COMPLEMENTARY LEARNWARE REUSE VIA DISTRIBUTION FILLING

Anonymous authors
Paper under double-blind review

ABSTRACT

The growing availability of AI models calls for systems that can organize and combine their capabilities for new tasks. The *learnware paradigm* supports this goal by pairing each model with a specification describing its capabilities, enabling identification and reuse without sharing the training data of developers. Since models are developed for different tasks and conditions, no single learnware may adequately serve a new user: each may cover only part of the required input distribution, together with regions irrelevant to the task. When different learnwares cover complementary parts, combining their useful portions can enable the system to solve tasks that no single learnware can adequately handle. The challenge is to determine how learnwares can jointly serve different parts of the user distribution while reducing the influence of irrelevant regions. By modeling learnware utility at individual user inputs, we characterize how well a learnware set serves the user distribution and where an additional learnware can fill gaps in its coverage. This perspective motivates distribution filling, which jointly reweights regions across learnware specifications, so that they collectively match the user distribution while irrelevant regions are down-weighted. For each learnware, the resulting weights quantify its overall relevance, which is combined with its compatibility with individual inputs to determine how much it contributes to the prediction on each input. Experiments show improved predictive performance over existing multi-learnware reuse methods and further demonstrate favorable accuracy–efficiency trade-offs for our method.

1 INTRODUCTION

The growing number and diversity of AI models call for systems that can organize and combine their capabilities for new tasks. The *learnware paradigm* provides a basis for such AI model management (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026). A learnware pairs a trained model with a *specification*, making the model a unit for systematic management, identification, and reuse. Natural-language descriptions alone may not capture the fine-grained differences in model capabilities needed for reuse. Specifications provide a computable characterization and are generated locally without sharing the original training data. Users express their requirements through task descriptions and specifications constructed from locally available data. By matching these requirements against the submitted specifications, the system identifies suitable learnwares for users to apply directly, combine, or further adapt with local labels when available. Beimingwu provides an open-source implementation of this workflow (Tan et al., 2024b).

This reuse workflow connects models developed for their original tasks with users whose needs may differ from those tasks. Such differences make partial relevance a common concern: a learnware can serve some regions of the user distribution while also covering conditions irrelevant to the current task. Consider fault diagnosis at a factory operating under low and high loads (Figure 1). Two learnwares predict the same faults from the same sensors, but one was developed under low/medium loads and the other under medium/high loads. Each covers a required regime together with an unneeded medium-load regime, while neither alone covers both required regimes. The same issue arises across models developed for different populations, operating environments, or class ranges: what was useful for the original task need not coincide with what a new user needs. Identifying and combining the relevant capabilities within such learnwares can therefore enable a system to address a broader range of tasks than individual models adequately serve.

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.

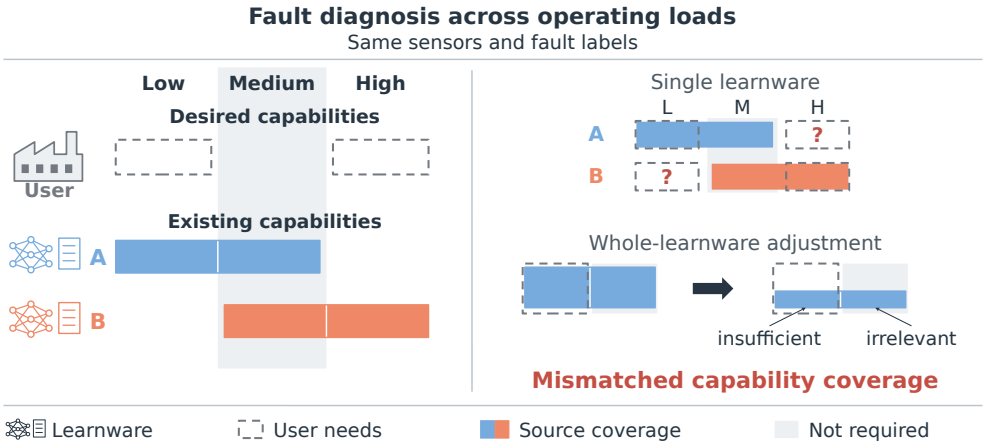


Figure 1: **Mismatch between desired and existing capabilities.** In factory fault diagnosis with the same sensors and fault labels, the user needs low- and high-load regimes, while learnwares A and B cover low/medium and medium/high loads. Neither suffices alone, and scaling either source as a whole reduces useful coverage together with irrelevant coverage.

Realizing this potential requires understanding which parts of each learnware serve the user and what they add to other available learnwares. We focus on *subsumption*, where portions of several source distributions jointly cover the user distribution, while the sources also include regions outside the user task. In the factory example, the desired low- and high-load capabilities come from different learnwares, but each is accompanied by unneeded medium-load coverage. Treating the coverage of a learnware as a whole makes it difficult to retain its useful part while reducing the influence of its irrelevant part. Existing reuse approaches exploit different forms of evidence about model usefulness. Ensemble selection evaluates model combinations through aggregate validation performance (Caruana et al., 2004). MoreBoost (Ding & Zhou, 2020) combines models using input-wise reusability indicators, but errors in these indicators can impair reuse. JobSelector, a representative specification-based reuse method (Wu et al., 2023), matches the user distribution through a mixture of complete source distributions, coupling the useful and irrelevant regions of each candidate through one mixture coefficient. These approaches do not explicitly optimize how regions within different source specifications jointly match the user distribution. The central challenge is thus to characterize, from compact specifications, how regions within and across learnwares contribute to user needs, so that their complementary capabilities can be combined while the influence of irrelevant coverage is reduced. Appendix A discusses related learnware, transfer learning, and model combination methods.

Our insight is to characterize the usefulness of a learnware at individual user inputs and assess its value to a set by the additional capabilities it provides where the set falls short. We use the best available utility at each input to characterize the coverage of a learnware set. A new learnware contributes where it improves this coverage, rather than simply duplicating capabilities already available. Thus, a learnware with limited overall relevance can still be valuable when its useful regions fill gaps left by others. This view suggests examining how different regions within learnwares jointly meet user needs, instead of judging each learnware only as a whole. True predictive utility cannot be read directly from the available specifications, so we use the distributional information they provide to seek evidence of such complementary coverage. Motivated by this view, we propose *distribution filling* for complementary learnware reuse: it jointly adjusts the contributions of regions within different learnware specifications to match the user distribution. This yields the overall relevance of each learnware to the task. Overall relevance alone, however, does not indicate which user inputs a learnware can serve. We therefore combine it with local compatibility from the original specification, relating the contribution of the learnware to the task as a whole to its suitability for each input. In this way, the method translates complementary distribution coverage into input-wise reuse.

Our contributions are as follows:

- We develop a unified utility view of complementary learnware reuse, connecting the usefulness of a learnware at individual user inputs, the coverage provided by a learnware

set, and the additional value of a new member. The same formulation encompasses individual learnwares and learnware sets, explaining why a partially relevant learnware can be important when it supplies capabilities missing from others.

- We propose distribution filling for complementary learnware reuse, which jointly reweights regions within candidate specifications rather than scaling each specification as a whole. Our analysis relates the optimal weight of each region to the joint residual; combining the resulting global relevance with local compatibility guides input-wise reuse.
- At least one variant of our method achieves the best mean accuracy among the evaluated methods in 54/60 dataset–scenario pairs, including ties. Controlled comparisons with existing multi-learnware reuse methods also show favorable accuracy–efficiency trade-offs.

2 PRELIMINARIES

A learnware pairs a trained model f_i with a specification σ_i describing its capabilities. We use reduced kernel mean embedding (RKME) specifications to represent input distributions (Wu et al., 2023; Tan et al., 2024b). Let $\mathcal{X}$ be an input space and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ a positive-definite kernel with reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$. For a distribution P on $\mathcal{X}$, its kernel mean embedding (KME) (Smola et al., 2007) and empirical estimate from i.i.d. samples $\{x_n\}_{n=1}^N$ are

$$\mu_P = \mathbb{E}_{x\sim P}[k(x,\cdot)] \in \mathcal{H}_k, \qquad \widehat{\mu}_P = \frac{1}{N}\sum_{n=1}^{N} k(x_n,\cdot).$$

Under standard conditions, $\widehat{\mu}_P$ converges to μ_P at rate $O(N^{-1/2})$ (Smola et al., 2007). The distance $\|\mu_P - \mu_Q\|_{\mathcal{H}_k}$ is the maximum mean discrepancy (MMD) between P and Q (Gretton et al., 2012).

RKME approximates the empirical embedding using representative points $Z_i = \{z_{ij}\}_{j=1}^{m_i}$ and coefficients β_{ij}. We construct nonnegative coefficients summing to one and omit zero-coefficient terms. Indexing the remaining points gives

$$\sigma_i = \sum_{j=1}^{m_i} \beta_{ij} k(z_{ij},\cdot), \qquad z_{ij} \in \mathcal{X}, \quad \beta_{ij} > 0, \quad \sum_j \beta_{ij} = 1. \tag{1}$$

Appendix B details RKME construction. Specifications are constructed locally without releasing raw training samples. Under the stated assumptions and suitable specification sizes, RKME resists original-sample disclosure, linkage, and inference attacks (Lei et al., 2024), and adds negligible inference risk beyond releasing the model alone (Lei et al., 2026).

3 PROBLEM SETUP

A learnware dock system connects developers and users in two stages.

Submitting stage. Developers submit trained models f_i together with locally constructed specifications σ_i, forming a pool of M learnwares, $\mathcal{L} = \{L_i = (f_i, \sigma_i)\}_{i=1}^M$. The system receives the models and specifications without access to the original training data of developers.

Deploying stage. A user seeks a predictor for an unknown joint distribution $\mathcal{D}_u$ over input–label pairs (x, y). From N_u unlabeled inputs $\mathcal{U}_u = \{x_n^u\}_{n=1}^{N_u}$ drawn from its input marginal $\mathcal{D}_u^X$, the user constructs an RKME σ_u with points z_{uj}, coefficients β_{uj}, and point count m_u. The user submits this specification and task requirements without sharing raw inputs or instance labels. Users may hold labels locally; our specification construction and reuse do not use them. Candidate outputs are aligned with the user task through known task-level correspondences; a classification candidate may cover only part of the user label set. Given a candidate budget B, the goal is to identify and combine at most B learnwares into a predictor $\widehat{f}_u$ with low expected loss $\mathbb{E}_{(x,y)\sim\mathcal{D}_u}[\ell(\widehat{f}_u(x), y)]$, where ℓ is the task loss. We focus on the homogeneous setting, where learnwares and user inputs share the feature space $\mathcal{X}$ and kernel k. Existing specification mapping and feature alignment methods support extension to heterogeneous feature spaces (Tan et al., 2026); Section 5.4 evaluates this extension within Beimingwu.

4 PROPOSED APPROACH

4.1 A UTILITY VIEW OF COMPLEMENTARY REUSE

A learnware can be valuable because it serves user inputs that other candidates do not adequately cover. Let $U(x, L_i) \in [0, 1]$ denote its utility at input x, accounting for predictive usefulness under the task and deployment requirements of the user. For fixed measurable utilities, the expected coverage of a set $\mathcal{S} \subseteq \mathcal{L}$ is

$$F_u(\mathcal{S}) = \mathbb{E}_{x\sim\mathcal{D}_u^X}\left[\max_{L_i\in\mathcal{S}} U(x, L_i)\right], \qquad F_u(\emptyset) = 0. \tag{2}$$

The maximum rewards the best available utility at each input without counting repeated coverage. This gives a common utility view of reuse with one or several learnwares: a singleton is evaluated by its expected utility, while a larger set is evaluated by the best utility its members can provide at each input. Writing $v_{\mathcal{S}}(x) = \max_{L_j\in\mathcal{S}} U(x, L_j)$, with $v_\emptyset(x) = 0$, the contribution of $L_i \notin \mathcal{S}$ is

$$F_u(\mathcal{S} \cup \{L_i\}) - F_u(\mathcal{S}) = \mathbb{E}_{x\sim\mathcal{D}_u^X}\left[\big(U(x, L_i) - v_{\mathcal{S}}(x)\big)_+\right], \tag{3}$$

where $(a)_+ = \max\{a, 0\}$. A partially relevant learnware can thus add value by improving coverage where the existing set is weak. Appendix C.1 gives the ideal budgeted selection objective and the proof of diminishing returns. This utility concerns choosing a model per input; it does not characterize the utility of prediction aggregation.

The available specifications describe input distributions, while predictive utility is unobserved. With task correspondences given, we use distributional coverage as statistical evidence of complementary capabilities. This leads to two requirements: contributions within a source must be adjustable so that irrelevant regions do not constrain useful ones, and reuse must distinguish which candidate is compatible with each user input. DF addresses the first through joint pointwise matching and the second through local compatibility. The utility view motivates these choices without asserting that distribution matching directly optimizes F_u.

Method overview. RKME-based retrieval first returns K compatible candidates, restricting subsequent matching to their specifications:

$$\mathcal{C} = \{L_i = (f_i, \sigma_i)\}_{i=1}^K \subseteq \mathcal{L}, \qquad K \le B, \quad K \ll M, \tag{4}$$

Candidates are reindexed within $\mathcal{C}$. We jointly reweight their representative points to match the user specification, implementing the regional view in Figure 1. Within-candidate weight sums give task-level relevance, which combines with compatibility from the original specification to guide input-wise model selection or prediction aggregation (Figure 2).

4.2 POINTWISE DISTRIBUTION REWEIGHTING

Whole-specification matching in JobSelector (Wu et al., 2023) assigns one mixture weight θ_i to each candidate L_i. With the original point coefficients β_{ij} and locations z_{ij} from Equation 1, its representation is

$$\sum_{i=1}^K \theta_i\sigma_i = \sum_{i=1}^K\sum_{j=1}^{m_i} \theta_i\beta_{ij}k(z_{ij}, \cdot), \qquad \theta_i \ge 0, \quad \sum_i \theta_i = 1. \tag{5}$$

Whole-specification weighting couples useful and irrelevant regions. Distribution filling instead adjusts the weight w_{ij} of each point at its fixed location z_{ij}. Setting $w_{ij} = \theta_i\beta_{ij}$ recovers any whole-specification mixture; independent adjustment allows the regions to contribute differently. It can thus combine partial coverage and reduce excess mass without identifying the scenario in advance.

Let $\mathcal{A} = \{(i, j) : 1 \le i \le K,\ 1 \le j \le m_i\}$ index the retrieved RKME points. Reference masses $\rho_{ij} = \beta_{ij}/K$ give each learnware equal total mass while preserving its internal proportions. We regularize deviation from this reference to discourage concentration:

$$\min_{\substack{w_{ij}\ge 0\\ \sum_{(i,j)\in\mathcal{A}} w_{ij}=1}} \frac{1}{2}\left\|\sigma_u - \sum_{(i,j)\in\mathcal{A}} w_{ij}k(z_{ij}, \cdot)\right\|_{\mathcal{H}_k}^2 + \frac{\lambda}{2}\sum_{(i,j)\in\mathcal{A}} \frac{w_{ij}^2}{\rho_{ij}}, \qquad \lambda > 0. \tag{6}$$

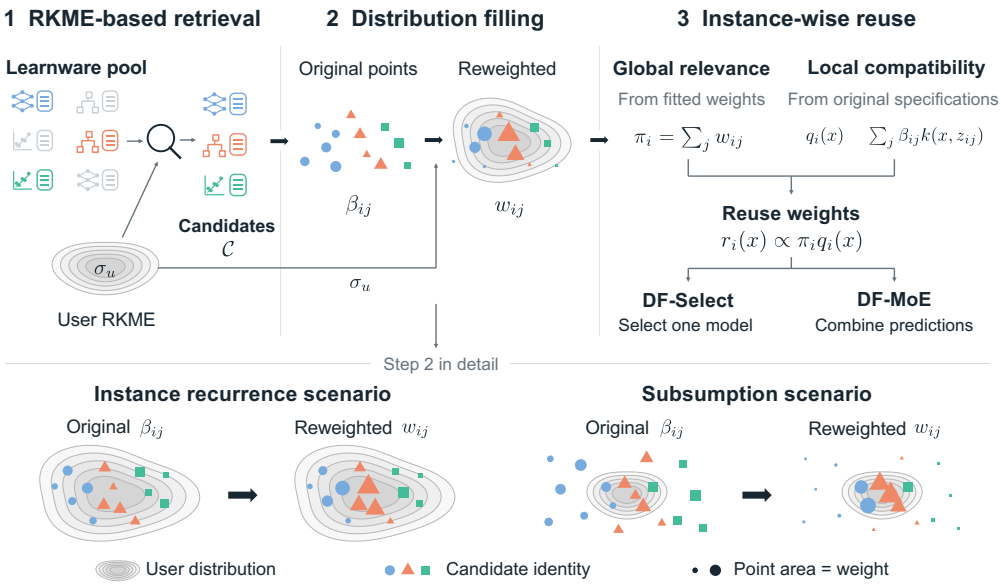


Figure 2: **Distribution filling and reuse.** The upper row shows RKME-based retrieval, joint filling, and input-wise reuse combining global relevance with local compatibility. The lower row expands the filling step in two scenarios. Under instance recurrence (Wu et al., 2023), the user distribution is a mixture of source distributions. Under subsumption, filling also reduces mass in source regions outside user needs. Point locations remain unchanged; areas encode weights, colors and marker shapes identify candidates, and gray shading and contours indicate the user distribution.

