Integrated Learnware Identification and Reuse via Reusability-Aware Metric Learning

Hai-Tian Liu
Nanjing University
Nanjing, China

Peng Tan
Nanjing University
Nanjing, China

Jian-Dong Liu†
SinapisAI
Nanjing, China

Zhi-Hao Tan
Nanjing University
Nanjing, China

Zhi-Hua Zhou*
Nanjing University
Nanjing, China

Abstract

The *learnware paradigm* aims to help users solve machine learning tasks by reusing existing well-trained models instead of building from scratch. These models are accommodated in a *learnware dock system*, where each *learnware* consists of a model and a *specification* that characterizes the model's capability, enabling the learnware to be effectively identified and reused for future tasks. Based on specifications, the system identifies learnwares whose training distributions align with the user task. However, whether the identified learnwares benefit the user depends not only on task similarity but also on how they perform after reuse, making it important that identification reflects expected reuse performance. To this end, this paper leverages the reuse experience among hosted learnwares to make specifications characterize learnware reusability for new user tasks. Specifically, the system treats each hosted learnware as a pseudo-user task and evaluates the task performance achieved by identifying and reusing other learnwares for these tasks. Under the supervision of these system-wide evaluations, the system evolves specifications into a reusability-aware specification space where shorter distances correspond to higher expected reuse performance. With this specification evolvement, learnware identification is guided by expected reuse performance rather than original task similarity alone. Experiments demonstrate that, with reusability-aware identification, the system can effectively recommend learnwares that benefit new user tasks after reuse.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Information systems** → **Information systems applications**.

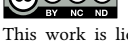
Keywords

Machine Learning, Learnware, Learnware Dock System, Model Recommendation, Metric Learning

ACM Reference Format:

Hai-Tian Liu, Peng Tan, Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. 2026. Integrated Learnware Identification and Reuse via Reusability-Aware Metric Learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD 2026), August 9–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3770855.3818210

*Corresponding author (email: zhouzh@lamda.nju.edu.cn).
†Also affiliated with Nanjing University.
Institution details: Nanjing University (National Key Laboratory for Novel Software Technology and School of Artificial Intelligence).



KDD 2026, Jeju Island, Republic of Korea.

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3818210

Resource Availability:

The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.20541297 and https://github.com/liuht-0807/Reuse-Aware-Learnware-Identification.

1 Introduction

"Learnware = Model + Specification" [31, 32]. The learnware paradigm considers a practical scenario where machine learning developers are willing to share their trained models while keeping their original training data private. In such a scenario, one can envision a *learnware dock system* that accommodates a wide range of models voluntarily submitted by developers worldwide. These models may be trained on different data, for different tasks, and with different learning algorithms and objectives. As the system accommodates a large number of such models, a future user who aims to build a model for her own task can submit her task requirement to the system. The system then identifies a helpful model or even reassembles a set of models to help solve the user task, enabling the user to obtain a high-quality model without starting from scratch. Notably, throughout this process, neither model developers nor users need to disclose their original data to the learnware dock system. Given that the system has no access to the original training data and that submitted models appear as black boxes whose functional behaviors are difficult to inspect, how can the system still identify models that are helpful for the user task, and how can diverse models, even those developed for seemingly irrelevant purposes, be identified and reassembled to benefit the user task?

The learnware paradigm makes this possible. A key component is the *specification*, which enables a trained model to be adequately identified and reused according to a future user's task requirement, even when the user knows nothing about the model, without disclosing the developer's original training data. Specifically, when a model is submitted to the learnware dock system, a specification is generated with assistance from the system and associated with the model, thereby forming a learnware. The specification can approximately characterize the model's utility and specialty without requiring access to its training data, and a future user's task requirement is also submitted in the form of a task specification. As the system continuously accommodates a large number of learnwares, it is expected to give rise to increasingly strong system capabilities, such as enabling "small models do big" [32] and applying models to tasks beyond their original development purposes [11]. Recent research has substantially advanced this paradigm. The reduced kernel mean embedding (RKME) specification [32] was proposed to capture model capabilities via training distribution information,

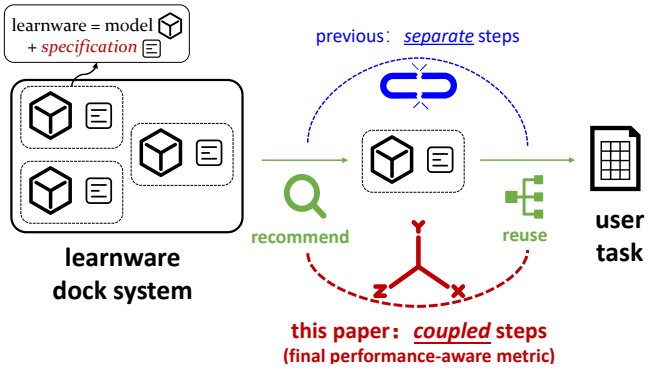


Figure 1: This paper proposes specification evolvement that makes identification guided by expected reuse performance, achieved through a reusability-aware specification metric.

and has been theoretically proven to protect the original training data of the model [10]. Building on RKME, subsequent studies have improved identification efficiency [11, 28] and extended the paradigm to heterogeneous feature spaces [22, 23] and heterogeneous label spaces [6]. More recently, the PAVE specification [20] has been proposed for identifying learnwares from unstructured data, and the paradigm has also been extended to language models [25] and plug-and-play learnware agents [14]. Alongside these advances, Beimingwu [24] has been released as the first learnware dock system, providing a research platform for the full submission-to-deployment process.

Based on specifications, recent studies have successfully identified helpful learnwares whose training distributions align with the user task. Nevertheless, task similarity is a proxy criterion computed prior to learnware reuse, whereas the effectiveness of the identified learnwares is ultimately determined by their performance after reuse. Indeed, since reuse performance depends on the user task, the candidate learnware, and the reuse strategy, learnwares close in the original specification space may not perform well after reuse, while those trained on distant tasks may still be beneficial. Since identification is guided by task similarity rather than reuse performance, the system may overlook such beneficial learnwares, leading to suboptimal recommendations. It is therefore important that learnware identification accounts for final reuse performance.

However, aligning identification with expected reuse performance, rather than relying solely on distributional similarity, remains fundamentally challenging: identification must occur before reuse, so the system cannot observe how candidate learnwares will actually perform on future user tasks. The key difficulty is how to construct a specification space whose distances not only capture statistical similarity but also reflect expected reuse performance. Our key insight is that the learnware dock system provides a unique foundation for this: it hosts numerous learnwares from diverse tasks, whose specifications incorporate the task information of each model. By systematically applying hosted learnwares to solve each other's tasks, the system can evaluate reuse performance across these tasks, obtaining system-wide reuse experience. With this experience, the specification space can be evolved so that shorter distances correspond to higher expected reuse performance, shifting learnware identification from task similarity matching toward reusability-aware identification. As the system accommodates learnwares from increasingly diverse tasks, this reuse experience becomes more comprehensive, providing a basis for evolving specifications to characterize learnware reusability.

In this paper, we propose to evolve specifications so that they characterize learnware reusability across tasks, enabling reusability-aware identification for new user tasks. Specifically, the system treats each hosted learnware as a pseudo-user task and evaluates the reuse performance of other learnwares on these tasks using a fixed reuse strategy. Based on these leave-one-out evaluations, the system learns a specification mapping that encodes the system-wide reuse experience into specification distances, constructing a reusability-aware specification space. When a new user task arrives, the system identifies learnwares in this evolved specification space, where closer learnwares are expected to perform better after reuse. The contributions are summarized as follows:

- This paper proposes to leverage the system-wide reuse experience among hosted learnwares to make specifications characterize learnware reusability for new user tasks. With such specification evolvement, the system can more effectively identify learnwares expected to benefit new user tasks after reuse.
- We design a leave-one-out evaluation scheme that treats each hosted learnware as a pseudo-user task and evaluates reuse performance under a fixed reuse strategy, obtaining reuse evaluations without requiring future user tasks. Based on these system-wide evaluations, the system learns a specification mapping that encodes the reuse experience into specification distances, constructing a reusability-aware specification space where shorter distances correspond to higher expected reuse performance.
- Experiments demonstrate that, with the evolved specification space, the system can effectively identify learnwares that benefit new user tasks after reuse, including learnwares that original task similarity would overlook.

