GRADIENT PARAMETER SPECIFICATIONS FOR LEARNWARE IDENTIFICATION AND REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

The *learnware paradigm* enables users to solve new tasks by identifying and reusing learnwares from a learnware dock system, where each learnware pairs a well-established model with a specification of its reuse-relevant capabilities. However, existing specification methods primarily summarize data distributions or feature statistics, limiting their ability to distinguish tasks with similar inputs but different predictive requirements. To address this limitation, we propose **G**radient **P**arameter **S**pecification (GPS), a learnware specification that encodes task-dependent discriminative structure and model-dependent predictive behavior through class-wise one-step gradient directions. Specifically, GPS optimizes a small proxy sample set to construct learnware specifications from developer tasks and model capabilities, analogously constructs user specifications from labeled user tasks, and identifies reusable learnwares through partial class-wise gradient direction matching. Our representation-level analysis characterizes how gradient responses depend jointly on input representations and predictive targets, and explains how class-wise responses mitigate cross-class cancellation. Extensive experiments on twelve image and text benchmarks across three reuse scenarios show that GPS achieves the highest average performance among data-centric specification methods.

1 INTRODUCTION

Machine learning has achieved remarkable success across domains ranging from computer vision and natural language processing to scientific discovery and decision-making (Krizhevsky et al., 2012; Brown et al., 2020). However, developing a high-quality model for a new task still requires abundant labeled data, substantial computation, and significant modeling expertise (Kaplan et al., 2020; Bommasani et al., 2021), making it prohibitively expensive for many users. Reusing existing models is equally challenging due to data privacy, heterogeneous architectures, limited task-specific supervision, and catastrophic forgetting. Consequently, enabling users to leverage existing model capabilities without accessing training data or retraining models remains a fundamental challenge.

The *learnware paradigm* (Zhou, 2016; Zhou & Tan, 2024) offers a promising alternative by organizing *well-established models* in a learnware dock system for effective reuse on new tasks. Each *learnware* pairs a well-established model with a compact specification characterizing its reuse-relevant capabilities (Lei et al., 2026a). Given a user task, the dock system identifies suitable learnwares by matching the user requirement against stored learnware specifications and reuses either a single learnware or multiple complementary learnwares when the task extends beyond the original functionality of any individual learnware (Liu et al., 2026; Tan et al., 2024a). Since the dock system cannot access developers' private data or evaluate every stored model on each user task, specification quality directly determines identification and reuse quality (Lei et al., 2024). This paradigm therefore raises a fundamental question: *How can we construct efficient specifications that characterize both task properties and model capabilities, while enabling reliable learnware identification and reuse across heterogeneous models in the dock system?*

Existing specifications typically summarize the input distributions (Wu et al., 2023; Liu et al., 2025b) or feature statistics (Chen et al., 2025b;a) of tasks mastered with well-established models. While effective when distributional similarity reflects reuse compatibility, such data-centric summaries may overlook the predictive behavior of the underlying model. Tasks with similar inputs can differ sub-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

stantially in label semantics or decision rules, and thus require different model capabilities. For example, the *Aquatic Mammals* and *Fish* superclasses of CIFAR-100 share similar visual patterns but exhibit distinct class structures. As shown in Fig. 1(a), distribution discrepancies provide limited separation between matched and semantically related yet mismatched task pairs. An effective specification should therefore capture not only where the task data lie, but also how the associated model responds to them—information naturally reflected by gradient responses.

To investigate this possibility, we adopt a shared randomly initialized three-layer CNN as a proxy network in our empirical work. Unlike data-centric statistics, gradients jointly reflect input-dependent representations and model-dependent predictive targets, providing a functional signal from a single backward pass through the classification head. As shown in Fig. 1(b), matched learnware–task pairs exhibit highly aligned gradient directions, whereas mismatched pairs show substantially weaker or even opposing alignment. Notably, this discriminative signal emerges without auxiliary training.

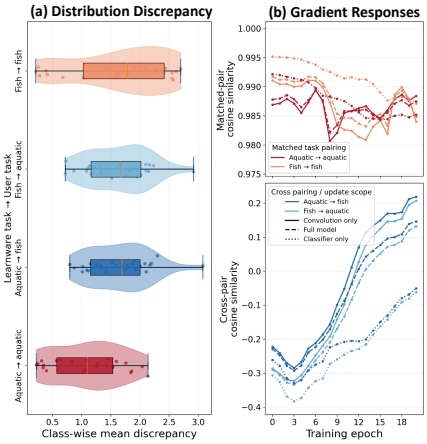

Figure 1: **Distribution discrepancy and gradient responses.** (a) Distribution statistics provide only weak separation between matched and cross-task pairs. (b) Under a shared proxy network, matched pairs exhibit higher gradient similarity than cross-task pairs, and this distinction remains evident even when using only one-step gradients from the classifier.

Motivated by this observation, we propose **G**radient **P**arameter **S**pecification (GPS), which captures task-dependent discriminative structure and model-dependent predictive behavior through class-wise one-step gradient directions. GPS employs a shared proxy network with a fixed feature extractor and a prediction head resampled at each optimization iteration, establishing a common coordinate system for gradient alignment. During submission, it optimizes a compact proxy set to match the class-wise gradients induced by private developer data and model predictions, yielding the learnware specification. During deployment, it constructs the user specification from labeled task data and performs partial class-wise matching to identify complementary learnwares and establish class–model correspondences for reuse. Our contributions are summarized as follows:

- We propose GPS, a learnware specification that encodes task-dependent discriminative structure and model-dependent predictive behavior through class-wise one-step gradient directions under a shared proxy network.
- We develop a partial class-wise gradient direction matching mechanism between user and learnware specifications, establishing class–model correspondences to identify complementary learnwares for reuse.
- We provide a representation-level analysis that characterizes how gradient responses depend jointly on input representations and predictive targets, and explains how class-wise responses mitigate cross-class cancellation.
- Experiments on twelve image and text benchmarks across three reuse scenarios demonstrate that GPS achieves the highest average performance among the evaluated data-centric specification methods.

2 Related Work

The learnware paradigm (Zhou, 2016) offers a systematic approach for managing and reusing well-established models, enabling users to solve new tasks without designing and training models from scratch. A key component of this paradigm is the learnware specification (Liu et al., 2026), which serves as a compact capability identifier for each learnware and provides a comparable representation for assessing its relevance to user tasks, thereby facilitating learnware identification and reuse. Existing learnware specification methods are primarily data-centric, characterizing learnwares by sketching the distributions or feature statistics of the tasks mastered by their well-established mod-

els (Zhou & Tan, 2024). Representative approaches include Reduced Kernel Mean Embedding (RKME) (Wu et al., 2023), Label-Aware Neural Embedding (LANE) (Chen et al., 2025b), and their extensions (Liu et al., 2025b; Chen et al., 2025a). Beyond specification design, subsequent studies have extended the learnware paradigm to heterogeneous reuse (Tan et al., 2024b; 2023; 2024a; Guo et al., 2023), unseen-task scenarios (Zhang et al., 2021; Tan et al., 2026), and efficient learnware identification (Xie et al., 2023; Liu et al., 2024; 2025a), while recent theoretical studies have further investigated the foundations of learnware specifications (Lei et al., 2024; 2026a). The release of the Beimingwu learnware dock system (Tan et al., 2024c) further provides a practical platform for learnware management, identification, and reuse. More recently, learnware specifications have characterized model capabilities through task-specific parameter changes in large pre-trained foundation models (Shi et al., 2026; Lei et al., 2026b). While moving beyond data-centric specifications, these approaches rely on large pre-trained foundation models and task-specific adaptation. In contrast, GPS provides a learnware specification without such dependence, characterizing predictive behavior through class-wise one-step gradient directions under a shared proxy network.

3 METHODOLOGY

Problem Setup The learnware paradigm consists of a developer-oriented submitting stage and a user-oriented deploying stage (Tan et al., 2024c). Developer i submits a private task $\mathcal{D}_i = \{(\boldsymbol{x}_{i,n}, y_{i,n})\}_{n=1}^{N_i}$ together with a well-established model $f_{\boldsymbol{\theta}_i}$. The dock system assigns a specification Φ_i and stores the learnware $\{f_{\boldsymbol{\theta}_i}, \Phi_i\}$, while the task data remain private. Given a user task $\hat{\mathcal{D}}$, the dock system constructs a user specification $\hat{\Phi}$ and compares it with the stored learnware specifications $\{\Phi_i\}_{i=1}^{c}$ to identify relevant learnwares for reuse. Thus, the key challenge is to design specifications whose similarity reliably reflects learnware reuse compatibility. Further background on the learnware paradigm and related data-centric specifications is deferred to Appendix A.

Building on the above setup, the proposed GPS encodes task-dependent discriminative structure and model-dependent predictive behavior through class-wise one-step gradient directions in a shared proxy network. Then, the dock system constructs learnware specifications from developer tasks and model predictions, constructs user specifications from labeled user tasks, and identifies reusable learnwares through partial class-wise gradient direction matching.

3.1 GRADIENT PARAMETER SPECIFICATION

The gradient response of a model on a specific task jointly depends on the input representation and the predictive target used to evaluate it (Zhao et al., 2021; Jiang et al., 2023; Ba et al., 2022; Yin et al., 2024; Cazenavette et al., 2025). This property holds even for one-step gradients and is further empirically supported by Fig. 1(b). Motivated by this observation, GPS introduces a shared proxy network $\phi_{\boldsymbol{\xi}} : \mathcal{X} \to \mathcal{Y}$ to provide a common gradient coordinate system across all learnwares and user tasks. Specifically, the shared proxy network is defined as

$$\phi_{\boldsymbol{\xi}} = g_{\boldsymbol{\omega}_0} \circ \psi_{\boldsymbol{\vartheta}}, \qquad \boldsymbol{\xi} = (\boldsymbol{\vartheta}, \boldsymbol{\omega}_0), \tag{1}$$

where $\psi_{\boldsymbol{\vartheta}} : \mathcal{X} \to \mathbb{R}^d$ denotes a globally fixed feature extractor deployed in the dock system, and $g_{\boldsymbol{\omega}_0} : \mathbb{R}^d \to \mathcal{Y}$ denotes a prediction head. At iteration t, we sample $\boldsymbol{\omega}_0^{(t)} \sim P_{\boldsymbol{\omega}}$ and set $\boldsymbol{\xi}^{(t)} = (\boldsymbol{\vartheta}, \boldsymbol{\omega}_0^{(t)})$. The feature extractor remains fixed throughout, while the sampled head is shared across all task–proxy gradient evaluations within the same iteration and resampled at the next iteration. For brevity, we omit the superscript (t) when referring to a single iteration. Thus, $\phi_{\boldsymbol{\xi}}$ serves purely as a probing mechanism rather than an auxiliary model to be trained.

One-Step Gradient Responses. Let $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ denote a well-established model associated with a private developer task $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. GPS represents a task-model pair through the gradient response it induces under a shared proxy network. For the developer task, the prediction of the submitted model, $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, serves as the supervision signal for probing its predictive behavior. Given a mini-batch $\mathbf{B}_{\boldsymbol{x}} \subset \mathcal{D}$, we define the following capability-probing objective under the shared proxy network:

$$\mathcal{L}_{\mathrm{cap}}(\boldsymbol{\omega}; \mathbf{B}_{\boldsymbol{x}}) = \frac{1}{|\mathbf{B}_{\boldsymbol{x}}|} \sum_{\boldsymbol{x} \in \mathbf{B}_{\boldsymbol{x}}} \ell\left(g_{\boldsymbol{\omega}} \circ \psi_{\boldsymbol{\vartheta}}(\boldsymbol{x}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right), \tag{2}$$

where ℓ denotes the cross-entropy loss. Because the supervision target is provided by the submitted model rather than the ground-truth labels, this objective captures both the task inputs and the model's predictive behavior. We characterize their interaction through the resulting one-step gradient with respect to the randomly initialized proxy-head parameters $\boldsymbol{\omega}_0$:

$$\boldsymbol{G}_{\mathcal{D}}(\mathbf{B}_{\boldsymbol{x}};\boldsymbol{\xi}) = \nabla_{\boldsymbol{\omega}}\mathcal{L}_{\text{cap}}(\boldsymbol{\omega};\mathbf{B}_{\boldsymbol{x}})|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}\,. \tag{3}$$

The resulting gradient therefore jointly captures task structure and the predictive behavior of the submitted model. Evaluating all gradient responses under the same proxy network places heterogeneous learnwares in a common gradient space, enabling direct specification comparison. To reduce sensitivity to prediction confidence, representation scale, and batch composition, we discard gradient magnitude and retain only its orientation via the polar factor (Bo & Wang, 2026). For a gradient matrix $\boldsymbol{G}_{\mathcal{D}}$ with thin SVD, we define its direction as the orthogonal polar factor:

$$\text{Dir}(\boldsymbol{G}_{\mathcal{D}}) = \boldsymbol{U}\boldsymbol{V}^{\top}, \qquad s.t. \quad \boldsymbol{G}_{\mathcal{D}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}. \tag{4}$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ have orthonormal columns. By discarding the singular values $\boldsymbol{\Sigma}$, $\text{Dir}(\boldsymbol{G}_{\mathcal{D}})$ removes scale information while preserving the orthogonal structure of the gradient response, thereby making the representation invariant to positive rescaling (Jiang et al., 2023).

Class-Wise Gradient Directions. Aggregating gradients across classes may obscure class-specific responses due to gradient cancellation and class imbalance. We therefore compute gradient responses separately for each class. Let $\mathcal{D}^k := \{(\boldsymbol{x}, y) \in \mathcal{D} \mid y = k\}$ denote the subset associated with class k, where $k \in [K]$. Using the submitted model prediction $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ as the supervision target, the corresponding class-wise gradient is defined as

$$\begin{aligned} &\boldsymbol{G}_{\mathcal{D}^k}(\mathbf{B}^k_{\boldsymbol{x}};\boldsymbol{\xi}) = \nabla_{\boldsymbol{\omega}}\mathcal{L}_{\text{cap}}(\boldsymbol{\omega};\mathbf{B}^k_{\boldsymbol{x}})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}\,, \\ &s.t. \quad \mathcal{L}_{\text{cap}}(\boldsymbol{\omega};\mathbf{B}^k_{\boldsymbol{x}}) = \frac{1}{|\mathbf{B}^k_{\boldsymbol{x}}|}\sum_{(\boldsymbol{x},y)\in\mathbf{B}^k_{\boldsymbol{x}}} \ell\left(g_{\boldsymbol{\omega}}\circ\psi_{\boldsymbol{\vartheta}}(\boldsymbol{x}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right), \end{aligned} \tag{5}$$

where $\mathbf{B}^k_{\boldsymbol{x}} \subseteq \mathcal{D}^k$, and the orthogonal polar factor of its direction is defined as:

$$\text{Dir}\left(\boldsymbol{G}_{\mathcal{D}^k}\right) = \boldsymbol{U}^k(\boldsymbol{V}^k)^{\top} \qquad s.t. \quad \boldsymbol{G}_{\mathcal{D}^k} = \boldsymbol{U}^k\boldsymbol{\Sigma}^k(\boldsymbol{V}^k)^{\top}. \tag{6}$$

