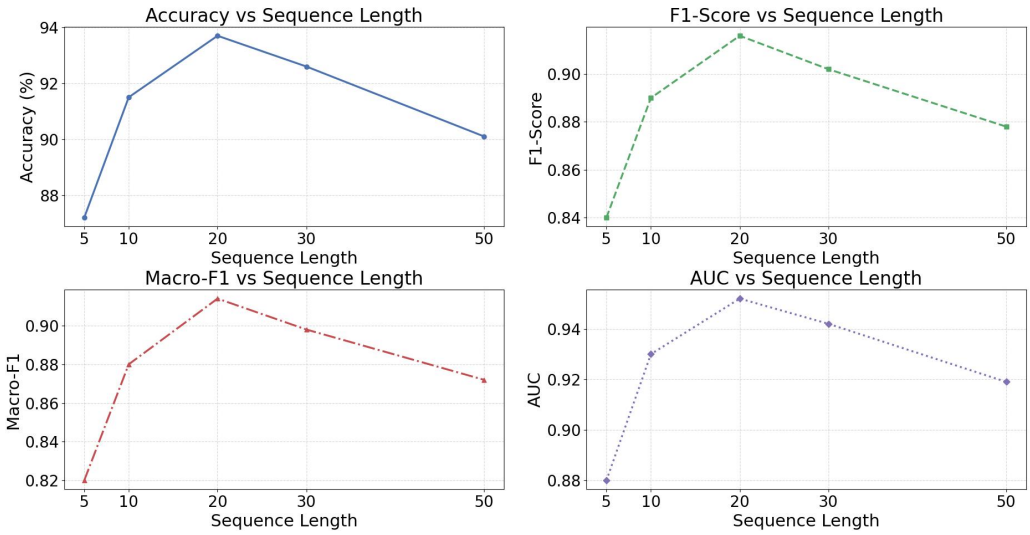

Figure 7. Test of the impact of time series feature truncation length on model performance

Overall, this experiment shows that proper configuration of the temporal feature window is crucial in elastic scaling strategy modeling. It is important to ensure the completeness of input information while controlling the impact of redundancy. The truncation strategy should match the service patterns and system dynamics to fully leverage the model's temporal modeling capability.

5. Conclusion

This paper addresses the intelligent modeling of elastic scaling strategies for service instances. It proposes a cross-service generalization framework that combines meta-learning with deep prediction mechanisms. The framework effectively overcomes the limitations of existing methods in handling heterogeneous service environments, particularly in terms of poor transferability and weak adaptability. By introducing a service-aware task construction mechanism and a dual-stage strategy prediction model, the proposed method achieves rapid adaptation to new services with few samples. It outperforms mainstream methods across multiple metrics, demonstrating its feasibility and efficiency in complex scheduling tasks. Experimental results show that the model maintains good stability and generalization across diverse service scenarios. It performs reliably even when handling highly dynamic and heterogeneous service types. This capability directly supports automation, resource optimization, and service-level QoS assurance in cloud-native architectures. It also provides an intelligent solution for real-world deployments involving data center resource scheduling and microservice elasticity control.

In addition, this study conducts detailed experiments on support set size, temporal window length, and service heterogeneity. These analyses reveal the sensitivity of the meta-learning framework to task configurations and offer insights into tuning strategies. The findings contribute both theoretical foundations and experimental references for future research. The proposed method has methodological value and strong application potential in areas such as service computing, intelligent operations, and edge resource management. It expands the practical boundaries of AI algorithms in system engineering. Looking ahead, the modeling approach proposed in this work still has room for improvement. One direction is to enhance the task construction process with automatic task discovery and task difficulty modeling. Another is to adopt more lightweight or federated architectures to improve deployment efficiency and privacy protection in large-scale and multi-tenant environments. With the continued development of cloud computing and edge intelligence, the demand for highly adaptive and generalizable strategy models will keep increasing. The methods and ideas presented in this paper are expected to play a broader role in future system optimization tasks.

References

[1] Vettoruzzo, Anna, et al. "Advances and challenges in meta-learning: A technical review." IEEE transactions on pattern analysis and machine intelligence 46.7 (2024): 4763-4779.

[2] Yang, Jin, et al. "On a Meta Learning-Based Scheduler for Deep Learning Clusters." IEEE Transactions on Cloud Computing 11.4 (2023): 3631-3642.

[3] Chen, Hao, Ruiping Yin, and Zhen Yang. "Edge-Cloud Collaborative High-Quality Recommendation: A Meta-Learning Approach." 2023 18th International Conference on Intelligent Systems and Knowledge Engineering (ISKE). IEEE, 2023.

[4] Sharma, Nelson, et al. "Deep meta q-learning based multi-task offloading in edge-cloud systems." IEEE Transactions on Mobile Computing 23.4 (2023): 2583-2598.

[5] Belarbi, O., Spyridopoulos, T., Anthi, E., Mavromatis, I., Carnelli, P., & Khan, A. (2023, December). Federated deep learning for intrusion detection in IoT networks. In GLOBECOM 2023-2023 IEEE Global Communications Conference (pp. 237-242). IEEE.

