COMPOSITIONAL LEARNWARE IDENTIFICATION AND GRADIENT-GUIDED ASSEMBLY

Anonymous authors
Paper under double-blind review

ABSTRACT

Tasks and query batches often require diverse capabilities, motivating complementary model reuse from growing repositories. The learnware paradigm—*Learnware = Model + Specification*—provides reusable capability representations, supporting such reuse without evaluating every candidate on the target task. PAVE (Parameter Vector) specifications encode model capabilities and user requirements as parameter updates in a common space. However, independently ranking models by PAVE similarity can favor overlapping capabilities, while collection-level PAVEs provide limited evidence for assigning individual inputs. We address these challenges through complementary identification and gradient-guided sample-level assignment. Inspired by task arithmetic, we formulate identification as sparse PAVE reconstruction, using the unexplained user-PAVE residual to guide complementary model selection. For assignment, we exploit sample gradients as local adaptation signals, calibrated using collection-level statistics, to match individual inputs with selected model PAVEs. Under a local update approximation, our analysis relates gradient–PAVE matching to average gradient-kernel similarity between a sample and a specification's fitting distribution. Across 72 mixed LLM batches, 720 mixed CV/NLP runs, and 41 single-dataset benchmarks, our method consistently improves model reuse without training a parametric selector. It exceeds the strongest compared learning-based router by 1.06 points on mixed LLM workloads and improves over hindsight Best-Single by 10.62/2.52 points on mixed CV/NLP tasks, while invoking one candidate per input. Extensive controlled ablations further validate the contributions of complementary identification and gradient-guided assembly, as well as the effects of gradient calibration and common-direction removal.

1 INTRODUCTION

Model repositories offer a growing collection of independently developed models whose strengths vary across inputs (Hu et al., 2024; Li et al., 2026). Meanwhile, real-world applications often involve tasks or query batches requiring multiple capabilities; financial workloads, for example, may combine sentiment assessment, numerical reasoning, and risk identification. A single model may therefore leave some requirements unmet, whereas independently matching each input against a large candidate pool can be costly and may rely on insufficient sample-level evidence. It can also overlook information shared across the workload that reveals its broader capability needs. These observations motivate leveraging workload-level evidence to identify a small set of complementary models for different subsets of inputs. Doing so at scale calls for reusable capability representations that support model identification without evaluating every candidate on each new workload.

The learnware paradigm provides such representations by pairing each model with a specification that characterizes its functionality: *Learnware = Model + Specification* (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026). PAVE (**Pa**rameter **Ve**ctor) specifications encode model behavior and user requirements as parameter updates in a shared space (Shi et al., 2026; Tan et al., 2025), enabling model identification through similarity between their specifications. Constructive specifications further organize PAVEs derived from different datasets and data partitions to characterize black-box model capabilities at multiple granularities (Liu et al., 2026b). These advances make PAVE a natural basis for workload-level model reuse, but extending PAVE-based identification to multi-model reuse raises two challenges: identifying complementary models and assigning inputs among them.

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.

The first challenge is identifying complementary models, as ranking candidates independently by their similarity to the user PAVE may favor models with overlapping capabilities. Inspired by the linear compositionality of task arithmetic (Ilharco et al., 2023; Ortiz-Jiménez et al., 2023), we identify a small set of models whose PAVEs jointly reconstruct the user PAVE through a sparse linear combination. The second challenge is assigning individual inputs among these models. While PAVEs summarize collection-level capabilities, assignment requires sample-specific evidence. We address this gap by querying the selected PAVEs with sample gradients calibrated using collection-level statistics, enabling assignment for both unlabeled query batches and tasks with limited labeled data. Under a local update approximation, we further show that gradient–PAVE inner products reflect a sample's average gradient-kernel similarity to the fitting distribution represented by each specification.

We evaluate our method on systems of 10 black-box LLMs, 12 CV models, and 15 NLP models, covering 72 mixed LLM batches, 720 mixed-dataset CV/NLP runs, and 41 single datasets. Balanced and long-tail mixtures, out-of-system LLM benchmarks, and leave-one-out CV/NLP settings test robustness and generalization across diverse capability compositions and unseen datasets. Controlled ablations further isolate the contributions of identification and assembly by matching candidate counts and fixing candidate sets and labeled user data, while additional analyses examine the effects of gradient calibration and common-space removal for our method. Our main contributions are:

- **Complementary model identification.** A model's complementary value depends on requirements not yet explained by the selected set, rather than on its standalone similarity to the user PAVE. Inspired by task arithmetic, we capture this dependence through sparse PAVE reconstruction, using the residual to guide complementary model selection.
- **Sample-level model assignment.** While PAVEs encode aggregate capabilities, sample gradients capture the local adaptation direction of individual inputs. We use these local signals, calibrated by collection-level statistics, to assign inputs among selected models by gradient–PAVE matching. Our kernel-query analysis explains how this matching reflects a sample's compatibility with the fitting distribution represented by a specification.
- **Broad empirical gains with controlled validation.** Without training a parametric selector, our method exceeds the strongest compared learning-based router by 1.06 points on mixed LLM workloads and 2.62 points across single-dataset LLM benchmarks. On mixed CV/NLP tasks, it improves over hindsight Best-Single by 10.62/2.52 points and PAVE Top-8 Ensemble by 9.28/9.67 points, executing one candidate per input. Extensive controlled ablations further isolate the gains from complementary identification and gradient-guided assembly, while further analyses examine gradient calibration and common-direction removal.

2 Problem Formulation

We consider a model repository $\{f_m\}_{m=1}^{M}$ with index set $\mathcal{M} = [M]$, and user inputs $\mathcal{U} = \{x_i\}_{i=1}^{N_q}$.

Two scenarios. We study two scenarios under a unified formulation. In the *batch-query scenario*, we assign unlabeled queries $\mathcal{U}$ to black-box LLMs without target fine-tuning. Model-side evidence comprises successful examples on system-maintained datasets $\mathcal{D} = \{D_d\}_d$, termed performance profiles. In the *task scenario*, users reuse developer-uploaded learnwares trained on datasets D_m^{train} (Shi et al., 2026), providing limited labeled data D_u^l for selection and reserving $\mathcal{U}$ for evaluation.

Joint identification and assignment. Using model-side evidence and either unlabeled $\mathcal{U}$ or labeled D_u^l, we select a candidate set $\mathcal{C}$ and an assignment function $a : \mathcal{U} \to \mathcal{C}$. Leaving their dependence on this shared evidence implicit, we maximize expected utility under a model budget K:

$$\max_{\mathcal{C},a} \quad \mathbb{E}\left[\frac{1}{N_q}\sum_{i=1}^{N_q} u(f_{a_i}(x_i), y_i)\right] \qquad (1)$$
$$\text{s.t.} \quad \mathcal{C} \subseteq \mathcal{M}, \quad 1 \le |\mathcal{C}| \le K, \quad a_i = a(x_i) \in \mathcal{C}, \quad i = 1, \ldots, N_q.$$

Here y_i is the target and u is a higher-is-better utility; for accuracy, $u(f_{a_i}(x_i), y_i) = \mathbf{1}\{f_{a_i}(x_i) = y_i\}$. The expectation covers evaluation samples and stochastic predictions, and f_m includes its output interface and permitted adaptation. Each input uses one model, with at most K models overall.

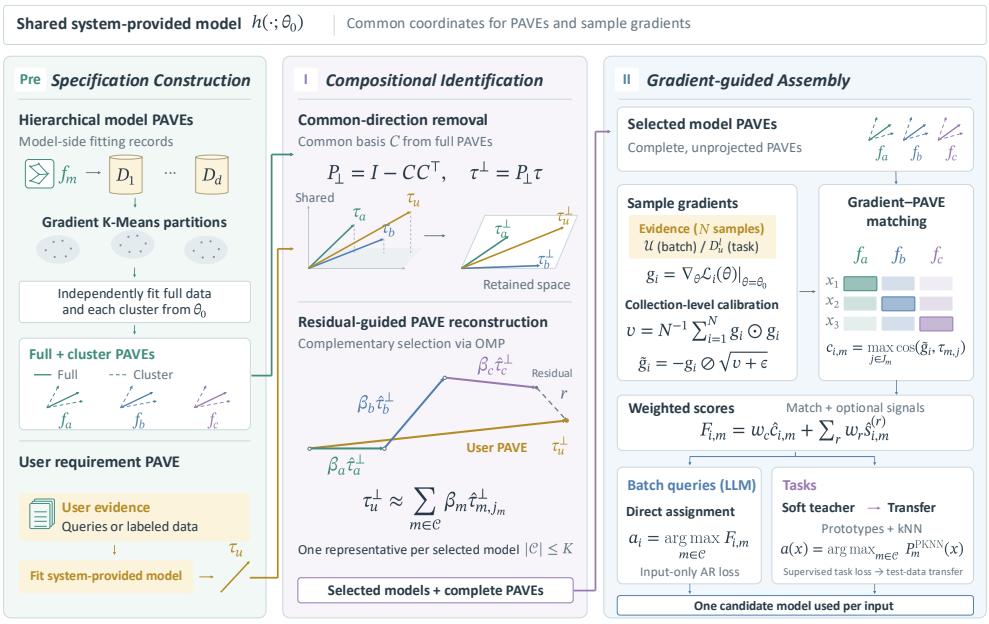

Figure 1: Method overview. **(Pre)** Full and gradient-clustered records yield hierarchical model PAVEs; user evidence yields a task PAVE. **(I)** Common-direction removal and sparse reconstruction identify candidates $\mathcal{C}$. **(II)** Calibrated gradients guide direct batch-query assignment or teacher transfer to unlabeled task inputs, with one candidate model serving each input.

3 Methodology

Figure 1 illustrates the overall workflow of our method. Hierarchical model PAVEs are prepared within the learnware system. At task arrival, we construct a user PAVE to identify complementary models, then use calibrated sample gradients to guide assignment within the selected candidate set.

3.1 Specification Construction

PAVE. A PAVE (**Pa**rameter **Ve**ctor) encodes model capability or task requirements through the parameter update obtained by fitting a system-provided pre-trained model $h(\cdot;\theta_0)$ to model predictions or user data (Shi et al., 2026; Tan et al., 2025). For a dataset D, the corresponding PAVE is

$$\tau(D) = \arg\min_{\tau} \sum_{(x,y)\in D} w(x)\mathcal{L}(h(x;\theta_0+\tau), y), \tag{2}$$

where y is a model prediction or ground-truth target, $w(x) \ge 0$ is a sample weight, and $\mathcal{L}$ the fitting loss. In practice, PAVEs are constructed via LoRA fine-tuning, reducing computational and storage costs (Hu et al., 2022). PAVE cosine similarity reflects capability–requirement alignment.

Hierarchical specifications. Following constructive specification (Liu et al., 2026b), we capture within-dataset capability variation at two levels: a *full PAVE* fitted to all records of a dataset and *cluster PAVEs* independently fitted to subsets obtained by clustering. These vectors form the hierarchy

$$\mathcal{T}_m(D) = \{\tau_m(D)\} \cup \{\tau_m(D,c)\}_c, \qquad \mathcal{S}_m(\mathcal{D}_m) = \{\mathcal{T}_m(D) \mid D \in \mathcal{D}_m,\ \mathcal{T}_m(D) \neq \emptyset\}.$$

K-Means partitions per-example gradients under the PAVE fitting loss, grouping samples with compatible first-order update requirements in the shared system-provided model.

In the *batch-query scenario*, each black-box LLM's specification fits the input texts of its successful examples from profiled datasets $\mathcal{D}_m \subseteq \mathcal{D}$. The user PAVE $\tau_u = \tau(\mathcal{U})$ fits the unlabeled query collection. Both use the input-only autoregressive loss, which also supplies assembly gradients.

In the *task scenario*, model PAVEs fit prediction records $D_m^{\text{pred}} = \{(x, f_m(x)) \mid (x,y) \in D_m^{\text{train}}\}$ from the developer's training set, with $\mathcal{D}_m = \{D_m^{\text{pred}}\}$. The user PAVE $\tau_u = \tau(D_u^l)$ instead fits labeled user data with a supervised loss. Construction details are in Appendix B.1.

We index model m's PAVEs by $j = (D, c) \in \mathcal{J}_m$, where $c = 0$ denotes the full dataset and $c \geq 1$ a retained cluster, yielding the vectors used for identification and assembly:

$$\{\tau_{m,j} \mid j \in \mathcal{J}_m\} = \bigcup_{D \in \mathcal{D}_m} \mathcal{T}_m(D). \tag{3}$$

3.2 Compositional Learnware Identification

Inspired by task arithmetic (Ilharco et al., 2023; Ortiz-Jiménez et al., 2023), we identify complementary learnwares through sparse linear reconstruction of the user PAVE to capture joint capability requirements. Each selected PAVE identifies its model as a candidate for sample-level assembly.

Shared PAVE directions can obscure model differences, motivating common-component removal in light of task-vector interference (Ortiz-Jiménez et al., 2023; Gargiulo et al., 2025). We estimate an orthonormal basis C by SVD of one normalized, query-matched full PAVE per model, avoiding unequal weighting from different cluster counts while retaining cluster PAVEs for selection.

We reconstruct only the component orthogonal to C. With $P_\perp = I - CC^\top$, decompose $\tau_u = CC^\top \tau_u + \tau_u^\perp$, where $\tau_u^\perp = P_\perp \tau_u$, and normalize nonzero projected PAVEs as $\widehat{\tau}_{m,j}^\perp = P_\perp \tau_{m,j} / \|P_\perp \tau_{m,j}\|_2$. Under a model budget K, we select at most one representative per model:

$$\min_{\substack{\mathcal{C} \subseteq \mathcal{M},\ |\mathcal{C}| \leq K \\ j_m \in \mathcal{J}_m,\ \beta_m \in \mathbb{R}}} \left\| \tau_u^\perp - \sum_{m \in \mathcal{C}} \beta_m \widehat{\tau}_{m,j_m}^\perp \right\|_2^2. \tag{4}$$

Here j_m indexes model m's selected PAVE, and $\beta_m \in \mathbb{R}$ permits directional addition or cancellation. Restoring the user's unchanged common component gives the full reconstruction

$$\widehat{\tau}_u = CC^\top \tau_u + \sum_{m \in \mathcal{C}} \beta_m \widehat{\tau}_{m,j_m}^\perp \approx \tau_u. \tag{5}$$

We use Orthogonal Matching Pursuit (OMP) (Pati et al., 1993) to approximately solve Eq. (4): each iteration selects a PAVE from an unselected model by absolute residual correlation, then jointly refits all coefficients. It returns the corresponding model set $\mathcal{C}$ of at most K models, with optional early stopping based on a relative-explainability threshold (Algorithm 1; Appendix B.2).