Proxy Gradient Alignment in the Common Coordinate System. Directly releasing task gradients may expose sensitive information about the developer data and model behavior. Instead, GPS optimizes a small proxy set whose class-wise gradient directions align with those of the original task in the shared gradient space. The resulting proxy-induced directions provide a compact surrogate for the original functional responses without directly exposing raw developer data or task gradients. Therefore, let the proxy sample set be $\mathcal{S} = \{(\boldsymbol{s}_m, y_{\boldsymbol{s}_m})\}_{m=1}^{M}$, where $M \ll N$ and each proxy sample $\boldsymbol{s}_m \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ at initialization. For each class $k \in [K]$, the proxy gradient response is computed from a mini-batch $\mathbf{B}^k_{\boldsymbol{s}} \subset \mathcal{S}^k$, where $\mathcal{S}^k := \{(\boldsymbol{s}, y_{\boldsymbol{s}}) \in \mathcal{S} \mid y_{\boldsymbol{s}} = k\}$, as

$$\begin{aligned} &\boldsymbol{G}_{\mathcal{S}^k}(\mathbf{B}^k_{\boldsymbol{s}};\boldsymbol{\xi}) = \nabla_{\boldsymbol{\omega}}\mathcal{L}_{\text{cap}}(\boldsymbol{\omega};\mathbf{B}^k_{\boldsymbol{s}})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}\,, \\ &s.t. \quad \mathcal{L}_{\text{cap}}(\boldsymbol{\omega};\mathbf{B}^k_{\boldsymbol{s}}) = \frac{1}{|\mathbf{B}^k_{\boldsymbol{s}}|}\sum_{(\boldsymbol{s},y_{\boldsymbol{s}})\in\mathbf{B}^k_{\boldsymbol{s}}} \ell\left(g_{\boldsymbol{\omega}}\circ\psi_{\boldsymbol{\vartheta}}(\boldsymbol{s}), y_{\boldsymbol{s}}\right). \end{aligned} \tag{7}$$

Correspondingly, $\text{Dir}(\boldsymbol{G}_{\mathcal{S}^k})$ denotes the gradient direction induced by the proxy samples of class k. We optimize $\mathcal{S}$ by minimizing the average angular discrepancy of gradient directions between the task and proxy sets across all classes:

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathcal{L}_{\text{GPS}}(\mathcal{D},\mathcal{S}) = \frac{1}{K}\sum_{k=1}^{K}\left[1 - \cos\left(\text{Dir}(\boldsymbol{G}_{\mathcal{D}^k}), \text{Dir}(\boldsymbol{G}_{\mathcal{S}^k})\right)\right]. \tag{8}$$

Here, $\cos(\cdot,\cdot)$ denotes cosine similarity between the vectorized gradient-direction matrices. At each iteration, $\boldsymbol{G}_{\mathcal{D}^k}$ and $\boldsymbol{G}_{\mathcal{S}^k}$ are computed using the same proxy-head initialization $\boldsymbol{\omega}_0$, ensuring a common gradient space. The proxy head is resampled across iterations, while the feature extractor remains fixed and only the proxy set $\mathcal{S}$ is optimized. It aims to reproduce the class-wise functional responses of the original task rather than its data distribution. As shown in Fig. 2, gradient discrepancy decreases rapidly while data discrepancy remains large, accompanied by increasing prediction similarity. This observation suggests that class-wise one-step gradient alignment can preserve reuse-relevant functional information without reconstructing the original data distribution. Accordingly, the GPS specification is defined as $\Phi^{\text{GPS}} = \{\text{Dir}(\boldsymbol{G}_{\mathcal{S}^k})\}_{k=1}^{K}$, where $\mathcal{S}$ is optimized according to Eq. 8 and $\Phi^{\text{GPS}} \in \mathbb{R}^{K\times d}$. The full optimization procedure is provided in Algorithm 1.

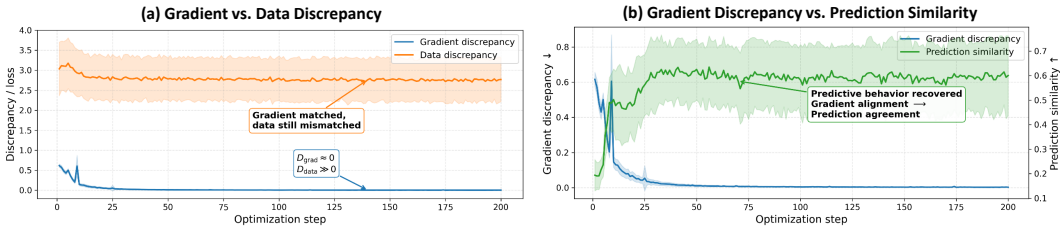

Figure 2: **Gradient alignment preserves predictive behavior without data reconstruction.** (a) Gradient discrepancy rapidly decreases during optimization while data discrepancy remains large. (b) Prediction similarity increases as gradient discrepancy decreases, indicating that class-wise one-step gradient alignment preserves task-relevant predictive behavior.

3.2 Developer-Oriented Submitting Stage

For developer i, the targets in Eq. 8 are given by the predictive distributions of the submitted well-established model $f_{\boldsymbol{\theta}_i}$ on its private task $\mathcal{D}_i$. Optimizing the corresponding proxy set $\mathcal{S}_i$ therefore yields a learnware specification $\Phi_i^{\text{GPS}} = \{\text{Dir}(\boldsymbol{G}_{\mathcal{S}_i^k})\}_{k=1}^{K_i}$ of the model's predictive behavior:

$$\mathcal{S}_i^* = \arg\min_{\mathcal{S}_i} \mathcal{L}_{\text{GPS}}\left(\mathcal{D}_i := \{(\boldsymbol{x}, f_{\boldsymbol{\theta}_i}(\boldsymbol{x}))\}, \mathcal{S}_i\right). \tag{9}$$

The dock system stores only $\{f_{\boldsymbol{\theta}_i}, \Phi_i^{\text{GPS}}\}$, without requiring the private developer data or original task gradients. Since all learnwares construct their specifications using the shared proxy network, their gradient direction responses are represented in compatible coordinates for subsequent specification matching. The complete submitting procedure is provided in Algorithm 2.

3.3 User-Oriented Deploying Stage

Given a user task with limited labeled data $\hat{\mathcal{D}} = \{(\hat{\boldsymbol{x}}_{\hat{n}}, \hat{y}_{\hat{n}})\}_{\hat{n}=1}^{\hat{N}}$, the dock system optimizes a proxy set $\hat{\mathcal{S}}$ according to Eq. 8 to construct the user task specification $\hat{\Phi}^{\text{GPS}} = \{\text{Dir}(\hat{\boldsymbol{G}}_{\hat{\mathcal{S}}^{\hat{k}}})\}_{\hat{k}=1}^{\hat{K}}$:

$$\hat{\mathcal{S}}^* = \arg\min_{\hat{\mathcal{S}}} \mathcal{L}_{\text{GPS}}(\hat{\mathcal{D}} := \{(\hat{\boldsymbol{x}}, \hat{y})\}, \hat{\mathcal{S}}). \tag{10}$$

Unlike the submitting stage, where Φ_i^{GPS} is constructed from developer inputs and the predictive distribution of the corresponding well-established model, $\hat{\Phi}^{\text{GPS}}$ is constructed from user inputs and ground-truth labels. Accordingly, the learnware specification characterizes the predictive capability provided by a learnware, whereas the user specification represents the predictive requirement of the user task. Since both are constructed under the shared proxy network, their class-wise gradient directions lie in compatible coordinates and can be directly compared. For learnware i, the dock system computes the following class-wise matching score:

$$\text{Score}\left(\hat{\Phi}^{\text{GPS}}, \Phi_i^{\text{GPS}}\right) = \frac{\sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k,*} \cos\left(\hat{\Phi}_{\hat{k}}^{\text{GPS}}, \Phi_{i,k}^{\text{GPS}}\right)}{\max\left(1, \sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k,*}\right)}, \tag{11}$$

where $\hat{\Phi}_{\hat{k}}^{\text{GPS}} \in \mathbb{R}^d$ and $\Phi_{i,k}^{\text{GPS}} \in \mathbb{R}^d$. Here, $e_i^{\hat{k},k} \in \{0,1\}$ indicates whether user class $\hat{k}$ is matched to class k of learnware i. The feasible set $\mathcal{E}_i$ contains partial one-to-one matchings satisfying $\sum_k e_i^{\hat{k},k} \le 1$, $\sum_{\hat{k}} e_i^{\hat{k},k} \le 1$ and $\sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k} = \min(\hat{K}, K_i)$. Its maximum-weight optimizer is $\boldsymbol{E}_i^* = \{e_i^{\hat{k},k,*}\} = \arg\max_{\boldsymbol{E}_i \in \mathcal{E}_i} \sum_{\hat{k},k} e_i^{\hat{k},k} \cos(\hat{\Phi}_{\hat{k}}^{\text{GPS}}, \Phi_{i,k}^{\text{GPS}})$; thus, $e_i^{\hat{k},k,*} = 1$ denotes a selected class pair. The normalization makes the scores comparable across learnwares with different label-space sizes. The highest-scoring learnware is

$$i^* = \arg\max_{i \in [c]} \text{Score}\left(\hat{\Phi}^{\text{GPS}}, \Phi_i^{\text{GPS}}\right). \tag{12}$$

Although i^* provides the best learnware at the task level, individual user classes can have different best matches. Accordingly, task-level identification uses i^*, whereas class-level routing considers

the matched candidates from all learnwares. To obtain an unambiguous class-level routing, let $\mathcal{C}_{\hat{k}} = \{(i,k) : e_i^{\hat{k},k,*} = 1,\ i \in [c],\ k \in [K_i]\}$ be the candidate learnware–class pairs for user class $\hat{k}$. For every nonempty candidate set, the dock system selects

$$(i_{\hat{k}}^*, k_{\hat{k}}^*) = \underset{(i,k)\in\mathcal{C}_{\hat{k}}}{\arg\max} \cos\left(\hat{\Phi}_{\hat{k}}^{\text{GPS}}, \Phi_{i,k}^{\text{GPS}}\right). \tag{13}$$

The dock system produces the resulting class–model correspondence $\mathcal{M}^* = \{(\hat{k}, i_{\hat{k}}^*, k_{\hat{k}}^*) : \mathcal{C}_{\hat{k}} \neq \emptyset,\ \hat{k} \in [\hat{K}]\}$ and the selected learnware indices $\mathcal{I}^* = \{i : \exists(\hat{k}, i, k) \in \mathcal{M}^*\}$. Each correspondence $(\hat{k}, i, k) \in \mathcal{M}^*$ routes the user-class data $\hat{\mathcal{D}}^{\hat{k}}$ to the corresponding learnware model $f_{\boldsymbol{\theta}_i}$ for reuse. This unified routing mechanism supports both single-learnware and multi-learnware reuse, with the former occurring when $|\mathcal{I}^*| = 1$. The overall deployment procedure is summarized in Algorithm 3.

4 Theoretical Analysis

We analyze GPS as a first-order task response under a shared probe. All results are conditioned on one optimization iteration, in which the feature extractor $\psi_{\boldsymbol{\vartheta}}$ and sampled proxy head $g_{\boldsymbol{\omega}_0}$ are shared by the compared responses. Let $\boldsymbol{h}(\boldsymbol{x}) = \psi_{\boldsymbol{\vartheta}}(\boldsymbol{x})$ and $\boldsymbol{p}_0(\boldsymbol{x}) = \mathrm{softmax}(g_{\boldsymbol{\omega}_0} \circ \boldsymbol{h}(\boldsymbol{x}))$. A task $\mathcal{T} = (P, \boldsymbol{q})$ comprises an input distribution and a target: submitted-model predictions for developers and one-hot labels for users. We assume $\|\boldsymbol{h}(\boldsymbol{x})\|_2 \leq C_h$ and, where responses are subtracted, a common output coordinate system. For a linear proxy head and cross-entropy loss, the population one-step head gradient is

$$\boldsymbol{G}(\mathcal{T}) = \mathbb{E}_{\boldsymbol{x}\sim P}\left[(\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top\right]. \tag{14}$$

This residual–feature moment combines input representations with the target behavior relative to the probe. If two tasks share P, then

$$\boldsymbol{G}(P, \boldsymbol{q}_1) - \boldsymbol{G}(P, \boldsymbol{q}_2) = \mathbb{E}_P\left[(\boldsymbol{q}_2 - \boldsymbol{q}_1)\boldsymbol{h}^\top\right]. \tag{15}$$

Thus, identical covariates can yield different responses when their predictive requirements differ; see Proposition 3 and Corollary 4.For general $\mathcal{T}_i = (P_i, \boldsymbol{q}_i)$, define $\boldsymbol{F}_2(\boldsymbol{x}) = (\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}_2(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top$. Adding and subtracting $\mathbb{E}_{P_1}[\boldsymbol{F}_2]$ yields the exact decomposition

$$\boldsymbol{G}(\mathcal{T}_1) - \boldsymbol{G}(\mathcal{T}_2) = \mathbb{E}_{P_1}\left[(\boldsymbol{q}_2 - \boldsymbol{q}_1)\boldsymbol{h}^\top\right] + \mathbb{E}_{P_1}[\boldsymbol{F}_2] - \mathbb{E}_{P_2}[\boldsymbol{F}_2]. \tag{16}$$

The two terms separate predictive-target mismatch and probe-induced distributional mismatch; Theorem 5 gives the corresponding upper bound. Class conditioning can reduce aggregation ambiguity: in $\boldsymbol{G} = \sum_k \pi_k \boldsymbol{G}^k$, class-specific differences may cancel and class-prior shifts may alter the aggregate. This motivates aligning each $\mathrm{Dir}(\boldsymbol{G}^k)$ separately (Proposition 6).

Finally, the polar factor is invariant to positive scaling. For matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the same dimensions whose smallest singular values are at least $\gamma > 0$,

$$\|\mathrm{Dir}(\boldsymbol{A}) - \mathrm{Dir}(\boldsymbol{B})\|_F \leq \frac{2}{\gamma}\|\boldsymbol{A} - \boldsymbol{B}\|_F. \tag{17}$$

Thus, under non-degeneracy, raw-gradient approximation preserves directional fidelity (Theorem 7 and Corollary 8). Moreover, if an intended class pair is retained as a candidate and its ideal cosine score exceeds every incorrect candidate by a margin larger than the specification-induced score perturbation, partial class-wise routing recovers that pair (Theorem 9). Finite batches and averages over independently resampled heads concentrate around their respective population responses (Appendix C.2). These are conditional representation and routing guarantees; they do not establish semantic correspondence, proxy-optimization convergence, or reuse accuracy without the stated assumptions. Further details are provided in Appendix C.

5 Experiment

In this section, we extensively evaluate GPS in the learnware paradigm. We first describe the experimental setup, including the datasets, compared methods, and experimental configuration. We then evaluate GPS and the compared methods under three learnware reuse scenarios: *Beyond Original Functionality*, *Few-Shot Beyond Original Functionality*, and *Within Original Functionality*. Finally, we conduct ablation studies and further analyses to evaluate the effectiveness of GPS for learnware identification and reuse.

Table 1: Performance comparison under the *Beyond Original Functionality* setting. Results are classification accuracy, reported as mean ± standard deviation over sampled user tasks. For each dataset, the best mean among non-oracle entries is highlighted in bold and underlined.