2 Preliminary

The specification is the central component of the learnware paradigm, playing a pivotal role in learnware organization, identification, and reuse. It captures the model's essential statistical properties while not disclosing the original training data. To this end, the Reduced Kernel Mean Embedding (RKME) specification [32] was proposed based on the Kernel Mean Embedding (KME) [21], and has demonstrated significant efficacy in recent learnware studies [11, 22]. The core idea of RKME is to characterize a trained model's capabilities by constructing a reduced set of weighted samples $s = \{(\beta_j, \boldsymbol{z}_j, y_j)\}_{j=1}^{n}$ ($n \ll m$) that statistically approximates the joint distribution of the original features and model outputs $\mathcal{D}_{\text{joint}} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{m}$. This reduced set is generated by minimizing the Maximum Mean Discrepancy (MMD) in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$, as defined by:

$$\min_{\boldsymbol{\beta},\mathbf{Z},\boldsymbol{y}} \left\| \frac{1}{m}\sum_{i=1}^{m} k((\boldsymbol{x}_i, f(\boldsymbol{x}_i)), \cdot) - \sum_{j=1}^{n} \beta_j k((\boldsymbol{z}_j, y_j), \cdot) \right\|_{\mathcal{H}_k}^2, \quad (1)$$

where $k(\cdot,\cdot)$ is a kernel defined on the joint space. This characterization captures the joint distribution of features and model outputs rather than the original training data distribution.

In simple cases where only covariate shift exists between tasks, it suffices to only approximate the marginal feature distribution $P(X)$ [26]. This specification is generated by solving the following optimization problem in an alternating manner:

$$\min_{\boldsymbol{\beta},\mathbf{Z}} \left\| \frac{1}{m}\sum_{i=1}^{m} k\left(\boldsymbol{x}_i,\cdot\right) - \sum_{j=1}^{n} \beta_j k\left(\boldsymbol{z}_j,\cdot\right) \right\|_{\mathcal{H}_k}^2, \tag{2}$$

with the non-negative coefficients $\{\beta_j\}_{j=1}^n$. The resulting RKME $\Phi(\boldsymbol{\beta},\mathbf{Z}) = \sum_{j=1}^n \beta_j k\left(\boldsymbol{z}_j,\cdot\right)$ converges to the empirical KME at a rate of $O(1/\sqrt{n})$ [2]. Theoretically, this specification is proven to scarcely contain any original data and robustly defend against common inference attacks [10].

However, in high-dimensional feature spaces, measuring solely on the joint distribution as in Eq. (1) can cause the label information to be overshadowed by complex feature structures. To address this, the RKME_L specification [22] proposes to minimize the distribution loss of the sum of the marginal distribution $P(X)$ and the conditional distributions $P(X|\widehat{Y})$. Formally, for a task with C classes, let $\mathcal{P}$ and $\mathcal{P}_c$ denote the overall and class-conditional data distributions. The objective minimizes the MMD between their empirical KMEs and the corresponding RKMEs:

$$\min_{\boldsymbol{\beta},\mathbf{Z}} \|\widehat{\mu}_k(\mathcal{P}) - \Phi(\boldsymbol{\beta},\mathbf{Z})\|_{\mathcal{H}_k}^2 + \lambda \sum_{c=1}^{C} \left\|\widehat{\mu}_k(\mathcal{P}_c) - \Phi(\boldsymbol{\beta}_c,\mathbf{Z}_c)\right\|_{\mathcal{H}_k}^2, \tag{3}$$

where $\lambda > 0$ balances the marginal and conditional alignment, and $\Phi(\boldsymbol{\beta}_c,\mathbf{Z}_c)$ represents the conditional RKME for class c, where $\boldsymbol{\beta}_c$ and $\mathbf{Z}_c$ denote the weights and points in the reduced set whose labels belong to class c. For regression tasks, the conditional term is omitted and the reduced set is labeled by the model's predictions $y_j = f(\boldsymbol{z}_j)$. We adopt the RKME_L specification in this paper.

3 Problem Setup

In this section, we formulate the coupling between learnware identification and reuse within the learnware paradigm.

Submitting Stage. In this stage, developers submit their well-trained models $\{f_i\}_{i=1}^N$, where each $f_i \in \mathcal{F} = \{f \mid f : \mathcal{X} \to \mathcal{Y}\}$ is trained on a task with data distribution $\mathcal{D}_i$ over $\mathcal{X}$ and approximates the ground-truth labeling function $h_i \in \mathcal{F}$. Along with the submitted model f_i, each developer also generates an RKME_L specification $s_i = \{(\beta_{ij}, \boldsymbol{z}_{ij}, y_{ij})\}_{j=1}^{n_i}$ with assistance from the system, forming a learnware as the pair $l_i = (f_i, s_i)$. The system then accommodates all submitted learnwares $\mathcal{L} = \{l_i\}_{i=1}^N$ through an organization process, which facilitates effective identification and reuse in the subsequent deploying stage.

Deploying Stage. In this stage, the user aims to solve her own task, defined by a data distribution $\mathcal{D}_u$. The user possesses a limited set of labeled data $D_u^L = \{(\widetilde{\boldsymbol{x}}_{u,j}, y_{u,j})\}_{j=1}^{m_l}$ and a set of unlabeled data $D_u^U = \{\boldsymbol{x}_{u,j}\}_{j=1}^{m_u}$, both drawn from $\mathcal{D}_u$. Without disclosing the original data, the user generates a corresponding RKME_L task requirement $s_u = \{(\beta_{uj}, \boldsymbol{z}_{uj}, y_{uj})\}_{j=1}^{n_u}$ and submits it to the learnware dock system. The system then identifies and returns a set of helpful learnwares $\{l_j \mid j \in I\}$ to the user, with the number constraint $|I| \le K$. The user can then reuse these learnwares to tackle her task. In this paper, for simplicity, we focus on the single-learnware case and fix a standard reuse strategy $\mathcal{R}$, where $\mathcal{R}(\boldsymbol{x}; l)$ denotes the prediction obtained by reusing learnware l on input $\boldsymbol{x}$.

Ideally, the system should identify learnwares that will perform well after reuse on the user's task, rather than relying solely on task similarity. We aim to identify the learnware that minimizes the expected reuse loss for the user's task:

$$i^\star = \arg\min_{i\in[N]} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_u} \left[\ell\left(\mathcal{R}(\boldsymbol{x}; l_i), y\right)\right], \tag{4}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is the loss function for the user's task. The central challenge is how to make learnware identification account for expected reuse performance, given that the system has no access to future user tasks.

4 Our Approach

In this section, we present our approach to evolving specifications using the reuse experience among hosted learnwares. Section 4.1 formulates the specification evolvement under a given reuse strategy, Section 4.2 describes how the system obtains reuse experience, and Section 4.3 details the specification metric construction. The overall workflow is summarized in Section 4.4. Figure 2 provides an overview.

4.1 Specification Evolvement

One way to account for reuse performance is to let the user evaluate candidate learnwares on her own data after reuse. However, the user's limited labeled data is insufficient to reliably evaluate reuse performance. On the system side, interactive protocols [11] can refine task requirements for more precise identification, but they still match specifications by task similarity rather than reuse performance. This motivates organizing learnwares on the system side by evolving their specifications into a new specification space, before any user task arrives.

To evolve specifications on the system side, the system must assess reuse performance across tasks, yet for future user tasks, no such evaluation is available. Our insight is that the hosted learnwares can themselves serve as pseudo-user tasks: since each learnware's specification incorporates its task information, the system can evaluate how candidate learnwares perform after reuse on these tasks. These system-wide evaluations yield empirical reuse estimates across diverse tasks, enabling the system to evolve specifications so that their distances reflect expected reuse performance.