3.3 Gradient-Guided Model Assembly

Gradient-guided assembly assigns inputs within $\mathcal{C}$ using the candidates' complete specifications. Let $\mathcal{E} = \mathcal{U}$ for batch queries and $\mathcal{E} = D_u^l$ for tasks, with $N = |\mathcal{E}|$. For each evidence record $i \in [N]$, we compute $g_i = \nabla_\theta \mathcal{L}_i(\theta)|_{\theta=\theta_0}$ for its fitting loss $\mathcal{L}_i$, in the shared PAVE coordinates.

To prevent consistently large gradient coordinates from dominating matching, we calibrate each coordinate by its root-mean-square magnitude across evidence records in $\mathcal{E}$:

$$v = \frac{1}{N} \sum_{i=1}^{N} g_i \odot g_i, \qquad \tilde{g}_i = -g_i \oslash \sqrt{v + \epsilon}, \tag{6}$$

where operations are element-wise and $\epsilon > 0$. This adaptive-optimization-inspired update (Kingma & Ba, 2015) is directly matched to candidate PAVEs by cosine similarity:

$$c_{i,m} = \max_{j \in \mathcal{J}_m} \cos(\tilde{g}_i, \tau_{m,j}), \qquad m \in \mathcal{C}. \tag{7}$$

Each sample matches the most aligned full or cluster PAVE. Reduced shared coordinates can lower computation (Appendix D.1.3); Section 3.4 explains gradient–PAVE matching and calibration.

We combine normalized matching with optional auxiliary scores to guide assignment. Batch queries use input-only gradients for direct assignment; tasks transfer a soft teacher derived from labeled gradients on D_u^l to unlabeled inputs through shared encoder features, prototypes, and KNN. Algorithm 2 gives the assembly procedure; Appendices B.3.1 and B.3.2 detail scoring and teacher transfer.

3.4 Understanding Gradient–PAVE Matching

Gradient–PAVE matching as a kernel query. Let z_i denote the fitting record used to compute g_i, including its input and loss-specific target. Write $g_i = g(z_i)$ and let $D_{m,j}$ contain the fitting records for a full or cluster PAVE. Following Shi et al. (2026), we view PAVE as an accumulation of gradients in the NTK regime. Approximating these gradients by their values at the shared initialization yields

$$\tau_{m,j} \approx -\frac{\eta_{m,j}}{|D_{m,j}|} \sum_{z \in D_{m,j}} g(z), \tag{8}$$

where $\eta_{m,j} > 0$ captures the accumulated step scale, neglecting gradient evolution and adaptive coordinate scaling. Defining $k(z, z') = g(z)^\top g(z')$, we obtain the kernel query

$$-g_i^\top \tau_{m,j} \approx \frac{\eta_{m,j}}{|D_{m,j}|} \sum_{z \in D_{m,j}} k(z_i, z). \tag{9}$$

Thus, a sample gradient queries a model PAVE's fitting distribution by mean gradient-kernel similarity.

Task-conditioned kernel queries. Calibration in Eq. (6) balances collection-level coordinate scales across parameter dimensions through $\tilde{g}_i = -H_u g_i$, where $H_u = \mathrm{diag}((v + \epsilon)^{-1/2})$. For a fixed user collection, it preserves the underlying query structure under the same local approximation:

$$k_u(z, z') = g(z)^\top H_u g(z'), \quad \tilde{g}_i^\top \tau_{m,j} \approx \frac{\eta_{m,j}}{|D_{m,j}|} \sum_{z \in D_{m,j}} k_u(z_i, z). \tag{10}$$

The feature map $H_u^{1/2} g(z)$ makes k_u positive semidefinite: calibration changes the geometry but preserves the sample-to-distribution comparison. A higher local cosine score thus suggests stronger alignment between the query's local requirements and the model capabilities represented by the matched specification. Appendices C.1–C.2 provide the supporting derivations and further details.

4 Experiments

To simulate the diverse capability requirements that can coexist within a task or query batch, we combine samples from different datasets into mixed workloads. We also evaluate single-dataset workloads to test whether multi-model reuse remains useful without an explicitly constructed mixture. Finally, we examine how gradient calibration affects sample–model matching and conduct ablations to assess the quality of the identified candidate sets and the effectiveness of gradient-guided assembly.

4.1 Mixed-Dataset Evaluation

4.1.1 Batch-Query Scenario: Black-Box LLMs

Setup. We construct a system of 10 black-box LLMs and evaluate 12 mixtures from 14 datasets spanning mathematics, healthcare, and finance. Balanced and long-tail proportions with 3 seeds yield 72 query batches of 64 queries each. We report Utility, the average per-query benchmark score on a 100-point scale, as mean $\pm$ SD over 3 seeds. Appendix D.1.1 describes the models and datasets; Appendix D.2.1 details the mixture compositions, sampling procedure, and metric calculations.

Baselines and configuration. Random, DescSim, K-NN, EmbedLLM (Zhuang et al., 2025), GraphRouter (Feng et al., 2025), and UniRoute (K-means) (Jitkrittum et al., 2025) assign one model per query. PAVE-Single fits a user PAVE to the full batch and selects one model for all queries. Cluster-PAVE instead partitions the batch with K-means, fits a PAVE per cluster, and selects one model per cluster. Appendix D.1.1 describes the baselines, and Table A6 reports our configuration.

Results and analysis. Figure 2(a) shows that our method achieves a mean Utility of 81.16, exceeding the strongest baseline, UniRoute, by 1.06 points. Cluster-PAVE improves upon PAVE-Single from 76.43 to 79.28, supporting the benefit of using different models for different parts of a mixed batch. Our additional 1.88-point gain over Cluster-PAVE highlights the benefits of both stages: capability-aware identification finds complementary models that jointly address batch-level requirements, while gradient-guided assembly exploits local sample information for finer model assignment. Their combination better aligns global complementarity with sample-specific needs.

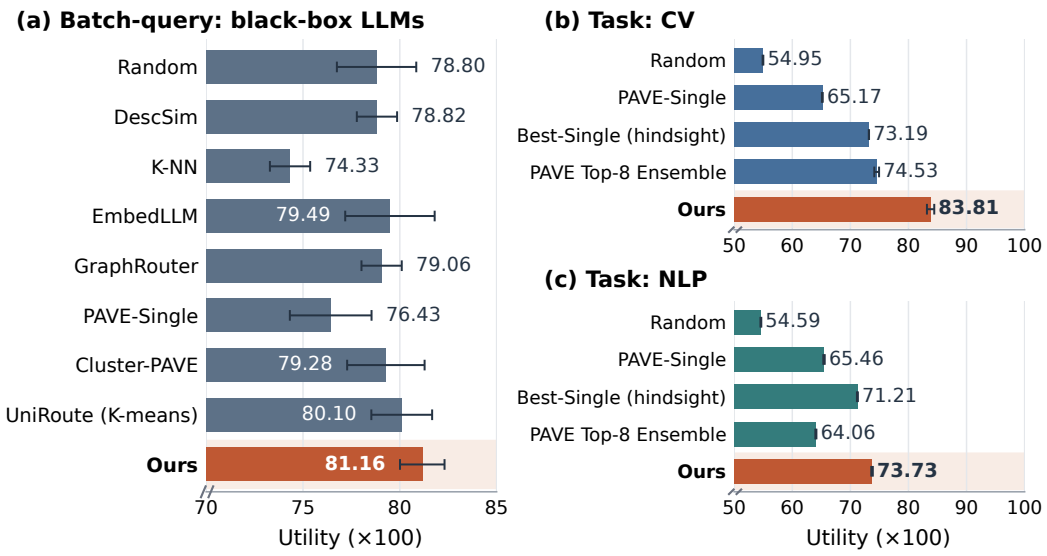

Figure 2: Mixed-dataset Utility ($\times 100$) for (a) black-box LLMs, (b) CV, and (c) NLP. Labels give means; error bars show sample SD over 3 seed means. CV uses accuracy and NLP uses native metric-row utility. Tables A11 and A12 report numerical results and aggregation details.

Identification overhead. Including the shared user-PAVE construction cost, our method increases identification time by only 7.02% relative to PAVE-Single on the same mixed workloads, indicating modest overhead for identifying multiple models (Table A9 in the appendix).

4.1.2 Task Scenario: CV/NLP

Setup. We construct separate systems of 12 CV models and 15 NLP models. Each dataset comprises a training split and a test split, with the latter reserved for evaluation. The training split is further divided into 2 subsets: one for developers to train a model and the other to provide a small amount of labeled user data. Each dataset contributes one model to the system. Each modality uses 60 mixtures: 20 Between-Group, 20 Across-All-Groups, and 20 Global-Random. Balanced and long-tail proportions and 3 sample seeds produce 360 runs per modality. Appendix D.1.2 describes the models and datasets; Appendix D.2.2 specifies the mixture designs and sampling procedure.

Baselines and configuration. We compare Random, PAVE-Single, hindsight Best-Single, PAVE Top-8 Ensemble, and our method. Our method uses a model budget of 8, a relative-explainability threshold of 95% and common rank $r_c = 1$ in both modalities. Appendix D.1.2 describes the compared methods, and Table A7 reports our method's configuration.

Results and analysis. Figure 2(b,c) shows that our method achieves the highest mean Utility in CV and NLP, exceeding single-model selection and 8-model ensembling while invoking one candidate per input. Gains of 10.62 points on CV and 2.52 on NLP over hindsight Best-Single highlight the value of assigning different models across a mixed workload. Table A15 in the appendix shows that identification retrieves over 80% of the models trained on the mixture's datasets in both modalities, while assembly assigns most inputs to the model trained on their respective dataset. These results suggest that the method preserves dataset-specific capability differences when selecting models and assigning samples. Table A13 reports results by mixture design and proportion regime.

Illustrative cases. As shown in Figure 3, the identified models serve different sample subsets within each mixed workload. Across these 6 selected cases, identification recovers every component-trained model in each seed, and assembly assigns most inputs to the models trained on their respective datasets. CV-2 and NLP-1 identify exactly the component-trained models; in several other cases, assembly remains aligned with the component sources despite additional identified candidates. These

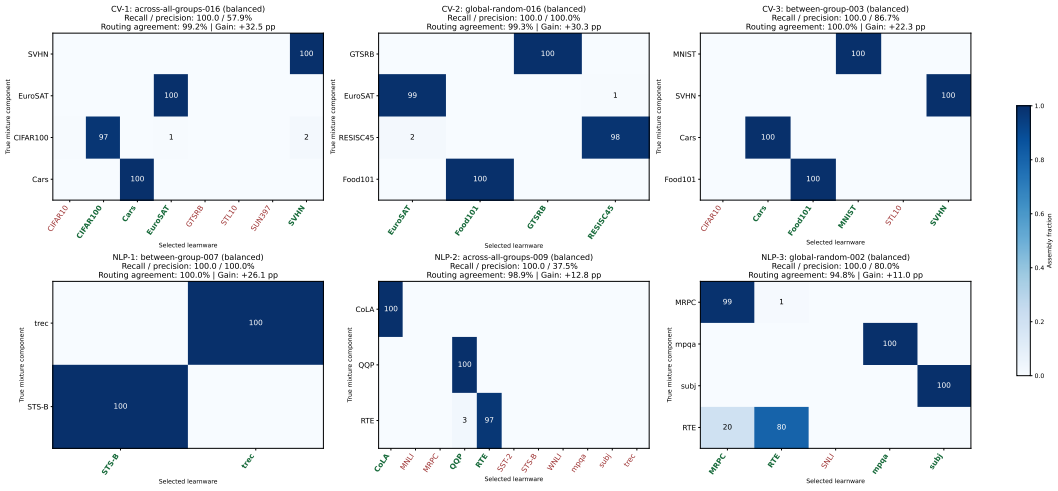

Figure 3: Selected CV (top) and NLP (bottom) mixtures. Rows are components; columns are the identified union; cells are pooled assembly fractions. Green labels denote component models, red labels other models. Assignment agreement measures how often inputs are assigned to models trained on their component datasets. Table A16 reports predictive Utility and candidate precision separately.

selected successes illustrate the mechanism, suggesting selective use of the retrieved candidate pool. Appendix E.1.2 gives the case-selection criteria, and Table A16 reports the full quantitative results.

4.2 Single-Dataset Evaluation

To complement the constructed mixtures, we evaluate individual datasets from the same benchmark pools as separate user workloads. For CV/NLP, we use leave-one-task-out evaluation: the model trained on the target dataset is excluded from the candidate pool, so the task must be served by models trained elsewhere. Dataset splits and protocol details are given in Appendix D.3.

Batch-query scenario (black-box LLMs). Our method achieves the highest macro-average on both in-system and out-of-system benchmarks, with 9 wins, 0 ties, and 5 losses against the strongest overall learning-based routing baseline. The out-of-system gains support reusing capability profiles for datasets absent from profiling. Tables A17 and A18 in the appendix report the complete results.

Task scenario (CV/NLP). Our method improves the dataset-average score over PAVE-Single in both modalities, with win/tie/loss counts of 5/2/5 for CV and 9/5/1 for NLP. It also exceeds PAVE Top-8 Ensemble on NLP while invoking one candidate per input; the ensemble retains higher average accuracy on CV. Tables A19 and A20 in the appendix report the complete results.

4.3 Mechanism Analysis and Ablation Studies

4.3.1 Understanding Gradient Calibration

We compare raw and calibrated gradient–PAVE rankings using 10 black-box LLMs and 100 queries from each of 8 in-system datasets. Scoring all 10 models without candidate identification or auxiliary signals isolates local matching; correctness is used only for evaluation. Table A21 in the appendix shows improvements in every reported ranking metric and in top-1 accuracy after calibration. The mean score gap between correct and incorrect candidates also becomes positive. These results support task-level rescaling as a way to better align local matching scores with observed model correctness and provide more informative signals for assembly. Appendix D.4.1 details sampling and metrics.

Table 1: Identification and assembly ablations on 720 CV/NLP mixed-task runs. Utility ($\times 100$) is mean $\pm$ sample SD over 3 seed means. Identification methods (column groups) match candidate counts per run: $\bar{K} = 4.49$ for CV and 4.90 for NLP. Assembly methods (rows) share the same candidate set and model order within each identification method. Oracle selects the best candidate per sample; "–" denotes unevaluated combinations.