The first term matches the reweighted points to the user specification in RKHS. The second penalizes concentration relative to the reference, especially large weights on points with little reference mass. Parameter λ balances matching against this penalty, a Pearson χ^2 divergence (Ali & Silvey, 1966) up to a constant, without imposing a predefined cutoff or fixing internal proportions. Positive ρ_{ij} and λ ensure a strictly convex quadratic program with a unique solution, computed once per task. Appendix C.2 gives the matrix formulation and derivation.

To see how DF accounts for coverage supplied by other candidates, write $\sigma_{w^\star} = \sum_{(i,j)\in\mathcal{A}} w^\star_{ij} k(z_{ij}, \cdot)$ for the jointly fitted specification.

Proposition 1 (Allocation by the joint residual) *Let $w^\star$ solve Equation 6. There exists $\tau \in \mathbb{R}$ such that*

$$w^\star_{ij} = \frac{\rho_{ij}}{\lambda} \left[\langle \sigma_u - \sigma_{w^\star}, k(z_{ij}, \cdot) \rangle_{\mathcal{H}_k} - \tau \right]_+ . \tag{7}$$

Appendix D provides the proof. Proposition 1 shows that a point receives positive mass exactly when its residual score exceeds the common threshold τ, with its reference mass ρ_{ij} scaling the allocation. Thus, a region that resembles user inputs can still receive little mass when its coverage is already supplied by other candidates. A useful region can also contribute even when other regions of the same source receive no mass. DF therefore assesses each region in the context of joint coverage, allowing contributions to vary within an original source task.

Discussion. Distribution filling can represent user specifications beyond mixtures of complete source specifications by allowing their regions to contribute in different proportions. Proposition 1 further explains how these contributions are coordinated: the residual compares user demand with coverage supplied by the joint allocation. Expanding it gives

$$\langle \sigma_u - \sigma_{w^\star}, k(z_{ij}, \cdot) \rangle_{\mathcal{H}_k} = \underbrace{\sum_{t=1}^{m_u} \beta_{ut} k(z_{ut}, z_{ij})}_{\text{user kernel response}} - \underbrace{\sum_{\ell=1}^{K} \sum_{t=1}^{m_\ell} w^\star_{\ell t} k(z_{\ell t}, z_{ij})}_{\text{joint candidate kernel response}} .$$

For a Gaussian kernel, the first term measures the concentration of user inputs around z_{ij}, while the second measures the coverage represented there by all weighted candidate points. A region's allocation thus responds to the mismatch remaining under the joint fit, rather than user similarity alone. Complementary contributions are determined through the joint allocation, without specifying in advance which candidates should cover which regions. The weights describe distributional coverage rather than predictive ability. Corollary 3 further shows that a learnware receives zero global weight exactly when none of its points has a residual score above τ.

4.3 Input-Wise Model Selection and Aggregation

The joint match assigns mass to representative points, while prediction uses the candidate models. We aggregate the fitted mass within each candidate into global relevance π_i. To distinguish where each model is suitable, we also compute its local compatibility q_i:

$$\pi_i = \sum_{j=1}^{m_i} w_{ij}, \qquad q_i(x) = \sum_{j=1}^{m_i} \beta_{ij} k(x, z_{ij}), \qquad i = 1, \ldots, K. \tag{8}$$

The π_i sum to one and express contributions to the task-level match. A small fitted point weight can indicate redundant coverage rather than low predictive ability near that point. We therefore use the original specification for local compatibility, retaining source-distribution information alongside the fitted task-level contribution. For a Gaussian kernel, $q_i(x)$ is proportional to a weighted kernel density estimate (KDE) (Rosenblatt, 1956). Normalizing the product of global relevance and local compatibility over all candidates gives the input-specific reuse weight $r_i(x)$:

$$r_i(x) = \frac{\pi_i q_i(x)}{\sum_{h=1}^{K} \pi_h q_h(x)}. \tag{9}$$

If the denominator is zero, set $r_i(x) = \pi_i$. If retrieval is empty, abstain before filling. Otherwise, reuse weights are nonnegative and sum to one.

Using these weights, DF–Select selects a model with the largest $r_i(x)$, indexed by $i^\star(x)$, while DF–MoE aggregates model outputs. For classification, DF–MoE combines aligned probabilities $\widehat{p}_i(y \mid x)$ from f_i for user label y, assuming they are comparable and calibrated on the same scale. The predictions in user-task coordinates are

$$\widehat{y}_{\mathrm{Select}}(x) = f_{i^\star(x)}(x), \qquad \widehat{y}_{\mathrm{MoE}}(x) = \arg\max_{y} \sum_{i=1}^{K} r_i(x) \widehat{p}_i(y \mid x). \tag{10}$$

For regression, DF–MoE returns $\widehat{y}_{\mathrm{MoE}}(x) = \sum_{i=1}^{K} r_i(x) f_i(x)$, with numerical predictions aligned to the user target.

Both rules use the same statistical evidence without training an additional selector. Appendix C.3 gives Algorithm 1; Appendix C.4 details the computational costs.

5 Experiments

We first demonstrate distribution filling in a synthetic example with irrelevant source coverage. The main benchmark evaluates predictive performance on 17 tabular and 13 image datasets under subsumption and instance recurrence. A sales-forecasting study tests heterogeneous reuse across 40 stores. Ablations isolate the effects of global relevance, local compatibility, and pointwise filling. Parameter scans assess sensitivity to regularization strength and specification size. Timing experiments assess accuracy–efficiency trade-offs across candidate budgets. Appendix G.3 reports a comparison with tree-based assembly using limited labeled user examples.

5.1 A Demonstration of Distribution Filling

Figure 3 makes pointwise reweighting directly visible in a controlled example with three user regions and three candidates, each covering two user regions and one extra region. We increase extra mass from 0 to 90% while fixing useful prototype locations. Whole-specification matching and DF share prototypes and user samples. Appendix G.1 gives the distributions and complete results.

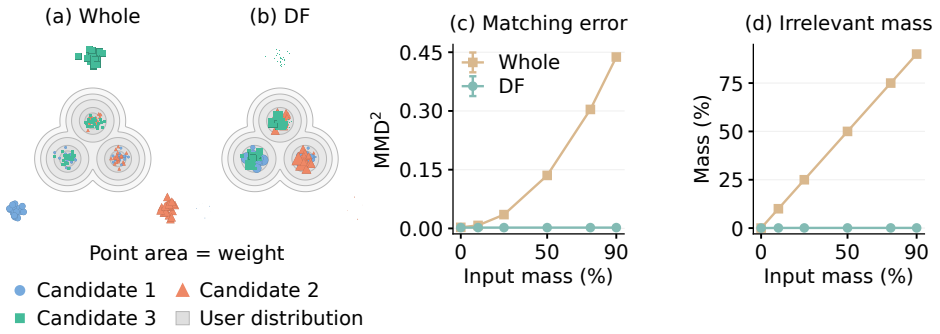


Figure 3: **Distribution filling recovers useful coverage.** Left: whole-specification (Whole) and DF weights at 75% extra mass. Colors and marker shapes identify candidates, point areas show weights, and gray shading shows the user distribution. Right: independent-sample MMD^2 and remaining irrelevant mass (mean $\pm$ SD, three seeds). Filling transfers mass toward complementary user regions.

Figure 3 shows that mixing complete specifications cannot reduce the extra-mass fraction shared by all candidates. DF breaks this coupling: the mean remaining irrelevant mass is only 0.073–0.089% across the 10–90% settings. At 90%, independent-sample MMD^2 drops from 0.438 to 0.0020. DF improves the match in all 15 nonzero-extra-mass configurations, recovering useful coverage despite the irrelevant mass in the candidate specifications.

5.2 Experimental Setup

Scenarios and models. The main benchmark uses 17 tabular datasets from OpenML and TALENT (Vanschoren et al., 2014; Liu et al., 2025b) and 13 image datasets. Under *subsumption*, each candidate covers part of the user class set and additional classes; under *instance recurrence*, candidates cover smaller class subsets that jointly cover the user set. No candidate covers the entire user set. User RKMEs use unlabeled task inputs; known class identities align outputs, and instance labels are used only for evaluation. Tabular and image learnwares use LightGBM (Ke et al., 2017) and ResNet-18 (He et al., 2016), respectively. Following prior learnware practice (Liu et al., 2025a), image specifications use features from an ImageNet-pretrained ViT-B/16 (Dosovitskiy et al., 2021; Russakovsky et al., 2015). Defaults are $K = 5$ candidates, $m = 100$ points per specification, and $s = 10^{-3}$, with λ set by Equation 16. Appendix E details task construction, models, and settings.

Comparisons and evaluation. *Random* chooses one eligible learnware without matching; *From-scratch* trains a new model using labeled user data and the same training algorithm used for the learnwares; *Averaging* uses equal-weight voting; *MoreBoost* combines models using input-wise reusability indicators (Ding & Zhou, 2020); *RKME-Select* uses the highest-ranked RKME candidate; *JobSelector* learns a selector from mimicked data (Wu et al., 2023). DF–Select selects one model per input; DF–MoE combines probabilities. Per-dataset results report means and standard deviations over three repetitions of task-averaged evaluation. Summary results weight datasets equally; accuracy gains are reported in percentage points.

Controlled analyses. Independent pools on CJS, Fourier, CIFAR-10, and EuroSAT cover both scenarios, using LightGBM and ResNet-18 models. Comparisons share tasks, candidates, models, and probability interfaces. Ablation and sensitivity analyses use ten tasks per seed over three seeds. Appendix G specifies these independent controls.

5.3 Performance across Two Reuse Scenarios

We test whether DF improves accuracy over existing reuse strategies across the benchmark tasks. Table 1 summarizes mean accuracy and average ranks in all four settings. At least one of DF–Select and DF–MoE attains the best mean, including ties, in 54 of 60 dataset–scenario pairs: 15/17 and 16/17 tabular cases, and 10/13 and 13/13 image cases, under subsumption and instance recurrence, respectively. Appendix F gives per-dataset results and gain plots.

DF–MoE improves over JobSelector by 6.00/4.46 points on tabular subsumption/instance recurrence and 8.02/12.63 on images. DF–MoE also improves over MoreBoost in all four settings,

Table 1: **Results across 60 dataset–scenario pairs.** Mean accuracy (%) and ranks are averaged equally over datasets; ties receive average ranks (1 is best). Bold marks the best entry per row. Full three-repetition means and standard deviations appear in Appendix F.

Metric	Random	From-scratch	Averaging	MoreBoost	RKME-Select	JobSelector	DF–Select	DF–MoE
Tabular / Subsumption (17 datasets)								
Mean accuracy ↑	60.48	33.56	81.01	67.12	70.12	83.27	86.54	**89.27**
Average rank ↓	6.76	8.00	3.12	5.47	5.24	3.35	2.53	**1.53**
Tabular / Instance recurrence (17 datasets)								
Mean accuracy ↑	34.93	65.51	61.49	58.35	49.36	77.62	77.81	**82.08**
Average rank ↓	7.65	5.41	4.41	5.41	6.12	2.88	2.82	**1.29**
Image / Subsumption (13 datasets)								
Mean accuracy ↑	67.88	51.13	82.91	77.36	68.50	81.81	87.08	**89.83**
Average rank ↓	6.85	7.54	2.69	4.23	6.31	4.31	2.58	**1.50**
Image / Instance recurrence (13 datasets)								
Mean accuracy ↑	32.76	33.47	57.40	54.38	45.30	62.80	74.05	**75.43**
Average rank ↓	7.46	6.92	3.77	4.69	6.08	3.92	1.77	**1.38**

extending the gains to boosting-based reuse. Both DF variants achieve higher mean accuracy than JobSelector on every image dataset in both scenarios; aggregation yields larger dataset-averaged gains, while DF–Select invokes one model per input.

5.4 Heterogeneous Reuse on a Real-World Benchmark

We evaluate heterogeneous reuse on Predict Future Sales (Kaggle, 2018) using the sales-forecasting and heterogeneous processing workflow of Beimingwu (Tan et al., 2024b). A shared pool contains 240 LightGBM learnwares with complementary category coverage. Five source feature views and a distinct user view use different attribute subsets, connected through specification mapping and feature alignment. All methods share retrieved candidates, user specifications, and feature-aligned predictions; Appendix H details the setup.

Table 17 in Appendix H reports the results for RKME-Select, JobSelector, DF–Select, and DF–MoE. DF–MoE reduces overall RMSE by 14.66% relative to JobSelector and 11.13% relative to RKME-Select, attaining the best mean on 30 of 40 stores among these four methods and an average rank of 1.25. DF–Select also reduces overall RMSE relative to JobSelector.

5.5 Understanding Distribution Filling

Ablation studies. We examine how global relevance, pointwise filling, and local compatibility contribute to reuse. The main benchmark compares DF–MoE with Uniform-local, which normalizes $q_i(x)$ instead of $\pi_i q_i(x)$, keeping candidates and probability fusion fixed. Independent controls use a 3×2 design at $K = 5$, crossing equal, whole-RKME, and DF global weights with or without local compatibility.

On the main benchmark, adding global relevance improves mean accuracy by 0.71/0.48 points on tabular subsumption/instance recurrence and 0.14/0.25 on images, with 49 improvements and two ties across 60 dataset–scenario pairs at the reported precision. Figure 4 summarizes the independent controls: local compatibility improves DF-global accuracy by 5.38–30.14 points on Fourier and images. At fixed local compatibility, pointwise filling improves over whole-RKME weighting by 18.40/7.36 points on CJS, 1.68 on Fourier subsumption, and 1.95/0.84 on EuroSAT; each gain holds in all three seeds. Paired values denote subsumption/instance recurrence. Together, these results support task-level weighting of locally compatible models and pointwise allocation that separates useful and irrelevant regions. Appendix G.2 gives the complete ablations and analyses.

Parameter stability. Separate scans vary the regularization scale s and the number m of points per candidate specification, keeping the user specification at 100 points and fixing other defaults. On Fourier and images, DF–MoE accuracy varies by less than 0.08 points over $s = 10^{-5}$–10^{-1}; using

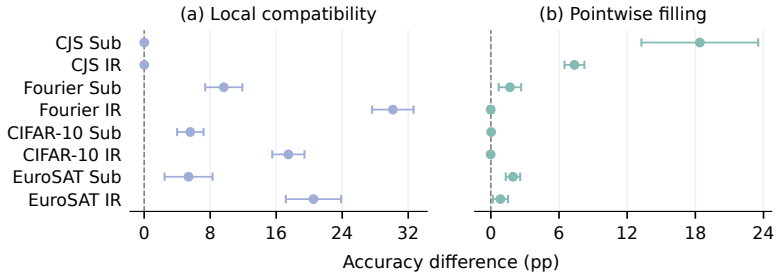


Figure 4: **Separating local compatibility and pointwise filling.** Left: adding local compatibility to DF global weights. Right: replacing whole-RKME with DF weights at fixed local compatibility. Points and bars show paired accuracy gains (percentage points), with mean and SD over three seeds.

50–100 prototypes recovers much of the accuracy lost with 25 in image instance recurrence. These trends support robustness to regularization and a moderate specification budget (Appendix G.4).

5.6 Efficiency and Candidate-Budget Trade-offs

Figure 5 shows the accuracy–time trade-off on images over feasible budgets $K \in \{1, 2, 3, 5, 10, 20\}$. Timing covers reuse with models and inputs loaded, on fixed hardware across methods; Appendix G.5 gives the protocol, all eight settings, and stage costs.