Formally, we realize this evolvement as a mapping $\mathcal{M}_\theta : \mathcal{S} \to \mathcal{S}'$ that transforms each specification into a reusability-aware specification space $\mathcal{S}'$. The mapping is learned so that distances in $\mathcal{S}'$ encode the reuse experience from the pseudo-user tasks. In this space, given a task requirement s_i, identification returns the learnware $l_\theta(s_i)$ whose evolved specification is closest to the evolved task requirement $\mathcal{M}_\theta(s_i)$. To learn θ toward the identification objective in Eq. (4), we optimize over the task distribution $\mathcal{T}$ induced by the hosted learnwares:

$$\theta^\star = \arg\min_{\theta} \mathbb{E}_{\mathcal{D}_i\sim\mathcal{T}}\left[\mathcal{L}_{\mathcal{D}_i}(l_\theta(s_i), \mathcal{R})\right], \tag{5}$$

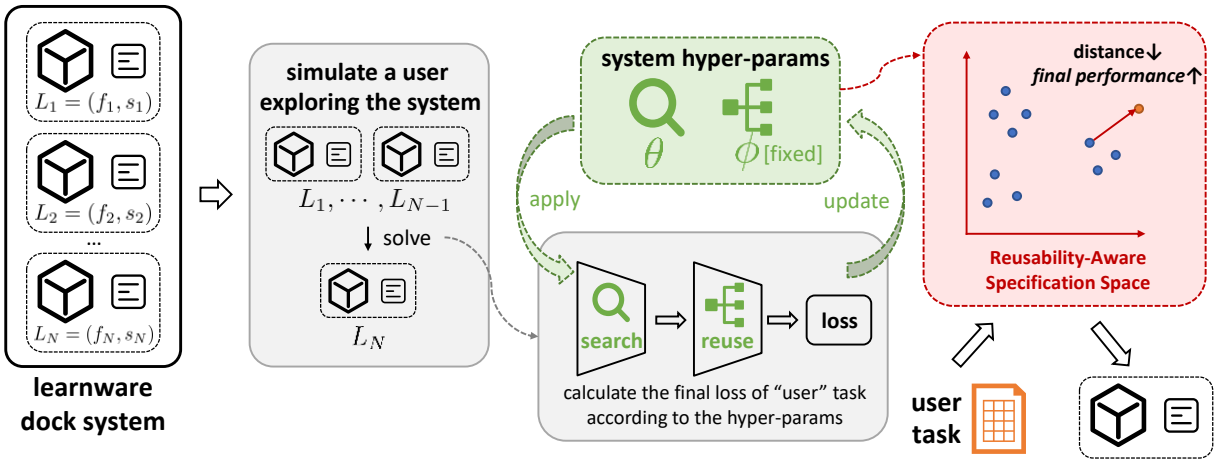


Figure 2: Overview of the proposed approach. The system evaluates reuse performance among hosted learnwares via leave-one-out evaluation (Section 4.2), treating each learnware as a pseudo-user task. The resulting reuse experience is encoded into a reusability-aware specification space through specification metric construction (Section 4.3), so that specification distances reflect expected reuse performance under a given reuse strategy.

where $\mathcal{L}_{\mathcal{D}_i}(l, \mathcal{R}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_i}\left[\ell(\mathcal{R}(\boldsymbol{x}; l), y)\right]$ is the reuse loss of the identified learnware under $\mathcal{R}$. In this way, the mapping $\mathcal{M}_\theta$ is guided by the reuse performance under $\mathcal{R}$, so that learnwares are organized by their empirical reuse performance rather than original task similarity.

Remark. In this paper, we focus on a fixed, single-learnware reuse strategy $\mathcal{R}$. More generally, $\mathcal{R}$ can be parameterized as $\mathcal{R}_\phi$ (e.g., feature augmentation [13]) and learned jointly with $\mathcal{M}_\theta$ under the same reuse-performance objective. In this case, the specification evolvement would simultaneously guide both which learnware to recommend and how to reuse it. Furthermore, when extending to learnware assembly, the specification evolvement may capture how learnwares complement each other, which task similarity alone does not account for, helping the system identify more effective learnware combinations. We leave the joint learning of $\mathcal{M}_\theta$ and $\mathcal{R}_\phi$, and the extension to multi-learnware assembly to future work.

4.2 Leave-One-Out Reuse Evaluation

Optimizing the objective in Eq. (5) requires evaluating the reuse performance of candidate learnwares across the task distribution $\mathcal{T}$. However, computing the true reuse loss requires the original data distribution and the ground-truth labeling function, neither of which is accessible to the system. To address this, the system conducts leave-one-out reuse evaluations among the hosted learnwares, as part of the system's organization in the submitting stage.

Leave-one-out construction. The system holds out each learnware $l_i = (f_i, s_i) \in \mathcal{L}$ in turn as a pseudo-user task:

- The specification s_i serves as the task requirement for identification. Furthermore, since the original data distribution $\mathcal{D}_i$ is not accessible, the reduced set $\{(\beta_{ik}, \boldsymbol{z}_{ik}, y_{ik})\}_{k=1}^{n_i}$ provides the test samples for evaluating reuse performance.
- Since the ground-truth labeling function h_i is unknown, the labels y_{ik} produced by the well-trained model f_i serve as the ground truth for evaluation.
- The remaining learnwares $\mathcal{L} \setminus \{l_i\}$ form the candidate pool for this pseudo-user task.

Surrogate reuse loss. For each pseudo-user task l_i, the system first identifies candidate learnwares from $\mathcal{L} \setminus \{l_i\}$ in the current evolved space $\mathcal{S}'$, simulating the deploying stage. The reuse performance of each identified candidate l_j on l_i's task is then evaluated under $\mathcal{R}$ through a surrogate reuse loss:

$$\widetilde{\mathcal{L}}_{s_i}(l_j, \mathcal{R}) = \sum_{k=1}^{n_i} \beta_{ik}\, \ell\big(\mathcal{R}(\boldsymbol{z}_{ik}; f_j),\, y_{ik}\big)\,. \tag{6}$$

By repeating this process for all pseudo-user tasks, we obtain the system-wide reuse experience $\{S_{ij}\}$ with $S_{ij} = \widetilde{\mathcal{L}}_{s_i}(l_j, \mathcal{R})$, which captures how each hosted learnware performs after reuse on other learnwares' tasks.

The following theorem establishes that S_{ij} approximates the true reuse loss $\mathcal{L}_{\mathcal{D}_i}(l_j, \mathcal{R})$. The proof is in Appendix A.

Theorem 4.1. *Assume $\sup_{\boldsymbol{x}\in\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x}) < \infty$ and the loss function ℓ satisfies the triangle inequality. Let $g_{ij}(\boldsymbol{x}) = \ell(\mathcal{R}(\boldsymbol{x}; f_j), f_i(\boldsymbol{x}))$ and assume $\|g_{ij}\|_{\mathcal{H}_k}$ is uniformly bounded for all $f_i, f_j \in \mathcal{F}$. Suppose the original model f_i satisfies the expected risk bound $\mathcal{L}_{\mathcal{D}_i}(f_i, h_i) \le \epsilon_i$ with respect to the ground truth h_i. Then, with probability at least $1-\delta$, $\delta \in (0,1)$, for all $l_j \in \mathcal{L} \setminus \{l_i\}$:*

$$\left|\mathcal{L}_{\mathcal{D}_i}(l_j, \mathcal{R}) - S_{ij}\right| \le \epsilon_i + O(m_i^{-\frac{1}{2}} + n_i^{-\frac{1}{2}}), \tag{7}$$

where m_i and n_i are the sizes of the original task data and the RKME specification, respectively.

This guarantee ensures that the system-wide reuse experience $\{S_{ij}\}$ reliably approximates the true reuse performance, providing supervision for learning the specification metric in the next section.