User Task	*Learnware Identification and Reuse*					*Manual Selection*	
Dataset	**RKME**	**CRKME**	**LANE**	**DALI**	**GPS**	**MAX**	**Oracle**
CIFAR-100	0.472 ± 0.116	0.871 ± 0.065	0.858 ± 0.077	$\underline{\mathbf{0.894 \pm 0.035}}$	$\underline{\mathbf{0.894 \pm 0.035}}$	0.500 ± 0.114	0.894 ± 0.035
Food101	0.206 ± 0.092	0.608 ± 0.191	0.381 ± 0.173	$\underline{\mathbf{0.813 \pm 0.049}}$	$\underline{\mathbf{0.813 \pm 0.049}}$	0.440 ± 0.099	0.813 ± 0.049
DTD	0.210 ± 0.103	0.103 ± 0.142	0.445 ± 0.212	$\underline{\mathbf{0.808 \pm 0.065}}$	0.793 ± 0.094	0.445 ± 0.090	0.808 ± 0.065
Cars	0.059 ± 0.057	0.055 ± 0.070	0.107 ± 0.069	0.203 ± 0.090	0.197 ± 0.090	$\underline{\mathbf{0.243 \pm 0.085}}$	0.402 ± 0.086
SUN397	0.162 ± 0.074	0.130 ± 0.067	0.371 ± 0.097	$\underline{\mathbf{0.503 \pm 0.094}}$	0.489 ± 0.100	0.328 ± 0.090	0.546 ± 0.085
GTSRB	0.513 ± 0.167	0.976 ± 0.041	0.949 ± 0.067	$\underline{\mathbf{0.997 \pm 0.002}}$	$\underline{\mathbf{0.997 \pm 0.002}}$	0.629 ± 0.149	0.997 ± 0.002
20Newsgroups	0.434 ± 0.084	0.781 ± 0.099	0.809 ± 0.094	0.649 ± 0.172	$\underline{\mathbf{0.851 \pm 0.049}}$	0.501 ± 0.101	0.860 ± 0.032
Amazon	0.489 ± 0.121	$\underline{\mathbf{0.856 \pm 0.042}}$	0.843 ± 0.056	0.845 ± 0.061	$\underline{\mathbf{0.856 \pm 0.042}}$	0.489 ± 0.099	0.856 ± 0.042
Bugs	0.218 ± 0.102	0.321 ± 0.199	0.407 ± 0.197	0.249 ± 0.160	$\underline{\mathbf{0.436 \pm 0.179}}$	0.364 ± 0.135	0.535 ± 0.150
Coronavirus	0.333 ± 0.122	0.517 ± 0.095	0.437 ± 0.186	0.407 ± 0.118	$\underline{\mathbf{0.524 \pm 0.127}}$	0.433 ± 0.059	0.669 ± 0.050
TREC	0.380 ± 0.146	0.375 ± 0.262	0.639 ± 0.147	0.566 ± 0.221	$\underline{\mathbf{0.694 \pm 0.108}}$	0.503 ± 0.124	0.694 ± 0.108
WoS	0.415 ± 0.087	0.640 ± 0.077	0.664 ± 0.065	0.672 ± 0.069	$\underline{\mathbf{0.683 \pm 0.058}}$	0.440 ± 0.105	0.692 ± 0.052
Avg. Metric (↑)	0.324	0.519	0.576	0.634	0.686	0.443	0.731
Avg. Rank (↓)	5.67	3.96	3.25	2.54	1.50	4.08	–
GPS (win/tie/loss)	12/0/0	11/1/0	12/0/0	6/3/3	–	11/0/1	0/4/8

5.1 EXPERIMENTAL SETUP

We summarize the evaluation datasets, compared methods, and experimental configuration in this experiment. Further details are provided in Appendix D.

Datasets. We evaluate GPS on twelve image and text classification benchmarks. The image datasets include CIFAR-100 (Krizhevsky & Hinton, 2009), Food101 (Bossard et al., 2014), DTD (Cimpoi et al., 2014), Cars (Krause et al., 2013), SUN397 (Xiao et al., 2016), and GTSRB (Stallkamp et al., 2012), covering object, fine-grained, texture, scene, and traffic-sign recognition. The text datasets include 20Newsgroups (Lang, 1995), Amazon (Ni et al., 2019), Bugs (Lee et al., 2024), Coronavirus (Wang et al., 2020), TREC (Li & Roth, 2002), and WoS (Kowsari et al., 2017), covering topic, review, technical-domain, question, and scientific document classification. Further dataset details are provided in Appendix D.1.

Compared methods. We compare GPS with four data-centric specification baselines: RKME (Wu et al., 2023), CRKME (Liu et al., 2025b), LANE (Chen et al., 2025b), and DALI (Chen et al., 2025a). We also report two manual selections: *MAX*, which selects the best-performing single learnware for each user task, and *Oracle*, which selects the best available reuse solution. Both references rely on downstream performance information and therefore do not constitute deployable learnware-identification methods. Further details are provided in Appendix D.2.

Experiment configuration. Each dataset provides predefined training and test splits organized by superclasses. Training superclasses define developer learnware tasks, whereas test superclasses are used to construct user tasks under the three reuse scenarios. We sample 100 user tasks per dataset and scenario, and report mean classification accuracy and standard deviation over these tasks. For a fair comparison, all methods use *ResNet110* and *Sentence Transformer* as the shared feature extractors for image and text tasks, respectively. The specification set contains $M = 10 \times K$ samples, where K is the number of classes in the corresponding task. Further implementation and experimental details are provided in Appendix D.3.

5.2 PERFORMANCES

We evaluate GPS and the compared methods under three learnware reuse scenarios: *Beyond Original Functionality*, *Few-Shot Beyond Original Functionality*, and *Within Original Functionality*. Table 1 reports the results for *Beyond Original Functionality*, with the remaining results and complete evaluation protocol provided in Appendix D.4.

Beyond Original Functionality. This setting randomly combines subclasses drawn from multiple superclasses. No single learnware covers the complete task, so effective reuse can require multiple complementary learnwares. In Table 1, GPS obtains the highest average accuracy (0.686) and the best average rank (1.50) among automatic methods. It improves on the second-best automatic method, DALI, by 8.2%, remains 6.2% below the *Oracle* accuracy (0.731), and exceeds the single-learnware *MAX* reference on 11 of 12 benchmarks.

Table 2: Performance comparison under the *Few-Shot Beyond Original Functionality* setting. Results are classification accuracy, reported as mean ± standard deviation over sampled user tasks. For each dataset, the best mean among non-oracle entries is highlighted in bold and underlined.

User Task	*Learnware Identification and Reuse*					*Manual Selection*	
Dataset	**RKME**	**CRKME**	**LANE**	**DALI**	**GPS**	**MAX**	**Oracle**
CIFAR-100	0.473 ± 0.125	0.721 ± 0.156	0.859 ± 0.091	0.893 ± 0.049	$\mathbf{0.895 \pm 0.048}$	0.509 ± 0.125	0.895 ± 0.048
Food101	0.119 ± 0.109	0.303 ± 0.237	0.389 ± 0.187	$\mathbf{0.813 \pm 0.086}$	0.785 ± 0.106	0.454 ± 0.106	0.827 ± 0.060
DTD	0.206 ± 0.108	0.119 ± 0.133	0.472 ± 0.216	$\mathbf{0.804 \pm 0.061}$	0.782 ± 0.091	0.449 ± 0.105	0.804 ± 0.061
Cars	0.062 ± 0.071	0.042 ± 0.050	0.103 ± 0.071	0.192 ± 0.083	0.195 ± 0.084	$\mathbf{0.241 \pm 0.079}$	0.413 ± 0.085
SUN397	0.173 ± 0.078	0.126 ± 0.072	0.374 ± 0.096	$\mathbf{0.508 \pm 0.090}$	0.500 ± 0.092	0.353 ± 0.090	0.554 ± 0.079
GTSRB	0.352 ± 0.132	0.877 ± 0.101	0.927 ± 0.075	$\mathbf{0.996 \pm 0.007}$	0.994 ± 0.012	0.599 ± 0.127	0.997 ± 0.004
20Newsgroups	0.424 ± 0.112	0.728 ± 0.127	0.475 ± 0.185	0.597 ± 0.187	$\mathbf{0.840 \pm 0.060}$	0.484 ± 0.099	0.850 ± 0.045
Amazon	0.564 ± 0.119	$\mathbf{0.852 \pm 0.047}$	0.776 ± 0.098	0.827 ± 0.070	$\mathbf{0.852 \pm 0.047}$	0.478 ± 0.106	0.852 ± 0.047
Bugs	0.153 ± 0.065	0.208 ± 0.095	0.162 ± 0.106	0.173 ± 0.105	$\mathbf{0.313 \pm 0.115}$	0.242 ± 0.097	0.366 ± 0.118
Coronavirus	0.226 ± 0.121	0.464 ± 0.151	0.397 ± 0.156	0.377 ± 0.151	$\mathbf{0.530 \pm 0.171}$	0.436 ± 0.076	0.695 ± 0.067
TREC	0.357 ± 0.129	0.332 ± 0.187	0.494 ± 0.169	0.556 ± 0.156	$\mathbf{0.646 \pm 0.083}$	0.411 ± 0.097	0.647 ± 0.082
WoS	0.400 ± 0.076	0.620 ± 0.093	0.536 ± 0.112	0.633 ± 0.077	$\mathbf{0.658 \pm 0.075}$	0.407 ± 0.086	0.664 ± 0.074
Avg. Metric (↑)	0.292	0.449	0.497	0.614	0.666	0.422	0.714
Avg. Rank (↓)	5.58	4.04	3.75	2.33	1.46	3.83	–
GPS (win/tie/loss)	12/0/0	11/1/0	12/0/0	8/0/4	–	11/0/1	0/2/10

Few-Shot Beyond Original Functionality. This setting uses the same multi-superclass construction as *Beyond Original Functionality*, but provides only 10 labeled user examples for every selected subclass. Despite this limited supervision, GPS achieves the highest average accuracy (0.666) and the best average rank (1.46) among automatic methods (Table 2). It surpasses DALI by 8.5% and is within 6.7% of the *Oracle* accuracy (0.714), indicating effective identification of complementary learnwares from few labeled examples.

Within Original Functionality. This setting forms each user task by randomly selecting subclasses from a single superclass, such that one learnware is sufficient for reuse. As shown in Appendix Table 4, GPS attains the highest average accuracy among the automatic identification methods (0.693), improving on DALI by 11.4% and remaining within 4.9% of the *Oracle* accuracy (0.729). These results show that GPS identifies suitable learnwares without accessing downstream performance on the user tasks.

Summary. Across all three reuse scenarios, GPS consistently achieves the strongest aggregate performance among the compared specification methods and remains close to the *Oracle*, demonstrating robust identification and reuse. However, its advantage is less pronounced on some image benchmarks. Unlike methods that initialize specifications from task samples, GPS initializes its proxy set from random noise and captures reuse-relevant information solely through gradient alignment. Optimizing such proxies can be more difficult in high-dimensional image spaces, potentially limiting their fidelity in reproducing class-wise gradient responses. This suggests a trade-off between reducing direct dependence on task samples and increasing proxy optimization difficulty, motivating more effective proxy initialization strategies in future work.

5.3 ABLATION STUDY

We conduct ablation studies to evaluate the contributions of key components in GPS, including model quality, proxy sample size, and efficiency.

Model Quality. Unlike data-centric specifications that primarily characterize data distributions or feature statistics, GPS represents the functional behavior of a learnware through responses jointly induced by its predictions and task inputs. To test whether this representation distinguishes learnwares with different levels of model quality, we add, for each well-established model, a corrupted counterpart with substantially degraded predictive performance to the dock system. This produces a candidate pool that is heterogeneous in both model identity and quality. As shown in Fig. 3 and Appendix Figs. 4–8, GPS more effectively identifies learnwares associated with well-performing models and avoids their corrupted counterparts.

Proxy Sample Size. The proxy sample size governs a fidelity–cost trade-off: larger proxy sets provide greater capacity for matching class-wise gradient directions but incur higher construction costs. As shown in Appendix Fig. 9, increasing the size from 5 to 10 samples per class generally improves

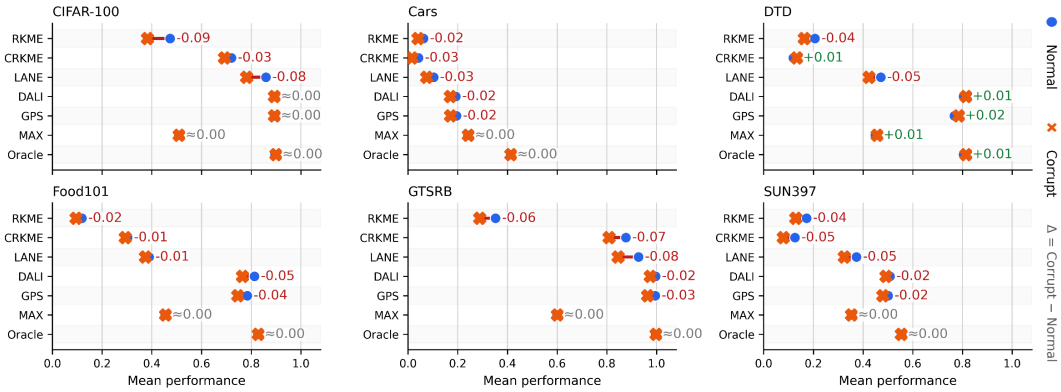

Figure 3: Ablation study on model quality under *Few-Shot Beyond Original Functionality*. *Normal* contains only the original well-established models, whereas *Corrupted* additionally includes their corrupted counterparts.

or preserves reuse performance, whereas increasing it further to 20 yields no consistent benefit. We therefore use 10 samples per class by default; detailed results are provided in Appendix D.6.

Efficiency Analysis. Table 5 compares efficiency across three stages: learnware specification generation, user specification generation, and learnware identification and reuse. Compared with DALI, the strongest competing specification method, GPS reduces the latency-critical identification-and-reuse time by 34.5%–43.5% across all three settings, while also requiring lower learnware-specification cost and fewer optimization rounds to converge. These efficiency gains, together with the superior reuse performance of GPS, demonstrate its favorable efficiency–performance trade-off for practical deployment. Detailed results are provided in Appendix D.7.

5.4 Further Analysis and Limitations

To examine whether the gains are consistent across benchmarks, we conduct cross-benchmark statistical tests in Appendix D.8. The Friedman test shows significant overall differences among specification methods for every modality–scenario pair, with pairwise effects generally favoring GPS. However, after Holm–Bonferroni correction, no post-hoc pairwise comparison remains significant at the 0.05 level, likely due to the small number of datasets per modality.

Although GPS achieves the strongest overall identification and reuse performance, its advantage is less consistent on fine-grained image benchmarks such as Cars, where subtle inter-class differences may not be sufficiently captured by the shared *ResNet110* representation. Additional limitations include difficult proxy optimization in high-dimensional spaces, the overhead of proxy construction, dependence on the shared representation, and the current restriction to classification. Unlike PAVE (Shi et al., 2026), GPS does not rely on large pre-trained foundation models or task-specific adaptation, making direct comparison under our controlled protocol less meaningful. These limitations motivate improved proxy optimization, broader task support, stronger privacy guarantees, and more transferable representations.

6 Conclusion

In this paper, we introduce GPS, a learnware specification that characterizes model capabilities and user requirements through class-wise one-step gradient directions under a shared proxy network. Our representation-level analysis clarifies how gradient responses jointly encode input representations and predictive targets, while class-wise modeling mitigates cross-class cancellation. Extensive experiments across diverse learnware reuse scenarios demonstrate that GPS enables effective and robust learnware identification and reuse, including complementary reuse beyond individual learnware functionality and reliable discrimination against corrupted models. These results establish gradient directions as a promising alternative to conventional data-centric specifications for capturing reuse-relevant model capabilities. Future work will explore more efficient proxy optimization, formal privacy guarantees, extensions beyond classification, and more transferable shared representations.

AI USE STATEMENT

Generative AI tools were used solely to aid and polish the writing of this paper. All AI-assisted text was reviewed and revised by the authors for accuracy, clarity, and consistency with the underlying research. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide the mathematical formulation and implementation details of GPS in Section 3 and Appendix B. The assumptions, formal statements, and proofs supporting the theoretical analysis are given in Section 4 and Appendix C. The datasets, task construction, compared methods, model configurations, and evaluation protocols are documented in Section 5.1 and Appendix D. These materials specify the information needed to reproduce the reported experiments and verify the theoretical results. The implementation of GPS used in this work is provided in the **Supplementary Material** for review, and the core code will be publicly released upon publication.

REFERENCES

Jimmy Ba, Murat A. Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. In *Advances in Neural Information Processing Systems 35*, pp. 37932–37946, New Orleans, USA, 2022.

Deyu Bo and Xinchao Wang. Beyond soft label: Dataset distillation via orthogonal gradient matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5596–5605, Denver, USA, 2026.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, Virtual Event, 2020.

George Cazenavette, Antonio Torralba, and Vincent Sitzmann. Dataset distillation for pre-trained self-supervised vision models. In *Advances in Neural Information Processing Systems 38*, pp. 67182–67223, San Diego, USA, 2025.

Wei Chen, Jun-Xiang Mao, Xiaozheng Wang, and Min-Ling Zhang. Learnware specification via dual alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 8683–8699, Vancouver, Canada, 2025a.