	Our identification		Similarity Top-K	
Assembly	CV	NLP	CV	NLP
Ours	83.81 ± 0.63	73.73 ± 0.13	78.01 ± 0.55	72.65 ± 0.32
Uniform ensemble	79.07 ± 0.47	67.57 ± 0.02	74.28 ± 0.24	67.84 ± 0.23
Supervised MLP	83.39 ± 0.41	69.67 ± 0.22	–	–
Supervised Linear	81.95 ± 0.41	68.74 ± 0.22	–	–
Oracle	89.03 ± 0.45	91.90 ± 0.14	85.62 ± 0.21	91.70 ± 0.18

Table 2: Gradient calibration and common-space ablations on the 72 LLM mixed-query conditions. Utility is $\times 100$, mean $\pm$ sample SD over 3 seed means. $\bar{K}$ is the mean identified candidate count.

Method	$\bar{K}$	Utility
Ours	4.88	$\mathbf{81.16 \pm 1.15}$
w/o gradient calibration	4.88	79.17 ± 2.10
w/o common-space removal	3.03	79.36 ± 2.28

4.3.2 Component ablations

We ablate identification and assembly on LLM and CV/NLP mixed-dataset settings (Sections 4.1.1 and 4.1.2), matching candidate counts for identification and fixing candidate sets for assembly. We additionally examine gradient calibration and common-space removal on LLM mixed queries.

Identification. On the 720 CV/NLP mixed-task runs, we compare our identification with cosine-similarity Top-K over the original full PAVEs, matching candidate counts per run. Table 1 compares the identification methods using a per-sample oracle, uniform ensembling, and our assembly with fixed settings to isolate the effect of candidate selection. Appendix D.4.3 details the controls; Table A22 in the appendix reports the corresponding LLM identification comparison.

Our identification yields higher mean oracle Utility in both modalities, suggesting better coverage of complementary capabilities. With the same gradient-guided assembly, the gains over Top-K widen to 5.81 and 1.08 points for CV and NLP. On NLP, despite similar oracle Utility, uniform ensembling slightly favors Top-K candidates, while gradient-guided assembly performs better with ours. This contrast highlights the interaction between identification and reuse: gradient-guided assignment can exploit complementary strengths that uniform ensembling leaves underused.

Assembly. On the same CV/NLP mixtures, we fix the identified candidates and their order and compare our assembly with supervised MLP and Linear selectors and uniform ensembling. The selectors share our labeled user data and input features and additionally use candidate predictions; training settings are in Appendix D.4.3. Under our identification in Table 1, our assembly achieves higher mean Utility than all three alternatives in both modalities, supporting gradient–PAVE guidance from limited labeled data while invoking one candidate per input. Table A22 in the appendix reports the LLM assembly comparison with majority voting over the same candidates.

Gradient calibration and common-space removal. Across 72 LLM mixed-query conditions, we separately disable calibration in assembly and common-space removal in identification. The former uses raw descent gradients with candidates fixed; the latter reruns identification with the same budget and explainability threshold (Appendix D.4.2). The complete method gains 2.00 and 1.80 mean Utility points, respectively (Table 2). Improvements under both balanced and long-tail proportions support calibrated sample–PAVE matching and shared-direction removal.

5 Related Work

Our work contributes to the learnware paradigm by identifying complementary models and assigning inputs according to their local strengths. It supports model routing in growing repositories, connects task arithmetic to complementary model identification, and applies gradient representations to sample assignment. Reusable specifications enable this identification without evaluating every candidate on each target task. Appendix A provides a detailed discussion of the related work.

Learnware paradigm and specifications. The learnware paradigm pairs models with specifications for identification and reuse (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026), as implemented in Beimingwu (Tan et al., 2024b). RKME characterizes training data distributions (Wu et al., 2023), with extensions for evolving capabilities (Liu et al., 2024), differing feature and label spaces (Tan et al., 2024a; 2023; Guo et al., 2023; Liu et al., 2025b), and improved specification matching and search (Chen et al., 2025; Xie et al., 2023; Liu et al., 2025a; 2026a). PAVE represents capabilities in parameter space (Shi et al., 2026; Tan et al., 2025), while constructive specifications organize black-box capabilities hierarchically (Liu et al., 2026b). Theoretical studies examine information retention and specification privacy (Lei et al., 2024; 2026a;b). Building on the PAVE specification, we further develop methods for compositional learnware identification and gradient-guided assembly.

Model routing and batch-level decisions. Benchmark studies characterize different model strengths and support cross-dataset routing (Hu et al., 2024; Li et al., 2026; Shnitzer et al., 2023). Routing approaches include cost-aware cascades (Chen et al., 2023), preference- and representation-based selection (Ong et al., 2024; Chen et al., 2024; Zhuang et al., 2025; Jitkrittum et al., 2025), and relational or workflow-aware decisions (Feng et al., 2025; 2026). Batch-level methods coordinate assignments under capacity constraints and performance uncertainty (Mei et al., 2025; Markovic-Voronov et al., 2026). Our method scales to growing model repositories by incorporating new models via PAVE specifications, supporting workload-level identification and sample-level assembly.

Parameter vectors and task arithmetic. Parameter updates, often termed task vectors, support multi-task model merging (Ilharco et al., 2023; Ortiz-Jiménez et al., 2023). Task Vector Bases compresses task vectors, while Task Singular Vectors exploits spectral structure to reduce interference (Zeng et al., 2025; Gargiulo et al., 2025). In contrast, we use parameter vectors as specifications and identify complementary models via sparse combinations that jointly explain user requirements.

Per-sample gradient representations. Fisher kernels represent samples with log-likelihood gradients (Jaakkola & Haussler, 1998), while Task2Vec and Grad2Task derive task representations from Fisher information or gradients (Achille et al., 2019; Wang et al., 2021). Gradient-based methods further support data attribution and selection, including influence functions, TracIn, TRAK, GraNd, BADGE, LESS, and LoGra, with LESS incorporating optimizer-aware matching and LoGra structured projection (Koh & Liang, 2017; Pruthi et al., 2020; Park et al., 2023; Paul et al., 2021; Ash et al., 2020; Xia et al., 2024; Choe et al., 2025). We similarly represent samples with gradients, using them to cluster data for specification construction and guide sample-to-model assignment.

6 Conclusion and Discussions

We presented a PAVE-based method for compositional learnware identification and gradient-guided assembly. Compositional identification selects complementary learnwares through sparse reconstruction of the user PAVE, while gradient-guided assembly matches calibrated sample gradients to the selected models' specifications. The method accommodates unlabeled query batches and tasks with limited labeled data without training a parametric selector. Experiments on mixed and single-dataset workloads, together with controlled ablations, support the value of both stages of our method. Future work will investigate cost-aware learnware identification and sample assignment, jointly considering predictive utility, computational overhead, and model invocation costs.

AI USE STATEMENT

In this work, we used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods, assist with translation, support qualitative and thematic data analysis and interpret results. We have not used generative AI tools to generate synthetic data sets, propose or refine hypotheses, or clean and reformat datasets. Additionally, we used generative AI tools to formulate questions for surveys or interviews, create or modify scientific figures or images, suggest experimental parameters, create or edit software code, draft parts of a research paper, transcribe recordings of research material, summarize or analyse existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, edit a research paper to improve readability and identify relevant literature. We have reviewed all AI-assisted work. We check mathematical formulations and derivations, reconcile reported results with experiment records, and test any AI-assisted code, among other verification steps. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. Task2Vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6430–6439, 2019.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv:1905.13319*, 2019.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *The 8th International Conference on Learning Representations (ICLR)*, 2020.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 632–642, 2015.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv:2305.05176*, 2023.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 66305–66328, 2024.

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

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff G. Schneider, Eduard H. Hovy, Roger Baker Grosse, and Eric P. Xing. What is your data worth to GPT? LLM-scale data valuation with influence functions. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 145944–145985, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for LLM selections. In *The 13th International Conference on Learning Representations (ICLR)*, 2025.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. Graphplanner: Graph memory-augmented agentic routing for multi-agent LLMs. In *The 14th International Conference on Learning Representations (ICLR)*, 2026.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18695–18705, 2025.

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 (ICML)*, pp. 12122–12131, 2023.

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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv:2103.03874*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The 10th International Conference on Learning Representations (ICLR)*, 2022.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv:2403.12031*, 2024.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The 11th International Conference on Learning Representations (ICLR)*, 2023.

Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11 (NeurIPS)*, pp. 487–493, 1998.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing*, pp. 2567–2577, 2019.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient LLM inference. *arXiv:2502.08773*, 2025.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations (ICLR)*, 2015.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1885–1894, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 4158–4169, 2019.

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 (NeurIPS)*, pp. 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 (ICML)*, 2026a.

Hao-Yi Lei, Jin-Hui Wu, Zhi-Hao Tan, and Zhi-Hua Zhou. On the privacy-preserving capabilities of PAVE specifications in learnware. In *Proceedings of the 35th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4465–4473, 2026b.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. Llmrouterbench: A massive benchmark and unified framework for LLM routing. In *Findings of the 64th Association for Computational Linguistics (ACL)*, pp. 37733–37754, 2026.

Hai-Tian Liu, Peng Tan, Jian-Dong Liu, Zhi-Hao Tan, and Zhi-Hua Zhou. Integrated learnware identification and reuse via reusability-aware metric learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD)*, pp. 3187–3197, 2026a.

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 (AAAI)*, pp. 13909–13917, 2024.

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

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

Jian-Dong Liu, Zi-Chen Zhao, Hao Sun, Lin-Xing Wu, Huan Zhang, Pengyuan Wang, Ming Zhao, Xinyu Chu, Shu Yan, Yongbei Zhu, Weijun Zhong, Zhi-Hao Tan, Jing Shang, Yang Yu, and Zhi-Hua Zhou. Constructive specification for plug-and-play learnware agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD)*, pp. 3222–3233, 2026b.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796, 2014.

Jelena Markovic-Voronov, Kayhan Behdin, Yuanda Xu, Zhengze Zhou, Zhipeng Wang, and Rahul Mazumder. Robust batch-level query routing for large language models under cost and capacity constraints. In *Proceedings of the ACM Conference on AI and Agentic Systems (CAIS)*, pp. 890–916, 2026.

Kai Mei, Wujiang Xu, Minghao Guo, Shuhang Lin, and Yongfeng Zhang. Omnirouter: Budget and performance controllable multi-llm routing. *ACM SIGKDD Explorations Newsletter*, 27(2): 107–116, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv:2406.18665*, 2024.

Guillermo Ortiz-Jiménez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pp. 66727–66754, 2023.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, pp. 248–260, 2022.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 27074–27113, 2023.

Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, 1993.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 20596–20607, 2021.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, volume 33, pp. 19920–19930, 2020.

Hao-Yu Shi, Zhi-Hao Tan, Zi-Chen Zhao, Yang Yu, and Zhi-Hua Zhou. A study on PAVE specification for learnware. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv:2309.15789*, 2023.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. Handling learnwares developed from heterogeneous feature spaces without auxiliary data. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 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 (NeurIPS)*, pp. 12767–12795, 2024a.

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 (KDD)*, pp. 5773–5782, 2024b.

Zhi-Hao Tan, Zi-Chen Zhao, Hao-Yu Shi, Xin-Yu Zhang, Peng Tan, Yang Yu, and Zhi-Hua Zhou. Learnware of language models: Specialized small language models can do big. *arXiv:2505.13425*, 2025.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *The 7th International Conference on Learning Representations (ICLR)*, 2019.

Jixuan Wang, Kuan-Chieh Wang, Frank Rudzicz, and Michael Brudno. Grad2Task: Improved few-shot text classification using gradients for task representation. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 6542–6554, 2021.

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 (TKDE)*, 35(1): 699–710, 2023.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 54104–54132, 2024.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3485–3492, 2010.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, Yijing Xu, Haoqiang Kang, Ziyan Kuang, Chenhan Yuan, Kailai Yang, Zheheng Luo, Tianlin Zhang, Zhiwei Liu, Guojun Xiong, Zhiyang Deng, Yuechen Jiang, Zhiyuan Yao, Haohang Li, Yangyang Yu, Gang Hu, Jiajia Huang, Xiao-Yang Liu, Alejandro Lopez-Lira, Benyou Wang, Yanzhao Lai, Hao Wang, Min Peng, Sophia Ananiadou, and Jimin Huang. The finben: An holistic financial benchmark for large language models. *arXiv:2402.12659*, 2024.

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 (ECAI)*, pp. 2752–2759, 2023.

Siqi Zeng, Yifei He, Weiqiu You, Yifan Hao, Yao-Hung Hubert Tsai, Makoto Yamada, and Han Zhao. Efficient model editing with task vector bases: A theoretical framework and scalable approach. *arXiv:2502.01015*, 2025.

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.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. In *The 13th International Conference on Learning Representations (ICLR)*, 2025.

A Extended Related Work

This appendix expands the connections summarized in Section 5, covering learnware specifications, model selection, compositional parameter representations, and sample gradients.

Learnware Paradigm and Specification. The learnware paradigm uses specifications to identify and reassemble well-trained models without access to their original training data, enabling system-level capabilities to emerge through model reuse (Zhou, 2016; Zhou & Tan, 2024; Zhou, 2026). A specification characterizes a model's capabilities for future identification; reduced kernel mean embedding (RKME) specifications approximate kernel mean embeddings of models' training data distributions with a small set of weighted synthetic points, supporting identification by matching these distributions to user data (Wu et al., 2023). On the theoretical side, the information about developers' training data retained by RKME specifications has been analyzed (Lei et al., 2024). Lei et al. (2026a) further formulate learnware-inversion as statistical decision games, measuring specification risk by the increase in optimal Bayes inference value beyond observing the model alone. For RKME, they derive risk bounds through an RKHS-smoothed total-variation bridge and reduced-set stability analysis, characterizing conditions under which specification size controls additional privacy risk while preserving identification utility.

Methodological work extends capability characterization and identification: evolvable specifications use task information from other hosted learnwares to characterize capabilities beyond a model's original task and build indexes for efficient identification (Liu et al., 2024). To handle heterogeneous feature spaces, conditional-distribution matching extends specifications beyond marginal distributions (Tan et al., 2024a), while another approach works without auxiliary data (Tan et al., 2023). Other work addresses efficient identification through anchors (Xie et al., 2023), dynamic filtering of redundant learnwares (Liu et al., 2025a), identification across heterogeneous label spaces (Guo et al., 2023), and reuse across different label spaces (Liu et al., 2025b). Dual alignment combines distribution alignment with discriminative performance when constructing specifications (Chen et al., 2025). Reusability-aware metric learning evaluates how hosted learnwares perform when reused for one another's tasks and learns specification distances that reflect expected reuse performance (Liu et al., 2026a). At the system level, Beimingwu provides a learnware dock for model submission, identification, and reuse (Tan et al., 2024b).