Algorithm 1 System-Level Organization

Input: Learnware dock system $\mathcal{L} = \{l_i\}_1^N$, Reuse strategy $\mathcal{R}$, Top-κ, Learning rate η
Output: Learned specification mapping $\mathcal{M}_\theta$

Initialize mapping parameters θ randomly
repeat
  Sample a batch of pseudo-user tasks $\mathcal{B} \subset \mathcal{L}$
  for each task $l_i \in \mathcal{B}$ **do**
    Retrieval: Retrieve candidate learnwares C_i closest to $\mathcal{M}_\theta(s_i)$ in $\mathcal{S}'$
    Evaluation: Compute scores $S_{ij} := \widetilde{\mathcal{L}}_{s_i}(l_j, \mathcal{R})$ for $l_j \in C_i$
    Optimization: $\mathcal{L}_{\text{batch}} \leftarrow \mathcal{L}_{\text{batch}} + \text{ListMLE}(\text{sort}(S_{ij}); \theta)$
  end for
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{batch}}$
until convergence
return $\mathcal{M}_\theta$

4.3 Specification Metric Construction

Section 4.2 provides the reuse experience $\{S_{ij}\}$, from which we learn the specification mapping $\mathcal{M}_\theta$. Directly optimizing the objective in Eq. (5) is intractable, as the discrete identification step is non-differentiable in θ and the reuse loss is available only on the pseudo-user tasks. We therefore formulate the learning of $\mathcal{M}_\theta$ as a ranking problem, aiming to place candidate learnwares that perform better after reuse closer to each pseudo-user task in the evolved space.

Specification mapping. We instantiate the mapping $\mathcal{M}_\theta : \mathcal{S} \to \mathcal{S}'$ from Section 4.1 to act point-wise on the reduced set of a specification. For a specification $s_j = \{(\beta_{jk}, \boldsymbol{z}_{jk}, y_{jk})\}_{k=1}^{n_j}$, it transforms each point $\boldsymbol{z}_{jk}$ while preserving the weights and labels:

$$s_j' = \mathcal{M}_\theta(s_j) = \{(\beta_{jk}, \mathcal{M}_\theta(\boldsymbol{z}_{jk}), y_{jk})\}_{k=1}^{n_j}. \tag{8}$$

The evolved specification s_j' thus retains the set structure of the RKME specification while residing in the reusability-aware specification space $\mathcal{S}'$.

Reusability-aware distance. In the evolved space, we measure the dissimilarity between a task requirement s_i and a candidate learnware's specification s_j by the distance between their evolved specifications:

$$d_\theta(s_i, s_j) = d(s_i', s_j'), \tag{9}$$

where $d(\cdot, \cdot)$ is the MMD distance between evolved reduced sets. The mapping $\mathcal{M}_\theta$ is trained so that smaller distances correspond to lower reuse loss S_{ij}. Together, the mapping $\mathcal{M}_\theta$ and the induced distance d_θ define the reusability-aware specification metric.

Ranking objective. The goal is to learn $\mathcal{M}_\theta$ such that candidate learnwares with lower reuse loss are placed closer to each pseudo-user task in the evolved space. We formulate this as a learning-to-rank problem, since identification selects learnwares by their distance ranking rather than by absolute distances. Specifically, we use the negative distance $-d_\theta(s_i, s_j)$ as the ranking score so that a smaller distance ranks a candidate learnware higher. For a pseudo-user task l_i with specification s_i, the system retrieves candidate learnwares C_i nearest to $\mathcal{M}_\theta(s_i)$ in the evolved space, and we denote by π^* the ranking that sorts them by ascending reuse loss $\{S_{ij}\}_{j \in C_i}$. We train the mapping $\mathcal{M}_\theta$ to match this ranking with the ListMLE loss [27], which maximizes the likelihood of π^* under the ranking scores $-d_\theta$:

$$P(\pi^* \mid s_i; \theta) = \prod_{k=1}^{|C_i|} \frac{\exp(-d_\theta(s_i, s_{\pi_k^*}))}{\sum_{j=k}^{|C_i|} \exp(-d_\theta(s_i, s_{\pi_j^*}))}. \tag{10}$$

To emphasize the most reusable candidate learnwares, we focus the loss on the top-κ positions of π^*:

$$\mathcal{L}_{\text{rank}}(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{\kappa} \log \frac{\exp(-d_\theta(s_i, s_{\pi_k^*}))}{\sum_{j=k}^{|C_i|} \exp(-d_\theta(s_i, s_{\pi_j^*}))}. \tag{11}$$

Minimizing $\mathcal{L}_{\text{rank}}$ over all pseudo-user tasks yields the mapping $\mathcal{M}_\theta$, which defines the reusability-aware specification space $\mathcal{S}'$.

Generalization to unseen users. The specification mapping $\mathcal{M}_\theta$ is optimized over the task distribution $\mathcal{T}$ within the system (Eq. (5)), so its generalization to unseen user tasks depends on how well the pseudo-user tasks cover future user tasks. As the learnware dock system accommodates learnwares from increasingly diverse tasks, the pseudo-user task distribution becomes more comprehensive, providing a broader basis for learning $\mathcal{M}_\theta$. Provided $\mathcal{T}$ is representative of future user tasks, the learned mapping that ranks candidate learnwares by reuse performance is expected to generalize to unseen user tasks, in line with the motivation that learned reusability metrics can transfer across tasks [4].

Discussion. Ranking each pseudo-user task against all N candidate learnwares costs $O(N)$ per task and $O(N^2)$ across the system, which becomes prohibitive for large-scale learnware dock systems. Since each evolved specification $s_i' = \mathcal{M}_\theta(s_i)$ remains a reduced set and d_θ is the RKME distance between such sets, the learnwares can be organized in an RKME cover tree [11] under d_θ, whose insertion and neighbor retrieval both cost $O(\log N)$. Each task is then ranked only against its retrieved neighbors, and the metric is updated based on this neighborhood. This not only avoids exhaustive search over the entire system but also focuses the metric update on the most informative candidate learnwares: the retrieved neighbors include both learnwares that are truly beneficial after reuse and hard negatives, i.e., those that the current specification metric ranks highly but perform poorly after reuse. As the specification metric evolves, the tree is updated via $O(\log N)$ re-insertion, keeping the organization aligned with the current specification metric and tractable as the system scales.

Algorithm 2 User-Level Deployment

Input: User data D_u, Learnware dock system $\mathcal{L} = \{l_j\}_{j=1}^N$, Specification mapping $\mathcal{M}_\theta$, Reuse strategy $\mathcal{R}$
Output: Predictions on D_u

Requirement generation:
  Generate task requirement s_u from D_u using the RKME_L mechanism
Identification:
  Map s_u to the evolved space: $s_u' \leftarrow \mathcal{M}_\theta(s_u)$
  Identify $l^\star \leftarrow \arg\min_{l_j \in \mathcal{L}} d(s_u', \mathcal{M}_\theta(s_j))$
return $\mathcal{R}(\cdot; l^\star)$ on the user data D_u

Table 1: Accuracy (%) (mean ± std) on user data's ground-truth labels. The best performance is emphasized in bold.

Scenario	Lightgbm	TabPFN	Random	RKME-task	RKME_L-task	GW	Ours
Bank	87.81 ± 0.56	88.94 ± 0.29	89.27 ± 0.00	88.92 ± 0.72	89.35 ± 0.12	**89.37 ± 0.32**	89.33 ± 0.27
Airlines	57.10 ± 1.55	57.95 ± 1.98	64.49 ± 0.00	**65.23 ± 0.81**	63.59 ± 0.55	64.86 ± 0.08	63.93 ± 0.81
Credit-a	83.03 ± 0.56	85.05 ± 0.39	83.11 ± 0.00	84.26 ± 0.08	84.39 ± 0.22	84.04 ± 0.92	**85.25 ± 0.20**
Insp-6	81.80 ± 0.87	83.38 ± 1.27	84.31 ± 0.00	34.52 ± 0.00	34.52 ± 0.00	88.54 ± 0.83	**88.80 ± 0.26**
Insp-12	80.05 ± 1.90	85.70 ± 2.10	88.42 ± 0.00	**90.03 ± 0.00**	89.56 ± 0.00	89.61 ± 0.60	89.33 ± 2.07
Census	85.29 ± 0.79	**88.13 ± 1.25**	81.19 ± 0.00	86.99 ± 0.37	86.78 ± 0.42	87.56 ± 0.46	88.09 ± 1.35
Income	86.17 ± 0.39	**87.63 ± 0.26**	83.34 ± 0.00	84.95 ± 0.00	85.31 ± 0.30	83.41 ± 0.61	85.87 ± 2.11
HR-Analytics	78.83 ± 1.08	78.68 ± 0.83	77.82 ± 0.00	**79.46 ± 1.25**	78.33 ± 0.78	77.06 ± 1.20	78.99 ± 1.72
Average	80.01	81.93	81.49	76.80	76.48	83.06	**83.70**
W/T/L	7/0/1	6/0/2	7/0/1	5/0/3	6/0/2	5/0/3	-
Avg Rank	5.50	3.62	5.12	3.50	4.12	3.50	**2.50**

Table 2: RMSE (mean ± std) on user data's ground-truth labels. The best performance is emphasized in bold.