Wei Chen, Jun-Xiang Mao, and Min-Ling Zhang. Learnware specification via label-aware neural embedding. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, volume 39, pp. 15857–15865, Philadelphia, Pennsylvania, 2025b.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, Columbus, USA, 2014.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

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

Nicholas J. Higham. Computing the polar decomposition-with applications. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1160–1174, 1986.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Zixuan Jiang, Jiaqi Gu, Mingjie Liu, and David Z. Pan. Delving into effective gradient matching for dataset condensation. In *2023 IEEE International Conference on Omni-layer Intelligent Systems*, pp. 1–6, Berlin, Germany, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. HDLTex: Hierarchical deep learning for text classification. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications*, pp. 364–371, Cancun, Mexico, 2017.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, Sydney, Australia, 2013.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, Lake Tahoe, USA, 2012.

Ken Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pp. 331–339, San Francisco, USA, 1995.

Jae Yong Lee, Sungmin Kang, Juyeon Yoon, and Shin Yoo. The github recent bugs dataset for evaluating LLM-based debugging applications. In *Proceedings of the IEEE Conference on Software Testing, Verification and Validation*, pp. 442–444, Toronto, ON, Canada, 2024.

Hao-Yi Lei, Zhi-Hao Tan, and Zhi-Hua Zhou. On the ability of developers' training data preservation of learnware. In *Advances in Neural Information Processing Systems 37*, pp. 36471–36513, Vancouver, Canada, 2024.

Hao-Yi Lei, Zhi-Hao Tan, and Zhi-Hua Zhou. A statistical framework for analyzing specification resistance to learnware-inversion risks. In *Proceedings of the 43rd International Conference on Machine Learning*, Seoul, South Korea, 2026a.

Hao-Yi Lei, Jin-Hui Wu, Zhi-Hao Tan, and Zhi-Hua Zhou. On the privacy-preserving capabilities of PAVE specifications in learnware. In *Proceedings of the 35th International Joint Conference on Artificial Intelligence*, Bremen, Germany, 2026b.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, volume 1, pp. 1–7, Taipei, Taiwan, 2002.

Hai-Tian Liu, Peng Tan, Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Integrated learnware identification and reuse via reusability-aware metric learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4465–4473, Jeju, South Korea, 2026.

Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Towards making learnware specification and market evolvable. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, volume 38, pp. 13909–13917, Vancouver, Canada, 2024.

Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Dynamic learnware filtering for efficient learnware identification and system slimming. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1811–1822, Toronto, Canada, 2025a.

Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Identifying and reusing learnwares across different label spaces. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence*, pp. 5734–5742, Montreal, Canada, 2025b.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 188–197, Hong Kong, China, 2019.

Hao-Yu Shi, Zhi-Hao Tan, Zi-Chen Zhao, Yang Yu, and Zhi-Hua Zhou. A study on PAVE specification for learnware. In *The 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. Computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

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

Peng Tan, Hai-Tian Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Handling learnwares from heterogeneous feature spaces with explicit label exploitation. In *Advances in Neural Information Processing Systems 37*, pp. 12767–12795, Vancouver, Canada, 2024a.

Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. Towards enabling learnware to handle heterogeneous feature spaces. *Machine Learning*, 113(4):1839–1860, 2024b.

Peng Tan, Feifan Yang, Zhi-Hao Tan, and Zhi-Hua Zhou. Tabular learnwares can be repurposed for seemingly irrelevant new tasks. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, volume 40, pp. 25778–25786, Singapore, 2026.

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. Beimingwu: A learnware dock system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5773–5782, Barcelona, Spain, 2024c.

Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. CORD-19: The COVID-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, 2020.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

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

Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119 (1):3–22, 2016.

Yi Xie, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. Identifying helpful learnwares without examining the whole market. In *Proceedings of the 26th European Conference on Artificial Intelligence*, volume 372, pp. 2752–2759, Kraków, Poland, 2023.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, Seattle, USA, 2024.

Yu-Jie Zhang, Yu-Hu Yan, Peng Zhao, and Zhi-Hua Zhou. Towards enabling learnware to handle unseen jobs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, volume 35, pp. 10964–10972, Virtual Event, 2021.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *Proceedings of the 9th International Conference on Learning Representations*, Virtual Event, 2021.

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

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

APPENDIX

A DETAILED PRELIMINARIES OF THE LEARNWARE PARADIGM

The *learnware paradigm* enables users to solve new tasks by retrieving and reusing suitable well-established models from a *learnware dock system*, avoiding the need to train models from scratch. Each learnware consists of a well-established model and a compact specification that captures reuse-relevant information, such as task properties and model predictive behavior. The paradigm comprises two stages: the *Developer-Oriented Submitting Stage*, which constructs and stores learnwares, and the *User-Oriented Deploying Stage*, which identifies task-relevant learnwares and reuses their associated models.

A.1 TASK AND LEARNWARE FORMULATION

Suppose c developers submit well-established models together with their associated training tasks. Developer i possesses a private dataset $\mathcal{D}_i = \{(\boldsymbol{x}_{i,n}, y_{i,n})\}_{n=1}^{N_i}$, where $\boldsymbol{x}_{i,n} \in \mathcal{X}$ and $y_{i,n} \in \mathcal{Y}$ denote the input and target label, respectively. We denote the corresponding developer task by $T_i = \{\mathcal{D}_i, f_{\boldsymbol{\theta}_i}\}$, where $f_{\boldsymbol{\theta}_i} : \mathcal{X} \to \mathcal{Y}$ is the well-established model associated with $\mathcal{D}_i$. A model is considered well-established if it achieves sufficiently low expected risk on its associated task:

$$\mathcal{R}_i\left(f_{\boldsymbol{\theta}_i}\right) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_i}\left[\ell\left(y, f_{\boldsymbol{\theta}_i}(\boldsymbol{x})\right)\right] \leq \epsilon, \tag{18}$$

where $\ell(\cdot,\cdot)$ denotes a task-specific loss function and ϵ is a sufficiently small tolerance. The capability of a well-established model is reflected in its predictive behavior over the task distribution. For classification, this behavior can be characterized by the model-induced predictive distribution $P\left(f_{\boldsymbol{\theta}_i}(\mathbf{X}) \mid \mathbf{X}, \boldsymbol{\theta}_i\right)$, where $\mathbf{X}$ denotes the task input random variable. Accordingly, a learnware specification should compactly encode both task characteristics and model capabilities so as to indicate the applicability of $f_{\boldsymbol{\theta}_i}$ to future user tasks.

A.2 DEVELOPER-ORIENTED SUBMITTING STAGE

When a developer submits a well-established model $f_{\boldsymbol{\theta}_i}$, the dock system automatically assigns a specification Φ_i and forms the corresponding learnware $\{f_{\boldsymbol{\theta}_i}, \Phi_i\}$. The specification is the compact information through which the dock system estimates the reusability of the submitted model for future user tasks.

A common design principle in existing learnware specification studies is to characterize the data distribution associated with the submitted well-established model. Let $\mu(\cdot)$ denote a representation operator that maps a distribution into a unique metric space. A generic distribution-based specification can be formulated as

$$\Phi_i = \arg\min_{\Phi_i} \left\|\mu(P_{\mathbf{X}_i}) - \mu(P_{\Phi_i})\right\|^2, \tag{19}$$

where $P_{\mathbf{X}_i}$ denotes the input distribution of the i-th task and P_{Φ_i} denotes the distributional representation induced by Φ_i. Furthermore, if the conditional predictive distribution $P\left(f_{\boldsymbol{\theta}_i}(\mathbf{X}) \mid \mathbf{X}, \boldsymbol{\theta}_i\right)$ is also considered, the specification can be extended to jointly characterize the conditional distribution of the i-th task. Eq. 19 can then be generalized to

$$\Phi_i = \arg\min_{\Phi_i} \sum_{k\in\mathcal{Y}} \left\|\mu(P_{\mathbf{X}_i|Y_i=k}) - \mu(P_{\Phi_i|Y_{\Phi_i}=k})\right\|^2, \tag{20}$$

where $P_{\mathbf{X}_i|Y_i=k}$ and $P_{\Phi_i|Y_{\Phi_i}=k}$ denote the class-conditional distributions of the i-th task and its corresponding specification, respectively, for class k. For example, the Reduced Kernel Mean Embedding (RKME) specification represents the task distribution in a reproducing kernel Hilbert space. Given a positive semi-definite kernel $\mathrm{k}(\cdot,\cdot)$, the empirical kernel mean embedding of the task data is

$$\mu(P_{\mathbf{X}_i}) = \frac{1}{N_i}\sum_{n=1}^{N_i} \mathrm{k}\left(\boldsymbol{x}_{i,n}, \cdot\right). \tag{21}$$

Correspondingly, the RKME approximates this embedding using a small number of reduced points:

$$\mu(P_{\Phi_i}) = \sum_{m=1}^{M_i} \beta_{i,m} \, \mathrm{k}\left(\boldsymbol{s}_{i,m}, \cdot\right), \tag{22}$$

where the specification is $\Phi_i = \{(\boldsymbol{s}_{i,m}, \beta_{i,m})\}_{m=1}^{M_i}$ and $M_i \ll N_i$. The corresponding specification generation objective is

$$\Phi_i^{RKME} = \underset{\Phi_i^{RKME}}{\arg\min} \left\| \frac{1}{N_i} \sum_{n=1}^{N_i} \mathrm{k}(\boldsymbol{x}_{i,n}, \cdot) - \sum_{m=1}^{M_i} \beta_{i,m} \, \mathrm{k}(\boldsymbol{s}_{i,m}, \cdot) \right\|_{\mathcal{H}}^2 . \tag{23}$$

More generally, learnware specifications may characterize reuse-relevant task properties, such as data distributions and label information. The original developer data remain local, while only the well-established model and its specification are stored in the dock system for future reuse.

A.3 User-Oriented Deploying Stage

Given a user requirement task, the dock system identifies relevant learnwares according to the relationship between the user task and the stored learnware specifications, and subsequently reuses the corresponding well-established models.

Distribution-Level Learnware Identification. Consider first an unlabeled user requirement task $\hat{T} : \hat{\mathcal{D}} = \{\hat{\boldsymbol{x}}_{\hat{n}}\}_{\hat{n}=1}^{\hat{N}}$, where $\hat{\boldsymbol{x}}_{\hat{n}}$ is sampled from an unknown user distribution $P_{\hat{\mathbf{X}}}$. Conceptually, the user-task distribution can be represented by a combination of the distributional characteristics encoded by relevant learnware specifications. In the specification space, this relationship can be expressed as

$$\mu(P_{\hat{\mathbf{X}}}) \approx \sum_{i=1}^{c} w_i \mu(P_{\Phi_i}), \qquad \boldsymbol{w} \in \Delta^{c-1}, \tag{24}$$

where

$$\Delta^{c-1} = \left\{ \boldsymbol{w} \in \mathbb{R}_+^c : \sum_{i=1}^{c} w_i = 1 \right\} \tag{25}$$

denotes the probability simplex, and w_i quantifies the relevance of the i-th learnware to the user task. Accordingly, distribution-level learnware identification can be formulated as

$$\boldsymbol{w}^* = \underset{\boldsymbol{w} \in \Delta^{c-1}}{\arg\min} \left\| \mu(P_{\hat{\mathbf{X}}}) - \sum_{i=1}^{c} w_i \mu(P_{\Phi_i}) \right\|^2 . \tag{26}$$

Learnwares associated with sufficiently large coefficients can then be regarded as candidate learnwares for reuse. This formulation enables the dock system to estimate learnware relevance through compact specifications without accessing the original developer datasets.

Class-Wise Learnware Identification. When the user has a limited labeled dataset $\hat{\mathcal{D}} = \{(\hat{\boldsymbol{x}}_{\hat{n}}, \hat{y}_{\hat{n}})\}_{\hat{n}=1}^{\hat{N}}$, with $(\hat{\boldsymbol{x}}_{\hat{n}}, \hat{y}_{\hat{n}}) \overset{\text{i.i.d.}}{\sim} P_{\hat{\mathbf{X}},\hat{\mathbf{Y}}}$, its labels can be used to establish class-level relationships between the user task and stored learnwares.

Suppose the dock system constructs a class-wise user specification $\hat{\Phi} = \{\hat{\Phi}_{\hat{k}}\}_{\hat{k}=1}^{\hat{K}}$, where $\hat{K}$ denotes the number of user-task classes and $\hat{\Phi}_{\hat{k}}$ represents the specification associated with the $\hat{k}$-th user class. Similarly, the specification of the i-th learnware is represented as $\Phi_i = \{\Phi_{i,k}\}_{k=1}^{K_i}$, where K_i denotes the number of classes associated with the i-th learnware.

Consistent with the class-wise directional matching in Section 3, we define the class-wise similarity as

$$a_i^{\hat{k},k} = \cos\left(\hat{\Phi}_{\hat{k}}^{\text{GPS}}, \Phi_{i,k}^{\text{GPS}}\right), \tag{27}$$

where the cosine is computed between vectorized directional specification matrices. Let $e_i^{\hat{k},k} \in \{0,1\}$ denote the matching variable, where $e_i^{\hat{k},k} = 1$ indicates that the $\hat{k}$-th user class is matched

with the k-th class of the i-th learnware. For each learnware, the optimal class-wise correspondence is obtained by

$$\boldsymbol{E}_i^* = \arg\max_{\boldsymbol{E}_i \in \mathcal{E}_i} \sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k} a_i^{\hat{k},k}, \tag{28}$$

subject to

$$\sum_{k=1}^{K_i} e_i^{\hat{k},k} \leq 1, \quad \sum_{\hat{k}=1}^{\hat{K}} e_i^{\hat{k},k} \leq 1 \quad \text{and} \quad \sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k} = \min(\hat{K}, K_i) \tag{29}$$

These constraints allow partial one-to-one matching, thereby accommodating heterogeneous label spaces with different numbers of classes. To make matching scores comparable across learnwares with different label-space sizes, we define the normalized matching score as

$$\text{Score}\left(\hat{\Phi}^{\text{GPS}}, \Phi_i^{\text{GPS}}\right) = \frac{\sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k,*} a_i^{\hat{k},k}}{\max\left(1, \sum_{\hat{k}=1}^{\hat{K}} \sum_{k=1}^{K_i} e_i^{\hat{k},k,*}\right)}. \tag{30}$$

A larger score indicates stronger specification-level relevance between the i-th learnware and the user task. For a user class $\hat{k}$, if $\sum_{k=1}^{K_i} e_i^{\hat{k},k,*} > 0$, the i-th learnware is considered relevant to this user class under the specification matching criterion. Otherwise, $\sum_{k=1}^{K_i} e_i^{\hat{k},k,*} = 0$ indicates that no class in the i-th learnware is matched to the corresponding user class.

Learnware Reuse. After class-wise matching, the dock constructs a unique class–model routing by retaining, for each user class, the matched learnware class with the largest specification similarity. Let $\mathcal{M}^* = \{(\hat{k}, i_{\hat{k}}^*, k_{\hat{k}}^*) : \mathcal{C}_{\hat{k}} \neq \emptyset\}$ denote the resulting user-class–learnware–class correspondence, and let $\mathcal{I}^* = \{i : \exists(\hat{k}, i, k) \in \mathcal{M}^*\}$ be the associated selected learnware indices. The dock retrieves $\{f_{\boldsymbol{\theta}_i}\}_{i \in \mathcal{I}^*}$ and applies each model to the user-class subset specified by its corresponding triple. This unified formulation covers both one-model and multi-model reuse.

When the selected models need to be integrated, their predictions can be combined through a task-dependent reuse strategy:

$$\hat{f} = \mathcal{R}\left(\{f_{\boldsymbol{\theta}_i}\}_{i \in \mathcal{I}^*}\right), \tag{31}$$

where $\mathcal{R}(\cdot)$ denotes a generic reuse operator, such as model selection, class-wise composition, weighted prediction, or other task-dependent aggregation strategies.