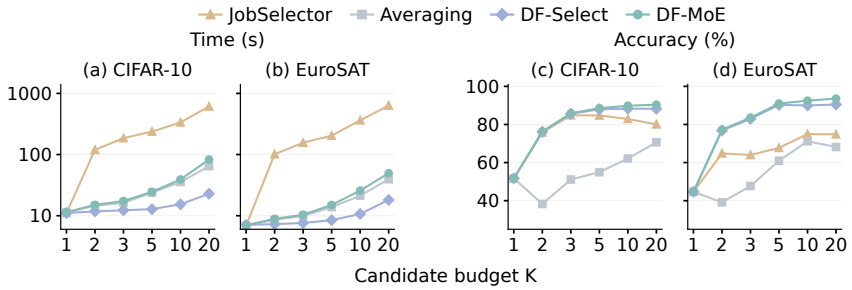


Figure 5: **Accuracy and reuse cost under increasing candidate budgets.** The plots show CIFAR-10 and EuroSAT under instance recurrence; all three tasks are eligible at every budget. Accuracy is averaged over tasks; total time (log scale) is the median of 15 timed measurements.

At $K = 5$, DF–MoE has higher mean accuracy and lower median total time than JobSelector in all eight settings, with 9.65–27.72-fold speedups on images. Avoiding mimicked-data generation and selector training saves preparation time. DF–Select uses about 52–56% of the image reuse time required by DF–MoE, while DF–MoE improves accuracy over DF–Select by 0.62–2.25 percentage points. Increasing K from 5 to 20 improves DF–MoE accuracy on both image datasets under instance recurrence, while DF–Select offers lower-cost prediction.

6 Conclusion

This paper studies how to identify and combine complementary capabilities within partially relevant learnwares. We characterize usefulness at individual user inputs and the additional coverage a learnware provides to an existing set. This utility view motivates distribution filling, which jointly adjusts contributions within candidate specifications to match user needs. The fitted task-level contributions combine with local compatibility from the original specifications to guide input-wise model selection or prediction aggregation. Experiments across both reuse scenarios and a shared system with heterogeneous feature spaces support the effectiveness of this approach, with controlled analyses clarifying its coverage and accuracy–efficiency trade-offs.

AI USE STATEMENT

We used generative AI tools to assist with editing and polishing the manuscript and with writing and debugging experimental code. Writing assistance focused on grammar, sentence structure, word choice, clarity, and terminology consistency. Code assistance supported the implementation of experimental procedures and evaluation scripts specified by the authors. The research ideas, core methodological design, theoretical claims, and interpretation of experimental results were developed by the authors. All AI-assisted text was reviewed and revised by the authors to ensure accuracy and consistency with the intended scientific meaning. AI-assisted code was reviewed and tested for correctness before use, and the authors checked the reported results against experimental records. The authors take responsibility for the final content of this work, including its text, code, analyses, and conclusions.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our research studies complementary learnware reuse through experiments on publicly available benchmark datasets. The reuse procedure operates on model specifications without requiring access to raw developer training data or user instance labels. This access restriction does not constitute a general guarantee of privacy or eliminate biases inherited from the reused models. Data permissions, model reliability, and potential impacts in the intended application should therefore be assessed before deployment. We follow established standards of research integrity in conducting the experiments and reporting the results.

REPRODUCIBILITY STATEMENT

The paper and its appendices document the method, theoretical analysis, and experimental procedures. Appendix B describes RKME construction. Appendices C.1 and D provide the utility analysis and the proof of the joint-residual characterization, while Appendices C.2 and C.3 give the optimization details and reuse algorithm. Appendix E specifies the datasets, task construction, model settings, and evaluation protocol. Complete benchmark results, additional controlled analyses, and the heterogeneous reuse experiment are reported in Appendices F, G, and H, respectively. We will publicly release the code used in our experiments to facilitate reproduction of the reported results.

REFERENCES

Sk Miraj Ahmed, Dripta S. Raychaudhuri, Sujoy Paul, Samet Oymak, and Amit K. Roy-Chowdhury. Unsupervised multi-source domain adaptation without access to source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10103–10112, 2021.

Syed Mumtaz Ali and Samuel David Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28 (1):131–142, 1966.

Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1355–1362, 2012.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101: Mining discriminative components with random forests. In *Proceedings of the 13th European Conference on Computer Vision*, pages 446–461, 2014.

Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the 15th European Conference on Computer Vision*, pages 139–155, 2018.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning*, page 18, 2004.

Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 109–116, 2010.

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*, pages 3606–3613, 2014.

Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

Yao-Xiang Ding and Zhi-Hua Zhou. Boosting-based reliable model reuse. In *Proceedings of the 12th Asian Conference on Machine Learning*, pages 145–160, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *The 9th International Conference on Learning Representations*, 2021.

Paul J. Goulart and Yuwen Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *Mathematical Programming Computation*, 2026. Published online.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

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*, pages 12122–12131, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems 19*, pages 601–608, 2006.

Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Kaggle. Predict Future Sales, 2018. URL `https://www.kaggle.com/c/competitive-data-science-predict-future-sales`.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pages 3146–3154, 2017.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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*, pages 36471–36513, 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*, 2026.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6028–6039, 2020.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 413–422, 2008.

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*, pages 13909–13917, 2024.

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*, pages 5734–5742, 2025a.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. TALENT: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226):1–16, 2025b.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *The 5th International Conference on Learning Representations*, 2017.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems 21*, pages 1041–1048, 2008.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the 6th Indian Conference on Computer Vision, Graphics and Image Processing*, pages 722–729, 2008.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, 2012.

Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

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*, 2026.

Alexander J. Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*, pages 13–31, 2007.

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 and Zhi-Hua Zhou. Decision tree-based assembly for learnwares from heterogeneous feature spaces. *Journal of Computer Research and Development*, 63(5):1249–1260, 2026. In Chinese.

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*, pages 4235–4243, 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*, pages 12767–12795, 2024a.

Peng Tan, Zhi-Hao Tan, Hai-Tian Liu, and Zhi-Hua Zhou. Handling learnwares from heterogeneous feature and label spaces with explicit label exploitation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. In press.

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*, pages 5773–5782, 2024b.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

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.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.

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*, pages 2752–2759, 2023.

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*, pages 10964–10972, 2021.

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

Zhi-Hua Zhou. Learnware and AI model management system. arXiv:2609.11656, 2026.

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

A Related Work

The learnware paradigm. The learnware paradigm supports AI model management by organizing trained models for identification and reuse on new tasks (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026). Each learnware pairs a trained model with a specification describing its capabilities. A learnware dock system organizes these models and matches their specifications against user requirements without accessing the raw training data of developers. Beimingwu implements this workflow, including specification generation, model organization, identification, and reuse (Tan et al., 2024b).

Specifications provide the statistical basis for identifying useful learnwares. RKME represents input distributions with weighted representative points, enabling distribution matching between learnwares and user tasks (Wu et al., 2023). Recently, PAVE, a parameter-vector specification for learnware, represents model capabilities and task requirements through parameter changes in a shared pretrained model (Shi et al., 2026). Privacy analyses examine resistance to original-sample disclosure and inference attacks under stated assumptions (Lei et al., 2024; 2026). Building on specifications, research on the learnware workflow extends identification to heterogeneous feature spaces (Tan et al., 2023; 2024a) and label spaces (Guo et al., 2023). For heterogeneous label spaces, Liu et al. (2025a) use class-wise specifications and class matching to identify helpful combinations, with labeled user examples supporting specification construction and reuse. Tan et al. (2026) further study learnwares with heterogeneous feature and label spaces through explicit label exploitation. As the learnware pool grows, anchor-based search improves identification efficiency (Xie et al., 2023), while evolvable specifications improve model characterization and identification (Liu et al., 2024). Our method uses input-distribution RKMEs and focuses on how their representative points jointly contribute to reuse.

Beyond identifying individual learnwares, solving a user task may require combining the complementary capabilities of several models. MoreBoost combines models using input-wise reusability indicators; inaccurate indicators can impair reuse, motivating an active-rectification extension that queries model providers (Ding & Zhou, 2020). JobSelector is a representative specification-based reuse method (Wu et al., 2023). It fits a mixture of complete RKMEs to the user distribution, generates mimicked inputs through kernel herding (Chen et al., 2010), and trains an input-wise selector with source-model identities as targets. This procedure requires neither raw developer samples nor user instance labels, but a single mixture coefficient scales all regions within each candidate specification. Tree-based assembly uses labeled user examples to organize high-confidence regions of heterogeneous learnwares into a routing tree (Tan & Zhou, 2026). These methods provide different ways to assess and combine model capabilities. With task correspondences given, distribution filling instead jointly reweights points within input-distribution specifications, allowing useful and irrelevant regions to contribute differently, and combines the resulting task-level relevance with local compatibility for input-wise reuse.

Transfer learning and domain adaptation. Transfer learning uses knowledge from source tasks to improve learning on a target task (Pan & Yang, 2010). Distribution matching provides one approach to handling differences between source and target domains. Kernel mean matching reweights source samples to match unlabeled target inputs in RKHS, supporting target-oriented predictor training under covariate shift (Huang et al., 2006). Distribution-weighted combination uses global coefficients and source densities at each input to combine source hypotheses, with guarantees under source-mixture assumptions (Mansour et al., 2008). This global–local weighting structure is closely related to our reuse rule. Distribution filling estimates global relevance by matching points within compact specifications, while computing local compatibility from the original specifications.

Partial domain adaptation addresses irrelevant source classes when the target label space is contained in the source label space (Cao et al., 2018). It shares the concern that transferring irrelevant source information can harm target performance. In our subsumption setting, useful regions from several candidates jointly cover the target task, while an individual candidate need not cover the entire target. Source-free adaptation also removes access to source samples: SHOT adapts a feature extractor while fixing the source classifier (Liang et al., 2020), and DECISION adapts and weights multiple source models using unlabeled target data (Ahmed et al., 2021). Our reuse procedure keeps candidate predictors fixed and uses their specifications to determine input-dependent contributions, without training a target predictor or an additional selector.

Model combination and input-dependent routing. Mixtures of experts coordinate specialized predictors through learned input-dependent gates (Jacobs et al., 1991). Ensemble selection constructs combinations from a model library using performance on labeled validation data (Caruana et al., 2004). These methods establish two relevant principles: different models can serve different inputs, and combining models can improve predictive performance. Our setting concerns independently supplied learnwares and unlabeled user inputs. DF–MoE combines predictions with input-dependent weights computed from specifications, rather than learning a gate from task labels or selecting an ensemble on labeled validation data.

B RKME Specifications and Construction

Section 2 introduces KME, its empirical estimate, and the RKME representation. Here we detail RKME construction.

Although an empirical KME gives a nonparametric distribution representation, it stores one kernel function for every sample. RKME replaces it with a much smaller collection of representative points and coefficients. Following the reduced-set notation in prior work (Wu et al., 2023), we denote the representative points of learnware L_i by $Z_i = \{z_{ij}\}_{j=1}^{m_i}$ and their coefficients by $\boldsymbol{\beta}_i = (\beta_{i1}, \ldots, \beta_{im_i})^\top$. Its RKME specification is

$$\sigma_i = \sum_{j=1}^{m_i} \beta_{ij} k(z_{ij}, \cdot), \qquad z_{ij} \in \mathcal{X}, \tag{11}$$

where m_i is the reduced-set size. Given the training inputs $\{x_{in}\}_{n=1}^{N_i}$ of developer i, with sample count N_i, we construct a nonnegative, normalized RKME by approximating their empirical KME:

$$\min_{\substack{\boldsymbol{\beta}_i \geq 0, \\ Z_i}} \left\| \frac{1}{N_i} \sum_{n=1}^{N_i} k(x_{in}, \cdot) - \sum_{j=1}^{m_i} \beta_{ij} k(z_{ij}, \cdot) \right\|_{\mathcal{H}_k}^2 . \tag{12}$$

Zero-coefficient terms do not contribute to σ_i and are omitted from its representation. We index the remaining points by $j = 1, \ldots, m_i$, so their coefficients are positive and still sum to one; no further normalization is needed during reuse. After any required alignment, user inputs and RKME points are compared in $\mathcal{X}$. For the reduced-set constructions analyzed in prior work, the RKHS-norm approximation error to the empirical KME decreases as $O(m_i^{-1/2})$ under the corresponding assumptions (Bach et al., 2012; Zhang et al., 2021).

For a user task, the user constructs an RKME σ_u locally from an unlabeled sample $\mathcal{U}_u = \{x_n^u\}_{n=1}^{N_u}$ drawn from the input marginal $\mathcal{D}_u^X$. After any required alignment, the user and learnware specifications are compared using the same kernel, without sharing the underlying samples.

C Utility Analysis and Optimization Details

This appendix explains the utility formulation, proves the marginal-utility result, and details the distribution-filling objective, optimization, and computational costs.

C.1 Utility interpretation and proof

The input-level utility $U(x, L_i)$ accounts for predictive usefulness under the task and deployment requirements of the user. For example, models receiving the same sensor inputs may predict failure occurrence or remaining operating life; a suitable predictor may also fail a model-size or inference-cost requirement. With a budget of B learnwares, the ideal identification objective is

$$\mathcal{S}^\star \in \underset{\mathcal{S} \subseteq \mathcal{L},\, |\mathcal{S}| \leq B}{\arg\max}\; F_u(\mathcal{S}). \tag{13}$$

For a finite, nonempty set, $F_u(\mathcal{S})$ is the best expected utility attainable by choosing one member for each input. Any randomized choice has utility no greater than the pointwise maximum, and choosing

a maximizer attains it. This interpretation concerns choosing one model for each input, rather than combining model outputs. Equation 13 therefore provides a common ideal objective for identifying a single learnware when $B = 1$ or a complementary set when $B > 1$, under this per-input selection interpretation. Summing utilities would instead reward repeated coverage of the same inputs.

Proposition 2 (Complementary local utility) *For fixed measurable utilities $U(\cdot, L_i) \in [0, 1]$, F_u is normalized, monotone, and submodular, and its marginal gain is given by Equation 3.*

Proof. Adding L_i replaces $v_{\mathcal{S}}(x)$ by $\max\{v_{\mathcal{S}}(x), U(x, L_i)\}$, giving the stated difference pointwise and hence in expectation. This difference is nonnegative. Moreover, if $\mathcal{S} \subseteq \mathcal{T}$, then $v_{\mathcal{S}}(x) \leq v_{\mathcal{T}}(x)$, so the marginal gain from adding any $L_i \notin \mathcal{T}$ cannot increase. Together with $F_u(\emptyset) = 0$, these facts establish the three properties. □

The proposition applies to fixed measurable utility functions, independent of which other learnwares are selected. Distribution filling instead provides candidate-dependent statistical evidence within the retrieved set. It therefore uses the utility view to motivate complementary reuse; the proposition is not a guarantee that distribution filling solves the ideal selection objective in Equation 13.

C.2 Distribution-filling objective and matrix form

The reference mass $\rho_{ij} = \beta_{ij}/K$ assigns equal total mass to each retrieved learnware while preserving its original relative point weights. All reference masses are positive under the RKME representation convention. They sum to one, as do feasible optimized weights. Consequently,

$$\sum_{(i,j)\in\mathcal{A}} \frac{w_{ij}^2}{\rho_{ij}} = 1 + \sum_{(i,j)\in\mathcal{A}} \frac{(w_{ij} - \rho_{ij})^2}{\rho_{ij}}. \tag{14}$$

The latter sum is the Pearson χ^2 divergence from the reference (Ali & Silvey, 1966). It penalizes assigning a large weight to a point with very little reference mass, while allowing the data-matching term to change the allocation when needed. The parameter λ controls this trade-off. This regularizer does not impose a predefined cutoff on irrelevant points or require the optimized weights to preserve the original proportions within each learnware.

For computation, enumerate the $n = \sum_i m_i$ points in $\mathcal{A}$ by $a \leftrightarrow (i, j)$, $a = 1, \ldots, n$. Write $z_a = z_{ij}$, and stack w_{ij} and ρ_{ij} as vectors $\mathbf{w}, \boldsymbol{\rho} \in \mathbb{R}^n$. Let the Gram matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ have entries $\mathbf{G}_{ab} = k(z_a, z_b)$, set $D_\rho = \mathrm{diag}(\boldsymbol{\rho})$, and define $\mathbf{c} \in \mathbb{R}^n$ by $c_a = \langle \sigma_u, k(z_a, \cdot)\rangle_{\mathcal{H}_k}$. With $\mathbf{1}$ the all-ones vector, expanding the squared RKHS distance in Equation 6 and dropping terms independent of $\mathbf{w}$ gives

$$\min_{\mathbf{w}\geq 0,\, \mathbf{1}^\top\mathbf{w}=1} \frac{1}{2}\mathbf{w}^\top(\mathbf{G} + \lambda D_\rho^{-1})\mathbf{w} - \mathbf{c}^\top\mathbf{w}. \tag{15}$$

All entries of $\mathbf{G}$ and $\mathbf{c}$ are computed from the candidate and user specifications. Since the reference masses are positive, D_ρ^{-1} is well defined and positive definite. The Gram matrix $\mathbf{G}$ is positive semidefinite, so $\lambda > 0$ makes the objective strictly convex and gives a unique solution on the simplex. This optimization is performed once for the user task; the resulting weights are then used for all its test inputs.