Scenario	Lightgbm	TabPFN	Random	RKME-task	RKME_L-task	GW	Ours
PFS	0.7002 ± 0.0594	**0.5777 ± 0.0646**	0.6300 ± 0.0000	0.6470 ± 0.0011	0.6400 ± 0.0164	0.6428 ± 0.0028	0.6209 ± 0.0157
M5	0.8059 ± 0.0074	**0.7963 ± 0.0056**	2.66 ± 0.00	5.42 ± 0.27	10.51 ± 0.23	4.86 ± 0.31	1.91 ± 0.87
Cook	0.6197 ± 0.0088	0.5863 ± 0.0080	0.5736 ± 0.0000	0.5848 ± 0.0000	0.5848 ± 0.0000	0.5692 ± 0.0000	**0.5664 ± 0.0046**
Route-0	0.1980 ± 0.0030	0.1788 ± 0.0052	0.1668 ± 0.0000	**0.1625 ± 0.0000**	**0.1625 ± 0.0000**	0.1669 ± 0.0010	0.1659 ± 0.0018
Route-1	0.1990 ± 0.0025	0.1774 ± 0.0047	0.1671 ± 0.0000	**0.1623 ± 0.0000**	**0.1623 ± 0.0000**	0.1675 ± 0.0005	0.1641 ± 0.0022
House	**0.2490 ± 0.0090**	0.2516 ± 0.0127	0.3337 ± 0.0000	0.2932 ± 0.0076	0.3251 ± 0.0165	0.3269 ± 0.0096	0.3106 ± 0.0228
Sales	0.2411 ± 0.0061	**0.2093 ± 0.0054**	0.3426 ± 0.0000	0.4127 ± 0.0099	0.4226 ± 0.0096	0.3774 ± 0.0339	0.3702 ± 0.0335
Average	0.4304	**0.3968**	0.6963	1.10	1.83	1.0158	0.5869
W/T/L	4/0/3	3/0/4	6/0/1	4/0/3	5/0/2	7/0/0	-
Avg Rank	4.71	3.29	4.00	4.07	4.36	4.71	**2.86**

4.4 Overall Workflow

This subsection summarizes the complete workflow, consisting of a system-level organization phase in the submitting stage and a user-level deployment phase in the deploying stage.

System-Level Organization. Algorithm 1 outlines the submitting stage in which the system learns the reusability-aware specification mapping $\mathcal{M}_\theta$. Iterating through hosted learnwares as pseudo-user tasks (Section 4.2), the system obtains the reuse experience $\{S_{ij}\}$ under the fixed reuse strategy $\mathcal{R}$, and optimizes θ so that the induced distances rank candidate learnwares by their reuse performance. This encodes the system's collective reuse experience into the mapping $\mathcal{M}_\theta$, organizing the system so that a smaller distance in the reusability-aware specification space $\mathcal{S}'$ indicates higher expected reuse performance.

User-Level Deployment. Algorithm 2 details the deploying stage. When a user submits a task requirement s_u, the system maps it to the reusability-aware specification space $s'_u = \mathcal{M}_\theta(s_u)$, identifies the nearest learnware $l^\star$ by distance to s'_u, and the user applies the recommended learnware $l^\star$ on her task under the reuse strategy $\mathcal{R}$. Notably, the user benefits from the system's organized reuse experience without participating in the system-level organization.

5 Experiments

In this section, we conduct experiments to validate the effectiveness of the proposed approach. We first detail the experimental setup, including the scenario construction, evaluation metrics, contender methods, and configuration details in Section 5.1. We then present and analyze the empirical results in the following sections.

5.1 Experiment Setup

Scenario construction. We evaluate our approach across diverse real-world classification and regression tasks from the Tabzilla [16] and TabReD [19] benchmarks. Representative scenarios include Census, Airlines, and Postures for classification (e.g., predicting income levels, flight delays, and human movements); and M5, Route, and House for regression (e.g., forecasting product sales, travel times, and property prices). These datasets vary significantly in scale, with instances ranging from approximately 400 to over 1.0 million and feature dimensions spanning from 6 to 985.

For each dataset, we naturally partition the data based on categorical attributes (e.g., store locations, census regions, or geographic blocks), where each partition represents a distinct but related task. We randomly select a majority of these partitions to construct the learnware dock system, while holding out the remaining partitions as user tasks for evaluation. For each partition in the system, we train a LightGBM model under a fixed hyperparameter search space

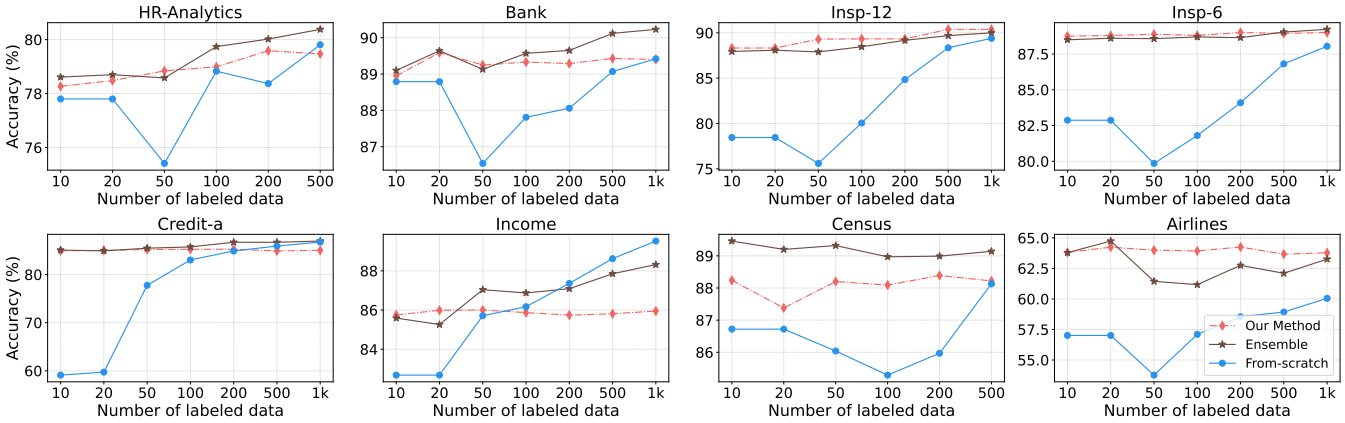


Figure 3: Classification accuracy as the number of user labeled instances varies. Each subplot corresponds to one scenario. `Our Method` denotes the identified learnware, `LightGBM` denotes the user self-trained model, and `Ensemble` combines their predictions.