For high-dimensional inputs, PAVE encodes model capabilities through parameter updates from a shared pre-trained model (Shi et al., 2026). Its privacy properties are studied theoretically by Lei et al. (2026b), who distinguish specification disclosure from amplification of attacks against the released model. They establish intrinsic differential privacy for compact PAVE releases under an unreleased Gaussian sketch, a fixed specification subspace, and covariance stability, with the cumulative update independent of the realized sketch. Conditional privacy guarantees then bound both disclosure and specification-side amplification risks, relating privacy to identification utility. Language-model learnwares extend the paradigm to specialized small language models (Tan et al., 2025). Constructive specification builds hierarchical capability representations from black-box performance profiles (Liu et al., 2026b). Our method uses PAVE specifications to identify a compact set of complementary models through joint reconstruction of user requirements and then assigns samples using gradient–PAVE matching.

Model Routing and Batch-Level Decisions. Model routing predicts which candidate should handle an input, often using past query–model outcomes or preferences. Benchmark-based routing learns from performance on other datasets even when a new task has no labels (Shnitzer et al., 2023); RouterBench and LLMRouterBench document variation in model strengths and provide common settings for comparing routing policies (Hu et al., 2024; Li et al., 2026). FrugalGPT uses model cascades to balance prediction quality and inference cost (Chen et al., 2023). RouteLLM learns a strong-versus-weak model decision from preference data (Ong et al., 2024), while RouterDC learns query and model representations with contrastive objectives (Chen et al., 2024). EmbedLLM learns reusable model embeddings from observed behavior (Zhuang et al., 2025), and UniRoute represents models through responses to representative prompts to handle previously unseen candidates (Jitkrittum et al., 2025). GraphRouter models task–query–model interactions for contextual selection, and GraphPlanner incorporates workflow history for agentic settings (Feng et al., 2025; 2026). At the collection level, OmniRouter coordinates predicted query–model quality with cost and concurrency

constraints (Mei et al., 2025); robust batch routing adds uncertainty-aware performance estimates and model-instance allocation (Markovic-Voronov et al., 2026). Our method uses the input collection to identify a small set of complementary capability specifications, then assigns individual samples within that set. This addresses a different decision from serving-capacity optimization; evaluations should match information access and account for specification construction, assignment, and model-invocation costs.

Task Arithmetic and Spectral Model Analysis. Task arithmetic combines parameter differences from a common initialization to edit or merge model functions (Ilharco et al., 2023); its tangent-space analysis identifies conditions for disentangled functional effects (Ortiz-Jiménez et al., 2023). Task Vector Bases studies compact basis representations for task arithmetic (Zeng et al., 2025). Task Singular Vectors analyzes layer-wise task matrices through SVD and uses their geometry to reduce interference in model merging (Gargiulo et al., 2025). We apply the compositional perspective to fixed PAVE specifications for identifying independently callable candidates. Our SVD acts on a matrix of query-matched full PAVEs across models. The spectral viewpoint motivates examining shared directions, whose effect on identification is tested through ablation.

Per-Sample Gradients and Gradient-Based Representations. Gradients describe how an example affects a model's fitting objective and have long served as learning-related representations. Fisher kernels use score gradients to define sample features, while Task2Vec and Grad2Task aggregate gradient-derived information into task representations (Jaakkola & Haussler, 1998; Achille et al., 2019; Wang et al., 2021). Influence functions and TracIn use gradient interactions to assess training-example effects, with TRAK scaling attribution through projected gradient features (Koh & Liang, 2017; Pruthi et al., 2020; Park et al., 2023). For data selection, GraNd ranks examples by gradient norm, BADGE selects diverse gradient embeddings, and LESS matches target gradients with optimizer-aware update features to identify relevant instruction-tuning examples (Paul et al., 2021; Ash et al., 2020; Xia et al., 2024). LoGra reduces the cost of per-example gradient projections (Choe et al., 2025). These studies support using gradients to express local fitting requirements, while their attribution and selection objectives differ from model assembly. We cluster model-side fitting records by gradient similarity, independently fit full and cluster PAVEs, and compare calibrated user gradients with these specifications to assign candidate models. Our analysis treats the resulting similarity as a query against a model's fitting distribution under an explicit parameter-update approximation.

B ALGORITHMIC DETAILS

We use the notation of Section 3.1: τ_u is the user PAVE, $\tau_{m,j}$ a model PAVE indexed by $j \in \mathcal{J}_m$, and $\mathcal{C} \subseteq \mathcal{M}$ the identified model set.

B.1 SPECIFICATION CONSTRUCTION

All full and cluster PAVEs below are fitted independently from the shared initialization θ_0, using the hierarchical organization in Section 3.1.

B.1.1 BATCH-QUERY SCENARIO: BLACK-BOX LLMS

For a black-box LLM f_m accessed through an API, model-side evidence is its *performance profile* on each system-maintained public dataset D, as in constructive specification (Liu et al., 2026b):

$$\mathcal{P}_{f_m}(D) = \{(x, y) \in D \mid Q(f_m(x), y) \geq \delta\},$$

where $Q(f_m(x), y) \in [0, 1]$ measures response quality and $\delta \in (0, 1]$ is the success threshold. Reference answers and model responses determine profile membership; PAVE fitting uses only the input texts. For input tokens $x = (x_1, \ldots, x_{T_x})$, the shared reference model uses the autoregressive loss

$$\mathcal{L}_{\mathrm{AR}}(x; \theta) = -\sum_{t=1}^{T_x} \omega_{x,t} \log h(x_t \mid x_{<t}; \theta), \tag{11}$$

where $\omega_{x,t} \geq 0$ specifies token masking and loss reduction. Inputs include the supplied context and answer options, but exclude reference answers and model responses.

We partition $\mathcal{P}_{f_m}(D)$ by K-Means on per-example gradients of $\mathcal{L}_{\mathrm{AR}}$ at θ_0, obtaining subsets $\mathcal{P}_{f_m}(D, c)$. Fitting the inputs of the full profile and each subset yields $\tau_m(D)$ and $\tau_m(D, c)$, respectively. Together, these vectors characterize the input regions where f_m meets the quality threshold.

The user supplies an unlabeled query collection $\mathcal{U}$. Fitting the same reference model on these inputs with $\mathcal{L}_{\mathrm{AR}}$ yields $\tau_u = \tau(\mathcal{U})$; the same loss provides the sample gradients for assembly, without requiring user labels.

B.1.2 TASK SCENARIO: DEVELOPER-SUBMITTED MODELS

For a developer-submitted model f_m trained on D_m^{train}, the developer constructs model-side evidence from its predictions on this training set: $D_m^{\mathrm{pred}} = \{(x, f_m(x)) \mid (x, y) \in D_m^{\mathrm{train}}\}$. The shared reference model fits these prediction targets using $\mathcal{L}(h(x; \theta), f_m(x))$, characterizing the submitted model's behavior. K-Means on the per-example fitting-loss gradients at θ_0 partitions these records; fitting the full set and each subset produces the corresponding full and cluster PAVEs. The developer pairs these specifications with the trained model to form a learnware. Appendix D.1.2 describes the experimental data allocation.

The user provides a small labeled set D_u^l. Fitting the same reference model to its ground-truth labels with the target-task loss yields $\tau_u = \tau(D_u^l)$. This supervised loss also supplies the labeled-sample gradients used to construct the assembly teacher; no labels are required for subsequent test inputs.

B.2 Compositional Learnware Identification

Algorithm 1 details the procedure for Eq. (4).

Choosing full PAVEs for the common subspace. A black-box LLM may have a full PAVE for each of several profiled datasets. For each eligible model m, we compare τ_u with all available full PAVEs $\tau_m(D)$, $D \in \mathcal{D}_m$, before common-direction removal, and select the dataset D_m^* with the largest cosine similarity. This comparison uses ordinary cosine, not the absolute correlation used during OMP. In the task scenario, $\mathcal{D}_m$ contains only the model-prediction dataset, so its full PAVE is used directly.

The selected full PAVEs are unit-normalized and stacked as rows of F_{full}. Uncentered SVD then yields the common basis C, which remains fixed throughout pursuit for this query. This preliminary choice only determines each model's contribution to common-space estimation: it neither selects models for assembly nor restricts their reconstruction representatives. All available full and cluster PAVEs remain eligible for OMP after projection.

Algorithm 1 Multi-Learnware Identification via OMP

Require: User PAVE τ_u, eligible learnwares $\{(f_m, \mathcal{S}_m(\mathcal{D}_m))\}_{m\in\mathcal{M}}$, budget $1 \le K \le |\mathcal{M}|$, common rank r_c
Require: Optional threshold $\rho \in (0, 1]$ and reference explainability $E_{\max} > 0$
Ensure: Model-index set $\mathcal{C}$ and corresponding representatives $\{(j_m, \beta_m)\}_{m\in\mathcal{C}}$

for each $m \in \mathcal{M}$ **do**
  Choose $D_m^* \in \mathcal{D}_m$ maximizing $\cos(\tau_u, \tau_m(D))$ among available full PAVEs
  $\widehat{\tau}_m^{\text{full}} \leftarrow \tau_m(D_m^*)/\|\tau_m(D_m^*)\|_2$
end for
Stack $(\widehat{\tau}_m^{\text{full}})^\top$ as rows of F_{full}; let C contain its leading r_c right singular vectors
$P_\perp \leftarrow I - CC^\top, \quad \tau_u^\perp \leftarrow P_\perp \tau_u$
$\widehat{\tau}_{m,j}^\perp \leftarrow P_\perp \tau_{m,j}/\|P_\perp \tau_{m,j}\|_2$ for each nonzero projected PAVE
$\mathcal{C}_0 \leftarrow \varnothing, \quad B_0 \leftarrow [\,], \quad r_0 \leftarrow \tau_u^\perp$
for $t = 1, \ldots, K$ **do**
  $(m_t, j_t) \leftarrow \underset{\substack{m\in\mathcal{M}\setminus\mathcal{C}_{t-1}\\ j\in\mathcal{J}_m}}{\arg\max} |r_{t-1}^\top \widehat{\tau}_{m,j}^\perp|$
  $j_{m_t} \leftarrow j_t, \quad B_t \leftarrow [B_{t-1}, \widehat{\tau}_{m_t,j_t}^\perp], \quad \mathcal{C}_t \leftarrow \mathcal{C}_{t-1} \cup \{m_t\}$
  $\beta^{(t)} \leftarrow B_t^\dagger \tau_u^\perp, \quad r_t \leftarrow \tau_u^\perp - B_t \beta^{(t)}$
  $T \leftarrow t$; compute the explained fraction E_t using Eq. (12)
  if threshold enabled and $E_t \ge \rho E_{\max}$ **then**
    break
  end if
end for
$\mathcal{C} \leftarrow \mathcal{C}_T, \quad \beta_{m_\ell} \leftarrow \beta_\ell^{(T)}$ for $\ell = 1, \ldots, T$
return $\mathcal{C}$, $\{(j_m, \beta_m)\}_{m\in\mathcal{C}}$

Reconstruction representatives and returned learnwares. Once OMP selects a representative j_m, it remains fixed; subsequent least-squares fits update only the coefficients of the selected representatives. The returned $\mathcal{C}$ identifies the callable learnwares $\{(f_m, \mathcal{S}_m(\mathcal{D}_m))\}_{m\in\mathcal{C}}$, with (j_m, β_m) recording each one's reconstruction contribution. Assembly receives their complete, unprojected specifications, rather than only the representatives used by OMP.

Optional stopping. For a nonzero projected query, the explained fraction is

$$E_t = 1 - \frac{\|r_t\|_2^2}{\|\tau_u^\perp\|_2^2}. \tag{12}$$

Here $E_{\max}$ is a reference explainability computed in the same projected space, not necessarily the optimal reconstruction value. When enabled, stopping returns the first greedy prefix satisfying $E_t \ge \rho E_{\max}$, or K models if the threshold is not reached.

B.3 GRADIENT-GUIDED ASSEMBLY

Algorithm 2 assembles the models returned by compositional identification. Gradient computation and calibration use the unlabeled query collection in the batch-query scenario and labeled user data in the task scenario. In the task scenario, we use shared encoder features to transfer gradient-derived assignment preferences from labeled examples to unlabeled inputs.

Calibration scope. The evidence collection $\mathcal{E}$ in Algorithm 2 supplies gradients at θ_0 and one second-moment vector v, shared across all sample–candidate comparisons. Scaling acts only on sample gradients; model PAVEs are unchanged. Unlabeled task inputs do not contribute labels or gradients to calibration and are assigned through teacher transfer.

Algorithm 2 Gradient-Guided Model Assembly

Require: Identified models $\{f_m\}_{m\in\mathcal{C}}$, complete PAVEs $\{\tau_{m,j}\}_{m\in\mathcal{C},j\in\mathcal{J}_m}$, user PAVE τ_u, inputs $\mathcal{U}$
Require: Reference model $h(\cdot;\theta_0)$, scoring settings, auxiliary evidence when used, $\epsilon > 0$
Require: For tasks: labeled user data D_u^l, encoder φ, and teacher-transfer settings
Ensure: One model assignment $a(x) \in \mathcal{C}$ and prediction $\hat{y}(x)$ for each $x \in \mathcal{U}$
if batch-query scenario **then**
  $\mathcal{E} \leftarrow \mathcal{U}$; use input-only autoregressive fitting losses
else
  $\mathcal{E} \leftarrow D_u^l$; use supervised fitting losses
end if
$n \leftarrow |\mathcal{E}|$; compute $g_i \leftarrow \nabla_\theta \mathcal{L}_i(\theta)|_{\theta=\theta_0}$ for each record in $\mathcal{E}$
$v \leftarrow n^{-1}\sum_{i=1}^n g_i \odot g_i, \quad \tilde{g}_i \leftarrow -g_i \oslash \sqrt{v+\epsilon}$
$c_{i,m} \leftarrow \max_{j\in\mathcal{J}_m} \cos(\tilde{g}_i, \tau_{m,j})$ for $i = 1,\dots,n$ and $m \in \mathcal{C}$
Compute $F_{i,m}$ from matching and enabled auxiliary signals using Eq. (13)
if batch-query scenario **then**
  $a(x_i) \leftarrow \arg\max_{m\in\mathcal{C}} F_{i,m}$ for each $x_i \in \mathcal{U}$
else
  Form labeled-sample teacher distributions $P_{i,m}^l$ using Eq. (19)
  Extract $\varphi(x)$ for inputs from the labeled user data and all $x \in \mathcal{U}$
  Construct teacher-weighted prototypes μ_m using Eq. (20)
  for each $x \in \mathcal{U}$ **do**
    Compute prototype probabilities $P_m^{\text{proto}}(x)$ from feature–prototype similarities
    Propagate teacher distributions from the κ nearest labeled features to obtain $P_m^{\text{knn}}(x)$
    Geometrically fuse both branches into $P_m^{\text{PKNN}}(x)$ using Eq. (21)
    $a(x) \leftarrow \arg\max_{m\in\mathcal{C}} P_m^{\text{PKNN}}(x)$
  end for
end if
Obtain $\hat{y}(x) \leftarrow f_{a(x)}(x)$ with one candidate invocation per $x \in \mathcal{U}$
return $\{(a(x), \hat{y}(x)) \mid x \in \mathcal{U}\}$

B.3.1 COMBINING MATCHING AND AUXILIARY SCORES

Given the local matches $c_{i,m}$ from Eq. (7), we combine normalized matching and optional auxiliary signals into a utility score:

$$F_{i,m} = w_c \widehat{c}_{i,m} + \sum_r w_r \widehat{s}_{i,m}^{(r)}. \tag{13}$$

Here $\widehat{c}_{i,m}$ is the normalized gradient–PAVE match and $\widehat{s}_{i,m}^{(r)}$ denotes an auxiliary signal. Direct assembly selects the highest-scoring candidate:

$$a_i = a(x_i) = \arg\max_{m\in\mathcal{C}} F_{i,m}, \qquad \hat{y}_i = f_{a_i}(x_i). \tag{14}$$

Identity normalization with $w_c = 1$ and auxiliary weights set to 0 recovers direct gradient–PAVE matching, $F_{i,m} = c_{i,m}$. A standardized instantiation combines 4 signals:

$$F_{i,m} = w_R R_{i,m} + w_S S_{i,m} + w_G G_m + w_Q Q_{i,m}. \tag{15}$$

Here $R_{i,m}$ supplies $\widehat{c}_{i,m}$, with $w_R = w_c$; the other terms instantiate the auxiliary scores. Let Z subtract a vector's mean and divide by its population standard deviation plus ϵ, yielding 0 for a

constant vector. Relative fit and sample specificity are

$$R_{i,m} = Z(c_{i,:})_m, \qquad S_{i,m} = Z(c_{:,m})_i. \tag{16}$$

Relative fit compares candidates for one sample; specificity compares a model's fit to that sample against its fit across the available collection.