The resulting predictor is expected to achieve a small risk on the user task:

$$\mathcal{R}_{\hat{\mathcal{D}}}(\hat{f}) = \mathbb{E}_{(\hat{\boldsymbol{x}}, \hat{y}) \sim P_{\hat{\mathbf{X}}, \hat{\mathbf{Y}}}}\left[\ell\left(\hat{y}, \hat{f}(\hat{\boldsymbol{x}})\right)\right]. \tag{32}$$

Therefore, learnware identification and learnware reuse are two connected but conceptually distinct processes. Specification matching provides a surrogate criterion for estimating the relevance of submitted models to a user requirement, while model reuse directly exploits the selected well-established models to solve the user task.

B Algorithmic Implementation of GPS Specification in the Learnware Paradigm

B.1 Algorithm of the GPS Specification

Algorithm 1 constructs a compact, model-aware GPS specification based on class-wise one-step gradient directions. At each optimization iteration, it uses the globally fixed feature extractor $\psi_{\boldsymbol{\vartheta}}$ and a newly sampled, fixed proxy head to evaluate task and proxy gradients in the same coordinate system. It updates only the proxy samples to align their class-wise one-step gradient directions with the responses induced by the task inputs and targets, and uses the directions induced by the optimized proxy set as Φ^{GPS}.

Algorithm 1 GPS specification construction

Require: Labeled task data with targets $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, globally fixed feature extractor $\psi_{\boldsymbol{\vartheta}}$, head-initialization distribution $P_{\boldsymbol{\omega}}$, proxy sample set size $K \times l$, and optimization iterations T
Ensure: GPS specification Φ^{GPS}

initialize a proxy sample set $\mathcal{S} = \{(\boldsymbol{s}_m, y_{\boldsymbol{s}_m})\}_{m=1}^{M}$ with $M = K \times l$
for $t = 1, \ldots, T$ **do**
  Sample $\boldsymbol{\omega}_0^{(t)} \sim P_{\boldsymbol{\omega}}$, set $\boldsymbol{\xi}^{(t)} = (\boldsymbol{\vartheta}, \boldsymbol{\omega}_0^{(t)})$, and define $\phi_{\boldsymbol{\xi}^{(t)}} = g_{\boldsymbol{\omega}_0^{(t)}} \circ \psi_{\boldsymbol{\vartheta}}$
  for $k = 1, \ldots, K$ **do**
    Compute $\boldsymbol{G}_{\mathcal{D}^k}$ and $\boldsymbol{G}_{\mathcal{S}^k}$ using Eqs. 5 and 7 under $\phi_{\boldsymbol{\xi}^{(t)}}$
    Compute $\mathrm{Dir}(\boldsymbol{G}_{\mathcal{D}^k})$ and $\mathrm{Dir}(\boldsymbol{G}_{\mathcal{S}^k})$ using Eq. 6
    Compute $\mathcal{L}^k \leftarrow 1 - \cos(\mathrm{Dir}(\boldsymbol{G}_{\mathcal{D}^k}), \mathrm{Dir}(\boldsymbol{G}_{\mathcal{S}^k}))$
  end for
  Compute $\mathcal{L}_{\text{GPS}}(\mathcal{D}, \mathcal{S}) \leftarrow \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}^k$ by Eq. 8
  Update $\mathcal{S} \leftarrow \mathcal{S} - \eta \nabla_{\mathcal{S}} \mathcal{L}_{\text{GPS}}(\mathcal{D}, \mathcal{S})$
end for
$\Phi^{\text{GPS}} \leftarrow \{\mathrm{Dir}(\boldsymbol{G}_{\mathcal{S}^k})\}_{k=1}^{K}$
return Φ^{GPS}

B.2 Algorithm of the Developer-Oriented Submitting Stage

As shown in Algorithm 2, the developer-side procedure first obtains the predictive targets of the submitted well-established model on its private data. These model-induced targets are used to construct Φ_i^{GPS} with Algorithm 1, so that the specification represents the model-aware, class-wise functional behavior jointly induced by the developer task and its model. The dock then stores $\{f_{\boldsymbol{\theta}_i}, \Phi_i^{\text{GPS}}\}$ without releasing the original task data.

Algorithm 2 Developer-oriented submission stage with GPS specification in the learnware paradigm

Require: Developer local training task $\mathcal{D}_i = \{(\boldsymbol{x}_{i,n}, y_{i,n})\}_{n=1}^{N_i}$ with K classes, well-established model $f_{\boldsymbol{\theta}_i}$, globally fixed feature extractor $\psi_{\boldsymbol{\vartheta}}$, head-initialization distribution $P_{\boldsymbol{\omega}}$, proxy sample size l, and optimization iterations T
Ensure: Stored learnware $\{f_{\boldsymbol{\theta}_i}, \Phi_i^{\text{GPS}}\}$

Evaluate $f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{D}_i$
$\Phi_i^{\text{GPS}} \leftarrow \text{GPS}(\{(\boldsymbol{x}_{i,n}, f_{\boldsymbol{\theta}_i}(\boldsymbol{x}_{i,n}))\}_{n=1}^{N_i}, \psi_{\boldsymbol{\vartheta}}, P_{\boldsymbol{\omega}}, K \times l, T)$ using Algorithm 1
Store learnware $\{f_{\boldsymbol{\theta}_i}, \Phi_i^{\text{GPS}}\}$ in the learnware dock system

B.3 Algorithm of the User-Oriented Deploying Stage

Algorithm 3 constructs a class-wise user specification from labeled user data, representing the supervision required by the user task, and compares it with stored model-aware learnware specifications. For each learnware, it first solves the partial one-to-one class matching; it then selects, for each user class, the candidate learnware–class pair with the highest class-wise similarity. The procedure returns the retrieved models and the class–model correspondence $\mathcal{M}^*$, which routes each matched user-class dataset to its corresponding model for reuse.

C Detailed Theoretical Analysis

This section provides formal statements and proofs for the representation-level analysis in Section 4. Unless stated otherwise, results are conditioned on one optimization iteration: $\psi_{\boldsymbol{\vartheta}}$ is globally fixed and the sampled head $g_{\boldsymbol{\omega}_0}$ is held fixed when evaluating task and proxy gradients. We additionally analyze empirical gradients and averages over independently sampled heads. These results do not alone guarantee semantic class correspondences, learnware ranking, proxy-optimization convergence, or downstream accuracy.

Algorithm 3 User-oriented deployment stage with GPS specification in the learnware paradigm

Require: User task $\hat{\mathcal{D}} = \{(\hat{\boldsymbol{x}}_{\hat{n}}, \hat{y}_{\hat{n}})\}_{\hat{n}=1}^{\hat{N}}$ with the $\hat{K}$ classes, globally fixed feature extractor $\psi_{\boldsymbol{\vartheta}}$, head-initialization distribution $P_{\boldsymbol{\omega}}$, proxy sample size l, optimization iterations T, and learnware dock system $\{(f_{\boldsymbol{\theta}_i}, \Phi_i^{\text{GPS}})\}_{i=1}^{c}$

Ensure: Selected learnware models $\{f_{\boldsymbol{\theta}_i}\}_{i\in\mathcal{I}^*}$ and class–model correspondence $\mathcal{M}^*$

$\boldsymbol{q}(\hat{\boldsymbol{x}}) \leftarrow \text{onehot}(\hat{y})$ for each $(\hat{\boldsymbol{x}}, \hat{y}) \in \hat{\mathcal{D}}$

$\hat{\Phi}^{\text{GPS}} \leftarrow \text{GPS}(\{(\hat{\boldsymbol{x}}_{\hat{n}}, \boldsymbol{q}(\hat{\boldsymbol{x}}_{\hat{n}}))\}_{\hat{n}=1}^{\hat{N}}, \psi_{\boldsymbol{\vartheta}}, P_{\boldsymbol{\omega}}, \hat{K} \times l, T)$ using Algorithm 1

for $i = 1, \ldots, c$ **do**

  Compute $A_{\hat{k},k}^{(i)} \leftarrow \cos(\hat{\Phi}_{\hat{k}}^{\text{GPS}}, \Phi_{i,k}^{\text{GPS}})$ for all $(\hat{k}, k) \in [\hat{K}] \times [K_i]$

  Solve the partial one-to-one maximum-weight matching over $\mathcal{E}_i$ to obtain $\boldsymbol{E}_i^* = \{e_i^{\hat{k},k,*}\}$

  Compute $\text{Score}(\hat{\Phi}^{\text{GPS}}, \Phi_i^{\text{GPS}})$ using Eq. 11

end for

$i^* \leftarrow \arg\max_{i\in[c]} \text{Score}(\hat{\Phi}^{\text{GPS}}, \Phi_i^{\text{GPS}})$

$\mathcal{C}_{\hat{k}} \leftarrow \{(i,k) : e_i^{\hat{k},k,*} = 1,\ i \in [c],\ k \in [K_i]\}$ for every $\hat{k} \in [\hat{K}]$

for each user class $\hat{k} \in [\hat{K}]$ with $\mathcal{C}_{\hat{k}} \neq \emptyset$ **do**

  $(i_{\hat{k}}^*, k_{\hat{k}}^*) \leftarrow \arg\max_{(i,k)\in\mathcal{C}_{\hat{k}}} \cos(\hat{\Phi}_{\hat{k}}^{\text{GPS}}, \Phi_{i,k}^{\text{GPS}})$

end for

$\mathcal{M}^* \leftarrow \{(\hat{k}, i_{\hat{k}}^*, k_{\hat{k}}^*) : \mathcal{C}_{\hat{k}} \neq \emptyset\}$

$\mathcal{I}^* \leftarrow \{i : \exists(\hat{k}, i, k) \in \mathcal{M}^*\}$

Retrieve $\{f_{\boldsymbol{\theta}_i}\}_{i\in\mathcal{I}^*}$

for each correspondence triple $(\hat{k}, i, k) \in \mathcal{M}^*$ **do**

  Assign $\hat{\mathcal{D}}^{\hat{k}}$ to $f_{\boldsymbol{\theta}_i}$ and solve this class-specific task by learnware reuse

end for

return $\{f_{\boldsymbol{\theta}_i}\}_{i\in\mathcal{I}^*}$ and $\mathcal{M}^*$

C.1 Setup and Assumptions

Let $\mathcal{T} = (P, \boldsymbol{q})$ denote a task, where P is a distribution on $\mathcal{X}$ and $\boldsymbol{q} : \mathcal{X} \to \Delta^{r-1}$ is its target distribution. For developer tasks, $\boldsymbol{q}$ is the predictive distribution of the submitted model; for user tasks, it is the one-hot ground-truth label. Let $\boldsymbol{h}(\boldsymbol{x}) = \psi_{\boldsymbol{\vartheta}}(\boldsymbol{x})$, $\boldsymbol{p}_0(\boldsymbol{x}) = \text{softmax}(g_{\boldsymbol{\omega}_0} \circ \boldsymbol{h}(\boldsymbol{x}))$, and define the population head gradient by

$$\boldsymbol{G}(\mathcal{T}) = \mathbb{E}_{\boldsymbol{x}\sim P}\left[(\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top\right]. \tag{33}$$

We assume $\|\boldsymbol{h}(\boldsymbol{x})\|_2 \le C_h$ almost surely. This is a standard bounded-feature assumption and can be enforced by feature normalization. All compared responses are assumed to have compatible output dimensions.

C.2 Concentration of Empirical and Random-Head Responses

For a fixed sampled head, let $\widehat{\boldsymbol{G}}_n(\mathcal{T})$ denote the empirical counterpart of Eq. 33 computed from n independent samples from P. The following result quantifies the gap between the mini-batch gradient used in practice and its population counterpart.

Proposition 1 (Empirical gradient concentration)**.** *Let the output and feature dimensions be r and d, respectively, and set $L_\delta = \log((r+d)/\delta)$. Under the bounded-feature assumption, with probability at least $1-\delta$,*

$$\left\|\widehat{\boldsymbol{G}}_n(\mathcal{T}) - \boldsymbol{G}(\mathcal{T})\right\|_2 \le 4C_h\sqrt{\frac{L_\delta}{n}} + \frac{2\sqrt{2}C_h L_\delta}{3n}. \tag{34}$$

Proof. For $\boldsymbol{Z}(\boldsymbol{x}) = (\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top$, both $\boldsymbol{p}_0(\boldsymbol{x})$ and $\boldsymbol{q}(\boldsymbol{x})$ are vectors. Hence $\|\boldsymbol{Z}(\boldsymbol{x})\|_2 \le \|\boldsymbol{Z}(\boldsymbol{x})\|_F \le \sqrt{2}C_h$. The centered summands have norm at most $2\sqrt{2}C_h$ and variance proxy at most $8nC_h^2$. Rectangular matrix Bernstein applied to their sum (Tropp, 2012) gives Eq. 34. □

The bound can be converted to a Frobenius-norm bound using $\|\boldsymbol{A}\|_F \le \sqrt{\min\{r,d\}}\|\boldsymbol{A}\|_2$ and then combined with Theorem 7 when its non-degeneracy condition holds.

We next isolate the effect of resampling the proxy head. For $\boldsymbol{\omega} \sim P_{\boldsymbol{\omega}}$, let $\boldsymbol{G}_{\boldsymbol{\omega}}(\mathcal{T})$ be the population response under that head, define $\overline{\boldsymbol{G}}(\mathcal{T}) = \mathbb{E}_{\boldsymbol{\omega}}[\boldsymbol{G}_{\boldsymbol{\omega}}(\mathcal{T})]$, and let $\widetilde{\boldsymbol{G}}_T(\mathcal{T})$ be the average of T independent head responses.

Proposition 2 (Random-head averaging)**.** *With $L_\delta = \log((r+d)/\delta)$, independent head draws satisfy, with probability at least $1-\delta$,*

$$\left\|\widetilde{\boldsymbol{G}}_T(\mathcal{T}) - \overline{\boldsymbol{G}}(\mathcal{T})\right\|_2 \le 4C_h\sqrt{\frac{L_\delta}{T}} + \frac{2\sqrt{2}C_h L_\delta}{3T}. \tag{35}$$

Proof. Each $\boldsymbol{G}_{\boldsymbol{\omega}}(\mathcal{T})$ is an expectation of matrices with norm at most $\sqrt{2}C_h$, and therefore has the same norm bound. Applying the argument of Proposition 1 to the independent centered head responses proves the result. □

Algorithm 1 does not explicitly release this head-average response: it uses independently resampled heads to optimize a single proxy set and releases its final directional response. Thus, Proposition 2 characterizes the stability of the stochastic probing signal, rather than proving convergence of the complete nonconvex proxy optimization.

C.3 Gradient Decomposition

Proposition 3 (One-step gradient representation)**.** *For a linear head $g_{\boldsymbol{\omega}}(\boldsymbol{h}) = \boldsymbol{W}\boldsymbol{h} + \boldsymbol{b}$ and cross-entropy loss, the population gradient with respect to $\boldsymbol{W}$ is exactly Eq. 33, and hence Eq. 14.*

Proof. Fix $\boldsymbol{x}$ and let $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{b}$, so that $\boldsymbol{p}_0(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{z})$. For a possibly soft target $\boldsymbol{q}(\boldsymbol{x}) \in \Delta^{r-1}$, the cross-entropy is $\ell(\boldsymbol{z}, \boldsymbol{q}) = -\sum_{a=1}^{r} q_a \log p_{0,a}$. Its logit gradient is $\nabla_{\boldsymbol{z}}\ell = \boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}(\boldsymbol{x})$. Since $\partial z_a / \partial W_{ab} = h_b(\boldsymbol{x})$, each entry satisfies

$$\frac{\partial \ell}{\partial W_{ab}} = \big(p_{0,a}(\boldsymbol{x}) - q_a(\boldsymbol{x})\big)h_b(\boldsymbol{x}). \tag{36}$$

Equivalently,

$$\nabla_{\boldsymbol{W}}\ell = \big(\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}(\boldsymbol{x})\big)\boldsymbol{h}(\boldsymbol{x})^\top. \tag{37}$$

The feature bound and the simplex constraints imply that this gradient is integrable. Taking expectation over P therefore proves Eq. 33. □

The proposition shows that the response is a residual–feature moment: it jointly depends on the input distribution through $\boldsymbol{h}(\boldsymbol{x})$ and on predictive behavior through $\boldsymbol{q}(\boldsymbol{x})$. Consequently, two tasks with the same covariates need not have the same response when their predictive targets differ.