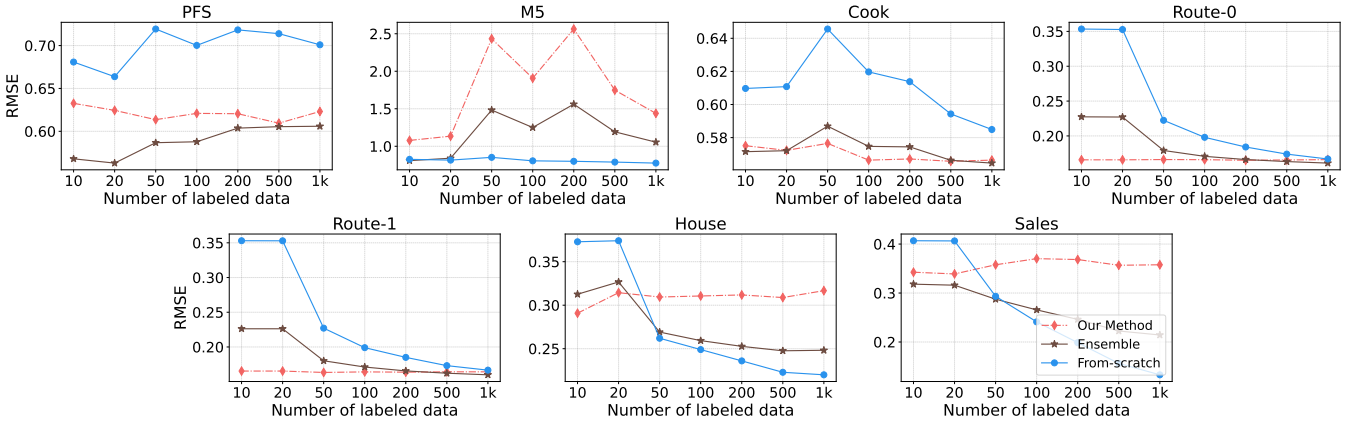


Figure 4: RMSE as the number of user labeled instances varies. Each subplot corresponds to one scenario. `Our Method` denotes the identified learnware, `LightGBM` denotes the user self-trained model, and `Ensemble` combines their predictions.

(see Appendix C). Each model and its corresponding RKME_L specification are submitted together as a learnware. For each held-out user task, all compared methods are evaluated with the same amount of labeled data: the user generates an RKME_L task requirement from a limited set of labeled instances and either reuses the identified learnware or trains a model from scratch. Both the classification and regression tables report results with 100 labeled instances per user task. Figures 3 and 4 further show how performance changes as the number of labeled instances varies.

Evaluation. We use classification accuracy and root-mean-square error (RMSE) as the evaluation metrics for classification and regression tasks, respectively, computed on the test data of each user task. For each scenario, we report the average performance across all held-out user tasks. The test instances of each user task are unseen by all learnwares in the system. All experiments are repeated 5 times with different random seeds, and we report the mean and standard deviation.

Contenders. We compare `Ours` with two categories of contenders. The first category consists of user self-trained methods that build a model from scratch using the user's own labeled data: the widely used gradient boosting decision tree `LightGBM`[9], and the prior-data fitted network `TabPFN`[7, 8], which performs in-context learning in a single forward pass without task-specific training. The second category consists of model hub-based methods, which identify learnwares from the system by comparing specifications and reuse them for the user task instead of training a new model. `Random` selects a learnware uniformly at random, included to confirm that effective identification is necessary. `RKME-task`[26] matches the marginal feature distribution through the MMD distance between RKME specifications. $\texttt{RKME}_\text{L}\texttt{-task}$[11] additionally matches the class-conditional distributions through the richer RKME_L specification. `GW` [17] compares specifications using the Gromov-Wasserstein distance, which captures structural patterns and is robust to feature permutations. These contenders all rely on specification similarity for identification, while our approach uses the reusability-aware specification metric learned from the system's reuse experience.

Configurations. All learnware specifications and user task requirements are generated as RKME_L specifications. The specification size is set as the piece-wise constant function $m = O(\log n)$ where n is the training data size, and we use a Gaussian kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\gamma\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2)$ with $\gamma = 0.1$. For all model hub-based methods including `Ours`, the identified learnware is directly applied to the user's task for prediction. For the proposed approach, the specification mapping $\mathcal{M}_\theta$ is implemented as a residual MLP and trained with the top-κ ListMLE loss under early stopping. Detailed hyperparameters are provided in Appendix C.

5.2 Performance on User Tasks

Tables 1 and 2 compare the performance of our approach with contenders on classification and regression tasks, respectively. Our approach achieves the lowest average rank across both settings, demonstrating the effectiveness of reusability-aware specification evolvement for learnware identification.

Among model hub-based methods, `RKME-task` and RKME_L`-task` identify learnwares by matching specification similarity but do not account for how learnwares will perform after reuse. On certain scenarios, specification similarity leads to suboptimal identification: on Insp-6, `RKME-task` achieves only 34.52% accuracy, well below `Random` (84.31%); on M5, `RKME-task` yields an RMSE of 5.42, more than double the error of `Random` (2.66), and RKME_L`-task` performs even worse (10.51), showing that richer specifications without reuse awareness can lead to worse identification. Our approach achieves 88.80% on Insp-6 and 1.91 on M5, identifying learnwares that specification similarity would overlook. `GW` is competitive among model hub-based contenders due to its ability to capture structural patterns, but it still does not account for reuse performance. By evolving specifications to reflect reuse performance, our approach outperforms all model hub-based contenders in average rank.

Compared with self-trained methods, leveraging well-trained models from the learnware dock system generally yields better performance under limited labeled data, as reflected by the lower average rank of our approach over `LightGBM` and `TabPFN`. On individual scenarios, `TabPFN` can be competitive (e.g., Census and Income for classification, PFS and Sales for regression). When the user's local data is already highly informative for the task, self-trained methods may achieve the best performance (e.g., House for regression), as system-identified learnwares provide less additional benefit in such cases.

5.3 Evaluation with Different Labeled Data Sizes

We now analyze how performance changes as the number of user labeled instances varies. Figures 3 and 4 illustrate the trends for classification and regression tasks, comparing the identified learnware (`Our Method`) against a model trained from scratch (`LightGBM`) and their ensemble (`Ensemble`), which combines the predictions of both.

With as few as 10 labeled instances, `Our Method` outperforms `LightGBM` on all 8 classification scenarios and 6 out of 7 regression scenarios (e.g., on Credit-a, 84.9% vs 59.1%; on Route, RMSE 0.166 vs 0.353). As the amount of labeled data increases, `LightGBM` benefits from more supervision and the gap narrows; at 500 labels, `LightGBM` catches up on about one-third of classification scenarios. The advantage of system-identified learnwares is larger when user supervision is limited. For regression, `Our Method` maintains a consistent advantage on scenarios such as Route and Cook, where the system's reuse experience may capture task structure that limited local data alone does not provide.

We further evaluate whether system-identified learnwares can complement user-trained models by ensembling their predictions with those of a `LightGBM` model trained on the user's labeled data. For classification, the ensemble outperforms `LightGBM` alone on nearly all scenario-and-data-size combinations, indicating that the reuse experience encoded in the identified learnware provides complementary knowledge to local supervision. For scenarios such as Cook and Insp-6, the identified learnware alone outperforms the self-trained model across all data sizes.

Table 3: Average rank of specification mapping and learning objective variants under the 100-label setting. The best result is emphasized in bold.

Variant	Map.	Objective	Avg Rank
`RKME-task`	–	Spec. distance	3.87
RKME_L`-task`	–	Spec. distance	4.53
`Linear+Reg`	Linear	Reg. on S_{ij}	3.57
`MLP+Reg`	MLP	Reg. on S_{ij}	2.83
`Linear+ListMLE`	Linear	top-κ ListMLE	3.53
`Full`	MLP	top-κ ListMLE	**2.67**

5.4 Ablation Study

To investigate the impact of the specification mapping and the learning objective, we evaluate variants that change either component while keeping the same learnware dock, reuse strategy, evaluation protocol, and 100-label setting. As shown in Table 3, we include `RKME-task` and RKME_L`-task` as baselines that identify learnwares directly from the original specification space without reusability-aware evolvement.

`Linear+Reg` replaces the residual MLP mapping with a linear transformation and replaces the ListMLE objective with a regression loss that directly fits the surrogate reuse scores S_{ij}. `MLP+Reg` keeps the nonlinear mapping but uses regression instead of ranking. `Linear+ListMLE` keeps the ranking objective but uses only a linear specification mapping. The full method combines the residual MLP mapping with top-κ ListMLE.

The full method obtains the best average rank among all variants. Comparing `Linear+Reg` with `MLP+Reg`, and `Linear+ListMLE` with the full method, shows that nonlinear specification evolvement improves identification under both supervision forms. In contrast, replacing regression with ranking supervision under a linear mapping (`Linear+Reg` vs `Linear+ListMLE`) yields little difference, suggesting that the ranking objective is most effective when combined with an expressive specification mapping. The full method further improves over `MLP+Reg`, showing that the ranking objective provides additional benefit on top of the nonlinear mapping by directly optimizing the ordering of candidate learnwares. Together, these results indicate that nonlinear specification evolvement is the

Table 4: Average rank under different top-κ values (same setting as Table 3).