Global fit compares the user specification with each candidate's full PAVEs:

$$G_m = Z\left(\left[\max_{D \in \mathcal{D}_n} \cos(\tau_u, \tau_n(D))\right]_{n \in \mathcal{C}}\right)_m. \tag{17}$$

Only available full PAVEs enter this maximum. This uses their original orientation and retains common directions.

If independent profile-quality measurements are available, let $D^*_{i,m}$ be the dataset associated with a maximizer $j^*_{i,m}$ in Eq. (7). The quality signal is

$$Q_{i,m} = Z\Big(\big[\mathrm{Acc}(f_n, D^*_{i,n})\big]_{n \in \mathcal{C}}\Big)_m. \tag{18}$$

Accuracy is measured on separate benchmark records containing both successes and failures, excluding user evaluation queries. Cross-benchmark difficulty remains a potential confound; without suitable measurements, the quality term is omitted.

Weights are nonnegative. Relative scoring alone preserves the direct-cosine argmax, although it changes softmax probabilities at a fixed temperature. These signals rank candidates by their estimated compatibility with the sample.

B.3.2 Transferring a Task-Specific Assembly Teacher

In the batch-query scenario, Eq. (14) directly assigns unlabeled inputs using their input-only gradients. In the task scenario, gradients and scores are available only on labeled user examples D^l_u. Using the supervised fitting loss, we compute gradients and matches as in Section 3.3, combine them into $F_{i,m}$ via Eq. (13), and form a soft teacher:

$$P^l_{i,m} = \frac{\exp(F_{i,m}/T_v)}{\sum_{n \in \mathcal{C}} \exp(F_{i,n}/T_v)}, \qquad T_v > 0. \tag{19}$$

These targets encode gradient-derived preferences; they do not use candidate predictions on the labeled user examples.

Let $\varphi(x)$ denote a shared base-encoder feature, distinct from the specification model $h(\cdot;\theta)$, and write $\varphi_i = \varphi(x_i)$. Each candidate has a teacher-weighted feature prototype

$$\mu_m = \mathrm{norm}\left(\frac{\sum_i P^l_{i,m}\varphi_i}{\sum_i P^l_{i,m}}\right), \tag{20}$$

where norm denotes unit normalization. For an unlabeled input x, the prototype branch applies a softmax to $\cos(\varphi(x), \mu_m)/T_p$ across candidates, giving $P^{\mathrm{proto}}_m(x)$. The neighbor branch finds the κ nearest labeled features under cosine similarity, weights them by a softmax of similarity divided by T_k, and averages their teacher distributions to obtain $P^{\mathrm{knn}}_m(x)$. The neighbor count κ and model budget K are separate hyperparameters.

The branches are combined by normalized geometric fusion:

$$P^{\mathrm{PKNN}}_m(x) \propto \big(P^{\mathrm{knn}}_m(x)\big)^{\beta_{\mathrm{knn}}} \big(P^{\mathrm{proto}}_m(x)\big)^{1-\beta_{\mathrm{knn}}}, \qquad \beta_{\mathrm{knn}} \in [0,1]. \tag{21}$$

The fusion weight is distinct from the reconstruction coefficients β_m. Using $\log P^{\mathrm{PKNN}}_m(x)$ as the test-time utility in Eq. (14) selects one model $a(x) \in \mathcal{C}$. Test assembly uses the shared features and the fitted teacher-transfer rule.

C SUPPLEMENTARY ANALYSIS OF GRADIENT–PAVE MATCHING

This appendix extends the analysis in Section 3.4.

C.1 SCALE BALANCING AND VIRTUAL UPDATES

Raw gradient matching can be dominated by coordinates with consistently large magnitudes. Calibration expresses each coordinate relative to its typical root-mean-square scale in the user collection, incorporating collection-level context into per-sample matching. Equation (6) gives each calibrated coordinate an empirical second moment of $v_j/(v_j+\epsilon)$, approaching one when $v_j \gg \epsilon$. This balances coordinate contributions and emphasizes relative fitting requirements. For negative log-likelihood gradients, v is the empirical Fisher diagonal, related to Fisher-based task summaries (Achille et al., 2019), with additional assumptions needed for equivalence to the model Fisher or Hessian (Kunstner et al., 2019).

From an optimization perspective, this scaling is analogous to Adam's coordinate-wise adaptation (Kingma & Ba, 2015) and defines a virtual parameter update whose step sizes are determined by the user collection. The calibrated negative gradient $\tilde{g}_i = -H_u g_i$ thus represents a sample's fitting requirements in a task-conditioned update geometry. Matching its direction with a PAVE compares this virtual update with a model-side parameter-update representation.

C.2 FROM CALIBRATED KERNEL QUERIES TO LOCAL MATCHING

For a fixed user collection, H_u is positive diagonal. The feature map $\phi_u(z) = -H_u^{1/2} g(z)$ gives $k_u(z, z') = \langle \phi_u(z), \phi_u(z') \rangle$, establishing positive semidefiniteness. Under Eq. (8), $H_u^{1/2}\tau_{m,j}/\eta_{m,j}$ approximates the empirical mean of these features over $D_{m,j}$, yielding the kernel-query interpretation in Eq. (10).

For nonzero gradients and PAVEs, Eq. (7) becomes

$$c_{i,m} \approx \max_{j \in \mathcal{J}_m} \frac{\eta_{m,j} \sum_{z \in D_{m,j}} k_u(z_i, z)}{|D_{m,j}| \, \|H_u g_i\|_2 \, \|\tau_{m,j}\|_2}. \tag{22}$$

Cosine normalization compares the directions of the calibrated sample update and the PAVE, while the maximum selects the best-matching full or cluster specification. Under the local approximation, a higher $c_{i,m}$ therefore suggests stronger alignment between the sample's fitting requirements and the capability-related distribution summarized by the matched specification.

D EXPERIMENTAL SETTINGS

D.1 DATASETS, COMPARED METHODS, AND METHOD CONFIGURATION

Table A3 summarizes the user evidence, candidate models, and adaptation protocols for the 2 scenarios.

Table A3: Evidence and prediction protocols. Profiling and labeled user data provide separate forms of evidence.

Setting	User evidence	Candidate eligibility	Target adaptation
Black-box LLMs	Unlabeled query inputs	Profiled 10-model library	None
CV/NLP single task	Labeled user data	Target-trained models excluded	Few-shot adaptation
CV/NLP mixtures	Pooled labeled user data	Component models retained	Frozen encoders with target-compatible heads

D.1.1 BATCH-QUERY SCENARIO: BLACK-BOX LLMS

We construct a system of 10 black-box LLMs accessed through APIs: GPT-3.5-Turbo, Gemini-2.5-Flash, DeepSeek-V3.2, Qwen3 Coder Flash, Qwen Long, Qwen3 Next 80B A3B Instruct, Qwen3

30B A3B Instruct (2507), Qwen Plus, Qwen Max, and Doubao Seed 1.6 Flash (250828). Both single-dataset and mixed-query results use this fixed system.

Table A4 summarizes the in-system and out-of-system benchmarks across mathematics, healthcare, and finance.

Table A4: Black-box LLM benchmarks: 8 in-system and 6 out-of-system evaluation datasets.

Role	Domain	Datasets
In-system	Mathematics	GSM8K, MATH, MathQA
In-system	Healthcare	MedMCQA, PubMedQA, MedQA
In-system	Finance	FPB, FiQA-SA
Out-of-system	Mathematics	MMLU-Pro (Math)
Out-of-system	Healthcare	MMLU-Pro (Health)
Out-of-system	Finance	Headlines, LendingClub, MultiFin, SC

In-System Benchmarks. We use 8 benchmarks across mathematics, healthcare, and finance. Their training splits support profiling and specification construction, while their test splits assess in-system performance.

- **GSM8K** (Cobbe et al., 2021): 8.5K high-quality grade-school math word problems, with 7,473 training and 1,319 test samples.
- **MATH** (Hendrycks et al., 2021): 12,500 challenging problems from high-school mathematics competitions, covering subjects such as algebra, geometry, and number theory, with 7,500 training and 5,000 test samples.
- **MathQA** (Amini et al., 2019): natural-language math problems paired with structured equation representations and final answers, with 29,837 training and 2,985 test samples.
- **MedMCQA** (Pal et al., 2022): medical multiple-choice questions from entrance examinations, covering 21 subjects and over 2.4K topics.
- **PubMedQA** (Jin et al., 2019): biomedical question answering over PubMed abstracts, with yes, no, or maybe answers.
- **MedQA**: medical licensing-examination questions assessing medical knowledge and clinical reasoning, using the 4-option configuration.
- **FPB** (Malo et al., 2014): financial sentiment classification over news and report sentences, with positive, neutral, and negative labels.
- **FiQA-SA** (Xie et al., 2024): financial sentiment analysis, evaluated with positive, neutral, and negative labels.

Out-of-System Benchmarks. To evaluate generalization, we use 6 held-out benchmarks from the same 3 domains but with different distributions and task characteristics. These benchmarks are reserved for out-of-system evaluation.

Mathematics Domain. We use the mathematics subset of MMLU-Pro:

- **MMLU-Pro (Math)**: multiple-choice problems assessing mathematical knowledge and reasoning.

Healthcare Domain. We use the health subset of MMLU-Pro:

- **MMLU-Pro (Health)**: multiple-choice questions assessing health-related knowledge and reasoning.

Finance Domain. We select 4 financial benchmarks with distinct task types from FinBen (Xie et al., 2024):

- **Headlines**: classify attributes of financial news headlines with yes/no decisions.

- **LendingClub**: classify credit risk from loan information.
- **MultiFin**: classify financial text into topic categories, using its English subset.
- **SC**: identify causal content in financial text using the FinCausal 2020 sentence-classification task.

Test Set Size. For single-dataset evaluation, we sample up to 1,000 queries per benchmark with seed 42, giving 12,069 queries across the 14 datasets. PubMedQA has 500, FPB 970, FiQA-SA 235, MMLU-Pro (Health) 818, and MultiFin 546; the other 9 datasets have 1,000 each.

Compared methods. The first 6 baselines make query-level model assignments. PAVE-Single uses one model for the batch, while Cluster-PAVE is a simple cluster-based comparison that selects one model per user-input subset.

- **Random**: select a candidate model uniformly for each query, providing a reference for the average capability of the system.
- **DescSim**: pair each model name with a natural-language capability summary, encode the descriptions and the user query in a shared embedding space, and select the model with the highest query–description similarity.
- **K-NN** (Shnitzer et al., 2023; Hu et al., 2024): retrieve the most similar in-system training queries and select the model with the highest average observed performance on these neighbors.
- **EmbedLLM** (Zhuang et al., 2025): learn model embeddings from the query–model performance matrix on in-system benchmarks. The learned query and model representations predict each candidate's performance on an incoming query and determine its assignment.
- **GraphRouter** (Feng et al., 2025): represent tasks, queries, and models in a heterogeneous graph, with observed query–model performance providing supervision. The graph-based predictor estimates candidate performance and selects a model for each query.
- **UniRoute (K-means)** (Jitkrittum et al., 2025): partition in-system prompt embeddings into clusters and estimate each model's mean error within each cluster. An incoming query is assigned to the lowest-error model associated with its nearest centroid. The centroids remain fixed across evaluation batches, and the cost penalty is set to 0.
- **PAVE-Single**: fit one input-only user PAVE to the complete query collection, compare it with each candidate's full PAVEs by cosine similarity, and select the highest-scoring model to answer the entire collection.
- **Cluster-PAVE**: a simple cluster-based baseline that applies K-means to frozen embeddings of the user inputs and constructs one input-only PAVE per cluster. Each cluster selects a model using the PAVE-Single rule. The number of user clusters is bounded by the candidate budget K, and different clusters may select the same model.

Learned baselines use outcomes from in-system profiling and development records. All methods are evaluated on the same held-out queries.

D.1.2 Task Scenario: CV/NLP

We construct separate systems of 12 CV models and 15 NLP models, with one developer-trained model per dataset. The CV models use a CLIP ViT-B/32 backbone, and the NLP models use a RoBERTa-base backbone.

Data splits and learnware construction. Each dataset is divided into a training split and a test split. The training split is further partitioned into developer data and labeled user data. For each classification dataset, we sample k labeled examples per class from the training split to form its k-shot user data pool, with $k = 2$ for CV and $k = 16$ for NLP. The STS-B regression pool contains 32 labeled examples, matching the number of labeled user examples in a binary NLP classification task. Single-dataset tasks use the complete corresponding pool as D_u^l. The remaining training examples are allocated to developers as D_m^{train}. Developers train f_m on D_m^{train}, construct its full and cluster PAVEs by fitting its predictions on D_m^{train}, and add the model together with these specifications to the system.