Corollary 4 (Sensitivity to predictive behavior)**.** *If $\mathcal{T}_1 = (P, \boldsymbol{q}_1)$ and $\mathcal{T}_2 = (P, \boldsymbol{q}_2)$ share the same input distribution, then*

$$\boldsymbol{G}(\mathcal{T}_1) - \boldsymbol{G}(\mathcal{T}_2) = \mathbb{E}_P\left[(\boldsymbol{q}_2(\boldsymbol{x}) - \boldsymbol{q}_1(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top\right]. \tag{38}$$

Hence the two responses differ whenever the residual–feature moment on the right-hand side is nonzero.

Proof. Subtract the two instances of Eq. 33; the proxy term $\boldsymbol{p}_0$ cancels because the proxy is shared. □

C.4 Predictive Sensitivity and Gradient Discrepancy

Theorem 5 (Task-gradient discrepancy)**.** *For $\mathcal{T}_i = (P_i, \boldsymbol{q}_i)$, $i \in \{1,2\}$, define $\boldsymbol{F}_2(\boldsymbol{x}) = (\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}_2(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top$. Then*

$$\boldsymbol{G}(\mathcal{T}_1) - \boldsymbol{G}(\mathcal{T}_2) = \mathbb{E}_{P_1}\left[(\boldsymbol{q}_2 - \boldsymbol{q}_1)\boldsymbol{h}^\top\right] + \mathbb{E}_{P_1}[\boldsymbol{F}_2] - \mathbb{E}_{P_2}[\boldsymbol{F}_2], \tag{39}$$

and, under the bounded-feature assumption,

$$\|\boldsymbol{G}(\mathcal{T}_1) - \boldsymbol{G}(\mathcal{T}_2)\|_F \le C_h\mathbb{E}_{P_1}\left[\|\boldsymbol{q}_1(\boldsymbol{x}) - \boldsymbol{q}_2(\boldsymbol{x})\|_2\right] + \|\mathbb{E}_{P_1}[\boldsymbol{F}_2(\boldsymbol{x})] - \mathbb{E}_{P_2}[\boldsymbol{F}_2(\boldsymbol{x})]\|_F. \tag{40}$$

Proof. Write $\boldsymbol{F}_i(\boldsymbol{x}) = (\boldsymbol{p}_0(\boldsymbol{x}) - \boldsymbol{q}_i(\boldsymbol{x}))\boldsymbol{h}(\boldsymbol{x})^\top$. Adding and subtracting $\mathbb{E}_{P_1}[\boldsymbol{F}_2]$ gives

$$\begin{aligned}\boldsymbol{G}(\mathcal{T}_1) - \boldsymbol{G}(\mathcal{T}_2) &= \mathbb{E}_{P_1}[\boldsymbol{F}_1 - \boldsymbol{F}_2] + \mathbb{E}_{P_1}[\boldsymbol{F}_2] - \mathbb{E}_{P_2}[\boldsymbol{F}_2] \\ &= \mathbb{E}_{P_1}\left[(\boldsymbol{q}_2 - \boldsymbol{q}_1)\boldsymbol{h}^\top\right] + \mathbb{E}_{P_1}[\boldsymbol{F}_2] - \mathbb{E}_{P_2}[\boldsymbol{F}_2].\end{aligned} \tag{41}$$

By the triangle inequality, Jensen's inequality, and $\|\boldsymbol{a}\boldsymbol{b}^\top\|_F = \|\boldsymbol{a}\|_2\|\boldsymbol{b}\|_2$, the first term satisfies

$$\begin{aligned}\left\|\mathbb{E}_{P_1}[(\boldsymbol{q}_2 - \boldsymbol{q}_1)\boldsymbol{h}^\top]\right\|_F &\le \mathbb{E}_{P_1}[\|(\boldsymbol{q}_2 - \boldsymbol{q}_1)\boldsymbol{h}^\top\|_F] \\ &\le C_h\mathbb{E}_{P_1}[\|\boldsymbol{q}_1 - \boldsymbol{q}_2\|_2].\end{aligned} \tag{42}$$

Combining the two bounds proves Eq. 40; the preceding equality is Eq. 39. □

The first term in Eq. 40 measures predictive-target mismatch, while the second is a probe-induced distribution discrepancy. When $P_1 = P_2$, the latter vanishes; thus, predictive-target differences can affect the gradient response even under identical input distributions, although they may cancel for particular feature–target configurations.

C.5 Class-Aware Alignment

Proposition 6 (Effect of class aggregation)**.** *Suppose a common class indexing is available and write $\boldsymbol{G}_i = \sum_k \pi_{ik}\boldsymbol{G}_i^k$, where $\pi_{ik} = P_i(Y = k)$ and $\boldsymbol{G}_i^k$ is the class-conditional response. Then*

$$\|\boldsymbol{G}_1 - \boldsymbol{G}_2\|_F \le \sum_k \pi_{1k}\|\boldsymbol{G}_1^k - \boldsymbol{G}_2^k\|_F + \sum_k |\pi_{1k} - \pi_{2k}|\|\boldsymbol{G}_2^k\|_F. \tag{43}$$

Proof. Adding and subtracting $\sum_k \pi_{1k}\boldsymbol{G}_2^k$ yields

$$\boldsymbol{G}_1 - \boldsymbol{G}_2 = \sum_k \pi_{1k}(\boldsymbol{G}_1^k - \boldsymbol{G}_2^k) + \sum_k (\pi_{1k} - \pi_{2k})\boldsymbol{G}_2^k. \tag{44}$$

Applying the triangle inequality separately to the two sums and using $\|a\boldsymbol{A}\|_F = |a|\|\boldsymbol{A}\|_F$ gives

$$\|\boldsymbol{G}_1 - \boldsymbol{G}_2\|_F \le \sum_k \pi_{1k}\|\boldsymbol{G}_1^k - \boldsymbol{G}_2^k\|_F + \sum_k |\pi_{1k} - \pi_{2k}|\|\boldsymbol{G}_2^k\|_F, \tag{45}$$

which is Eq. 43. □

Eq. 43 makes explicit that a task-level gradient entangles class-conditional responses with class priors. Moreover, cancellation can occur: for two classes with equal priors, $\boldsymbol{G}_1^1 - \boldsymbol{G}_2^1 = -(\boldsymbol{G}_1^2 - \boldsymbol{G}_2^2)$ yields identical aggregate gradients despite nonzero class-wise discrepancies. This motivates matching the released class-wise directions separately. The proposition is conditional on a correspondence and therefore motivates, but does not prove, the partial matching procedure in Section 3.

C.6 Stability of the Directional Specification

Theorem 7 (Polar-factor stability)**.** *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be matrices of the same dimensions with $\min\{\sigma_{\min}(\boldsymbol{A}), \sigma_{\min}(\boldsymbol{B})\} \ge \gamma > 0$. Then their polar factors satisfy*

$$\|\operatorname{Dir}(\boldsymbol{A}) - \operatorname{Dir}(\boldsymbol{B})\|_F \le \frac{2}{\gamma}\|\boldsymbol{A} - \boldsymbol{B}\|_F. \tag{46}$$

Proof. Let $\boldsymbol{A} = \boldsymbol{Q}_A\boldsymbol{H}_A$ and $\boldsymbol{B} = \boldsymbol{Q}_B\boldsymbol{H}_B$ be thin polar decompositions, where $\boldsymbol{Q}_A = \operatorname{Dir}(\boldsymbol{A})$ and $\boldsymbol{Q}_B = \operatorname{Dir}(\boldsymbol{B})$. The polar-factor perturbation inequality (Higham, 1986) states that

$$\|\boldsymbol{Q}_A - \boldsymbol{Q}_B\|_F \le \frac{2\|\boldsymbol{A} - \boldsymbol{B}\|_F}{\sigma_{\min}(\boldsymbol{A}) + \sigma_{\min}(\boldsymbol{B})}. \tag{47}$$

Since both smallest singular values are at least γ, this gives the stronger bound $\|\operatorname{Dir}(\boldsymbol{A}) - \operatorname{Dir}(\boldsymbol{B})\|_F \le \|\boldsymbol{A} - \boldsymbol{B}\|_F/\gamma$, which immediately implies Eq. 46. □

Corollary 8 (Proxy-specification fidelity)**.** *For every class k, if $\sigma_{\min}(\boldsymbol{G}_{\mathcal{D}^k})$ and $\sigma_{\min}(\boldsymbol{G}_{\mathcal{S}^k})$ are at least γ, then*

$$\|\operatorname{Dir}(\boldsymbol{G}_{\mathcal{D}^k}) - \operatorname{Dir}(\boldsymbol{G}_{\mathcal{S}^k})\|_F \le \frac{2}{\gamma}\|\boldsymbol{G}_{\mathcal{D}^k} - \boldsymbol{G}_{\mathcal{S}^k}\|_F. \tag{48}$$

Proof. Apply Theorem 7 with $\boldsymbol{A} = \boldsymbol{G}_{\mathcal{D}^k}$ and $\boldsymbol{B} = \boldsymbol{G}_{\mathcal{S}^k}$. For scale invariance, if $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is a thin singular value decomposition and $a > 0$, then $a\boldsymbol{G} = \boldsymbol{U}(a\boldsymbol{\Sigma})\boldsymbol{V}^\top$ has the same left and right singular vectors. Hence $\operatorname{Dir}(a\boldsymbol{G}) = \boldsymbol{U}\boldsymbol{V}^\top = \operatorname{Dir}(\boldsymbol{G})$. □

Thus, raw-gradient approximation is sufficient to preserve the released directional specifications. The non-degeneracy condition excludes rank-deficient gradients, for which the polar direction can be unstable. In practice, Eq. 8 directly minimizes the directional discrepancy; the corollary gives a complementary guarantee when an approximation is stated in the raw-gradient space.

C.7 Margin Condition for Class-Wise Routing

We next connect directional fidelity to the routing rule in Eq. 13. For a nonzero class-wise directional specification $\boldsymbol{\Phi}$, write $\overline{\boldsymbol{\Phi}} = \operatorname{vec}(\boldsymbol{\Phi})/\|\operatorname{vec}(\boldsymbol{\Phi})\|_2$. Thus, cosine similarity is the inner product between normalized specifications. Let $(i_k^\star, j_k^\star) \in \mathcal{C}_k$ be an intended candidate pair for user class k, and let $\boldsymbol{u}_k$ and $\boldsymbol{v}_{i,j}$ denote its ideal normalized user and learnware specifications, respectively. Their released counterparts are $\widehat{\boldsymbol{u}}_k$ and $\widehat{\boldsymbol{v}}_{i,j}$.

Theorem 9 (Margin-based class-wise routing recovery)**.** *Suppose that, for a user class k, the intended pair $(i_k^\star, j_k^\star)$ is retained in $\mathcal{C}_k$ and has an ideal-score margin $\Delta > 0$:*

$$\left\langle \boldsymbol{u}_k, \boldsymbol{v}_{i_k^\star, j_k^\star} \right\rangle \ge \langle \boldsymbol{u}_k, \boldsymbol{v}_{i,j} \rangle + \Delta, \quad \forall (i,j) \in \mathcal{C}_k \setminus \{(i_k^\star, j_k^\star)\}. \tag{49}$$

If the released specifications satisfy

$$\|\widehat{\boldsymbol{u}}_k - \boldsymbol{u}_k\|_2 \le \varepsilon, \qquad \|\widehat{\boldsymbol{v}}_{i,j} - \boldsymbol{v}_{i,j}\|_2 \le \varepsilon \quad \forall (i,j) \in \mathcal{C}_k, \tag{50}$$

and $\Delta > 4\varepsilon$, then Eq. 13 returns $(i_k^\star, j_k^\star)$.

Proof. For any $(i,j) \in \mathcal{C}_k$, unit norm and the Cauchy–Schwarz inequality give

$$|\langle \widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{v}}_{i,j} \rangle - \langle \boldsymbol{u}_k, \boldsymbol{v}_{i,j} \rangle| \le 2\varepsilon. \tag{51}$$

Hence, relative to any incorrect candidate, the released score of the intended pair exceeds its released score by at least $\Delta - 4\varepsilon > 0$. It is therefore the unique maximizer of the routing rule. □

The theorem is conditional on candidate retention and an intended class-level correspondence. It does not assert that the partial matching stage establishes semantic correspondence; rather, it shows that once such a candidate is present and separated by a margin, directional approximation does not change the routing decision.

D Experiment

D.1 Dataset

We evaluate the learnware identification methods on twelve benchmark datasets covering both image and text classification tasks. These datasets exhibit diverse data distributions, semantic granularities, and task complexities, providing a broad testbed for evaluating the effectiveness and robustness of learnware identification methods.

Image Classification Datasets. We consider six image classification benchmarks.

- **CIFAR-100.** CIFAR-100 (Krizhevsky & Hinton, 2009) is a widely used image classification benchmark consisting of 100 object categories. Each image has a relatively low spatial resolution, making the dataset suitable for evaluating multi-class visual recognition under limited image detail.

- **Food101.** Food101 (Bossard et al., 2014) is a fine-grained food recognition dataset containing 101 food categories. It exhibits substantial intra-class variation caused by different appearances, viewpoints, and presentation styles, while many categories also share considerable visual similarity.
- **DTD (Describable Textures Dataset).** DTD (Cimpoi et al., 2014) is a texture recognition benchmark containing 47 describable texture categories. The dataset mainly evaluates the ability of a model to capture local texture patterns and appearance characteristics rather than object-level semantic structures.
- **Cars (Stanford Cars).** Stanford Cars (Krause et al., 2013) is a fine-grained vehicle classification benchmark containing different car manufacturers, models, and production years. Since many categories exhibit highly similar visual appearances, the dataset presents a challenging fine-grained recognition scenario.
- **SUN397.** SUN397 (Xiao et al., 2016) is a large-scale scene recognition dataset consisting of 397 indoor and outdoor scene categories. It contains substantial visual diversity and complex background structures, making it suitable for evaluating scene-level semantic understanding.
- **GTSRB.** GTSRB (Stallkamp et al., 2012) is a traffic-sign recognition benchmark containing 43 traffic-sign categories. Although its categories are relatively well defined, the images exhibit considerable variations in scale, illumination, viewpoint, and acquisition conditions.

Text Classification Datasets. We further consider six text classification benchmarks.

- **20Newsgroups.** 20Newsgroups (Lang, 1995) is a classical text classification benchmark consisting of documents from 20 newsgroup topics, covering domains such as computers, science, politics, and religion. It is commonly used for evaluating topic classification and text representation methods.
- **Amazon.** The Amazon (Ni et al., 2019) dataset is constructed from product-review text and is commonly used for review-related or sentiment-oriented text classification tasks. The dataset contains diverse natural-language expressions across different products and user writing styles.
- **Bugs.** Bugs (Lee et al., 2024) consists primarily of software defect and bug-report text. Such documents usually contain domain-specific technical terminology and structured descriptions, making the dataset suitable for evaluating classification under specialized technical domains.
- **Coronavirus.** Coronavirus (Wang et al., 2020) contains text related to COVID-19 and coronavirus-related topics, including semantically diverse content associated with the pandemic. The dataset exhibits notable topical and linguistic heterogeneity.
- **TREC.** TREC (Li & Roth, 2002) is a classical question classification benchmark in which short questions are categorized according to their semantic types, such as person, location, numerical quantity, and entity. It primarily evaluates the ability to capture semantic information from short text sequences.
- **WoS (Web of Science).** WoS (Kowsari et al., 2017) is constructed from scientific document abstracts, where the categories correspond to different research areas or scientific disciplines. It provides a domain-specific text classification setting involving specialized terminology and hierarchical semantic structures.