κ	1	3	5	10	all
Avg Rank	3.27	2.95	**2.59**	3.18	3.00

primary contributor, while ranking supervision provides a complementary improvement for encoding the system's reuse experience into the specification metric.

5.5 Sensitivity to Top-κ

The ListMLE objective is applied to the top-κ candidate learnwares in the pseudo-user reuse ranking. This choice controls how much the learned specification metric focuses on highly reusable learnwares. A very small κ emphasizes only the best candidate learnware and can be unstable when the top candidate changes across seeds or when reuse scores are close. A very large κ approaches full-list ranking and spends optimization effort on low-ranked candidates that are unlikely to be selected at deployment. We therefore study $\kappa \in \{1, 3, 5, 10, \text{all}\}$ while keeping the same mapping architecture and evaluation setting.

Table 4 shows that the default $\kappa = 5$ gives the best average rank. $\kappa = 3$ yields a similar rank, suggesting that the method is not highly sensitive to the exact value as long as the objective remains focused on the leading candidates. This aligns with the design of top-κ ListMLE: the specification metric should preserve the ordering of the most reusable candidate learnwares rather than the full ranking of all learnwares in the system.

6 Related Work

The learnware paradigm [31, 32] leverages specifications to identify and reassemble well-trained models without accessing original training data, giving rise to emergent system capabilities. Central to this paradigm is the specification, which characterizes model capabilities without disclosing original data, enabling specification-based identification [26, 32]. Along this direction, Liu et al. [11] proposed evolving specifications by leveraging task information within hosted learnwares, computing each model's performance on other learnwares' specifications to capture capabilities beyond its original training task, and constructing specification indexes for efficient identification. Tan et al. [22] extended the specification by incorporating conditional distributions alongside marginal distributions, enabling the system to build a unified subspace across heterogeneous feature spaces and improve identification through conditional distribution matching. Lei et al. [10] theoretically established that the RKME specification scarcely contains original data and robustly defends against common inference attacks, providing a privacy foundation for the paradigm. More recently, learnware specifications have been extended toward unstructured data and language-model settings: the PAVE specification [20] encodes model capabilities through parameter changes relative to a shared pre-trained model for identifying learnwares from text and images, language-model learnwares [25] show how specialized small language models can be managed under the learnware paradigm, and constructive specification [14] builds hierarchical capability representations for plug-and-play identification of learnware agents without accessing agent internals. Research on the paradigm has also advanced along several complementary directions, including efficient identification via anchor-based retrieval [28], dynamic filtering of redundant learnwares [12], heterogeneous label spaces [6, 13], heterogeneous feature spaces without auxiliary data [23], and learnware specification generation via dual alignment [3]. Based on these advances, Beimingwu [24] has been released as the first learnware dock system, providing a research platform for the full submission-to-deployment process. Existing identification methods generally use task similarity as the identification criterion. Our work follows the specification evolvement perspective of Liu et al. [11], but focuses on making identification reflect expected reuse performance under a given reuse strategy.

Estimating whether a candidate model can be reused for a target task without exhaustively adapting all candidates has also been studied outside the learnware paradigm. Some methods design fine-tuning-free transferability measures, such as LogME [29], which estimates transferability from model-extracted features and labels on the target task, and Task2Vec [1], which represents a task using gradients of a probe network. Others learn to rank candidate models from historical ranking supervision, such as Model Spider [30]. Ding et al. [4] studied model reusability estimation in small-data transfer learning, addressing the challenge of selecting pre-trained models when target data is scarce. In contrast to these target-task-specific assessments, which rely on target-task data, model-derived representations, gradients, or historical transferability supervision, we treat hosted learnwares as pseudo-user tasks and evaluate reuse performance across these tasks, obtaining system-wide reuse experience that is encoded into a specification metric so that identification for future user tasks reflects expected reuse performance.

7 Conclusion

This paper evolves learnware specifications using the reuse experience among hosted learnwares, so that identification reflects expected reuse performance rather than distributional similarity alone. By treating each hosted learnware as a pseudo-user task, the system evaluates reuse performance across diverse tasks under a fixed reuse strategy. The resulting reuse experience is encoded into a reusability-aware specification space through a learned specification mapping, where specification distances reflect expected reuse performance. Experiments validate the proposed approach, showing that the system can effectively identify learnwares that benefit new user tasks after reuse, including those that original task similarity would overlook.

Acknowledgments

This work was supported by FIDBP of MoE (JYB2025XDXM118). Zhi-Hao Tan was supported by the Postdoctoral Fellowship Program of CPSF (GZB20250396) and Jiangsu Funding Program for Excellent Postdoctoral Talent (2025ZB482). The authors would like to thank Jia-Wei Shan, Zi-Chen Zhao and Hao-Yi Lei for helpful discussions. We are also grateful to the anonymous reviewers for their helpful comments.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2Vec: Task Embedding for Meta-Learning. In *Proceedings of the 17th IEEE/CVF International Conference on Computer Vision*. 6429–6438.

[2] Francis R. Bach, Simon Lacoste-Julien, and Guillaume Obozinski. 2012. On the Equivalence between Herding and Conditional Gradient Algorithms. In *Proceedings of the 29th International Conference on Machine Learning*. 1355–1362.

[3] Wei Chen, Jun-Xiang Mao, Xiaozheng Wang, and Min-Ling Zhang. 2025. Learnware Specification via Dual Alignment. In *Proceedings of the 42nd International Conference on Machine Learning*. 8683–8699.

[4] Yao-Xiang Ding, Xi-Zhu Wu, Kun Zhou, and Zhi-Hua Zhou. 2022. Pre-trained model reusability evaluation for small-data transfer learning. In *Advances in Neural Information Processing Systems 35*. 37389–37400.

[5] Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. 2024. TabR: Tabular Deep Learning Meets Nearest Neighbors. In *The 12th International Conference on Learning Representations*. 18209–18249.

[6] Lan-Zhe Guo, Zhi Zhou, Yu-Feng Li, and Zhi-Hua Zhou. 2023. Identifying useful learnwares for heterogeneous label spaces. In *Proceedings of the 40th International Conference on Machine Learning*. 12122–12131.

[7] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The 11th International Conference on Learning Representations*.

[8] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. 2025. Accurate predictions on small data with a tabular foundation model. *Nature* 637, 8045 (2025), 319–326.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*. 3146–3154.

[10] Hao-Yi Lei, Zhi-Hao Tan, and Zhi-Hua Zhou. 2024. On the Ability of Developers' Training Data Preservation of Learnware. In *Advances in Neural Information Processing Systems 37*. 36471–36513.

[11] Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. 2024. Towards Making Learnware Specification and Market Evolvable. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*. 13909–13917.

[12] Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. 2025. Dynamic Learnware Filtering for Efficient Learnware Identification and System Slimming. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1811–1822.

[13] Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. 2025. Identifying and Reusing Learnwares Across Different Label Spaces. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence*. 5734–5742.

[14] Jian-Dong Liu, Zi-Chen Zhao, Hao Sun, Lin-Xing Wu, Huan Zhang, Pengyuan Wang, Ming Zhao, Xinyu Chu, Shu Yan, Yongbei Zhu, Weijun Zhong, Zhi-Hao Tan, Jing Shang, Yang Yu, and Zhi-Hua Zhou. 2026. Constructive Specification for Plug-and-Play Learnware Agents. In *Workshop on Lifelong Agents: Learning, Aligning, Evolving, The 14th International Conference on Learning Representations*.

[15] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *The 7th International Conference on Learning Representations*.

[16] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. 2023. TabZilla: A Suite for Benchmarking Tabular AutoML. In *Advances in Neural Information Processing Systems 36*. 76336–76369.

[17] Facundo Mémoli. 2011. Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics* 11, 4 (2011), 417–487.

[18] Krikamol Muandet, Kenji Fukumizu, Bharath K. Sriperumbudur, and Bernhard Schölkopf. 2017. Kernel Mean Embedding of Distributions: A Review and Beyond. *Foundations and Trends in Machine Learning* 10, 1-2 (2017), 1–141.