Dataset inventory and sampling groups. The 12 CV and 15 NLP datasets form fixed sampling groups based on task attributes (Table A5), chosen before inspecting mixture results.

Table A5: Fixed, mutually exclusive dataset groups for randomized mixture construction.

Track	Group	Datasets
CV	Symbol/Street	MNIST, SVHN, GTSRB
CV	Remote/Scene	EuroSAT, RESISC45, SUN397
CV	Generic Object	CIFAR10, CIFAR100, STL10
CV	Fine-grained/Appearance	Cars, Food101, DTD
NLP	Single-sentence	CoLA, SST-2, cr, mr, mpqa, subj, trec
NLP	Pair Similarity	MRPC, QQP, STS-B
NLP	Textual Inference	MNLI, SNLI, QNLI, RTE, WNLI

Computer vision. The 12 image-classification datasets cover digit recognition, scenes, objects, textures, and fine-grained categories.

- **MNIST** (Modified National Institute of Standards and Technology Database): 70,000 grayscale images of handwritten digits from 0 to 9, at a resolution of 28×28, for 10-class digit recognition.
- **SVHN** (Street View House Numbers): real-world house-number images from Google Street View. We use cropped digit images for 10-class recognition in natural scenes.
- **GTSRB** (German Traffic Sign Recognition Benchmark) (Stallkamp et al., 2012): more than 50,000 images covering 43 German traffic-sign categories.
- **EuroSAT** (Helber et al., 2019): 27,000 Sentinel-2 satellite images in 10 land-use and land-cover categories, including residential areas, forests, and industrial zones. We use the RGB configuration.
- **RESISC45** (Remote Sensing Image Scene Classification): 31,500 aerial images from 45 scene types, including airports, forests, and residential areas.
- **SUN397** (Scene UNderstanding) (Xiao et al., 2010): 108,754 images from 397 indoor and outdoor scene categories, used for scene recognition.
- **CIFAR10** (Krizhevsky & Hinton, 2009): 60,000 color images of size 32×32 from 10 object categories, including airplanes, cars, and animals.
- **CIFAR100** (Krizhevsky & Hinton, 2009): 60,000 color images from 100 fine-grained categories organized into 20 superclasses. Evaluation uses the 100 fine labels.
- **STL10**: 10-class object recognition with 96×96 color images. We use the labeled portion of the dataset.
- **Cars** (Stanford Cars): 16,185 images of 196 vehicle categories distinguished by make, model, and year, assessing fine-grained visual recognition.
- **Food101**: 101,000 images from 101 food categories, including dishes such as sushi, pizza, and spaghetti.
- **DTD** (Describable Textures Dataset): 5,640 texture images organized into 47 descriptive categories, such as cracked and striped.

Classic NLP. The English-language suite includes 9 GLUE tasks (Wang et al., 2019), SNLI (Bowman et al., 2015), and 5 sentence-classification datasets (Kim, 2014).

- **CoLA** (Corpus of Linguistic Acceptability): English sentences from linguistic literature, labeled for grammatical acceptability. The metric is Matthews correlation coefficient (MCC).
- **MNLI** (Multi-Genre Natural Language Inference): premise–hypothesis pairs from diverse text genres, with entailment, contradiction, and neutral labels.
- **MRPC** (Microsoft Research Paraphrase Corpus): sentence pairs annotated for semantic equivalence, evaluated with accuracy and F1.

- **QNLI** (Question Natural Language Inference): a reformulation of SQuAD as binary classification, determining whether a sentence contains the answer to a given question.
- **QQP** (Quora Question Pairs): pairs of Quora questions labeled by whether they express the same meaning, evaluated with accuracy and F1.
- **RTE** (Recognizing Textual Entailment): premise–hypothesis pairs from textual-entailment challenges, with binary entailment labels.
- **SNLI** (Stanford Natural Language Inference): caption-derived premise–hypothesis pairs labeled as entailment, contradiction, or neutral (Bowman et al., 2015).
- **SST-2** (Stanford Sentiment Treebank): movie-review sentences labeled with positive or negative sentiment.
- **STS-B** (Semantic Textual Similarity Benchmark): sentence pairs with semantic-similarity scores from 0 to 5, evaluated with Pearson and Spearman correlations.
- **WNLI** (Winograd Natural Language Inference): a binary inference task derived from the Winograd Schema Challenge, requiring commonsense reasoning about ambiguous pronoun references.
- **cr** (Customer Reviews): product reviews labeled with positive or negative sentiment.
- **mpqa** (Multi-Perspective Question Answering): an opinion-polarity benchmark derived from the MPQA corpus, evaluated as binary sentiment classification.
- **mr** (Movie Reviews): movie-review snippets labeled with positive or negative sentiment.
- **subj** (Subjectivity): sentences labeled as subjective expressions of opinion or objective statements of fact.
- **trec** (Text REtrieval Conference): questions assigned to 6 coarse types, including person, location, and number.

Classification performance is measured by accuracy, with the additional or alternative metrics specified above.

Compared methods.

- **Random**: uniformly select one model for the whole task.
- **PAVE-Single** (Shi et al., 2026): construct a user specification from the labeled user data and identify one model by the largest cosine similarity between the user PAVE and the model's full PAVE.
- **PAVE Top-8 Ensemble**: select 8 models using the same full-PAVE similarity ranking. For classification, uniformly average their aligned class probabilities and predict the highest-probability label; for STS-B, average their scalar predictions.
- **Best-single**: select the model with the highest aggregate score on the evaluation workload, providing a hindsight reference for single-model reuse.

D.1.3 METHOD CONFIGURATION

Tables A6 and A7 report the hyperparameter configurations of our method for the batch-query (black-box LLM) and task (CV/NLP) scenarios, respectively.

For computational efficiency, assembly uses PAVE parameter changes and sample gradients restricted to the same selected parameter coordinates. For black-box LLMs, we retain the LoRA parameter coordinates in the final Transformer layers of the PAVE reference model, with the layer count specified in Table A6.

Table A6: Method configuration for black-box LLMs. Score weights correspond to Relative, Specificity, Global, and Quality in Eq. (15).

Configuration	Single dataset	Mixed datasets
Model budget K	5	8
Relative-explainability threshold ρ	Disabled	0.90
Common rank r_c	1	1
Last Transformer layers used	4	4
Requested clusters per profile	8	8
(w_R, w_S, w_G, w_Q)	$(1, 0, 0.075, 0.75)$	$(1, 0, 0.075, 0.75)$

Table A7: CV/NLP identification and assembly configuration. Model budgets and stopping criteria are specified separately for single- and mixed-dataset experiments.

Configuration	CV	NLP
Common rank r_c	1	1
Requested cluster PAVEs per model	20	8
Neighbors κ	2	14
KNN fusion weight β_{knn}	0.50	0.80
Relative-score weight w_R	1	1
Specificity weight w_S	0	0.5
Teacher temperature T_v	1.0	0.25
KNN temperature T_k	0.03	0.075
Prototype temperature T_p	0.1	0.1
Global-score weight w_G	0.4	0.2
Single-dataset model budget K	7	7
Single-dataset threshold stopping	Disabled	Disabled
Mixed-dataset model budget K	8	8
Mixed-dataset relative threshold ρ	0.95	0.95

D.2 Mixed-Dataset Settings

D.2.1 Batch-Query Scenario: Black-Box LLMs

Mixture design. With combination seed 2026, we construct 12 mixtures from the 14 evaluation datasets: 6 Between-Domain, 3 Across-All-Domains, and 3 Global-Random combinations (Table A8). Within each design, equal numbers of combinations contain only in-system datasets, both in-system and out-of-system datasets, or only out-of-system datasets. Balanced and long-tail proportions and seeds 42/43/44 give 72 batches of 64 queries, or 4,608 evaluated query positions per method. All methods share the fixed combinations, samples, and model specifications.

Balanced mixtures assign equal weight to each component. For long-tail mixtures with J components, draw $z \sim \mathrm{Dirichlet}(0.3\mathbf{1}_J)$ and set $\pi_j = 1/64 + (1 - J/64)z_j$. This floor reserves mass for small components. Mixture composition and sample seed determine the random draw. Integer quotas are allocated one position at a time to the available component with the largest deficit from $64\pi_j$, breaking ties by weight and then component order. Records are sampled without replacement within each component using the dataset, split, and sample seed; the same realized batch is used by every method. All adopted batches contain every specified component.

Identification and assembly settings. Table A6 gives the settings. Relative-explainability stopping yields a mean candidate-set size of 4.88, ranging from 2 to 8 models across the 72 query batches.

Evaluation metrics. Utility is the average benchmark score of the assigned models' predictions, with each query weighted equally. Within each seed, we average Utility equally across the 12 combinations and 2 proportion regimes, then report the mean and sample standard deviation of the 3 seed-level averages. Subgroup results use the same procedure over their corresponding combinations. Displayed Utility and score gaps are multiplied by 100.

Table A8: Black-box LLM mixture components. Coverage indicates whether all, some, or none of the component datasets are in-system benchmarks. Each combination uses balanced and long-tail proportions and seeds 42/43/44.

Design	Coverage	Component datasets
Between-Domain	All	MathQA, FPB, FiQA-SA
Between-Domain	All	MedQA, FPB, FiQA-SA
Between-Domain	Some	MedQA, SC
Between-Domain	Some	MathQA, MMLU-Pro (Health)
Between-Domain	None	Headlines, MMLU-Pro (Math), SC
Between-Domain	None	MMLU-Pro (Math), MMLU-Pro (Health)
Across-All-Domains	All	MathQA, MedQA, FiQA-SA
Across-All-Domains	Some	MATH, MMLU-Pro (Health), LendingClub
Across-All-Domains	None	MMLU-Pro (Math), MMLU-Pro (Health), MultiFin
Global-Random	All	GSM8K, MATH, MathQA, MedMCQA, PubMedQA, MedQA, FPB, FiQA-SA
Global-Random	Some	MATH, MathQA, MedMCQA, MedQA, Headlines, LendingClub
Global-Random	None	Headlines, MMLU-Pro (Math), MMLU-Pro (Health), LendingClub, MultiFin, SC

Identification timing. Table A9 reports timing on all 72 batches with two repetitions per condition after worker warmup, using one RTX A6000 GPU and eight CPU threads per worker. We average repetitions within each condition, conditions within each seed, and then the three seed means. The probe and model PAVEs remain resident. User-PAVE construction is measured afresh on each batch with batch size 1, 64 optimizer updates, and a sequence-length limit of 4,096. Identification uses the same frozen user PAVE for both methods to preserve the evaluated candidate sets; reported totals sum these separately measured phases. Timing excludes initialization, sample-gradient extraction, assembly, and candidate LLM inference.

Table A9: Mean identification-stage time per 64-query LLM batch, in seconds. Both methods include the same user-PAVE construction cost. Relative overhead is computed from the unrounded totals.

Method	PAVE construction	Identification	Total	Overhead
PAVE-Single	7.902	0.005	7.907	–
Ours	7.902	0.560	8.461	+7.02%

D.2.2 Task Scenario: CV/NLP

Each mixture combines 2 to 8 datasets from the same modality. Candidate model encoders are reused with fixed parameters and target-compatible output heads.

Outcome-independent sampling. Let $\mathcal{D}_{\text{mix}}$ denote the set of datasets selected for a mixture. With combination seed 2026, we sample 20 draws per design from the groups in Table A5. Sampling is without replacement within each draw; different draws may repeat dataset combinations. All methods share the fixed combinations and sample records.

1. **Between-Group (20 draws).** Select 2 distinct groups uniformly, then sample 1 or 2 datasets from each group, choosing the count uniformly and independently. This produces mixtures of 2 to 4 datasets. Balanced weights allocate $1/2$ of the mass to each group.
2. **Across-All-Groups (20 draws).** Draw 1 dataset uniformly from each group: 4 datasets for CV and 3 for NLP.
3. **Global-Random (20 draws).** Draw the number of datasets uniformly from $\{2, 4, 6, 8\}$, then sample that many datasets uniformly from the modality's library.

Proportions and sample allocation. Balanced mixtures assign equal weight to each selected dataset, except Between-Group mixtures, which allocate half the total weight to each group and

divide it equally among datasets within that group. Long-tail mixtures draw the requested dataset proportions

$$\boldsymbol{\pi} \sim \mathrm{Dirichlet}(0.3\mathbf{1}_{|\mathcal{D}_{\mathrm{mix}}|}). \tag{23}$$

For each selected dataset D_t, labeled user examples are drawn exclusively from its existing k-shot pool $D^l_{u,t}$ (Appendix D.1.2), while test examples are drawn from its test split. The nominal totals are $64|\mathcal{D}_{\mathrm{mix}}|$ labeled user examples and $256|\mathcal{D}_{\mathrm{mix}}|$ test examples, allocated across datasets according to π_t subject to available samples. The actual number of labeled user examples is

$$|D^l_u| = \min\left\{64|\mathcal{D}_{\mathrm{mix}}|, \sum_{D_t \in \mathcal{D}_{\mathrm{mix}}} |D^l_{u,t}|\right\}. \tag{24}$$

When the combined k-shot pools fit within the nominal budget, all available labeled user examples are used. Otherwise, allocations are capped by each pool's size, with unfilled allocations redistributed among pools with remaining examples, and subsets are sampled without replacement. Integer allocation and capacity limits may cause realized proportions to differ from the requested weights. All compared methods share the resulting labeled user data and test samples. The merged labeled user data D^l_u are used both to fit the mixed user PAVE and construct the assembly teacher. Appendix E.1.2 reports the realized labeled-data allocations.

Evaluation grid. The 60 draws, 2 proportion regimes, and 3 split seeds give 120 conditions and 360 runs per modality. Selected datasets and requested proportions stay fixed across seeds; the labeled user data, test samples, and training order vary.

Identification and assembly settings. Table A7 specifies the shared configuration and mixed-dataset stopping settings. Identification follows the stopping rule in Appendix B.2.

Task metrics and mixture utility. Let $\mathcal{K}_t$ index the native metrics of dataset D_t, and let $u_{t,k}(a)$ denote metric k computed from all predictions on that dataset under assignment a. Using the metrics in Appendix D.1.2 and requested dataset proportions π_t, mixed-task Utility is

$$\mathcal{U}_{\mathrm{mix}}(a) = \frac{\sum_{D_t \in \mathcal{D}_{\mathrm{mix}}} \sum_{k \in \mathcal{K}_t} \pi_t u_{t,k}(a)}{\sum_{D_t \in \mathcal{D}_{\mathrm{mix}}} \sum_{k \in \mathcal{K}_t} \pi_t}. \tag{25}$$

CV reduces to proportion-weighted accuracy. NLP weights metric rows, so MRPC, QQP, and STS-B each receive twice the effective weight of a single-metric dataset with the same π_t. Correlations retain their native ranges; displayed scores are multiplied by 100. Best-single maximizes this same score over individual candidate models.