[6] Liu, Xiaonan, et al. "Federated learning and meta learning: Approaches, applications, and directions." IEEE Communications Surveys & Tutorials 26.1 (2023): 571-618.

[7] Liu, Hongyun, et al. "Robustness challenges in reinforcement learning based time-critical cloud resource scheduling: A meta-learning based solution." Future Generation Computer Systems 146 (2023): 18-33.

[8] Lee, R., Kim, M., Li, D., Qiu, X., Hospedales, T., Huszár, F., & Lane, N. (2023). Fedl2p: Federated learning to personalize. Advances in Neural Information Processing Systems, 36, 14818-14836.

[9] Mo, Wenshuai, et al. "A meta-learning based resource allocation algorithm in vehicular edge computing networks." 2024 IEEE 3rd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA). IEEE, 2024.

[10] Chang, Bao Rong, Hsiu-Fen Tsai, and Guan-Ru Chen. "Self-adaptive server anomaly detection using ensemble meta-reinforcement learning." Electronics 13.12 (2024): 2348.

[11] Gui, Jie, et al. "A survey on self-supervised learning: Algorithms, applications, and future trends." IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

[12] Rani, Veenu, et al. "Self-supervised learning: A succinct review." Archives of Computational Methods in Engineering 30.4 (2023): 2761-2775.

[13] Jiao, Rushi, et al. "Learning with limited annotations: a survey on deep semi-supervised learning for medical image segmentation." Computers in Biology and Medicine 169 (2024): 107840.

[14] Han, Meng, et al. "A survey of multi-label classification based on supervised and semi-supervised learning." International Journal of Machine Learning and Cybernetics 14.3 (2023): 697-724.

[15] Cavasotto, Claudio N., and Juan I. Di Filippo. "The impact of supervised learning methods in ultralarge high-throughput docking." Journal of Chemical Information and Modeling 63.8 (2023): 2267-2280.

[16] Vanschoren, Joaquin. "Meta-learning: A survey." arXiv preprint arXiv:1810.03548 (2018).

[17] Lake, Brenden M., and Marco Baroni. "Human-like systematic generalization through a meta-learning neural network." Nature 623.7985 (2023): 115-121.

[18] Kim, Kyun Kyu, et al. "A substrate-less nanomesh receptor with meta-learning for rapid hand task recognition." Nature Electronics 6.1 (2023): 64-75.

[19] Luo, Jingjie, et al. "Meta-learning with elastic prototypical network for fault transfer diagnosis of bearings under unstable speeds." Reliability Engineering & System Safety 245 (2024): 110001.

[20] Zhang, Baoquan, et al. "Metadiff: Meta-learning with conditional diffusion for few-shot learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 38. No. 15. 2024.

[21] Lv, Qiujie, et al. "Meta learning with graph attention networks for low-data drug discovery." IEEE transactions on neural networks and learning systems (2023).

[22] Nichol, Alex, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms." arXiv preprint arXiv:1803.02999 (2018).

[23] Tao, Yiyi. "Meta learning enabled adversarial defense." 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE). IEEE, 2023.

[24] Yang, Lei, et al. "Personalized federated learning on non-IID data via group-based meta-learning." ACM Transactions on Knowledge Discovery from Data 17.4 (2023): 1-20.

[25] Verma, R., & Nalisnick, E. (2022, June). Calibrated learning to defer with one-vs-all classifiers. In International Conference on Machine Learning (pp. 22184-22202). PMLR.

[26] Zoppi, Tommaso, et al. "Which algorithm can detect unknown attacks? Comparison of supervised, unsupervised and meta-learning algorithms for intrusion detection." Computers & Security 127 (2023): 103107.

[27] Hao, Xiaoyang, et al. "Automatic modulation classification via meta-learning." IEEE Internet of Things Journal 10.14 (2023): 12276-12292.

[28] Tan, Chenmien, Ge Zhang, and Jie Fu. "Massive editing for large language models via meta learning." arXiv preprint arXiv:2311.04661 (2023).

[29] Zhu, Yuqicheng, et al. "Active Transfer Prototypical Network: An Efficient Labeling Algorithm for Time-Series Data." Procedia Computer Science 217 (2023): 1427-1436.

[30] Shao, Yuanjie, et al. "Improving the generalization of MAML in few-shot classification via bi-level constraint." IEEE Transactions on Circuits and Systems for Video Technology 33.7 (2022): 3284-3295.

[31] Hospedales, Timothy, et al. "Meta-learning in neural networks: A survey." IEEE transactions on pattern analysis and machine intelligence 44.9 (2021): 5149-5169.

[32] Jamal, Muhammad Abdullah, and Guo-Jun Qi. "Task agnostic meta-learning for few-shot learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[33] Girdhar, Rohit. Learning to Understand People via Local, Global and Temporal Reasoning. Diss. Carnegie Mellon University, 2019.

[34] Yao, Xin, et al. "Federated learning with unbiased gradient aggregation and controllable meta updating." arXiv preprint arXiv:1910.08234 (2019).