Overall, these benchmarks cover generic object recognition, fine-grained classification, texture recognition, scene understanding, traffic-sign recognition, topic classification, domain-specific text classification, short-question understanding, and scientific document categorization. Such diversity enables us to systematically evaluate the effectiveness and robustness of learnware identification methods across heterogeneous data modalities and task characteristics.

D.2 Compared methods

We compare GPS with four data-centric specification baselines and two manual selection methods. All automatic methods use the same developer/user splits, shared feature extractor and specification size as described in Appendix D.3. For a fair comparison, all methods use the same developer

tasks, well-established models, user tasks, and reuse evaluation protocol. Each method constructs its own learnware and user specifications and applies the corresponding matching strategy for learnware identification and reuse. Therefore, performance differences primarily reflect the information captured by the specifications and their corresponding matching strategies. The compared methods are described in detail as follows.

- **RKME.** (Wu et al., 2023) Reduced Kernel Mean Embedding (RKME) combines kernel mean embedding with reduced-set approximation to characterize task distributions in a Reproducing Kernel Hilbert Space (RKHS). A compact weighted reduced set preserves the dominant distributional information of the task mastered by a learnware, enabling developer and user tasks to be compared through their RKME specifications. Thus, RKME provides a distribution-based characterization of learnware capability, but does not explicitly incorporate class labels or the predictive behavior of the submitted model. We use a Gaussian kernel with $\gamma = 0.1$ in all experiments.

- **CRKME.** (Liu et al., 2025b) Class-Conditional Reduced Kernel Mean Embedding (CRKME) extends RKME by constructing a separate reduced kernel mean embedding for each class. It therefore characterizes the class-conditional distributions of a task, preserving the dominant distributional information within each class as well as class-level structure that is absent from an unconditional RKME specification. Learnware identification compares these class-wise distributional summaries between developer and user tasks. CRKME thus provides a finer-grained, label-aware indirect characterization of learnware capability, but it does not explicitly model the predictive behavior of the submitted learnware model.

- **LANE.** (Chen et al., 2025b) Label-Aware Neural Embedding (LANE) replaces the fixed RKHS used by RKME with a neural embedding space, thereby avoiding dependence on a hand-specified kernel. It uses both feature and label information to model intra-class feature distributions and learns a compact specification by distribution matching in this label-aware embedding space. Consequently, LANE provides a class-aware distributional characterization of the task, but it does not explicitly encode the predictive response of the submitted learnware model.

- **DALI.** (Chen et al., 2025a) Learnware Specification via Dual ALIgnment (DALI) constructs a compact learnable specification by jointly performing distribution alignment and discriminative alignment. The former preserves the task data distribution, whereas the latter incorporates the discriminative performance of the associated learnware model into specification generation. It is therefore more model-aware than purely distribution-based specifications. In contrast to GPS, however, DALI does not release class-wise one-step gradient directions in a shared proxy coordinate system as its matching representation.

- **MAX and Oracle.** *MAX* selects, for each user task, the single learnware with the highest downstream accuracy. *Oracle* selects the best available reuse result and may use multiple learnwares whenever the scenario permits. Both selectors use downstream evaluation information unavailable to an operational dock and are therefore non-deployable references, reported only to contextualize the performance attainable from the available learnware dock system.

D.3 EXPERIMENT CONFIGURATION

Dataset splits and task construction. Following the experimental configuration in Section 5.1, each benchmark uses its predefined training and test splits. We regard every training superclass as one developer learnware task; the fine-grained labels nested under that superclass are its within-task classes. Test superclasses are used to sample user tasks for the three reuse scenarios, with 100 user tasks sampled per dataset and scenario. Table 3 reports the resulting 131 developer tasks across the 12 evaluated benchmarks. Its class totals count the subclasses retained by these mappings, rather than necessarily all labels in the original public datasets.

Table 3: Learnware-task partitions used in the experiments (131 tasks in total). Each entry in the final column is formatted as *superclass* (number of retained subclasses).

Dataset	Tasks	Classes	Superclass partition
Image benchmarks			
CIFAR-100	20	100	aquatic_mammals (5); electrical_devices (5); fish (5); flowers (5); food-containers (5); fruit-vegetable (5); furniture (5); insects (5); invertebrates (5); large_carnivores (5); medium_sized_mammals (5); omnivores (5); outdoor_scenes (5); outdoor_things (5); people (5); reptiles (5); small_mammals (5); trees (5); vehicles_1 (5); vehicles_2 (5)
DTD	11	34	straight_line_grid (4); dot_bump (3); textile_weave (3); organic_groove_dynamic (3); layered (3); stochastic_granular (4); porous (3); damaged_irregular (3); surface_relief (3); complex_natural (3); fine_delicate (2)
Food101	24	66	cakes (3); desserts_fine (3); desserts_fried (3); desserts_cold (2); desserts_bakery (3); bread_based (3); sandwiches (4); pasta (3); rice_dishes (3); asian_noodles (3); beef (3); pork (2); chicken (2); seafood_main (3); raw_seafood (2); fried_food (3); mexican (3); middle_eastern (2); japanese (3); chinese (3); soups (2); salads (3); appetizers (3); breakfast (2)
GTSRB	4	43	speed_limits (9); prohibitions-derestrictions (8); mandatory (9); danger (17)
Cars	19	192	compact_sedans (10); mid_size_sedans (10); luxury_sedans (12); executive_sedans (18); sports_coupes (10); supercars (11); luxury_convertibles (11); compact_convertibles (15); compact_hatchbacks (14); wagons (5); compact_suvs (8); mid_size_suvs (11); full_size_suvs (15); light_pickups (10); heavy_pickups (8); cargo_vans (6); passenger_vans (6); muscle_cars (5); classic_sports (7)
SUN397	18	313	indoor_residential (33); indoor_commercial (35); indoor_office (13); indoor_education (8); indoor_medical (9); indoor_religious (10); indoor_entertainment (13); indoor_sports (12); indoor_industrial (13); outdoor_urban (16); outdoor_architecture (19); outdoor_transportation (20); outdoor_sports (20); outdoor_agriculture (15); outdoor_natural (38); outdoor_water (12); outdoor_industrial (16); transport_interior (11)
Text benchmarks			
20Newsgroups	5	20	comp (5); misc (3); rec (4); sci (4); talk (4)
TREC	6	50	ABBR (2); ENTY (22); DESC (4); HUM (4); LOC (5); NUM (13)
Coronavirus	2	5	Extremely (2); Bool (3)
WoS	7	134	CS (17); ECE (16); Psychology (19); MAE (9); Civil (11); Medical (53); Biochemistry (9)
Bugs	10	85	ACPI (12); UNKNOW (9); Drivers (23); File-Systems (10); IO-Storage (7); Memory-Management (4); Networking (5); Other (4); Platform-Specific-Hardware (6); Management (5)
Amazon	5	25	arts-crafts-and-sewing (5); electronics (5); grocery-and-gourmet-food (5); musical-instruments (5); video-games (5)

Feature extractors and learnware models. For specification construction, all compared methods use a pre-trained *ResNet110*[1] for image tasks and a pre-trained *Sentence Transformer*[2] for text tasks as shared feature extractors, as described in the main text. These feature extractors are instantiated independently of the well-established models contained in the learnwares and are used solely for specification construction. For each standard image learnware, we independently fine-tune a pre-trained *ResNet110* on the samples from its corresponding superclass, whereas for each standard text learnware, we train a classifier using *TfidfVectorizer* features. For a consistent specification budget, we set $M = 10 \times K$ for all methods, where K denotes the number of classes within a task.

[1]Code of ResNet110: https://github.com/bearpaw/pytorch-classification/blob/master/TRAINING.md

[2]Code of Sentence Transformer: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2/tree/main

Table 4: Performance comparison under the *Within Original Functionality* setting. Results are classification accuracy, reported as mean ± standard deviation over sampled user tasks. For each dataset, the best mean among non-oracle entries is highlighted in bold and underlined.

User Task	*Learnware Identification and Reuse*					*Manual Selection*	
Dataset	**RKME**	**CRKME**	**LANE**	**DALI**	**GPS**	**MAX**	**Oracle**
CIFAR-100	0.461 ± 0.170	0.850 ± 0.137	0.848 ± 0.166	$\underline{\mathbf{0.882 \pm 0.063}}$	$\underline{\mathbf{0.882 \pm 0.063}}$	$\underline{\mathbf{0.882 \pm 0.063}}$	0.882 ± 0.063
Food101	0.229 ± 0.132	0.615 ± 0.298	0.448 ± 0.344	$\underline{\mathbf{0.808 \pm 0.079}}$	$\underline{\mathbf{0.808 \pm 0.079}}$	$\underline{\mathbf{0.808 \pm 0.079}}$	0.808 ± 0.079
DTD	0.218 ± 0.170	0.117 ± 0.232	0.490 ± 0.331	$\underline{\mathbf{0.802 \pm 0.119}}$	0.792 ± 0.145	$\underline{\mathbf{0.802 \pm 0.119}}$	0.802 ± 0.119
Cars	0.092 ± 0.140	0.057 ± 0.126	0.113 ± 0.149	0.210 ± 0.195	0.236 ± 0.208	$\underline{\mathbf{0.441 \pm 0.163}}$	0.441 ± 0.163
SUN397	0.166 ± 0.124	0.111 ± 0.104	0.354 ± 0.175	0.500 ± 0.150	0.494 ± 0.148	$\underline{\mathbf{0.551 \pm 0.124}}$	0.551 ± 0.124
GTSRB	0.427 ± 0.231	0.971 ± 0.093	0.938 ± 0.152	$\underline{\mathbf{0.997 \pm 0.003}}$	$\underline{\mathbf{0.997 \pm 0.003}}$	$\underline{\mathbf{0.997 \pm 0.003}}$	0.997 ± 0.003
20Newsgroups	0.500 ± 0.113	0.771 ± 0.174	0.821 ± 0.138	0.658 ± 0.278	0.858 ± 0.071	$\underline{\mathbf{0.864 \pm 0.057}}$	0.864 ± 0.057
Amazon	0.780 ± 0.132	$\underline{\mathbf{0.856 \pm 0.077}}$	0.846 ± 0.092	$\underline{\mathbf{0.856 \pm 0.077}}$	$\underline{\mathbf{0.856 \pm 0.077}}$	$\underline{\mathbf{0.856 \pm 0.077}}$	0.856 ± 0.077
Bugs	0.259 ± 0.199	0.283 ± 0.295	0.330 ± 0.298	0.183 ± 0.247	0.440 ± 0.284	$\underline{\mathbf{0.483 \pm 0.280}}$	0.483 ± 0.280
Coronavirus	0.516 ± 0.211	0.497 ± 0.246	0.508 ± 0.368	0.343 ± 0.289	0.615 ± 0.323	$\underline{\mathbf{0.716 \pm 0.179}}$	0.716 ± 0.179
TREC	0.355 ± 0.205	0.342 ± 0.341	0.592 ± 0.222	0.528 ± 0.284	0.619 ± 0.204	$\underline{\mathbf{0.626 \pm 0.195}}$	0.626 ± 0.195
WoS	0.501 ± 0.145	0.677 ± 0.139	0.690 ± 0.143	0.702 ± 0.131	0.716 ± 0.110	$\underline{\mathbf{0.722 \pm 0.103}}$	0.722 ± 0.103
Avg. Metric (↑)	0.375	0.512	0.582	0.622	0.693	0.729	0.729
Avg. Rank (↓)	5.33	4.71	4.08	3.25	2.21	1.42	–
GPS (win/tie/loss)	12/0/0	11/1/0	12/0/0	6/4/2	–	0/4/8	0/4/8

Corrupted learnwares. To form corrupted learnwares, we replace the standard model construction with a *CNN* on image tasks and a *MultinomialNB* on text tasks. We also introduce label errors into their training data. Consequently, these learnwares differ from the standard pool in both predictive model family and label reliability, providing a controlled source of unreliable learnwares.

D.4 LEARNWARE REUSE SCENARIO

We generate user tasks by sampling subclasses from the superclass partitions in Table 3. For every benchmark and scenario, we sample 100 user tasks. The three scenarios differ only in how the selected subclasses are distributed across superclasses and in the available user supervision:

- **Beyond Original Functionality.** We randomly select and combine subclasses from multiple superclasses. No individual learnware covers the complete user task, motivating multi-learnware reuse.
- **Few-Shot Beyond Original Functionality.** We use the same multi-superclass subclass sampling as in the preceding scenario, but retain only 10 labeled user examples for each selected subclass when constructing the user specification.
- **Within Original Functionality.** We randomly select subclasses from one superclass to form each user task. The task is thus covered by the functionality of a single learnware.

Performance analysis. The three settings consistently demonstrate the advantage of GPS across different learnware reuse scenarios. Under *Beyond Original Functionality* (Table 1), GPS achieves the highest average accuracy (0.686) and best average rank (1.50), outperforming DALI by 0.052 and exceeding the single-learnware *MAX* reference on 11 of 12 datasets. The gains are particularly pronounced on text benchmarks, while GPS remains competitive with the strongest baselines on image tasks. Its remaining 0.045 gap to the *Oracle* (0.731) highlights the challenge of identifying the optimal combination of complementary learnwares.

The advantage persists under *Few-Shot Beyond Original Functionality* (Table 2), where only ten labeled examples per class are available. GPS again achieves the best average accuracy (0.666) and rank (1.46), wins or ties against RKME, CRKME, and LANE on all 12 benchmarks, and remains within 0.048 of the *Oracle* (0.714). The modest decrease from 0.686 to 0.666 suggests that its gradient-based specification remains effective for identifying complementary learnwares under limited user supervision.

Under *Within Original Functionality* (Table 4), GPS achieves the highest average accuracy (0.693) among deployable specification methods and the best average rank (2.21), winning or tying against every specification baseline on at least 10 of 12 datasets. Although the performance-based *MAX* and *Oracle* references remain stronger (0.729), GPS approaches them without access to downstream user-task performance. Overall, these results demonstrate robust learnware identification and reuse across both single- and multi-learnware settings.

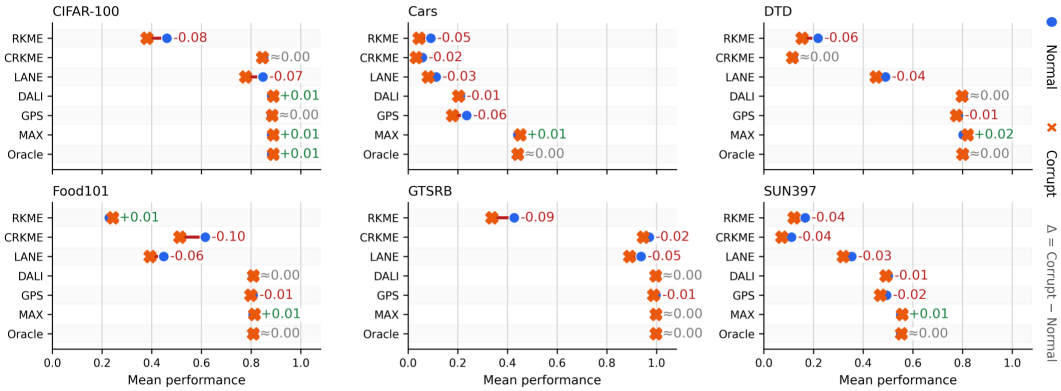

Figure 4: Ablation study on model quality under the *Within Original Functionality* setting on image benchmarks. *Normal* contains only the original well-established models, whereas *Corrupted* additionally includes their corrupted counterparts.

D.5 Model quality

This ablation evaluates robustness to unreliable learnwares. *Normal* uses only the standard learnware dock system described in Appendix D.3; *Corrupted* augments that dock system with one corrupted counterpart per standard learnware. The counterparts use the alternative model families and noisy labels described above, while the user task sampling and evaluation method remain unchanged. Figs. 3–8 show that enlarging the pool with corrupted learnwares has little effect on GPS in most benchmark–scenario pairs. On the image benchmarks, its performance changes are consistently small (within 0.06 in absolute value), and it remains essentially unchanged on the high-performing DALI/GPS and Oracle-level cases. The same trend largely carries over to text: although the more challenging beyond-original settings can cause a larger drop (e.g., Coronavirus and TREC), GPS is markedly more stable than RKME, CRKME, and LANE, whose performance often decreases by 0.1–0.34 after the corrupted models are introduced. In contrast, several methods exhibit occasional gains under the corrupted pool, indicating that their retrieval can be sensitive to incidental changes in the candidate composition rather than consistently preferring reliable models. These results show that the class-wise gradient responses used by GPS encode model-dependent predictive quality in addition to task similarity, allowing the retrieval procedure to suppress most unreliable counterparts while retaining the benefit of the original learnwares.