Aggregation and uncertainty. Within each seed, we average Utility equally across the 120 conditions per modality, then report the mean and sample standard deviation of the 3 seed-level averages. Subgroup results use the same procedure over their corresponding conditions. Displayed score gaps are multiplied by 100.

Identification and assignment diagnostics. For these diagnostics, a component model is the model trained on one of the mixture's datasets. Table A15 reports:

- **Component recall / weighted recall**: the fraction of component models retrieved, weighted equally or by the requested dataset proportions.
- **Candidate precision / exact identified set**: the fraction of retrieved models that are component-trained models, and whether the 2 sets coincide.
- **Dataset–model assignment agreement / component-equal agreement**: the fraction of test inputs assigned to the model trained on their dataset, averaged over all inputs or equally over the mixture's datasets.
- **Usage recall / exact model usage**: the fraction of component-trained models assigned at least one input, and whether the used-model set equals the set of component-trained models.

These statistics are computed per run, averaged equally across conditions and seeds, and reported as percentages.

D.3 Single-Dataset Settings

Scores and averages are computed at full precision and displayed on a 100-point scale.

D.3.1 Batch-Query Scenario: Black-Box LLMs

Evaluation uses one run per dataset with seed 42 and the single-dataset configuration in Table A6. We report equal-weight dataset averages for the 8 in-system benchmarks, the 6 out-of-system benchmarks, and all 14 benchmarks combined.

D.3.2 Task Scenario: CV/NLP

Evaluation protocol. Leave-one-task-out excludes the target-trained model and its specifications before identification, leaving 11 CV or 14 NLP candidates. Candidate models are adapted using the labeled user data in Appendix D.1.2, while their specifications remain fixed. Identification and assembly use the single-dataset configuration in Table A7.

Target adaptation. For CV, the encoder and target head are adapted for 20 epochs with learning rate 10^{-5}, batch size 128, weight decay 0.1, and 500 warmup steps. NLP uses full-parameter adaptation for 400 steps with learning rate $2 \cdot 10^{-5}$, training/evaluation batch sizes 8/16, and maximum sequence length 128. Compared methods share the labeled user data and adaptation settings.

Scoring and aggregation. Each target is evaluated once with the fixed seed-42 split and adaptation. CV averages accuracy equally across 12 datasets. NLP uses the metrics in Appendix D.1.2, first averaging metrics within each dataset and then equally weighting all 15 datasets. Win/tie/loss counts use these dataset-level scores.

D.4 Mechanism Analysis and Ablation Settings

D.4.1 Gradient-Calibration Diagnostic

Datasets and candidate models. This diagnostic uses the 8 in-system benchmarks and 10 black-box LLMs listed in Appendix D.1.1. Ranking ties follow the listed model order.

We sample 100 distinct queries per dataset with seeds derived from seed 42, by uniform sampling. Queries require valid binary outcomes for all candidates; conflicting duplicates are excluded. Among 800 queries, 309 contain both correct and incorrect candidates, 457 have all candidates correct, and 34 have none correct. Recorded correctness is used only for evaluation.

Paired matching conditions. We compare $-g_i$ with $\tilde{g}_i$ from Eq. (6), using $\epsilon = 10^{-8}$ and all 100 queries per dataset to estimate second moments. Both conditions use the same parameter coordinates and input length. All 10 models are ranked directly by the gradient–PAVE similarity in Eq. (7). The paired intervention changes the gradient scaling while keeping model PAVEs fixed.

The fixed collection has 593 PAVEs covering 68 of 80 model–dataset pairs; missing pairs are in MathQA and PubMedQA, and every model has a specification. The same PAVEs are used in both matching conditions, isolating the contribution of gradient calibration in the paired comparison.

Ranking metrics and aggregation. Ranking metrics use the 309 informative queries with both correct and incorrect candidates. AUC gives half credit to ties; NDCG@10 uses binary relevance and discount $1/\log_2(r+1)$; MRR uses the first correct candidate; Recall@k measures the fraction of correct candidates in the top k. Top-1 accuracy also uses all 800 queries.

Metrics are first averaged within datasets, then equally across all 8 datasets.

The Relative-score gap subtracts the mean score of incorrect candidates from that of correct candidates for each informative query. Scores are standardized across candidates with a standard-deviation floor of 10^{-6} and clipped to $[-5, 5]$. Gaps are averaged within datasets and then across datasets. The reported differences are descriptive estimates for this fixed-seed paired comparison.

D.4.2 BLACK-BOX LLM COMPONENT ABLATIONS

Identification and assembly. We use all 72 conditions in Appendix D.2.1. Similarity Top-K ranks the original full PAVEs by cosine similarity, taking each model's highest similarity across its profiling datasets, and matches our method's candidate count in each condition. Both identified candidate sets use the same assembly coordinates, full and cluster PAVEs, gradient calibration, global scores, and score weights. Assembly is compared with unweighted majority voting over the same identified models; voting ties follow identification order. No supervised selector is trained because the query batches provide no labeled user data. Oracle takes the highest per-query score among the corresponding identified models and uses evaluation labels only as a diagnostic reference. Utility and seed aggregation follow Appendix D.2.1.

Gradient calibration and common-space removal. Using the same 72 conditions, we disable the two components separately. Removing calibration replaces $\tilde{g}_i$ with $-g_i$, keeping the identified candidate set, PAVEs, and coordinates fixed. Removing common-space projection sets $P_\perp = I$ and reruns identification with $K = 8$ and $\rho = 0.90$, leaving assembly unchanged. Only the latter intervention can alter the candidate set. All remaining settings follow Table A6.

D.4.3 CV/NLP IDENTIFICATION AND ASSEMBLY ABLATIONS

Evaluation and interventions. The ablations use all mixed-task conditions in Appendix D.2.2, with identical labeled user data, test samples, and model predictions across methods. Hyperparameters follow Table A7.

- **Identification controls.** Similarity Top-K ranks original full PAVEs by cosine similarity and matches our method's candidate count in each condition. Both candidate sets are evaluated with the same gradient-guided assembly, uniform ensemble, and per-sample oracle. Assembly coordinates, full and cluster specifications, gradient calibration, and teacher-transfer settings are held fixed.
- **Assembly controls.** The supervised MLP, supervised Linear selector, and uniform ensemble receive the candidate set identified by our method in the same order. The ensemble averages aligned class probabilities or scalar regression predictions over that set.

Supervised selectors. MLP and Linear are trained separately for each condition using the same labeled user data and shared encoder features as our method. Feature standardization uses means and scales computed from the labeled user data. Candidate predictions on these data provide cross-entropy losses for classification and squared losses for STS-B; a Softmax over negative losses gives the training targets. Table A10 lists the fixed training settings. Each test input is assigned using its features, and evaluation labels are used to compute the reported scores. Conditions with a single candidate assign every input directly to that model.

Table A10: Training settings for the supervised assembly controls.

Setting	MLP	Linear
Hidden layer	64 units, ReLU	–
Dropout	0.1	0
Output	Softmax	Softmax
Target temperature	0.1	0.1
Optimizer	AdamW	AdamW
Learning rate	10^{-3}	10^{-3}
Weight decay	10^{-4}	10^{-4}
Batch size	64	64
Training epochs	200	200
Training seeds	42/43/44	42/43/44

Oracle diagnostics and aggregation. For classification, the oracle selects a correct candidate whenever one is available and breaks ties by supervised loss; otherwise it minimizes that loss. STS-B selects the candidate with minimum squared error. The resulting native NLP Utility is a

label-informed diagnostic; F1, MCC, and correlation are computed from the complete set of selected predictions. We also report classification correct coverage and STS-B minimum MSE. Coverage first averages realized classification inputs within each condition; MSE averages STS-B samples within each applicable condition. Both then average conditions within seed and the 3 seeds equally; MSE uses the 120 runs containing STS-B. Utility follows Eq. (25). Main means and sample SDs use 3 seed means, each over 120 conditions; paired differences use the same runs, and W/T/L counts all 360 runs per modality.

E Additional Experimental Results

All score differences in this appendix and the main text are computed before rounding.

E.1 Mixed-Dataset Results

E.1.1 Batch-Query Scenario: Black-Box LLMs

Table A11 reports the mixed-query comparison summarized in Figure 2(a).

Table A11: Mixed-query Utility ($\times 100$) with 10 black-box LLMs. Mean $\pm$ sample SD over 3 seeds, each equally weighting 12 mixtures and 2 proportion regimes.

Method	Utility (mean $\pm$ SD) $\uparrow$
Random	78.80 ± 2.05
DescSim	78.82 ± 1.04
K-NN	74.33 ± 1.04
EmbedLLM	79.49 ± 2.31
GraphRouter	79.06 ± 1.04
PAVE-Single	76.43 ± 2.11
Cluster-PAVE	79.28 ± 2.00
UniRoute (K-means)	80.10 ± 1.57
Ours	$\mathbf{81.16 \pm 1.15}$

E.1.2 Task Scenario: CV/NLP

We report performance by mixture design and proportion regime, followed by candidate-recovery diagnostics and illustrative cases.

Table A12: Mixed-task Utility ($\times 100$). Mean $\pm$ sample SD over 3 seeds, each equally weighting 3 designs and 2 proportion regimes (120 conditions). CV: accuracy; NLP: native metric-row utility. Ensemble executes 8 candidates per input; ours executes one.

Method	CV	NLP
Random	54.95 ± 0.06	54.59 ± 0.07
PAVE-Single	65.17 ± 0.07	65.46 ± 0.12
Best-Single (hindsight)	73.19 ± 0.02	71.21 ± 0.08
PAVE Top-8 Ensemble	74.53 ± 0.41	64.06 ± 0.09
Ours	$\mathbf{83.81 \pm 0.63}$	$\mathbf{73.73 \pm 0.13}$

Table A13 covers all sampled conditions under both balanced and long-tail proportions. The method supports several forms of heterogeneity: both CV and NLP attain their highest subgroup Utility on long-tail Across-All-Groups mixtures. Together with the overall comparison, these results demonstrate applicability across different mixture designs and component proportions.

Realized labeled-data allocation. Across 360 runs per modality, the complete k-shot user data pools are merged in 48 CV runs and 348 NLP runs; the remaining 312 CV runs and 12 NLP runs sample subsets within those pools. For example, cr, MRPC, and WNLI each provide 32 labeled

Table A13: Our method by mixture design and proportion regime ($\times 100$). Each row contains 20 draws per seed; SD is the sample SD of the 3 subgroup seed means.

Modality	Design	Proportions	Utility	Seed SD
CV	between-group	balanced	82.95	1.11
CV	between-group	long-tail	80.38	0.62
CV	across-all-groups	balanced	83.49	0.65
CV	across-all-groups	long-tail	86.86	1.92
CV	global-random	balanced	85.54	0.26
CV	global-random	long-tail	83.66	1.76
NLP	between-group	balanced	73.93	0.05
NLP	between-group	long-tail	73.23	0.17
NLP	across-all-groups	balanced	72.02	0.27
NLP	across-all-groups	long-tail	77.71	0.26
NLP	global-random	balanced	72.15	0.22
NLP	global-random	long-tail	73.37	0.86

examples, yielding 96 examples against a nominal budget of 192. A balanced SVHN–EuroSAT–STL10–Food101 mixture draws 20, 20, 20, and 196 examples, respectively, using 256 of the 262 available labeled examples. These allocations reflect the capacity-constrained sampling rule in Appendix D.2.2.

Model usage. Table A14 reports the resulting candidate-set sizes and the numbers of models used by assembly.

Table A14: Stopping and model usage over 360 runs per modality. Counts refer to recorded stopping reasons. Active is the mean number of models assigned at least one test input.

	ρ	Mean $\lvert\mathcal{C}\rvert$	Range	Active	Threshold stops	Cap stops
CV	95%	4.49	1–8	3.77	339	21
NLP	95%	4.90	2–8	3.49	354	6

Same-name identification and assembly diagnostics. Table A15 reports the 8 statistics defined in Appendix D.2.2 on final test assignments.

Table A15: Mixed-task identification and test-assignment diagnostics (%), using all 360 runs per modality. Appendix D.2.2 defines each metric. Identity agreement is distinct from prediction accuracy.

Metric ↑	CV	NLP
Component recall	80.37	85.00
Weighted recall	86.71	84.17
Candidate precision	77.52	62.99
Exact identified set	13.61	3.33
Dataset–model assignment agreement	84.69	73.36
Component-equal agreement	78.03	73.69
Usage recall	80.24	79.94
Exact model usage	26.94	38.89

CV identification recovers 80.37% of component-trained models on average, with 77.52% candidate precision; its higher weighted recall of 86.71% reflects stronger recovery of components accounting for more of the workload. CV dataset–model assignment agreement reaches 84.69%. NLP also recovers a large fraction of component models. The mean numbers of active models are smaller than the retained pools in both modalities (Table A14), illustrating selective use of identified candidates. These diagnostics describe recovery and use of component-trained models, complementing the predictive Utility results.

Illustrative cases and selection. The cases in Figure 3 require full component recall in every seed, mean dataset–model assignment agreement at least 90%, and mean Utility above Best-Single. Of 120 conditions per modality, 32 CV and 35 NLP cases qualify. We prioritize different designs and larger gains, avoiding duplicate component sets. 6 CV conditions additionally recover exactly the component-model set in all 3 seeds. We reserve one CV slot for the largest-gain example among these 6, preserving coverage of the 3 mixture designs. These cases are selected after evaluation for illustration; the aggregate tables retain all conditions.

Table A16: Selected mixed-task cases. Precision and agreement are percentages; Utility and gains use a 100-point scale. All have full component recall. Statistics average per-seed results; cross-seed unions only align the heatmap columns.

Case	Components	Precision	Agreement	Ours	Best-Single	Gain
CV-1	SVHN + EuroSAT + CIFAR100 + Cars	57.94	99.19	87.50	54.98	+32.52
CV-2	GTSRB + EuroSAT + RESISC45 + Food101	100.00	99.35	94.66	64.32	+30.34
CV-3	MNIST + SVHN + Cars + Food101	86.67	99.97	87.27	65.01	+22.27
NLP-1	trec + STS-B	100.00	100.00	89.42	63.34	+26.09
NLP-2	CoLA + QQP + RTE	37.50	98.87	67.24	54.44	+12.80
NLP-3	MRPC + mpqa + subj + RTE	80.00	94.79	81.14	70.17	+10.97

CV-1/2/3 are balanced Across-All-Groups draw 016, Global-Random draw 016, and Between-Group draw 003, respectively. Their candidate counts across seeds 42/43/44 are 8/7/6, 4/4/4, and 4/5/5. CV-2 identifies exactly the component-model set in every seed; CV-1 and CV-3 retain additional candidates while recovering every component-trained model. NLP-1 is balanced Between-Group draw 007; NLP-2 is balanced Across-All-Groups draw 009; NLP-3 is balanced Global-Random draw 002, with 2, 8, and 5 candidates per seed, respectively. Heatmap columns use the cross-seed model union for alignment; candidate-set sizes are measured within each run.