Regularization scale. To express regularization relative to the kernel matrix, the experiments use a dimensionless scale s. With $\beta_a = \beta_{ij}$ under the enumeration above, the implementation coefficient and its equivalent reference-mass coefficient are

$$\lambda_{\mathrm{impl}} = s\frac{\mathrm{tr}(\mathbf{G})}{\sum_a \beta_a^{-1}}, \qquad \lambda = \lambda_{\mathrm{impl}}/K. \tag{16}$$

Here $\mathrm{tr}(\mathbf{G})$ is the sum of the diagonal entries of $\mathbf{G}$. This choice gives $\mathrm{tr}(\lambda D_\rho^{-1}) = s\,\mathrm{tr}(\mathbf{G})$, so s controls the total diagonal contribution of the regularizer relative to that of the kernel matrix. The normalization accounts for the scale of $\mathbf{G}$ and the RKME coefficients when setting the regularization strength. Since $\rho_a = \beta_a/K$, the penalty $\frac{1}{2}\lambda_{\mathrm{impl}}\sum_a w_a^2/\beta_a$ equals the penalty in Equation 15. Thus, λ_{impl} and λ express the same regularization in different notation and yield the same optimized weights. The quadratic program is solved with Clarabel (Goulart & Chen, 2026).

C.3 REUSE ALGORITHM

Algorithm 1 summarizes the reuse procedure. Filling is performed once per user task, after retrieval; the resulting global weights are shared across its prediction inputs.

Algorithm 1 Distribution filling and instance-wise reuse

Input: Pool $\mathcal{L}$, user RKME σ_u, task requirements $\mathcal{T}_u$, budget B, kernel k, $\lambda > 0$, reuse rule (DF–Select or DF–MoE).

Output: Predictor $\widehat{f}_u$, or abstention; classification with DF–MoE requires aligned, comparable probabilities.

$\mathcal{C} \leftarrow \text{RKMEIDENTIFY}(\mathcal{L}, \sigma_u, \mathcal{T}_u, B)$
if $\mathcal{C} = \emptyset$ **then return** abstention
Reindex $\mathcal{C} = \{L_i\}_{i=1}^K$; set $\rho_{ij} \leftarrow \beta_{ij}/K$.
Solve Equation 6 once for $\{w_{ij}\}$; set $\pi_i \leftarrow \sum_j w_{ij}$.
function PREDICT(x)
    $q_i \leftarrow \sum_j \beta_{ij} k(x, z_{ij})$ for all i; $d_x \leftarrow \sum_i \pi_i q_i$.
    $r_i \leftarrow \pi_i q_i / d_x$ if $d_x > 0$, and $r_i \leftarrow \pi_i$ otherwise, for all i.
    if DF–Select **then return** $f_{i^\star}(x)$, where $i^\star \in \arg\max_i r_i$.
    return AGGREGATE($\{f_i\}_{i=1}^K, x, \{r_i\}_{i=1}^K, \mathcal{T}_u$).
end function
return PREDICT as $\widehat{f}_u$.

AGGREGATE combines model outputs according to the task requirements $\mathcal{T}_u$. For classification, it returns $\arg\max_y \sum_i r_i \widehat{p}_i(y \mid x)$, requiring the aligned, comparable probabilities stated above. For regression, it returns $\sum_i r_i f_i(x)$, with numerical predictions aligned to the user target.

This reuse stage permits subsequent user-side adaptation with locally held labels; such optional post-training lies outside the procedure and evaluation studied here.

C.4 COMPUTATIONAL ACCOUNTING

Let $n = \sum_{i=1}^K m_i$ be the number of retrieved RKME points. The dense Gram matrix requires $O(n^2)$ storage and kernel evaluations, which motivates limiting the candidate set before filling. Once the task-level quadratic program is solved, each test input requires n kernel evaluations and $O(n)$ arithmetic to form the local scores, followed by $O(K)$ arithmetic for model selection or weighting, in addition to the forward predictions of the selected models. These counts exclude retrieval, the quadratic-program solver, and the cost of an individual kernel evaluation. DF–Select invokes one model per input, whereas DF–MoE combines the available aligned candidate outputs.

D ANALYSIS OF JOINT DISTRIBUTION FILLING

Fix the retrieved candidates, representative points, reference masses, and $\lambda > 0$. Enumerate the points by $a \leftrightarrow (i, j)$, and write $\phi_a = k(z_a, \cdot)$, $\rho_a = \rho_{ij}$, and $\sigma_w = \sum_a w_a \phi_a$. Define

$$J(w) = \frac{1}{2}\|\sigma_u - \sigma_w\|^2_{\mathcal{H}_k} + \frac{\lambda}{2}\sum_a \frac{w_a^2}{\rho_a}, \qquad h_w(z) = \langle \sigma_u - \sigma_w, k(z, \cdot)\rangle_{\mathcal{H}_k}.$$

The feasible set is $\{w \geq 0 : \sum_a w_a = 1\}$.

Proof of Proposition 1. The objective is strictly convex because $\lambda > 0$ and every $\rho_a > 0$. The reference vector ρ is feasible and strictly positive, so the KKT conditions are necessary and sufficient. Let τ be the multiplier for unit total mass and $\nu_a \geq 0$ the multipliers for nonnegativity. The Lagrangian is

$$\mathcal{J}(w, \tau, \nu) = J(w) + \tau\left(\sum_a w_a - 1\right) - \sum_a \nu_a w_a.$$

Stationarity and complementary slackness give

$$-h_{w^\star}(z_a) + \lambda\frac{w_a^\star}{\rho_a} + \tau - \nu_a = 0, \qquad \nu_a w_a^\star = 0.$$

If $w_a^\star > 0$, then $\nu_a = 0$ and $w_a^\star = \rho_a[h_{w^\star}(z_a) - \tau]/\lambda$. If $w_a^\star = 0$, then $h_{w^\star}(z_a) \le \tau$. Combining both cases yields

$$w_a^\star = \frac{\rho_a}{\lambda}\big[h_{w^\star}(z_a) - \tau\big]_+,$$

which is Equation 7 after restoring the point indices and the definition of $h_{w^\star}$. The multiplier τ enforces unit total mass. □

Here, the subscript w denotes the current joint weight vector. Section 4.2 gives the kernel-response interpretation. The joint candidate kernel response also includes the contribution of point z_a itself. The residual score depends on the entire joint allocation and reflects distributional coverage rather than directly measuring model accuracy or predictive utility. Because the residual depends on the optimized weights, the proposition characterizes the solution implicitly and does not replace the quadratic-program solver.

The role of this residual response in the objective follows from

$$\frac{\partial J(w)}{\partial w_a} = -h_w(z_a) + \lambda\frac{w_a}{\rho_a}.$$

The first term gives the marginal effect on matching error, and the second gives the marginal concentration cost. Under the unit-mass constraint, increasing one weight requires decreasing others, so a transfer compares these total marginal costs at the two points.

Corollary 3 (Overall relevance from regional allocations) *The global relevance of candidate i satisfies*

$$\pi_i = \sum_j w_{ij}^\star = \frac{1}{\lambda}\sum_j \rho_{ij}\big[h_{w^\star}(z_{ij}) - \tau\big]_+. \tag{17}$$

Moreover,

$$\pi_i = 0 \quad \Longleftrightarrow \quad \max_{1\le j\le m_i} h_{w^\star}(z_{ij}) \le \tau.$$

Proof. Summing Equation 7 over candidate i gives the first identity. Every summand is nonnegative and every $\rho_{ij} > 0$, so the sum is zero exactly when all residual scores are at most τ. □

Thus, a candidate contributes to the joint match through the regions that receive mass. Its other regions need not contribute in the same proportion. This is a statement about fitted distributional coverage; zero fitted mass does not establish that a model is inaccurate there.

E Experimental Setup

This appendix specifies the datasets and task construction, models and comparison methods, and evaluation protocol for the main benchmark. The independent controls used in the additional analyses are described in Appendix G.

E.1 Datasets and Task Construction

Datasets. The benchmark comprises 17 tabular datasets from OpenML (Vanschoren et al., 2014) and TALENT (Liu et al., 2025b) and 13 image datasets. The tabular datasets are CJS, First-Order, Product, Volkert, Indian-Pines, Dry-Bean, Factors, Fourier, Karhunen, Pixel, Zernike, Mice-Protein, Optdigits, Satimage, Segment, Semeion, and Wine-Quality.

The image datasets are CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), Oxford Flowers (Nilsback & Zisserman, 2008), Food-101 (Bossard et al., 2014), Oxford-IIIT Pets (Parkhi et al., 2012), SVHN (Netzer et al., 2011), Fashion-MNIST (Xiao et al., 2017), GTSRB (Stallkamp et al., 2012), MNIST (LeCun et al., 1998), STL-10 (Coates et al., 2011), and USPS (Hull, 1994). The tables abbreviate Food-101 as Food and Oxford-IIIT Pets as Pet. For each dataset, candidate models and user tasks share the class identities of the dataset. Tabular specifications are compared in the same feature coordinates; image specifications use the common representation described in Appendix E.2.

Task construction. A user task is specified by a subset of the classes in the dataset and the corresponding batch of evaluation inputs. It therefore contains many prediction instances. Candidate learnwares are trained on other class subsets of the same dataset, and each model is paired with its statistical specification. Class identities define the correspondence between the outputs of a candidate and the user label space.

Table 2 summarizes the class-subset construction. Subsumption places classes outside the user task in the source distributions. We exclude any candidate whose class set contains the entire user class set, so solving the task requires assembling several partially relevant models. Instance recurrence uses smaller source class subsets and larger user subsets to test complementary coverage. For image datasets, the protocol samples 25 candidate subsets and ten user tasks per dataset. In class-combination construction, a pool-size quota is an upper bound: the number of available combinations can yield fewer distinct tasks or learnwares.

Table 2: **Class-subset construction.** For tabular datasets, C is the number of classes in the dataset. Image entries give proportions of the classes in the dataset.

Modality	Scenario	Candidate classes	User classes
Tabular	Subsumption	$C-1$ or $C-2$	3
Tabular	Instance recurrence	3	$C-1$ or $C-2$
Image	Subsumption	$[0.7, 0.9]$	$[0.2, 0.5]$
Image	Instance recurrence	$[0.2, 0.5]$	$[0.7, 0.9]$

The user RKME is constructed from the unlabeled inputs of the batch to be predicted, after shared feature extraction for images. Thus, the reuse methods can use the entire batch of unlabeled inputs to characterize the user distribution. The labels of the evaluation inputs are used only to compute accuracy; known class identities are used to define the task and align model outputs.

E.2 Models and Compared Methods

Main benchmark. Tabular learnwares use LightGBM (Ke et al., 2017) with five hyperparameter configurations. The combinations of (learning rate, number of leaves, maximum depth) are $\{(0.015, 224, 66), (0.005, 300, 50), (0.01, 128, 80), (0.15, 224, 80), (0.01, 300, 66)\}$. Image learnwares use ResNet-18 (He et al., 2016) trained for 100 epochs with SGD and cosine learning-rate decay (Loshchilov & Hutter, 2017), with initial learning rates in $\{0.1, 0.01, 0.001\}$. For image specification generation, we follow prior learnware practice (Liu et al., 2025a) and extract features with an ImageNet-pretrained ViT-B/16 (Dosovitskiy et al., 2021; Russakovsky et al., 2015). The same feature extractor is used for candidate and user specifications. The candidate-pool configurations yield approximately 2,000 tabular models and 650 image models across the two scenarios.

Each specification contains 100 RKME prototypes and uses the Gaussian kernel

$$k(x, x') = \exp\big(-0.1\|x - x'\|_2^2\big) .$$

The nonnegative RKME coefficients are normalized within each learnware. Retrieval uses a budget of five candidates, and the regularization scale is $s = 10^{-3}$ in Equation 16. Candidate models, their RKME specifications, and the user RKME are shared by DF–Select and DF–MoE; the two variants differ in whether they select one candidate or aggregate predictive probabilities.

Compared methods. *Random* selects one learnware from the eligible pool without distribution matching. *From-scratch* trains a new model using the same training algorithm as the learnwares, with ten labeled examples per user class. This supervised reference measures what can be learned with a small user-label budget; the reuse methods do not receive these labels. *Averaging* combines the selected candidates by equal-weight majority voting. *MoreBoost* (Ding & Zhou, 2020) uses boosting to combine input-wise reusability indicators and model predictions. Following Ding & Zhou (2020), we construct the reusability indicators using Isolation Forest (Liu et al., 2008) fitted to the training inputs of each model. We run 50 boosting rounds with $\eta = 1$ and without active rectification; image indicators use the shared ViT-B/16 features. *RKME-Select* deploys the highest-ranked candidate from RKME identification. *JobSelector* generates mimicked inputs from candidate specifications and trains a selector whose targets are the source learnware identities (Wu et al., 2023); Appendix A explains the procedure. *DF–Select* selects the candidate with the largest input-specific

reuse weight, whereas *DF–MoE* averages candidate probabilities with those weights. From-scratch trains a task predictor and JobSelector trains a selector; the candidate predictors are not updated during reuse.

For probability fusion, candidate outputs are mapped through the shared dataset class identities to the user classes and renormalized. The original RKME coefficients determine local compatibility $q_i(x)$; optimized point weights affect input-wise reuse through their learnware-level sums π_i. The six ablation variants in Appendix G.2 hold these interfaces fixed and change only the global weights and whether local compatibility is used. Their *Uniform-global* variant averages probabilities, which differs from the majority-vote *Averaging* baseline.

E.3 Evaluation Protocol

For task u, with N_u evaluation instances, accuracy is

$$A_u = \frac{100}{N_u} \sum_{n=1}^{N_u} \mathbf{1}[\widehat{y}_n^u = y_n^u] \,.$$

Here $\widehat{y}_n^u$ and y_n^u are the predicted and true labels of the n-th evaluation input, and $\mathbf{1}[\cdot]$ is the indicator function. We first average accuracies over the user tasks of a dataset within each repetition, giving each task equal weight. The complete per-dataset tables in Appendix F report means and standard deviations across three repetitions. The aggregate ablation table averages dataset-level accuracies within each modality and scenario, giving each dataset equal weight.

Bold entries indicate the highest reported mean in each row, including ties. The method-family counts identify datasets where at least one of DF–Select and DF–MoE attains the best mean. Both variants are presented separately in the analysis figures. A plotted difference of one percentage point is an absolute difference of one in accuracy measured in percent; it is not a relative percentage improvement. Gain plots use the reported means and a zero reference line, with no significance annotations.

F Complete Benchmark Results

This appendix gives the full dataset-level results underlying Table 1. The main-text summary averages the dataset-level means for each method equally within each setting.

We test whether distribution filling improves accuracy over existing reuse strategies in both scenarios, using the task construction and models described in Appendix E. Tables 3–6 report all results. At least one DF variant achieves the best mean, including ties, on 15/17 and 16/17 tabular datasets under subsumption and instance recurrence, and on 10/13 and 13/13 image datasets in the same scenario order. Figure 6 shows the gains of each variant over JobSelector: DF–MoE has the larger dataset-averaged improvement in all four settings, while DF–Select also achieves leading results with one model per input.

Tabular datasets. Under subsumption, DF–MoE and DF–Select improve the average over JobSelector by 6.00 and 3.27 points, respectively. DF–MoE gains 20.39 points on CJS and 17.50 on Segment, demonstrating effective reuse despite candidate classes outside the user task. Under instance recurrence, DF–MoE improves the average by 4.46 points and leads on 12 datasets; DF–Select leads on four, including Fourier and Karhunen. Both prediction rules exploit complementary coverage, with aggregation providing the stronger overall result.

Table 3: **Average accuracy (%) in the tabular subsumption scenario.** Results are means and standard deviations over three repetitions; the highest mean in each row is bold.