[19] Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. 2025. TabReD: Analyzing Pitfalls and Filling the Gaps in Tabular Deep Learning Benchmarks. In *The 13th International Conference on Learning Representations*.

[20] Hao-Yu Shi, Zhi-Hao Tan, Zi-Chen Zhao, Yang Yu, and Zhi-Hua Zhou. 2026. A Study on PAVE Specification for Learnware. In *The 14th International Conference on Learning Representations*.

[21] Alexander J. Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. 2007. A Hilbert Space Embedding for Distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*. 13–31.

[22] Peng Tan, Hai-Tian Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. 2024. Handling Learnwares from Heterogeneous Feature Spaces with Explicit Label Exploitation. In *Advances in Neural Information Processing Systems 37*. 12767–12795.

[23] Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. 2023. Handling learnwares developed from heterogeneous feature spaces without auxiliary data. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*. 4235–4243.

[24] Zhi-Hao Tan, Jian-Dong Liu, Xiao-Dong Bi, Peng Tan, Qin-Cheng Zheng, Hai-Tian Liu, Yi Xie, Xiao-Chuan Zou, Yang Yu, and Zhi-Hua Zhou. 2024. Beimingwu: A Learnware Dock System. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5773–5782.

[25] Zhi-Hao Tan, Zi-Chen Zhao, Hao-Yu Shi, Xin-Yu Zhang, Peng Tan, Yang Yu, and Zhi-Hua Zhou. 2025. Learnware of Language Models: Specialized Small Language Models Can Do Big. *arXiv preprint arXiv:2505.13425* (2025).

[26] Xi-Zhu Wu, Wenkai Xu, Song Liu, and Zhi-Hua Zhou. 2023. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2023), 699–710.

[27] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*. 1192–1199.

[28] Yi Xie, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. 2023. Identifying helpful learnwares without examining the whole market. In *Proceedings of the 26th European Conference on Artificial Intelligence*. 2752–2759.

[29] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. LogME: Practical Assessment of Pre-trained Models for Transfer Learning. In *Proceedings of the 38th International Conference on Machine Learning*. 12133–12143.

[30] Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. 2023. Model Spider: Learning to Rank Pre-Trained Models Efficiently. In *Advances in Neural Information Processing Systems 36*. 13692–13719.

[31] Zhi-Hua Zhou. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science* 10, 4 (2016), 589–590.

[32] Zhi-Hua Zhou and Zhi-Hao Tan. 2024. Learnware: small models do big. *Science China Information Sciences* 67, 1 (2024), 112102.

A Proof of Theorem 4.1

Here we provide the proof for Theorem 4.1. We aim to bound the difference between the surrogate reuse loss $S_{ij} = \widetilde{\mathcal{L}}_{s_i}(l_j, \mathcal{R})$ and the true reuse loss $\mathcal{L}_{\mathcal{D}_i}(l_j, \mathcal{R})$.

The true reuse loss is $\mathcal{L}_{\mathcal{D}_i}(l_j, \mathcal{R}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_i}[\ell(\mathcal{R}(\boldsymbol{x}; f_j), h_i(\boldsymbol{x}))]$. The surrogate reuse loss is $S_{ij} = \sum_{k=1}^{n_i} \beta_{ik}\ell(\mathcal{R}(\boldsymbol{z}_{ik}; f_j), f_i(\boldsymbol{z}_{ik}))$, computed on the RKME specification s_i using f_i as labels.

Using the triangle inequality of ℓ, we decompose the error as:

$$\begin{aligned}&\left|\mathcal{L}_{\mathcal{D}_i}(l_j, \mathcal{R}) - S_{ij}\right| \\ &\leq \underbrace{\mathbb{E}_{\mathcal{D}_i}[\ell(f_i, h_i)]}_{\text{(I) Model error}} + \underbrace{\left|\mathbb{E}_{\mathcal{D}_i}[g_{ij}] - S_{ij}\right|}_{\text{(II) RKME approx. error}}.\end{aligned} \tag{12}$$

Term (I): By assumption, $\mathbb{E}_{\mathcal{D}_i}[\ell(f_i(\boldsymbol{x}), h_i(\boldsymbol{x}))] \leq \epsilon_i$.

Term (II): This is the error of approximating the expectation of g_{ij} over $\mathcal{D}_i$ using the RKME specification. By the convergence of the empirical KME to the population KME [18] and the convergence of the RKME to the empirical KME [12], combined with the uniform boundedness of $\|g_{ij}\|_{\mathcal{H}_k}$, we have (II) $\leq O(m_i^{-\frac{1}{2}} + n_i^{-\frac{1}{2}})$.

Combining (I) and (II):

$$\left|\mathcal{L}_{\mathcal{D}_i}(l_j, \mathcal{R}) - S_{ij}\right| \leq \epsilon_i + O(m_i^{-\frac{1}{2}} + n_i^{-\frac{1}{2}}). \tag{13}$$

This completes the proof.

B Notations

The major notations used in this work are summarized in Table 5.

C Experiment Details

Model training details. Tree-based models are effective and efficient for tabular tasks, so we use `LightGBM` [9] to train all developer models. The same model family is also used as the user self-trained baseline. The hyperparameter search space consists of combinations over `learning_rate`, `num_leaves`, and `max_depth`: $(0.015, 224, 66)$, $(0.005, 300, 50)$, $(0.01, 128, 80)$, $(0.15, 224, 80)$, and $(0.01, 300, 66)$.

Table 5: Major notations used in this work.

Category	Notation	Description
Learnware System	$l_i = (f_i, s_i)$	The i-th learnware, consisting of a model f_i and its RKME_L specification s_i.
	$\mathcal{D}_i$	The data distribution of the i-th developer's task over the input space $\mathcal{X}$.
	$\mathcal{L} = \{l_i\}_{i=1}^N$	The learnware dock system accommodating N learnwares.
	$\mathcal{T}$	The distribution of tasks induced by the hosted learnwares.
User Task	$\mathcal{D}_u$	Data distribution of the user task.
	s_u	The user task requirement, generated as an RKME_L specification.
Reuse & Eval.	$\mathcal{R}(\boldsymbol{x}; l)$	Prediction by reusing learnware l on input $\boldsymbol{x}$ under fixed $\mathcal{R}$.
	$\mathcal{L}_{\mathcal{D}_i}(l, \mathcal{R})$	The expected reuse loss of learnware l under $\mathcal{R}$ on distribution $\mathcal{D}_i$.
	S_{ij}	Surrogate reuse loss of l_j on pseudo-user task l_i, i.e., $\widetilde{\mathcal{L}}_{s_i}(l_j, \mathcal{R})$.
Spec. Evolve.	$\mathcal{M}_\theta$	The learned specification mapping from the original space $\mathcal{S}$ to the reusability-aware space $\mathcal{S}'$.
	s'_j	Evolved specification of l_j, i.e., $\mathcal{M}_\theta(s_j)$.
	$d_\theta(s_i, s_j)$	The reusability-aware distance between evolved specifications in $\mathcal{S}'$.

Specification mapping details. The specification mapping $\mathcal{M}_\theta$ is implemented as a stack of 2 residual blocks [5], each consisting of a linear layer, batch normalization, ReLU activation, and a dropout rate of 0.1, with a skip connection. The main embedding dimension is 64 with a hidden expansion factor of 2. The mapping is trained with AdamW [15] using a learning rate of 10^{-3}, weight decay of 10^{-4}, and a cosine learning rate schedule for 50 epochs. Early stopping is based on NDCG@5 with a patience of 15. The ListMLE loss is focused on the top-5 candidate learnwares.

Baseline details. For `TabPFN` [7, 8], we use the official checkpoint and perform in-context learning on the user's labeled data. When the user's labeled data exceeds 1,000 instances or 100 features, we randomly subsample up to 1,000 instances and 100 features, repeating this process three times, and average the predictions from these runs.

Computational resources. Experiments were conducted on a server with a Tesla A100 80GB GPU, two Intel Xeon Platinum 8358 CPUs (32 cores each, 2.6 GHz base, 3.4 GHz turbo), and 512 GB of RAM.