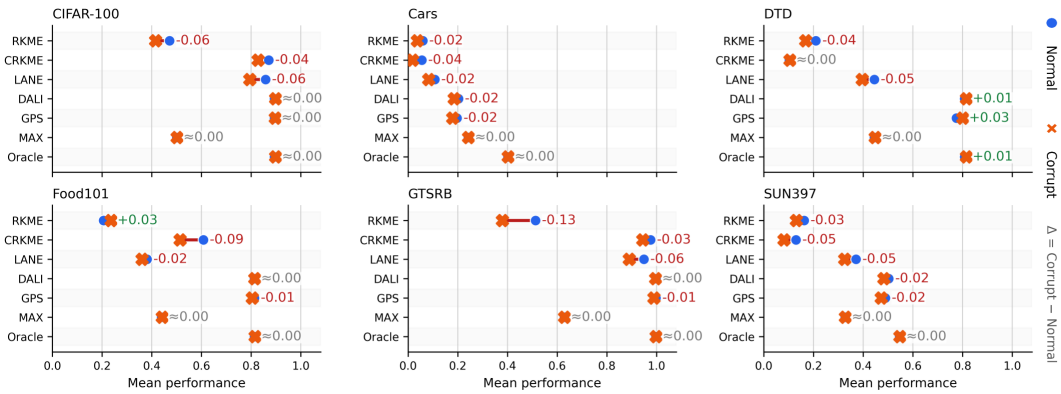

Figure 5: Ablation study on model quality under the *Beyond Original Functionality* setting on image benchmarks. *Normal* contains only the original well-established models, whereas *Corrupted* additionally includes their corrupted counterparts.

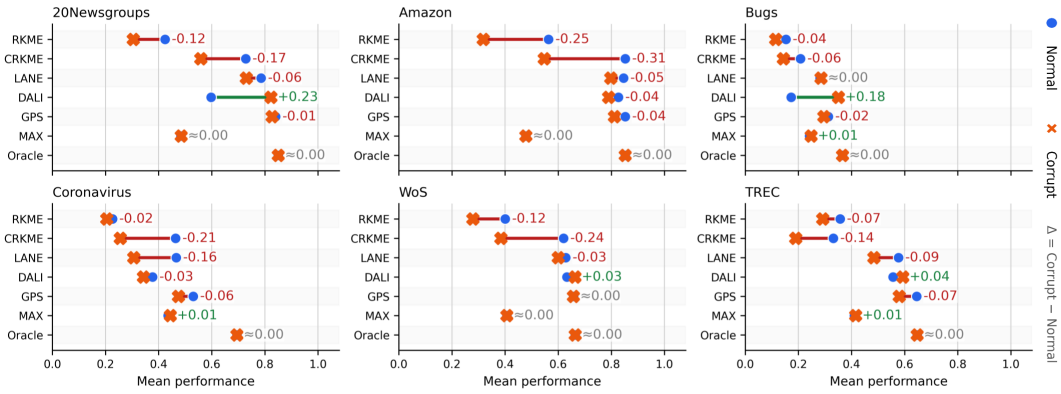

Figure 6: Ablation study on model quality under the *Few-Shot Beyond Original Functionality* setting on text benchmarks. *Normal* contains only the original well-established models, whereas *Corrupted* additionally includes their corrupted counterparts.

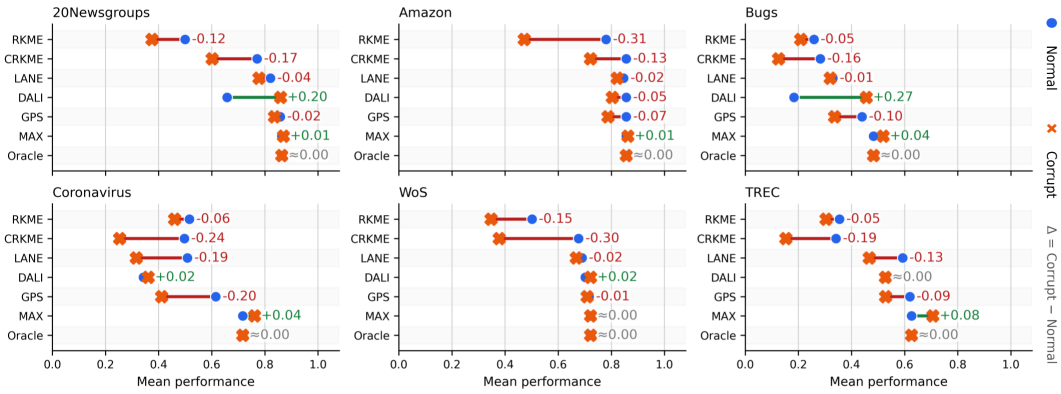

Figure 7: Ablation study on model quality under the *Within Original Functionality* setting on text benchmarks. *Normal* contains only the original well-established models, whereas *Corrupted* additionally includes their corrupted counterparts.

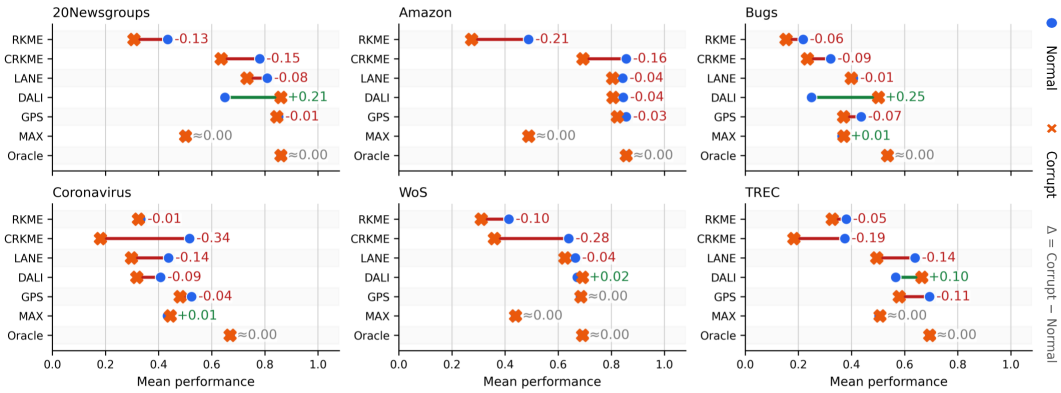

Figure 8: Ablation study on model quality under the *Beyond Original Functionality* setting on text benchmarks. *Normal* contains only the original well-established models, whereas *Corrupted* additionally includes their corrupted counterparts.

D.6 PROXY SAMPLE SIZE

Fig. 9 examines the sensitivity of GPS to the proxy sample size. Increasing the size from 5 to 10 samples per class generally improves or maintains reuse performance, with more pronounced gains on challenging benchmarks such as Bugs and Coronavirus. Further increasing the size to 20, however, yields no consistent improvement across modalities or reuse scenarios, with most results remaining close to those obtained with 10 samples per class and differences often falling within the displayed standard deviations. These results suggest that 10 proxy samples per class are generally sufficient for effective reuse, while larger proxy sets provide diminishing and task-dependent returns. We therefore adopt 10 samples per class as the default setting, providing a favorable trade-off between reuse performance and proxy construction cost.

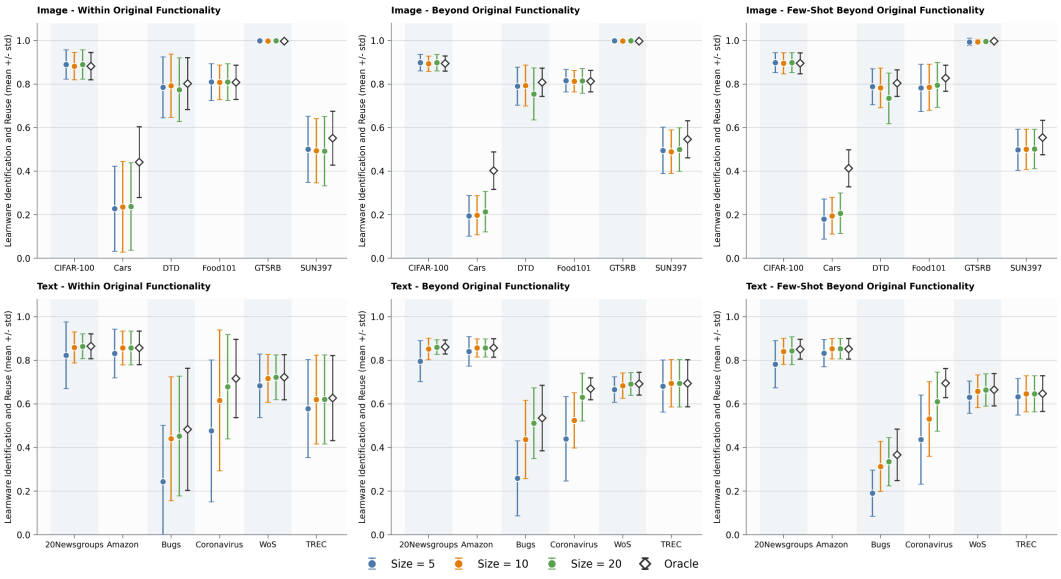

Figure 9: Effect of the proxy sample size on learnware identification and reuse. Points and error bars indicate the mean and standard deviation over different user tasks, respectively. The three colored markers correspond to proxy sizes of 5, 10, and 20 samples per class; the black diamond denotes the Oracle reference.

D.7 EFFICIENCY ANALYSIS

We measure wall-clock time (in seconds) for three stages: learnware specification generation, user specification generation, and learnware identification and reuse. The final stage includes both retrieving suitable learnwares from the dock system and generating predictions with the selected learnwares. Table 5 reports the mean runtime across the standard benchmark configurations, using a specification size of 10. Learnware specification generation is an offline cost incurred once when a learnware is submitted to the dock system, whereas user specification generation and learnware identification and reuse are online costs incurred for each user task. We focus particularly on the comparison between GPS and DALI, the strongest competing specification method in the main experiments. For learnware specification generation, GPS requires 0.0492 seconds, reducing the runtime by 42.5% relative to DALI's 0.0855 seconds. This demonstrates that GPS reduces the one-time specification construction cost before serving user requests. Its runtime also remains within the same order of magnitude as RKME (0.0252 seconds) and LANE (0.0279 seconds).

For user specification generation, GPS requires 0.0263, 0.0697, and 0.0661 seconds under the *Within Original Functionality*, *Beyond Original Functionality*, and *Few-Shot Beyond Original Functionality* settings, respectively. This pre-retrieval overhead is higher than that of DALI in the within-original and few-shot settings, while being nearly identical in the beyond-original setting. Nevertheless, these costs remain small in absolute terms relative to the subsequent identification-and-reuse stage. The main efficiency advantage of GPS lies in learnware identification and reuse. It requires only 0.23, 0.53, and 0.13 seconds across the three settings, compared with 0.35, 0.94, and 0.22 sec-

Table 5: Efficiency comparison in terms of mean wall-clock time (seconds) across standard benchmark configurations. "User Specification / Learnware Identification and Reuse" reports the time for user-specification generation and the total time for learnware identification and subsequent reuse, respectively. Here, d_{data} denotes the data dimensionality, and d_{feature} denotes the input feature dimensionality of the randomly initialized prediction head $g_{\boldsymbol{\omega}_0}$.

Method	Specification size M	Learnware specification	Within original	Beyond original	Few-shot beyond
RKME	$K \times d_{\text{data}}$	0.0252	0.0212 / 6.75	0.0122 / 9.09	0.0088 / 2.68
CRKME	$(K \times l) \times d_{\text{data}}$	0.0589	0.0340 / 1.03	0.0920 / 2.44	0.0699 / 0.97
LANE	$(K \times l) \times d_{\text{data}}$	0.0279	0.0059 / 0.33	0.0188 / 0.75	0.0183 / 0.35
DALI	$(K \times l) \times d_{\text{data}}$	0.0855	0.0138 / 0.35	0.0694 / 0.94	0.0149 / 0.22
GPS	$K \times d_{\text{feature}}$	0.0492	0.0263 / 0.23	0.0697 / 0.53	0.0661 / 0.13

Table 6: Modality-specific cross-benchmark significance analysis based on six dataset-level mean accuracies per modality. Δ is the mean accuracy gain of GPS over the indicated baseline. Parenthesized values are Holm–Bonferroni-adjusted one-sided Wilcoxon signed-rank p-values. Friedman p-values use the standard chi-square approximation.

Modality	Scenario	Friedman χ^2	Friedman p	RKME	CRKME	LANE	DALI
Image	Within	19.73	< 0.001	$+0.436$ (0.063)	$+0.248$ (0.063)	$+0.170$ (0.063)	$+0.002$ (0.500)
Image	Beyond	20.00	< 0.001	$+0.427$ (0.063)	$+0.240$ (0.063)	$+0.179$ (0.063)	-0.006 (0.946)
Image	Few-Shot Beyond	21.73	< 0.001	$+0.461$ (0.063)	$+0.327$ (0.063)	$+0.171$ (0.063)	-0.009 (0.953)
Text	Within	13.24	0.010	$+0.199$ (0.063)	$+0.113$ (0.063)	$+0.053$ (0.063)	$+0.139$ (0.063)
Text	Beyond	17.58	0.001	$+0.296$ (0.063)	$+0.092$ (0.063)	$+0.041$ (0.063)	$+0.109$ (0.063)
Text	Few-Shot Beyond	18.59	< 0.001	$+0.286$ (0.063)	$+0.106$ (0.063)	$+0.167$ (0.063)	$+0.113$ (0.063)

onds for DALI, corresponding to runtime reductions of 34.5%, 43.5%, and 39.6%, respectively. GPS is also the fastest method at this stage across all three settings. Moreover, GPS reaches a stable specification with fewer optimization rounds, whereas DALI requires at least 100 rounds to converge. Together with its lower one-time learnware specification cost, these results indicate that GPS provides a favorable efficiency–performance trade-off, substantially accelerating learnware identification and reuse while incurring only modest additional user-specification overhead in some settings.

D.8 STATISTICAL SIGNIFICANCE

Rather than treating tasks sampled from the same benchmark and learnware dock as independent observations, we use dataset-level mean accuracies, yielding six paired units per modality and scenario. This analysis evaluates consistency across benchmarks and complements task-level paired tests based on the 100 user-task accuracies.

For each modality and scenario, we apply the Friedman test (Friedman, 1940; Demšar, 2006) to the five specification methods (RKME, CRKME, LANE, DALI, and GPS). We then compare GPS with each baseline using a one-sided Wilcoxon signed-rank test, with the alternative hypothesis that GPS has higher accuracy (Wilcoxon, 1945). Tied dataset-level means yield zero paired differences and are omitted from the Wilcoxon ranks. We adjust the four post-hoc comparisons within each modality–scenario pair using Holm–Bonferroni (Holm, 1979). *MAX* and *Oracle* are excluded because they use downstream evaluation information and are not deployable identification methods.

Table 6 reports the modality-specific results. The standard chi-square approximation to the Friedman test is significant for every modality–scenario pair, although the result should be interpreted with caution given the small sample size. With only six datasets per modality, the smallest attainable exact one-sided Wilcoxon p-value is 0.015625; after correcting four comparisons, it becomes 0.0625. Thus, none of the pairwise post-hoc tests reaches the 0.05 threshold after Holm correction, including cases in which all six datasets favor GPS. We therefore report effect directions and adjusted p-values transparently. In future work, we will consider a finer-grained task-level paired analysis based on per-user-task accuracy vectors.