Dataset	Random	From-scratch	Averaging	MoreBoost	RKME-Select	JobSelector	DF–Select	DF–MoE
CJS	61.63 ± 4.34	37.20 ± 0.00	87.13 ± 0.73	60.24 ± 0.39	65.49 ± 0.29	67.56 ± 0.72	65.92 ± 0.48	**87.95 ± 1.20**
First-Order	51.50 ± 1.20	29.79 ± 0.00	66.92 ± 0.07	63.82 ± 0.25	61.57 ± 0.00	62.55 ± 0.30	60.33 ± 0.81	**66.94 ± 0.36**
Product	61.83 ± 2.20	33.00 ± 3.77	**81.32 ± 0.85**	75.21 ± 0.50	77.76 ± 1.10	77.25 ± 0.82	75.89 ± 1.14	80.65 ± 1.89
Volkert	59.98 ± 1.25	35.12 ± 0.94	**78.20 ± 0.93**	73.26 ± 0.78	72.56 ± 1.77	73.63 ± 1.50	73.46 ± 1.01	77.37 ± 0.35
Indian-Pines	65.05 ± 3.85	28.43 ± 0.47	96.54 ± 1.18	84.79 ± 3.09	91.96 ± 1.72	80.37 ± 0.86	94.86 ± 0.30	**97.36 ± 0.61**
Dry-Bean	59.14 ± 2.70	23.25 ± 0.69	87.59 ± 0.38	82.81 ± 0.55	79.14 ± 0.57	87.88 ± 0.44	91.85 ± 0.30	**95.09 ± 0.27**
Factors	62.88 ± 2.06	33.33 ± 0.00	81.47 ± 1.54	74.22 ± 2.25	66.02 ± 0.06	96.86 ± 0.13	**97.68 ± 0.21**	97.67 ± 0.13
Fourier	61.78 ± 2.33	33.33 ± 0.00	78.30 ± 0.62	69.22 ± 1.57	65.64 ± 0.17	86.70 ± 0.50	**91.38 ± 0.52**	90.91 ± 0.41
Karhunen	62.68 ± 2.02	33.33 ± 0.00	77.89 ± 2.81	68.15 ± 1.46	65.57 ± 0.10	93.01 ± 0.15	**97.18 ± 0.16**	95.64 ± 0.25
Pixel	62.94 ± 2.10	33.33 ± 0.00	93.65 ± 0.70	47.24 ± 1.12	66.07 ± 0.06	91.28 ± 0.53	97.82 ± 0.03	**97.88 ± 0.04**
Zernike	60.83 ± 2.75	33.33 ± 0.00	69.34 ± 1.02	60.31 ± 0.91	64.92 ± 0.72	83.58 ± 1.10	**89.83 ± 2.08**	84.92 ± 1.71
Mice-Protein	59.83 ± 2.72	35.69 ± 0.15	87.03 ± 2.31	63.66 ± 1.13	67.08 ± 0.44	97.91 ± 0.06	**99.56 ± 0.11**	98.22 ± 0.31
Optdigits	62.32 ± 1.10	33.35 ± 0.07	71.05 ± 1.91	71.86 ± 0.70	66.60 ± 0.03	89.65 ± 1.41	**94.42 ± 0.47**	89.23 ± 1.93
Satimage	60.45 ± 4.07	37.54 ± 0.00	83.39 ± 1.67	77.51 ± 0.04	75.93 ± 0.10	89.38 ± 0.10	92.60 ± 0.06	**93.86 ± 0.32**
Segment	60.22 ± 2.75	33.25 ± 0.00	79.71 ± 0.48	73.32 ± 1.10	65.55 ± 0.06	74.40 ± 0.65	84.96 ± 1.74	**91.90 ± 1.67**
Semeion	61.54 ± 1.57	33.26 ± 0.01	81.27 ± 0.49	34.81 ± 0.16	65.19 ± 0.05	87.75 ± 0.56	94.25 ± 0.41	**94.63 ± 0.29**
Wine-Quality	53.55 ± 8.56	43.91 ± 0.00	76.30 ± 0.02	60.58 ± 0.16	75.04 ± 0.00	75.83 ± 0.17	69.22 ± 0.75	**77.35 ± 0.50**

Table 4: **Average accuracy (%) in the tabular instance-recurrent scenario.** Other reporting details follow Table 3.

Dataset	Random	From-scratch	Averaging	MoreBoost	RKME-Select	JobSelector	DF–Select	DF–MoE
CJS	47.42 ± 1.33	33.00 ± 0.97	71.71 ± 0.84	50.05 ± 0.56	59.13 ± 0.66	61.56 ± 1.43	61.55 ± 1.00	**76.06 ± 1.42**
First-Order	37.40 ± 2.79	31.27 ± 0.89	59.51 ± 0.06	52.74 ± 0.35	57.33 ± 0.29	57.53 ± 0.22	54.51 ± 1.92	**61.23 ± 0.10**
Product	30.29 ± 1.73	40.74 ± 0.68	64.18 ± 2.16	62.96 ± 1.02	56.55 ± 0.97	57.42 ± 2.06	56.18 ± 1.77	**71.87 ± 1.14**
Volkert	26.60 ± 2.45	38.41 ± 0.93	54.76 ± 2.37	57.23 ± 1.02	46.91 ± 1.90	55.79 ± 0.15	48.08 ± 1.17	**59.98 ± 0.62**
Indian-Pines	36.97 ± 2.65	69.35 ± 0.95	87.96 ± 0.11	78.25 ± 2.20	78.74 ± 1.93	80.12 ± 0.85	90.79 ± 0.87	**92.67 ± 0.69**
Dry-Bean	45.85 ± 3.32	88.63 ± 0.51	84.79 ± 1.00	83.36 ± 1.18	66.51 ± 0.41	**93.77 ± 0.04**	91.08 ± 0.94	92.88 ± 0.41
Factors	28.95 ± 0.54	87.57 ± 0.59	48.06 ± 0.72	67.12 ± 0.21	34.62 ± 0.83	91.52 ± 0.20	92.40 ± 0.11	**92.47 ± 0.11**
Fourier	28.56 ± 0.49	71.67 ± 0.13	37.24 ± 1.89	43.49 ± 0.78	35.73 ± 0.14	72.21 ± 1.33	**73.87 ± 1.35**	73.79 ± 1.33
Karhunen	28.73 ± 0.45	80.70 ± 1.43	48.86 ± 1.40	43.34 ± 1.31	35.11 ± 0.45	86.99 ± 1.30	**90.59 ± 1.44**	89.14 ± 1.42
Pixel	28.64 ± 0.52	85.76 ± 0.91	46.31 ± 0.52	53.61 ± 0.17	28.63 ± 1.20	88.54 ± 0.50	95.85 ± 0.03	**95.92 ± 0.03**
Zernike	28.14 ± 0.40	68.66 ± 1.66	48.11 ± 0.74	50.07 ± 0.42	34.33 ± 0.58	74.87 ± 0.88	75.97 ± 0.75	**76.45 ± 0.83**
Mice-Protein	37.28 ± 1.56	74.41 ± 1.16	71.35 ± 2.11	55.32 ± 1.41	45.88 ± 0.46	96.42 ± 0.46	**97.30 ± 0.43**	97.02 ± 0.56
Optdigits	28.75 ± 0.77	84.24 ± 0.03	53.83 ± 1.75	73.41 ± 1.35	36.59 ± 0.16	87.23 ± 1.15	**89.31 ± 1.22**	88.61 ± 1.28
Satimage	46.34 ± 4.71	76.87 ± 1.53	92.31 ± 0.20	85.63 ± 0.64	77.41 ± 1.12	93.25 ± 0.04	92.89 ± 0.11	**93.63 ± 0.08**
Segment	41.77 ± 0.36	81.83 ± 2.45	82.71 ± 1.01	79.67 ± 0.57	56.24 ± 0.14	92.79 ± 0.20	84.32 ± 0.44	**92.82 ± 0.19**
Semeion	29.99 ± 1.24	70.12 ± 1.22	36.21 ± 0.38	23.51 ± 0.05	30.08 ± 0.55	67.66 ± 0.35	72.12 ± 0.47	**72.75 ± 0.34**
Wine-Quality	42.16 ± 1.22	30.51 ± 2.56	57.39 ± 1.81	32.21 ± 3.62	59.35 ± 0.07	61.87 ± 0.47	56.02 ± 2.25	**68.03 ± 0.53**

Image datasets. Both DF variants outperform JobSelector on every image dataset in both scenarios. Under subsumption, DF–Select and DF–MoE gain 5.27 and 8.02 points on average; both reach 93.07% on EuroSAT, 18.96 points above JobSelector. Under instance recurrence, at least one DF variant achieves the best mean on each of the 13 datasets. DF–MoE and DF–Select gain 12.63 and 11.25 points on average, respectively; DF–MoE improves USPS from 50.76% to 89.29%. These broad gains demonstrate effective assembly of complementary visual learnwares.

Table 5: **Average accuracy (%) in the image subsumption scenario.** Other reporting details follow Table 3.

Dataset	Random	From-scratch	Averaging	MoreBoost	RKME-Select	JobSelector	DF–Select	DF–MoE
CIFAR10	64.26 ± 3.46	52.88 ± 3.63	79.13 ± 4.15	67.43 ± 1.92	67.06 ± 1.34	92.04 ± 0.68	93.35 ± 0.07	**94.49 ± 0.02**
CIFAR100	69.28 ± 4.93	16.96 ± 3.07	**86.90 ± 0.25**	72.46 ± 1.86	69.80 ± 2.57	78.58 ± 0.25	83.35 ± 0.27	85.75 ± 0.33
DTD	64.88 ± 2.84	53.80 ± 0.90	74.89 ± 0.24	58.55 ± 2.68	56.34 ± 4.70	68.50 ± 1.08	72.68 ± 0.91	**76.80 ± 0.47**
EuroSAT	64.45 ± 3.49	46.94 ± 2.43	77.99 ± 0.93	78.16 ± 3.16	67.93 ± 1.66	74.11 ± 2.73	**93.07 ± 0.85**	**93.07 ± 0.30**
Flowers	66.51 ± 3.98	57.78 ± 0.63	84.88 ± 0.72	82.61 ± 0.89	71.18 ± 1.81	77.14 ± 0.68	79.79 ± 0.90	**86.34 ± 0.56**
Food	67.78 ± 2.94	18.07 ± 1.07	**83.87 ± 0.18**	83.20 ± 0.21	67.99 ± 0.25	72.47 ± 0.11	77.07 ± 0.07	81.25 ± 1.41
Pet	71.81 ± 6.24	80.83 ± 1.27	89.81 ± 0.20	80.43 ± 1.71	65.54 ± 0.83	88.58 ± 0.67	92.17 ± 1.07	**95.13 ± 0.73**
SVHN	64.39 ± 1.72	34.89 ± 2.58	78.11 ± 3.43	76.84 ± 6.40	68.27 ± 1.90	74.73 ± 1.28	75.98 ± 1.34	**81.36 ± 4.99**
FashionMNIST	65.30 ± 1.73	67.36 ± 1.69	78.11 ± 2.77	74.55 ± 3.54	67.86 ± 1.30	92.40 ± 1.38	93.00 ± 1.06	**93.34 ± 0.71**
GTSRB	83.23 ± 1.82	43.59 ± 3.92	**95.54 ± 0.94**	93.48 ± 0.71	77.86 ± 1.34	82.07 ± 1.69	84.62 ± 0.58	94.30 ± 0.11
MNIST	66.67 ± 1.63	62.57 ± 3.73	83.65 ± 5.97	78.11 ± 2.44	69.99 ± 1.41	95.15 ± 0.37	**97.34 ± 0.06**	96.58 ± 0.39
STL10	66.00 ± 1.81	77.35 ± 4.48	83.97 ± 2.99	80.69 ± 3.06	68.73 ± 1.71	96.11 ± 0.69	97.82 ± 0.17	**98.05 ± 0.14**
USPS	67.93 ± 2.11	51.73 ± 5.62	80.95 ± 3.08	79.23 ± 5.76	71.96 ± 1.36	71.60 ± 1.15	**91.80 ± 3.71**	91.32 ± 4.12

Table 6: **Average accuracy (%) in the image instance-recurrent scenario.** Other reporting details follow Table 3.

Dataset	Random	From-scratch	Averaging	MoreBoost	RKME-Select	JobSelector	DF–Select	DF–MoE
CIFAR10	33.63 ± 2.33	36.20 ± 0.94	62.85 ± 2.17	58.49 ± 1.17	53.61 ± 1.49	81.54 ± 1.42	84.91 ± 1.88	**85.44 ± 2.03**
CIFAR100	35.54 ± 2.04	14.76 ± 0.13	55.67 ± 0.97	43.40 ± 1.40	42.50 ± 0.96	56.72 ± 0.44	67.38 ± 0.37	**67.41 ± 0.62**
DTD	28.61 ± 2.20	33.52 ± 0.87	38.13 ± 1.18	27.04 ± 1.40	31.25 ± 1.43	46.04 ± 1.06	55.37 ± 0.57	**56.69 ± 0.92**
EuroSAT	32.32 ± 1.81	33.83 ± 2.91	62.85 ± 2.03	61.66 ± 3.87	54.79 ± 0.26	58.68 ± 3.13	**88.11 ± 0.35**	87.85 ± 0.75
Flowers	25.09 ± 3.93	17.10 ± 0.61	46.72 ± 0.48	44.61 ± 1.48	31.55 ± 0.48	42.57 ± 0.12	51.48 ± 0.13	**58.54 ± 0.09**
Food	31.88 ± 2.55	14.39 ± 0.44	49.27 ± 1.06	49.91 ± 0.18	37.60 ± 1.97	45.49 ± 0.36	**55.15 ± 0.40**	54.74 ± 1.31
Pet	29.08 ± 4.67	52.91 ± 1.21	58.23 ± 1.59	58.08 ± 1.75	43.99 ± 1.56	78.47 ± 0.78	84.42 ± 1.10	**85.90 ± 0.98**
SVHN	31.22 ± 1.90	15.46 ± 1.04	56.74 ± 2.75	57.63 ± 2.83	45.65 ± 1.71	44.48 ± 1.11	51.37 ± 1.37	**60.90 ± 2.79**
FashionMNIST	33.59 ± 1.84	49.73 ± 1.70	64.66 ± 1.98	60.69 ± 3.57	54.69 ± 1.87	85.11 ± 0.39	**86.56 ± 0.65**	86.44 ± 0.89
GTSRB	42.32 ± 3.86	19.64 ± 1.98	49.60 ± 2.27	47.01 ± 2.50	27.76 ± 2.31	52.02 ± 1.00	**63.32 ± 1.03**	63.28 ± 2.07
MNIST	36.58 ± 1.44	37.49 ± 4.57	69.01 ± 0.91	67.58 ± 2.93	55.48 ± 4.25	84.65 ± 0.53	**93.15 ± 0.84**	91.78 ± 0.98
STL10	33.54 ± 1.87	73.19 ± 1.33	65.69 ± 2.67	64.27 ± 1.54	56.91 ± 1.22	89.91 ± 1.11	92.22 ± 1.00	**92.37 ± 1.03**
USPS	32.46 ± 2.33	36.93 ± 3.37	66.80 ± 3.92	66.50 ± 3.83	53.07 ± 5.24	50.76 ± 1.19	89.25 ± 0.82	**89.29 ± 1.12**

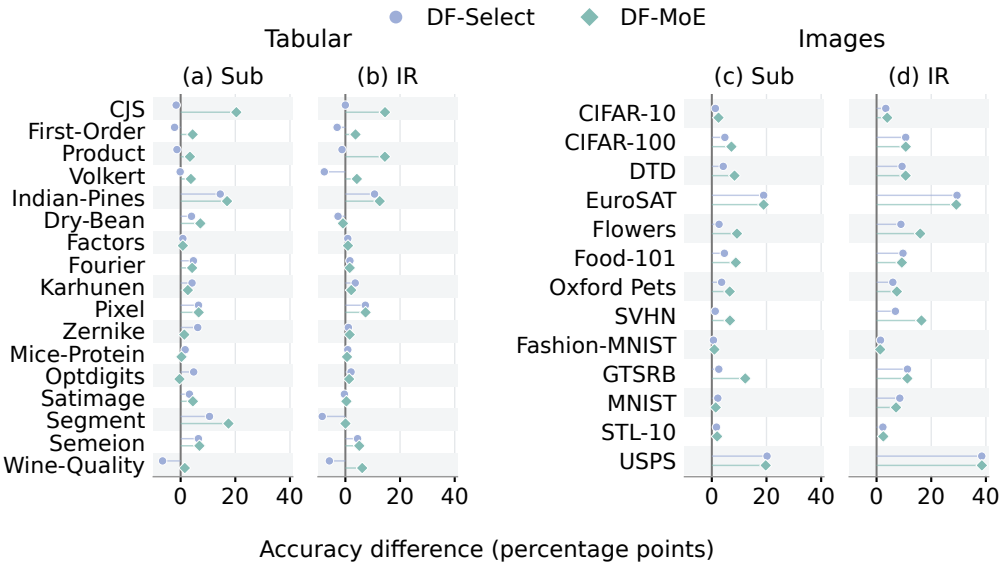


Figure 6: **Accuracy gains over JobSelector.** Differences from the mean accuracies in Tables 3–6, in percentage points. Sub: subsumption; IR: instance recurrence. Paired panels share dataset labels.

G Additional Experimental Analyses

These analyses examine whether pointwise allocation recovers useful coverage in the presence of irrelevant source mass, how global relevance and local compatibility contribute to prediction, how regularization and specification size affect accuracy, and how candidate budgets change accuracy and reuse cost. We also compare specification-guided reuse with tree-based assembly using limited labeled user examples. The global-relevance ablation uses the main benchmark in Appendix E; the remaining real-data analyses use the independent controls specified below. The synthetic demonstration has its own construction in Appendix G.1.

Datasets and tasks. The controlled ablation, sensitivity, and cost experiments use CJS, Fourier, CIFAR-10, and EuroSAT under both scenarios. Candidate training data and evaluation data are disjoint. The evaluation pools contain 562 CJS, 720 Fourier, 10,000 CIFAR-10, and 5,400 EuroSAT inputs. A task consists of all evaluation instances in its selected classes; Table 7 gives the class counts, candidate-pool sizes, and resulting task sizes. The same trained models are used across repetitions.

Models and defaults. The controlled experiments use LightGBM for tabular learnwares and ResNet-18 for image learnwares. Image specifications use features from a shared ImageNet-pretrained ViT-B/16, as described in Appendix E.2. Their model pools are independent of the main benchmark. Unless varied explicitly, all controlled methods use the same candidates, user inputs, specifications, and output mapping, with $K = 5$, $m = m_u = 100$, and $s = 10^{-3}$ in Equation 16; m_u denotes the user-specification size.

Evaluation and reporting. For the controlled ablation and sensitivity experiments, seeds 101, 102, and 103 each select ten class-subset tasks. We average the ten task accuracies within a seed and report the mean and sample standard deviation of the three seed means. Cost experiments use one predetermined task per seed, with one warm-up followed by five timed repetitions for each eligible budget. Accuracy gives each eligible task equal weight; time is the median over all timed repetitions. Each method receives the same tasks and candidates at a given budget. Budgets larger than the eligible pool are omitted. Appendix G.5 specifies the measured stages and hardware. Accuracy gains and time ratios are computed from unrounded statistics before rounding for display; differences of separately rounded table entries can differ by 0.01 percentage points.

Table 7: **Learnware pools and user tasks for the controlled real-data experiments.** Sub: subsumption; IR: instance recurrence. Class counts give the construction ranges; input counts give the observed ranges across 30 tasks (three seeds, ten tasks per seed).

Dataset / scenario	Learnwares in pool	Candidate classes	User classes	Evaluation instances per task
CJS Sub	21	4–5	3	207–355
CJS IR	20	3	4–5	310–493
Fourier Sub	55	8–9	3	216
Fourier IR	120	3	8–9	576–648
CIFAR-10 Sub	20	7–9	2–5	3000–5000
CIFAR-10 IR	20	2–5	7–9	7000–9000
EuroSAT Sub	20	7–9	2–5	1500–2800
EuroSAT IR	20	2–5	7–9	3600–4800

G.1 Distribution-Filling Example

The two-dimensional demonstration isolates the freedom to adjust mass *inside* a learnware. The user distribution is an equal mixture of Gaussian regions centered at $(-2, 0)$, $(2, 0)$, and $(0, 3)$, each with standard deviation 0.35. Numbering these regions in the order listed, the three candidates respectively cover region pairs $(1, 2)$, $(2, 3)$, and $(3, 1)$, with an additional region at $(-6, -4)$, $(6, -4)$, and $(0, 8)$, respectively. Each region contributes 24 fixed prototypes. Within each seed, we fix the prototype locations and vary only the unrelated-region mass $\eta \in \{0, 0.1, 0.25, 0.5, 0.75, 0.9\}$; the remaining mass is divided equally between the two useful regions. Zero-mass prototypes are removed, leaving 144 active prototypes at $\eta = 0$ and 216 otherwise.

We compare the original uniform candidate mixture, a fitted whole-RKME mixture (Whole), and DF, using 600 user samples for fitting and another 1,200 independent samples for evaluation. The kernel is $\exp(-\|x - x'\|_2^2/2)$ and $s = 10^{-3}$. We evaluate the squared RKHS discrepancy using the biased empirical MMD^2, including diagonal self-similarities, and sum the fitted weights assigned to unrelated regions. Region identities are used for generation and this final accounting only.

Table 8: **Controlled suppression of unrelated mass.** Entries are means $\pm$ sample standard deviations across three seeds. MMD^2 values are in units of 10^{-3} and use 1,200 independent user samples. The original uniform mixture and Whole assign the input fraction to unrelated regions by construction; DF mass is computed from the optimized prototype weights.

Input mass (%)	Uniform MMD^2	Whole MMD^2	DF MMD^2	DF mass (%)
0	1.558 ± 0.767	2.708 ± 0.590	2.017 ± 0.748	0.000 ± 0.000
10	6.509 ± 0.389	7.656 ± 0.269	2.017 ± 0.748	0.073 ± 0.048
25	34.342 ± 0.600	35.206 ± 0.429	2.017 ± 0.748	0.080 ± 0.052
50	135.148 ± 2.136	135.355 ± 2.050	2.017 ± 0.747	0.086 ± 0.055
75	303.977 ± 4.595	303.884 ± 4.555	2.017 ± 0.747	0.088 ± 0.056
90	437.926 ± 6.569	437.743 ± 6.516	2.017 ± 0.747	0.089 ± 0.057

Table 8 quantifies the behavior illustrated in the main text. Because every candidate has unrelated mass η, every whole-candidate convex mixture has that same fraction. DF adjusts the individual prototype weights and reduces the average unrelated mass below 0.1% at every positive η. Its independent MMD^2 stays near 0.002017 as the original unrelated mass grows to 90%, where the error is 99.54% lower than Whole. All 15 configurations with positive unrelated mass exhibit a lower independent error under DF. At zero unrelated mass, the original uniform mixture already gives accurate complementary coverage, with $\text{MMD}^2 = 0.001558$. The positive-mass sweep directly demonstrates the additional capability supplied by prototype-level allocation.

G.2 Ablation Studies

Global relevance on the main benchmark. We compare DF–MoE with Uniform-local while holding local compatibility and probability fusion fixed (Table 9).

Table 9: **Ablation of global relevance.** Mean accuracy (%) over datasets. Win-or-tie/Total counts datasets where DF–MoE matches or exceeds Uniform-local at the reported precision.

Scenario	Uniform-local	DF–MoE	Win-or-tie/Total
Tabular / Subsumption	88.56	**89.27**	14/17
Tabular / Instance recurrence	81.60	**82.08**	17/17
Image / Subsumption	89.69	**89.83**	10/13
Image / Instance recurrence	75.18	**75.43**	10/13

At the reported precision, DF–MoE improves on Uniform-local in 14/17 and 16/17 tabular dataset–scenario pairs under subsumption and instance recurrence, respectively, and in 10/13 and 9/13 image pairs. Two additional pairs are tied, yielding 49 improvements and two ties across the 60 pairs.

Figure 7 complements the aggregate results in Table 9 by showing the accuracy contribution of task-level relevance across datasets. The gains are particularly clear on tabular subsumption: adding global relevance improves Segment by 4.51 points, CJS by 2.25 points, and Zernike by 2.24 points. These examples show how the task-level contribution can improve reuse beyond local compatibility. Values tied at the displayed precision appear at zero in the difference plot.

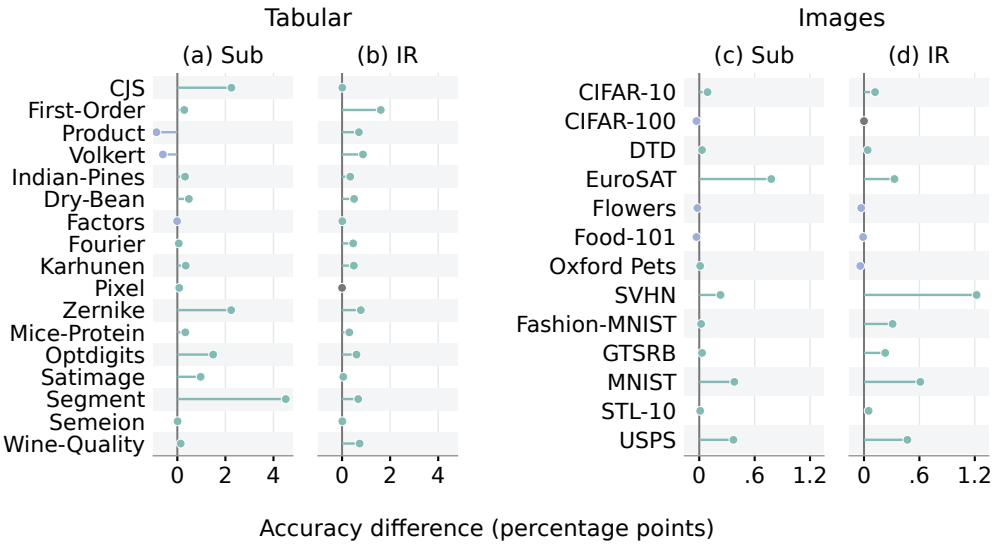


Figure 7: **Contribution of global relevance.** Dataset-level accuracy of DF–MoE minus Uniform-local, in percentage points, computed from the available rounded means. Positive values favor combining global relevance with local compatibility. All datasets are shown, including negative and zero differences; paired panels share dataset labels and a horizontal scale. Sub: subsumption; IR: instance recurrence.

Controlled global–local ablations. To isolate point-level allocation and local compatibility, we cross three global-weight choices with the presence or absence of local compatibility. Let $g_i \geq 0$ denote the global weight of candidate i, with $\sum_i g_i = 1$. Without the local term, model probabilities are averaged with weights g_i; with it, the weights are $g_i q_i(x) / \sum_{\ell=1}^{K} g_\ell q_\ell(x)$. The local score $q_i(x)$ always comes from the original specification. Table 10 summarizes the six resulting variants.

All six variants use the same tasks, candidates, and probability mapping. DF–Select is reported separately: it uses the same reuse scores as DF–MoE but invokes only the highest-weight model. Thus, comparing each global variant with its local counterpart isolates local compatibility. Comparing Whole-global with DF-global, or Whole-local with DF–MoE, isolates the effect of adjusting contributions within each specification.

Table 10: **The six probability-fusion ablations.** Whole-RKME weights are refitted on the same selected candidates; DF weights sum optimized point masses within each learnware.

Global weights g_i	**Without local compatibility**	**With local compatibility**
Equal weights $1/K$	*Uniform-global*	*Uniform-local*
Whole-RKME mixture	*Whole-global*	*Whole-local*
Point-level filling π_i	*DF-global*	*DF–MoE*

Table 11: **Controlled component results on tabular data.** Accuracy (%), mean ± sample standard deviation of three seed-level means, each over ten tasks. $K = 5$, $m = m_u = 100$, and $s = 10^{-3}$. Bold marks the highest mean before rounding, including numerical ties. DF–Select selects a single model and is separated from the six probability-fusion controls.

Method	**CJS Sub**	**CJS IR**	**Fourier Sub**	**Fourier IR**
Uniform-global	**88.64 ±3.47**	70.41 ±3.45	76.42 ±1.90	47.06 ±3.01
Uniform-local	**88.64 ±3.47**	**71.81 ±3.22**	85.69 ±0.37	74.04 ±4.47
Whole-global	67.68 ±2.75	63.37 ±2.05	74.26 ±1.57	45.31 ±1.33
Whole-local	67.70 ±2.75	63.40 ±2.03	84.34 ±1.09	**74.33 ±4.50**
DF-global	86.08 ±3.47	70.74 ±1.42	76.37 ±1.74	44.18 ±1.77
DF–MoE	86.09 ±3.41	70.76 ±1.46	86.02 ±0.53	74.32 ±4.28
DF–Select	69.19 ±1.12	59.44 ±3.10	**86.73 ±1.32**	74.29 ±4.08

Table 12: **Controlled component results on image data.** Accuracy (%), mean ± sample standard deviation of three seed-level means, each over ten tasks. $K = 5$, $m = m_u = 100$, and $s = 10^{-3}$. Bold marks the highest mean before rounding, including numerical ties. DF–Select selects a single model and is separated from the six probability-fusion controls.

Method	**CIFAR-10 Sub**	**CIFAR-10 IR**	**EuroSAT Sub**	**EuroSAT IR**
Uniform-global	89.77 ±3.60	69.81 ±0.96	92.46 ±5.56	70.45 ±6.42
Uniform-local	95.22 ±1.81	88.64 ±1.86	96.31 ±1.85	89.81 ±2.99
Whole-global	89.82 ±3.16	71.84 ±1.51	85.33 ±5.86	70.58 ±6.82
Whole-local	95.19 ±1.80	**88.78 ±1.92**	94.52 ±1.77	89.25 ±3.65
DF-global	89.62 ±3.35	71.28 ±1.11	91.09 ±4.18	69.57 ±6.26
DF–MoE	**95.22 ±1.82**	88.76 ±1.88	**96.47 ±1.66**	**90.09 ±2.98**
DF–Select	93.89 ±1.50	87.93 ±1.83	95.43 ±0.77	88.57 ±2.61

Tables 11 and 12 show the complete paired comparison. Local compatibility raises DF-global accuracy by 5.38–30.14 percentage points across the six Fourier and image settings. With local compatibility held fixed, DF–MoE improves on Whole-local by 18.40 and 7.36 points on CJS, 1.68 points on Fourier subsumption, and 1.95 and 0.84 points on EuroSAT; all three seeds have positive paired gains in each of these five settings. The two methods achieve similar mean accuracy in the remaining three settings. These results identify input-wise reuse and within-candidate mass allocation as distinct sources of improvement.

The CJS controls also explain the role of the local scores. Across the baseline tasks, the largest normalized local share averages only 20.07% in subsumption and 20.10% in instance recurrence among five candidates. The scores are positive and nearly equal, so the local term changes few predictions. DF lowers the average largest global weight from 73.95% to 29.18% in subsumption, and from 45.45% to 27.92% in instance recurrence, improving substantially over the whole-RKME weighting. Uniform-global and Uniform-local attain 88.64% on CJS subsumption, compared with 86.09% for DF–MoE; Uniform-local also leads on CJS instance recurrence (71.81% versus 70.76%). Because the candidates receive nearly equal local compatibility scores on CJS, their global weights largely determine how much each prediction contributes. DF makes these weights less concentrated than whole-RKME matching, but the resulting combination is still less accurate than equal weighting in these two CJS settings. These results suggest that equal global weights can be more effective in these CJS settings: unlabeled distribution matching does not necessarily identify the most accurate predictive combination. On CIFAR-10 and EuroSAT instance recurrence, adding DF to Uniform-

local yields smaller positive gains of 0.12 and 0.28 points, respectively, with consistent direction across seeds.

On Fourier instance recurrence, Whole-local and DF–MoE have nearly identical mean accuracy (74.33% and 74.32%). This is consistent with the instance-recurrent construction, where mixtures of whole source distributions already represent user coverage well; the finer allocation is more useful under subsumption, where DF–MoE gains 1.68 points over Whole-local.

G.3 Comparison with Tree-Based Assembly under Limited Labels

We compare DF with the tree-based assembly method of Tan & Zhou (2026), which uses labeled examples to partition the input space and assign suitable learnwares to different regions. This comparison uses the CJS and Fourier controls under both scenarios, with the same five candidates, evaluation inputs, and output mapping. We apply the tree assembly component in their shared feature space. It receives ten labeled examples per user class from the training split, separate from the evaluation inputs, while DF uses unlabeled inputs for reuse. Each setting comprises ten tasks per seed over three seeds. The tree has a maximum depth of three and uses linear classifiers for its splits.

Table 13: **Comparison with tree-based assembly under limited labels.** Accuracy (%), mean ± sample standard deviation over three seeds. Tree-based assembly uses ten labeled examples per user class; DF uses no user instance labels. Bold marks the highest mean in each row.

Dataset	Scenario	Tree-based Assembly	DF–Select	DF–MoE
CJS	Subsumption	69.05 ±1.68	69.19 ±1.12	**86.09 ±3.41**
CJS	Instance recurrence	68.10 ±2.87	59.44 ±3.10	**70.76 ±1.46**
Fourier	Subsumption	84.80 ±1.08	**86.73 ±1.32**	86.02 ±0.53
Fourier	Instance recurrence	59.31 ±4.30	74.29 ±4.08	**74.32 ±4.28**

Table 13 shows that DF–MoE achieves higher mean accuracy than tree-based assembly in all four settings. The gains are 17.04 and 2.66 percentage points on CJS under subsumption and instance recurrence, and 1.22 and 15.01 points on Fourier. DF–Select also exceeds tree-based assembly in three settings and attains the highest mean on Fourier subsumption. These results support the effectiveness of distribution filling and input-wise reuse in combining complementary capabilities across the four evaluated settings.

G.4 Parameter Sensitivity

Both scans use the same 30 tasks per setting and $K = 5$, varying one parameter at a time.

Regularization strength. We scan $s \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ with 100-point candidate and user specifications. DF–MoE is stable across the regularization grid on Fourier and the image datasets: the largest accuracy range among these six settings is 0.071 percentage points (Figure 8). CJS exhibits greater sensitivity, with ranges of 4.10 and 3.05 points; larger s improves its subsumption result to 88.31%.

Specification size. We scan $m \in \{25, 50, 100, 200\}$ at $s = 10^{-3}$, keeping the user RKME at 100 points. All sizes use the same training data for each candidate and the same construction rule; retrieval and local compatibility use the resulting specifications. For images, up to 500 training examples per class are sampled with a common seed, retaining examples correctly classified by the candidate model. More than 200 examples are retained for each candidate model.

The size scan (Figure 9) shows that moderate specifications already provide useful resolution for image reuse: 50–100 points recover much of the accuracy lost with 25 points in instance recurrence, while 200 points bring no further gain. The response is dataset dependent; Fourier subsumption achieves 89.77% with 25 points and 81.53% with 200. Because specification size affects retrieval, matching, and local scores together, these curves characterize the resulting reuse pipeline and support a moderate default budget.

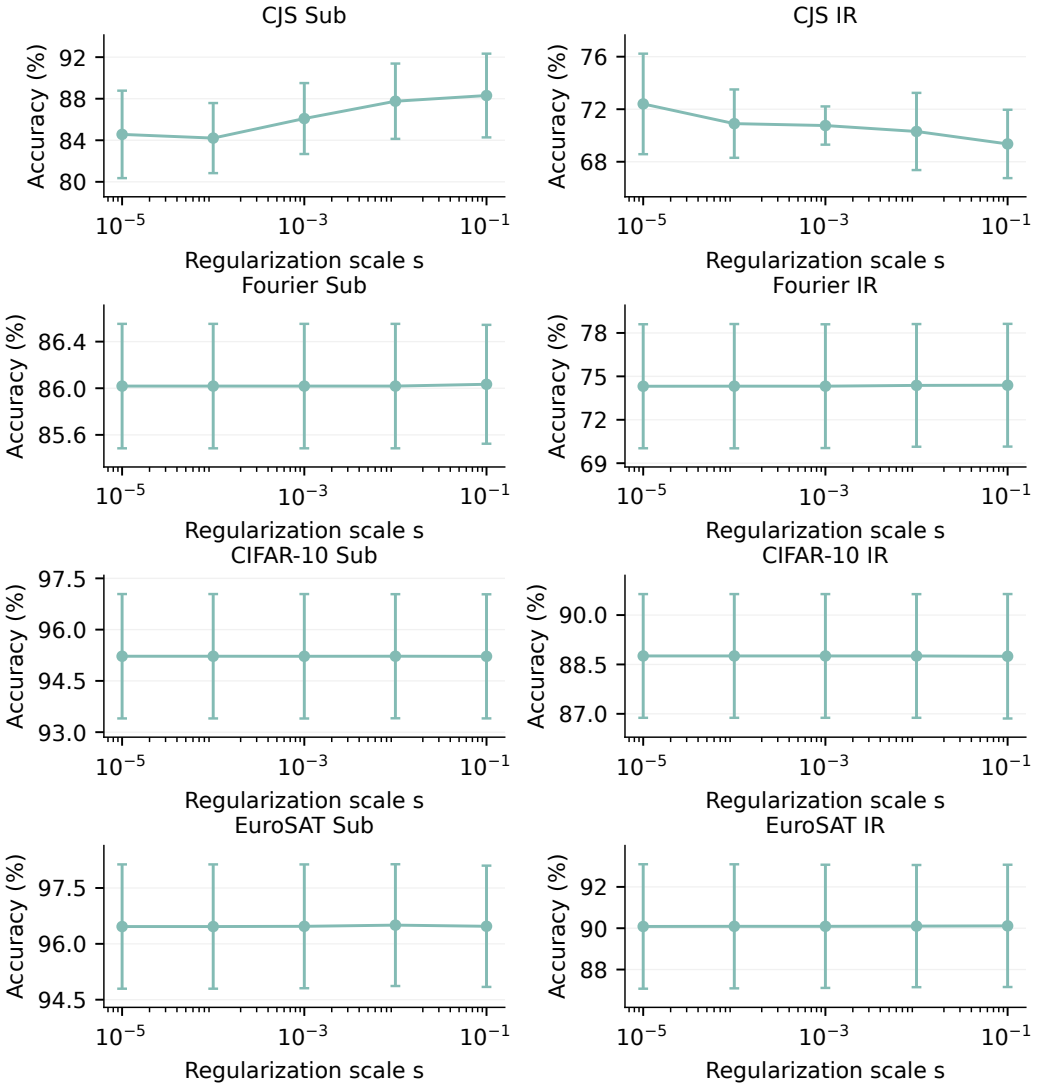


Figure 8: **DF–MoE sensitivity to regularization scale.** All eight controlled settings use the 100-point candidate specifications, $K = 5$, and a 100-point user RKME. Error bars denote one sample standard deviation over three seed-level means, each averaging ten tasks.

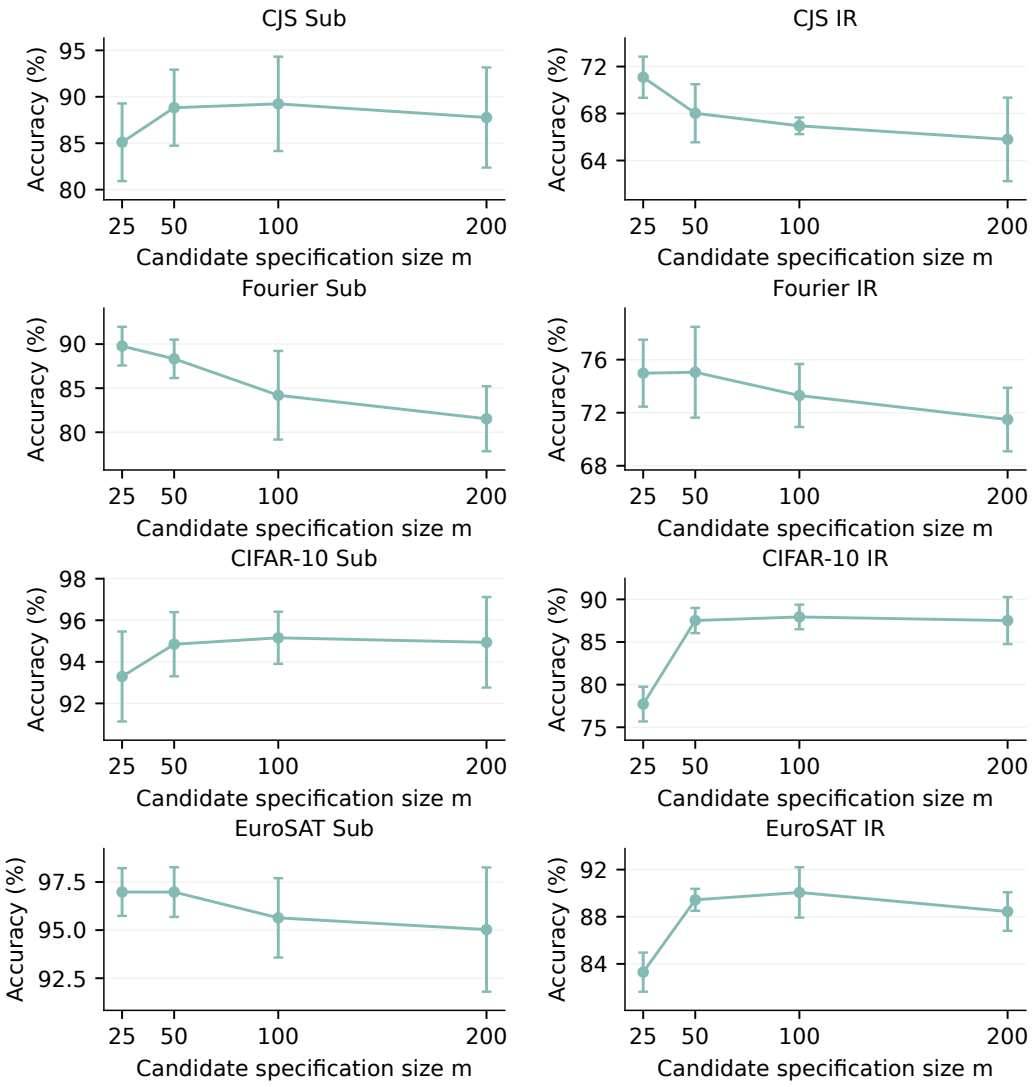


Figure 9: **DF–MoE sensitivity to candidate-specification size.** All specification sizes use the same training inputs for each candidate and the same construction rule. The user RKME stays at 100 points, $K = 5$, and $s = 10^{-3}$. Error bars are seed-level sample standard deviations as in Figure 8.

G.5 Reuse Cost and Candidate Budget

Measurement protocol. We assess the accuracy–cost trade-off over $K \in \{1, 2, 3, 5, 10, 20\}$ using the paired tasks and timing repetitions specified at the start of this appendix. Tabular experiments use a CPU with four computational threads; image experiments use an NVIDIA RTX A6000 (48 GB) with four CPU threads. GPU timings are synchronized. Total reuse time includes shared image feature extraction, user-RKME construction, and full-pool mixture retrieval; method-specific preparation; model selection or weighting; classifier inference; and probability projection and combination. Models and inputs are already loaded. DF preparation constructs and solves its quadratic program; JobSelector preparation performs kernel herding and selector training, with 1,000 mimicked inputs per learnware and the same representation and kernel. These boundaries measure the computation required to prepare and apply reuse on a user batch.

Table 14: **Accuracy and reuse cost at $K = 5$.** Accuracy (%) averages three tasks; time (s) is the median over five timings per task. Bold indicates only the highest accuracy, including ties. Sub: subsumption; IR: instance recurrence.

	Averaging		JobSelector		DF–Select		DF–MoE	
Setting	**Acc.↑**	**Time↓**	**Acc.↑**	**Time↓**	**Acc.↑**	**Time↓**	**Acc.↑**	**Time↓**
CJS Sub	**88.89**	0.185	71.89	477.040	67.86	0.608	**88.89**	0.643
CJS IR	60.24	0.215	61.30	64.867	51.01	0.652	**62.51**	0.684
Fourier Sub	64.97	0.393	88.89	34.156	**93.98**	0.659	91.20	0.687
Fourier IR	41.56	4.958	64.98	40.691	**67.01**	5.269	65.29	5.303
CIFAR-10 Sub	85.65	10.169	92.98	237.500	94.00	5.959	**95.61**	11.218
CIFAR-10 IR	54.96	23.832	84.71	237.356	87.93	12.796	**88.55**	24.591
EuroSAT Sub	88.45	5.681	88.00	178.144	96.06	3.560	**98.32**	6.427
EuroSAT IR	60.96	13.835	67.71	203.635	90.28	8.454	**90.96**	14.975

At $K = 5$, DF–MoE achieves higher mean accuracy and lower median time than JobSelector in all eight controlled settings (Table 14). Across the four image settings, it achieves 9.65–27.72-fold speedups. Preparation accounts for much of this difference: at $K = 20$, the median DF preparation is 5.52 s on CIFAR-10 instance recurrence and 7.36 s on EuroSAT instance recurrence, compared with 585.19 s and 619.85 s for JobSelector. Tables 15 and 16 provide the stage breakdown at the default and largest scanned budgets.

The complete budget curves appear in Figures 10 and 11. Increasing K broadens the available coverage: on the same three Fourier instance-recurrence tasks, DF–MoE rises from 65.29% at $K = 5$ to 83.60% at $K = 20$, with total time increasing from 5.30 s to 9.44 s. The image tasks exhibit a clear choice between two deployment variants. At $K = 5$, DF–Select takes about 52–56% of the total reuse time required by DF–MoE, while DF–MoE improves accuracy over DF–Select by 0.62–2.25 percentage points. At $K = 20$, CIFAR-10 instance recurrence gives 88.22% in 22.89 s with DF–Select and 90.39% in 83.01 s with DF–MoE; EuroSAT gives 90.47% in 18.12 s and 93.58% in 49.49 s. At $K = 10$, only one task in each of CIFAR-10 Sub and EuroSAT Sub has enough eligible candidates, whereas budgets up to five average three tasks; $K = 20$ is unavailable in these two settings. These single-task points are shown with open markers and are not connected to the three-task means. Overall, the eight $K = 5$ comparisons show that DF–MoE improves accuracy over JobSelector while reducing preparation cost without selector training. DF–Select offers a lower-cost prediction option, while DF–MoE benefits from combining model outputs.

Table 15: **Measured stage times on tabular data (seconds).** Medians over three tasks and five repetitions per task at $K = 5$ and, where eligible, $K = 20$. Shared combines feature processing, user RKME construction, and full-pool retrieval. Prepare includes DF matrix construction and solving or JobSelector herding and fitting. Assign denotes model selection or input-specific weighting. Each column is aggregated separately, so stage medians need not sum to the median total.

Method	Shared	Prepare	Assign	Inference	Merge	Total
CJS Sub, $K = 5$						
JobSelector	0.123	476.823	0.062	0.017	0.000	477.040
DF–Select	0.123	0.466	0.003	0.016	0.000	0.608
DF–MoE	0.123	0.466	0.003	0.051	0.000	0.643
CJS IR, $K = 5$						
JobSelector	0.161	64.594	0.113	0.019	0.000	64.867
DF–Select	0.161	0.472	0.005	0.018	0.000	0.652
DF–MoE	0.161	0.472	0.004	0.051	0.000	0.684
CJS IR, $K = 20$						
JobSelector	0.161	81.574	0.288	0.020	0.000	82.024
DF–Select	0.161	6.331	0.015	0.018	0.000	6.531
DF–MoE	0.161	6.331	0.015	0.132	0.000	6.643
Fourier Sub, $K = 5$						
JobSelector	0.349	33.726	0.082	0.016	0.000	34.156
DF–Select	0.349	0.299	0.004	0.016	0.000	0.659
DF–MoE	0.349	0.299	0.005	0.042	0.000	0.687
Fourier Sub, $K = 20$						
JobSelector	0.349	48.016	0.222	0.022	0.000	48.603
DF–Select	0.349	4.995	0.016	0.019	0.000	5.425
DF–MoE	0.349	4.995	0.016	0.100	0.000	5.528
Fourier IR, $K = 5$						
JobSelector	4.898	35.372	0.162	0.024	0.000	40.691
DF–Select	4.898	0.298	0.015	0.024	0.000	5.269
DF–MoE	4.898	0.298	0.011	0.056	0.000	5.303
Fourier IR, $K = 20$						
JobSelector	4.898	46.143	0.570	0.031	0.000	51.782
DF–Select	4.898	4.367	0.042	0.029	0.000	9.327
DF–MoE	4.898	4.367	0.044	0.158	0.001	9.441

Table 16: **Measured stage times on image data (seconds).** Hardware and timing boundaries are given in Appendix G.5; stage definitions and aggregation follow Table 15.

Method	Shared	Prepare	Assign	Inference	Merge	Total
CIFAR-10 Sub, $K = 5$						
JobSelector	3.527	230.234	1.720	1.295	0.000	237.500
DF–Select	3.527	0.326	0.667	1.313	0.000	5.959
DF–MoE	3.527	0.326	0.662	6.500	0.001	11.218
CIFAR-10 IR, $K = 5$						
JobSelector	8.182	221.475	2.412	2.806	0.001	237.356
DF–Select	8.182	0.328	1.356	2.965	0.001	12.796
DF–MoE	8.182	0.328	1.318	14.437	0.003	24.591
CIFAR-10 IR, $K = 20$						
JobSelector	8.182	585.192	12.088	3.155	0.001	609.489
DF–Select	8.182	5.515	5.434	2.861	0.001	22.890
DF–MoE	8.182	5.515	5.519	58.771	0.017	83.009
EuroSAT Sub, $K = 5$						
JobSelector	2.087	173.570	0.490	0.728	0.000	178.144
DF–Select	2.087	0.348	0.421	0.744	0.000	3.560
DF–MoE	2.087	0.348	0.420	3.629	0.001	6.427
EuroSAT IR, $K = 5$						
JobSelector	5.392	195.473	1.123	1.516	0.001	203.635
DF–Select	5.392	0.318	0.776	1.669	0.001	8.454
DF–MoE	5.392	0.318	0.700	8.388	0.002	14.975
EuroSAT IR, $K = 20$						
JobSelector	5.392	619.853	6.459	1.513	0.001	632.744
DF–Select	5.392	7.356	3.218	1.522	0.001	18.124
DF–MoE	5.392	7.356	3.192	32.840	0.008	49.494

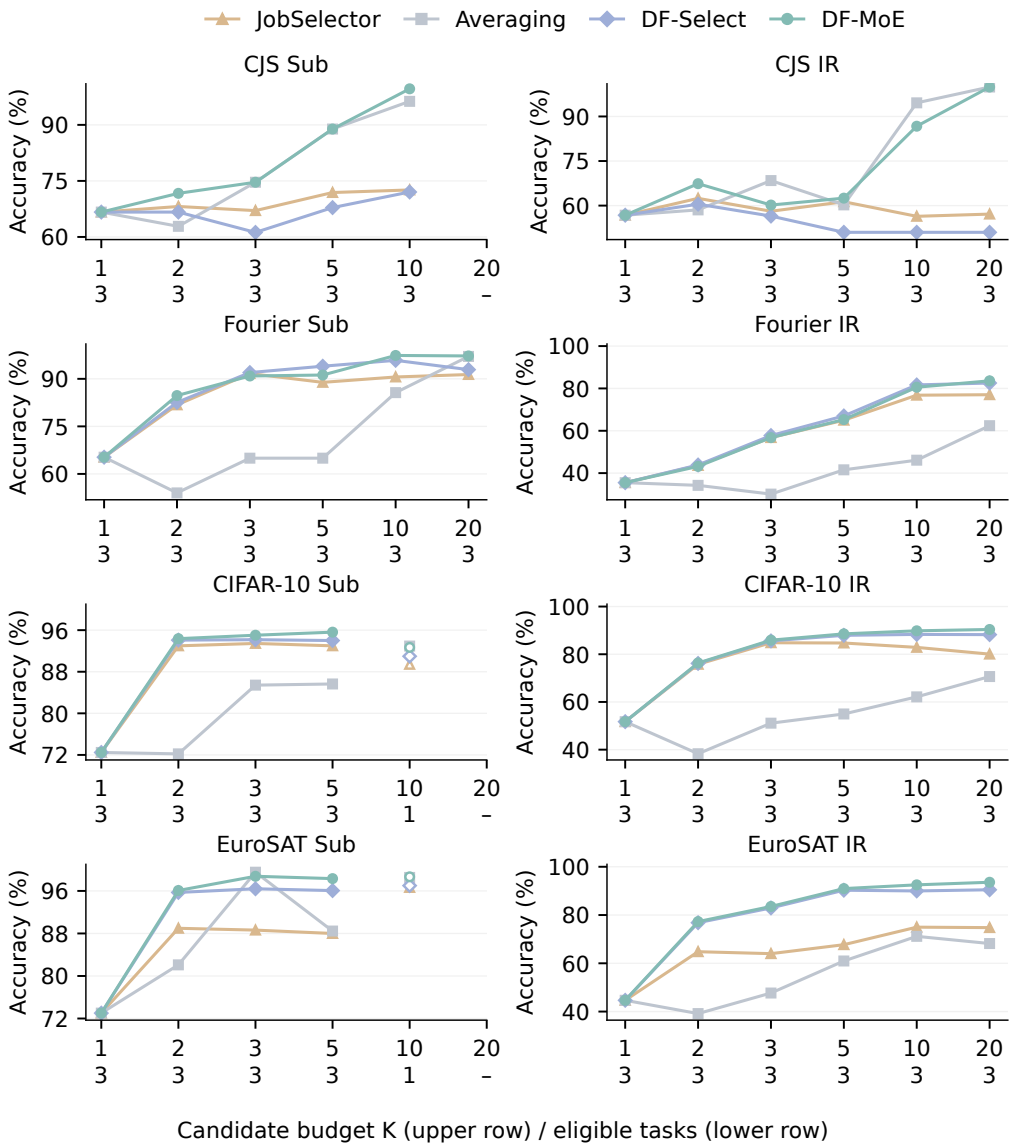


Figure 10: **Accuracy across the full candidate-budget grid.** All four methods share the eligible tasks and candidates at each budget. The task count is shown below each K; an open marker identifies a single-task point and is not joined to points averaging three tasks. Averaging denotes majority voting.

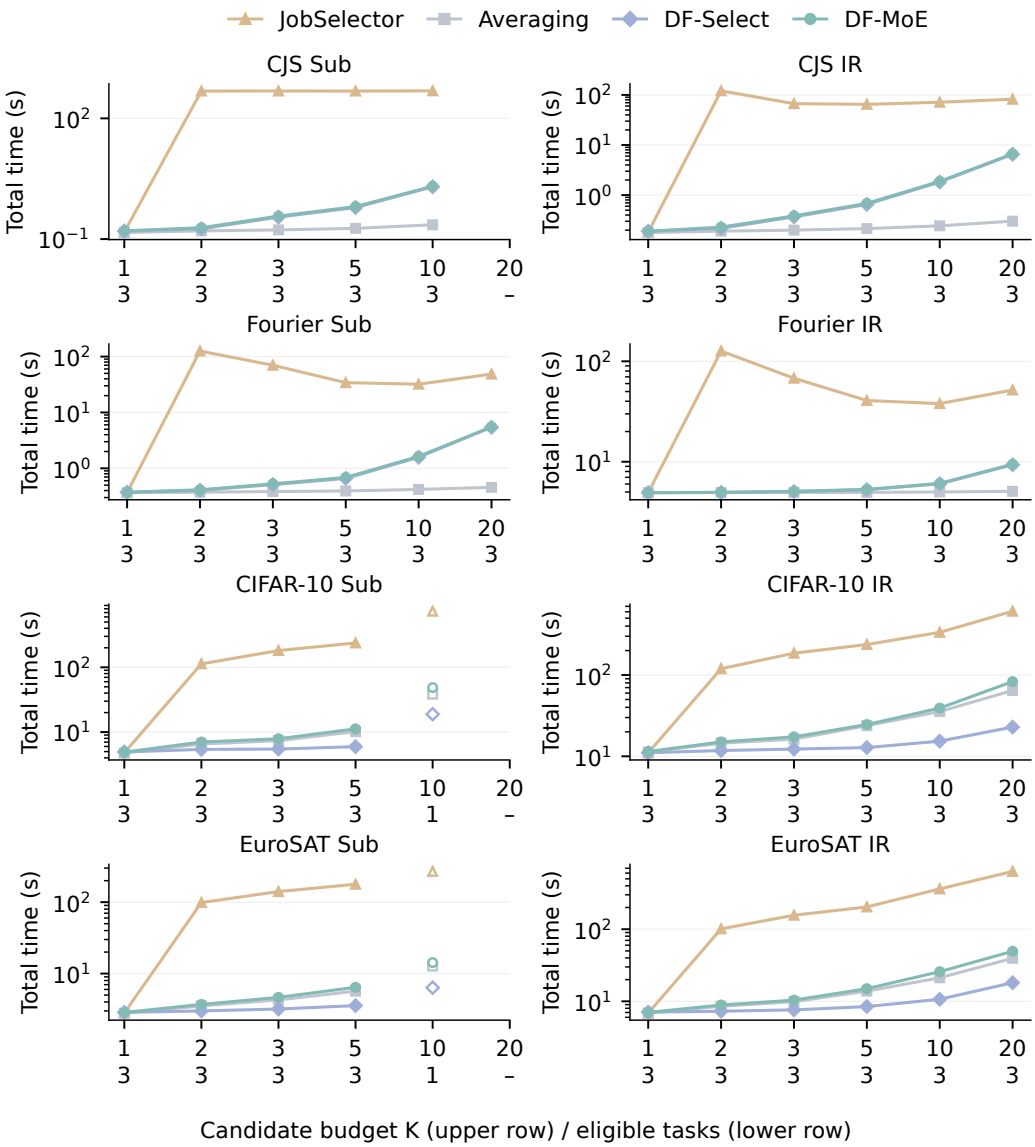


Figure 11: **Reuse time across the full candidate-budget grid.** Median total measured time after one warm-up, with five measured repetitions per eligible task; task counts appear below each K. Comparisons are paired within a setting on the same hardware. The included stages are specified in Appendix G.5. Open markers indicate single-task estimates, as explained in the text.

H Details of the Heterogeneous Reuse Experiment

We examine whether distribution filling can support reuse in a shared learnware dock system (LDS) with heterogeneous feature spaces. Following the sales-forecasting evaluation of Beimingwu (Tan et al., 2024b), we construct a pool of 240 learnwares from public sales data and integrate distribution filling with its heterogeneous specification-based retrieval and reuse workflow.

H.1 Experimental Setup

Dataset and learnware pool. Predict Future Sales (PFS) (Kaggle, 2018), used in the Beimingwu benchmark, provides sales records for store-level regression tasks. We construct six LightGBM learnwares for each of 40 stores, giving a shared pool of 240. One additional store is used only to fit preprocessing parameters and validate the workflow; it is excluded from the formal pool and the 40-store evaluation. Five source feature views and a distinct user view comprise different subsets of the original attributes. Source models use earlier months and user tasks use later months, separating training from evaluation.

To create complementary coverage, we partition product categories into five groups, denoted A–E. Each user requires A/B/C, while each source covers two of these groups together with D/E. The three coverage patterns are ABDE, BCDE, and ACDE, each represented by 80 learnwares. Extra-group sample proportions of 20%, 50%, and 80% are distributed evenly across the pool. Thus, sources contain both useful and additional regions, and several sources can jointly cover the task of a user.

User tasks and heterogeneous reuse. Each store has a balanced profile with equal A/B/C proportions and a skewed profile with proportions 60/20/20%, rotating the dominant group across stores. Each task uses 1,200 unlabeled calibration inputs and 1,200 independent evaluation inputs. Source and user RKMEs contain 100 points; the user RKME is formed in its original feature space before its representative points are mapped to the common space. We use the heterogeneous specification mapping, RKME retrieval, and feature alignment provided by Beimingwu (Tan et al., 2024b), then apply distribution filling to the retrieved specifications and adapted predictions. Retrieval considers the full pool with a budget of at most five candidates; every returned candidate set is retained, including single-candidate results.

Comparisons and evaluation. We compare DF–Select and DF–MoE with RKME-Select, which uses the best single retrieval result, and JobSelector, which learns an input-wise selector. The multi-learnware methods share the retrieved candidates, user specification, and feature-aligned model predictions. For regression, DF–Select selects one scalar prediction and DF–MoE combines scalar predictions with its reuse weights; no user labels enter retrieval or reuse. We repeat the mapping and user-side reuse with three seeds over the same source pool, yielding $40 \times 2 \times 3 = 240$ evaluations. In the following summaries, a task denotes one evaluation of a store and demand profile under a single seed. We report mean RMSE and standard deviation across the three seed-level task averages; lower RMSE is better. Overall scores average the two profiles equally.

H.2 Results and Discussion

Predictive performance across stores. Table 17 compares predictive performance and store-level rankings under balanced and skewed demand and overall. DF–MoE reduces overall RMSE by 14.66% relative to JobSelector and 11.13% relative to RKME-Select. Its reductions over JobSelector are 13.73% and 15.60% for balanced and skewed demand. After averaging over profiles and seeds within each store, DF–MoE attains the lowest RMSE among the four methods on 30 of 40 stores, with an average rank of 1.25. DF–Select also improves on JobSelector in both profiles, while aggregation provides the larger gains.

Gains from multi-learnware reuse. Figure 12b shows that the returned candidate counts $K = 1, 2, 3, 4, 5$ occur in 43, 16, 16, 9, and 156 tasks, respectively. Thus 197/240 tasks (82.1%) involve multiple learnwares, and 156/240 (65.0%) use the full five-candidate budget. The 43 single-candidate tasks give identical predictions across methods, with mean RMSE 1.0130. For the 197 multi-candidate tasks (Figure 12c), task-averaged RMSE is 1.2195 for RKME-Select, 1.2791 for JobSelector, 1.2214 for DF–Select, and 1.0591 for DF–MoE. DF–MoE therefore reduces RMSE by

Table 17: **Performance across 40 stores in a heterogeneous pool of 240 learnwares.** Mean RMSE ± SD over three seed-level averages. Ranks and best-store counts compare store means over seeds (and both profiles for Overall); ties receive average ranks and count for each tied method. Bold marks the best entry among the four methods.

Metric	RKME-Select	JobSelector	DF–Select	DF–MoE
Balanced demand (40 stores)				
Mean RMSE ↓	1.2043 ±0.1142	1.2354 ±0.1189	1.2002 ±0.1384	**1.0658 ±0.0620**
Average rank ↓	2.75	2.98	2.93	**1.35**
Best-store count ↑	12	0	0	**28**
Skewed demand (40 stores)				
Mean RMSE ↓	1.1607 ±0.0285	1.2274 ±0.0934	1.1679 ±0.0690	**1.0360 ±0.0264**
Average rank ↓	2.60	3.35	2.73	**1.33**
Best-store count ↑	11	0	1	**28**
Overall (40 stores)				
Mean RMSE ↓	1.1825 ±0.0691	1.2314 ±0.0973	1.1841 ±0.0994	**1.0509 ±0.0421**
Average rank ↓	2.68	3.10	2.98	**1.25**
Best-store count ↑	9	0	1	**30**

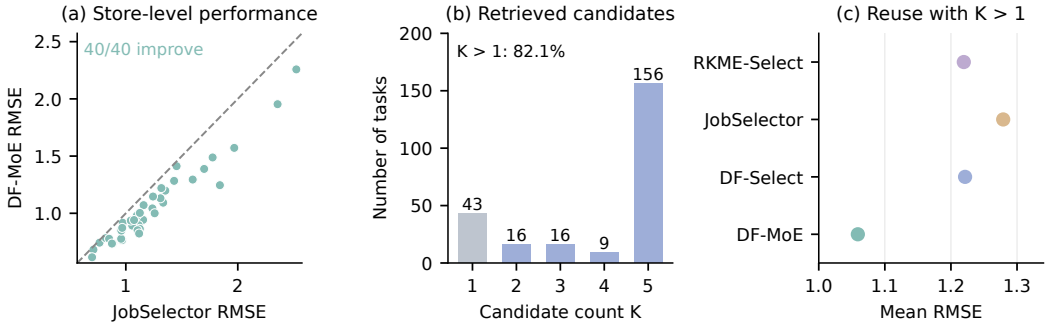


Figure 12: **Predictive gains and candidate multiplicity in the heterogeneous learnware system.** (a) Each point is one store, averaged over profiles and seeds; all 40 lie below the equality line. (b) Retrieved candidate counts across all 240 tasks. (c) Mean RMSE on the 197 tasks with multiple candidates. These are descriptive summaries of the evaluation in the shared LDS.

17.19% relative to JobSelector and 13.15% relative to RKME-Select on this subset. It improves on JobSelector in 180/197 tasks (91.4%), with 17 losses and no ties. These subset means weight tasks equally; they describe the candidate sets returned by the same retrieval procedure.

Across all tasks, averaging profiles and seeds gives lower DF–MoE RMSE than JobSelector for every store (Figure 12a). The multi-candidate results show that the aggregate improvement comes from combining learnwares rather than single-model returns. Together, these findings demonstrate effective integration of distribution filling with the existing heterogeneous specification mapping, retrieval, and feature alignment in the system for complementary reuse on real-world sales data.