E.2 Single-Dataset Results

E.2.1 Batch-Query Scenario: Black-Box LLMs

Tables A17 and A18 report the in-system and out-of-system results, respectively, with dataset-level macro-averages.

Table A17: In-system scores with 10 black-box LLMs on 8 benchmarks. Profiling and evaluation records are disjoint. UniRoute denotes its K-means variant. Bold marks the best score in each row; aggregates use unrounded scores.

Benchmark	Random	DescSim	K-NN	EmbedLLM	UniRoute	GraphRouter	PAVE-Single	Cluster-PAVE	Ours
Mathematics									
GSM8K	94.50	95.40	95.10	96.10	96.10	95.90	**96.30**	94.30	**96.30**
MATH	92.60	89.90	93.70	**95.90**	95.80	93.70	**95.90**	89.80	95.00
MathQA	88.90	87.90	90.50	88.70	88.80	90.60	85.80	**91.60**	89.80
Healthcare									
MedMCQA	71.20	72.40	75.10	**80.00**	79.10	75.70	**80.00**	74.50	78.60
PubMedQA	67.80	67.60	72.00	72.60	74.20	73.40	74.40	74.40	**75.20**
MedQA	87.20	86.20	87.50	**91.60**	90.70	89.60	**91.60**	84.70	90.30
Finance									
FPB	77.32	78.04	78.04	77.32	75.88	78.04	75.88	75.88	**78.76**
FiQA-SA	77.02	76.60	74.89	74.47	73.62	71.49	74.04	**80.43**	77.45
Dataset macro (8) ↑	82.07	81.75	83.35	84.59	84.27	83.55	84.24	83.20	**85.18**

Table A18: Out-of-system scores with 10 black-box LLMs on 6 datasets excluded from profiling and baseline training. The final row aggregates all 14 evaluated datasets. UniRoute denotes its K-means variant. Bold marks the best score in each row; aggregates use unrounded scores.

Benchmark	Random	DescSim	K-NN	EmbedLLM	UniRoute	GraphRouter	PAVE-Single	Cluster-PAVE	Ours
Mathematics									
MMLU-Pro (Math)	89.30	91.80	72.90	92.30	**93.90**	92.00	73.00	93.60	93.40
Healthcare									
MMLU-Pro (Health)	75.06	77.51	65.40	**78.24**	77.75	73.23	65.16	77.75	78.00
Finance									
Headlines	77.60	78.00	79.40	74.50	76.40	73.80	73.60	**81.90**	76.90
LendingClub	51.80	44.80	36.70	36.70	36.70	36.70	64.90	36.70	**65.00**
MultiFin	67.03	68.86	67.40	65.75	68.86	65.93	**71.06**	68.68	67.95
SC	73.80	72.00	66.90	80.60	80.40	79.80	78.90	78.90	**82.20**
Dataset macro (6) ↑	72.43	72.16	64.78	71.35	72.34	70.24	71.10	72.92	**77.24**
All 14 datasets ↑	77.94	77.64	75.40	78.91	79.16	77.85	78.61	78.80	**81.78**

E.2.2 Task Scenario: CV/NLP

Tables A19 and A20 report the CV and NLP leave-one-task-out results, respectively, using the adaptation protocol in Appendix D.3.2.

Table A19: CV leave-one-task-out accuracy ($\times 100$) on 12 targets with few-shot adaptation; our method uses $K = 7$, Ensemble $K = 8$. Bold marks the best non-oracle learnware score in each row. Best-single is a hindsight reference.

User Task	Single Learnware			Multiple Learnwares	
Dataset	Random	PAVE-Single	Best-single (hindsight)	PAVE Top-8 Ensemble	Ours
CIFAR10	79.46	90.19	91.90	**92.73**	90.84
CIFAR100	62.47	69.38	70.89	**74.15**	69.38
Cars	57.42	57.28	61.22	**61.92**	57.22
DTD	40.84	44.68	46.91	**45.90**	41.81
EuroSAT	57.07	**76.93**	76.93	76.55	75.13
Food101	77.99	80.44	83.00	**82.84**	78.24
GTSRB	36.53	36.33	41.66	**44.13**	38.27
MNIST	63.50	**85.29**	85.29	79.55	**85.29**
RESISC45	56.75	51.37	66.21	**65.83**	53.63
STL10	92.72	92.20	97.39	**96.66**	92.73
SUN397	61.86	62.17	63.94	**66.16**	61.89
SVHN	40.08	37.72	54.08	**58.88**	53.58
Dataset average ↑	60.56	65.33	69.95	**70.44**	66.50
Ours: win/tie/loss	10/0/2	5/2/5	0/1/11	1/0/11	–

Table A20: NLP leave-one-task-out scores ($\times 100$) on 15 targets with few-shot adaptation; our method uses $K = 7$, Ensemble $K = 8$. Configuration details appear in Appendix D.3.2. Averages and win/tie/loss counts weight the 15 datasets equally after averaging metrics within each dataset. Bold marks the best non-oracle learnware score; Best-single is a hindsight reference. MCC: Matthews correlation; PCC/SCC: Pearson/Spearman correlation.

User Task		Single Learnware			Multiple Learnwares	
Dataset	Metric	Random	PAVE-Single	Best-single (hindsight)	PAVE Top-8 Ensemble	Ours
CoLA	MCC	9.39	5.74	15.80	9.53	**15.80**
MNLI	Acc.	56.27	**65.30**	65.30	61.59	**65.30**
MRPC	Acc.	70.80	68.87	75.98	**73.77**	70.34
MRPC	F1	79.02	77.91	82.81	**81.83**	79.39
QNLI	Acc.	63.53	67.12	73.77	**69.78**	69.38
QQP	Acc.	66.80	71.36	74.92	**72.04**	71.36
QQP	F1	63.37	67.38	70.73	**68.70**	67.38
RTE	Acc.	58.72	55.96	70.04	64.98	**66.43**
SNLI	Acc.	67.50	**75.07**	75.07	73.09	**75.07**
SST-2	Acc.	87.45	88.53	90.14	89.45	**89.79**
STS-B	PCC	60.69	48.40	78.23	65.54	**67.14**
STS-B	SCC	59.70	46.80	78.81	64.69	**67.38**
WNLI	Acc.	46.88	**49.30**	50.70	**49.30**	**49.30**
cr	Acc.	88.42	88.90	90.00	**89.65**	89.25
mpqa	Acc.	80.75	82.15	84.25	**83.05**	82.35
mr	Acc.	82.96	83.50	86.05	**84.40**	83.50
subj	Acc.	89.17	88.75	91.25	**90.40**	90.20
trec	Acc.	73.01	**77.60**	77.60	75.20	75.60
Dataset average ↑		66.95	67.89	73.38	70.25	**70.90**
Ours: dataset win/tie/loss		14/0/1	9/5/1	0/3/12	7/1/7	–

E.3 Mechanism Analysis and Ablation Results

E.3.1 Gradient-Calibration Diagnostic

The paired diagnostic and aggregation protocol are described in Appendix D.4.1.

Table A21: Calibration diagnostics, macro-averaged over 8 datasets. Ranking metrics and Relative-score (R) gaps use queries with both correct and incorrect candidates; top-1 accuracy uses all 800 queries. Δ is calibrated minus raw before rounding; accuracy changes are percentage points.

Metric ↑	Raw	Calibrated	Δ
ROC-AUC	0.4294	**0.4976**	+0.0682
NDCG@10	0.8184	**0.8363**	+0.0180
MRR	0.7432	**0.7702**	+0.0270
Recall@1	0.0895	**0.1008**	+0.0113
Recall@3	0.2796	**0.3032**	+0.0236
Recall@5	0.4962	**0.5150**	+0.0188
Top-1 accuracy (%)	79.63	**81.50**	+1.88
R positive–negative mean gap	−0.1721	**0.0160**	+0.1881

On the 309 informative queries, pooled top-1 accuracy rises from 58.25% to 63.11% after calibration; these pooled figures use different weights from the dataset-macro results in Table A21.

Results and interpretation. Calibration improves every reported dataset-macro ranking metric, including AUC, NDCG, MRR, and Recall at each cutoff, as well as top-1 accuracy (Table A21). The Relative-score gap changes from negative to positive, indicating better average separation of correct and incorrect candidates. These complementary ranking and assignment results support second-moment scaling as a useful way to make gradient–PAVE matching more informative for model selection.

E.3.2 Black-Box LLM Identification and Assembly Ablations

Identification. We ablate identification by comparing our method with full-PAVE similarity Top-K at matched candidate counts. The controls follow Appendix D.4.2. Table A22 shows higher mean Utility with our identification under all three reuse strategies. The oracle gains 0.11 points, while the same gradient-guided assembly gains 0.39 points, supporting the value of complementary identification for unlabeled query batches.

Assembly. We ablate assembly by comparing our gradient-guided assembly method with majority voting over the same identified models. Table A22 reports both strategies and their candidate-set oracle. Our gradient-guided assembly method reaches 81.16 Utility while invoking one candidate per query; voting invokes all identified models, averaging 4.88 candidates.

Table A22: LLM identification and assembly ablations on the 72 mixed-query conditions. Utility ($\times 100$) is mean $\pm$ sample SD over 3 seed means. Candidate counts match per condition. Oracle denotes the per-sample oracle over the models identified by the corresponding identification method.

Identification	$\bar{K}$	Oracle	Ensemble	Our Assembly
Ours	4.88	91.45 ± 0.46	81.32 ± 2.26	81.16 ± 1.15
Similarity Top-K	4.88	91.34 ± 0.53	80.73 ± 1.36	80.77 ± 1.76

E.3.3 CV/NLP Identification and Assembly Ablations

The controls, selector configurations, and oracle metrics are specified in Appendix D.4.3.

Assembly results. With the identified candidate set fixed, our gradient-guided assembly method attains the highest mean Utility among the compared selectors and ensemble on both CV and NLP

(Table 1, under our identification). The paired results in Table A23 show wins over MLP in 233 of 360 CV runs and 332 of 360 NLP runs. On NLP, the mean gains over MLP, Linear, and uniform ensembling hold across all 6 combinations of mixture design and proportions (Table A24). These results support gradient–PAVE guidance and Prototype+KNN transfer for allocating inputs among complementary models in the evaluated mixtures.

Execution and costs. Per-sample selectors execute one candidate per test input; the ensemble executes all identified models, averaging 4.49 for CV and 4.90 for NLP. Supervised selector training additionally requires candidate predictions on the labeled user data. Candidate execution, feature extraction, and selector training contribute separately to the total computational cost.

Table A23: Paired module-ablation differences (reference minus baseline, $\times 100$). SD is across 3 paired seed means; W/T/L counts all 360 runs per modality.

Modality	Reference	Baseline	Difference	W/T/L
CV	Oracle (identified candidate set)	Top-K Oracle	3.40 ± 0.61	192/87/81
CV	Ours	Top-K + assembly	5.81 ± 1.11	205/117/38
CV	Ours	Supervised MLP	0.43 ± 0.22	233/35/92
CV	Ours	Supervised Linear	1.86 ± 0.23	306/18/36
CV	Ours	Ensemble (identified models)	4.74 ± 0.17	307/17/36
NLP	Oracle (identified candidate set)	Top-K Oracle	0.20 ± 0.17	144/85/131
NLP	Ours	Top-K + assembly	1.08 ± 0.36	135/138/87
NLP	Ours	Supervised MLP	4.06 ± 0.20	332/0/28
NLP	Ours	Supervised Linear	4.99 ± 0.09	338/0/22
NLP	Ours	Ensemble (identified models)	6.16 ± 0.12	301/1/58

Table A24: Module ablations by design and proportions. Utility is $\times 100$, mean $\pm$ seed SD; each row has 20 draws per seed. Top-K matches our candidate count and retains our assembly.

Modality	Design	Proportions	Ours	Top-K	MLP	Linear	Ensemble
CV	between-group	balanced	82.95 ± 1.11	76.88 ± 0.80	82.18 ± 1.29	81.03 ± 1.43	79.17 ± 0.86
CV	between-group	long-tail	80.38 ± 0.62	73.09 ± 0.85	79.28 ± 0.80	77.59 ± 1.23	78.13 ± 1.35
CV	across-all-groups	balanced	83.49 ± 0.65	78.45 ± 0.37	83.35 ± 0.55	82.01 ± 0.46	78.47 ± 0.44
CV	across-all-groups	long-tail	86.86 ± 1.92	78.51 ± 1.37	86.54 ± 1.90	85.50 ± 1.86	82.31 ± 2.30
CV	global-random	balanced	85.54 ± 0.26	80.85 ± 0.44	85.31 ± 0.36	83.65 ± 0.28	78.67 ± 0.43
CV	global-random	long-tail	83.66 ± 1.76	80.27 ± 0.59	83.66 ± 1.26	81.92 ± 1.13	77.69 ± 0.48
NLP	between-group	balanced	73.93 ± 0.05	72.88 ± 0.31	69.99 ± 0.14	69.52 ± 0.13	69.34 ± 0.21
NLP	between-group	long-tail	73.23 ± 0.17	72.11 ± 0.61	70.14 ± 0.62	69.75 ± 0.55	71.07 ± 0.34
NLP	across-all-groups	balanced	72.02 ± 0.27	72.16 ± 0.36	67.66 ± 0.23	66.83 ± 0.10	65.61 ± 0.49
NLP	across-all-groups	long-tail	77.71 ± 0.26	77.30 ± 0.18	72.83 ± 0.28	72.04 ± 0.26	66.84 ± 0.21
NLP	global-random	balanced	72.15 ± 0.22	70.14 ± 0.56	68.14 ± 0.29	66.57 ± 0.27	65.34 ± 0.22
NLP	global-random	long-tail	73.37 ± 0.86	71.30 ± 0.94	69.27 ± 0.34	67.73 ± 0.79	67.25 ± 0.44

Table A25: Candidate-set oracle diagnostics (mean $\pm$ seed SD). Coverage is in percent; STS-B minimum MSE retains its native scale and lower is better.

Identification	CV coverage ↑	NLP coverage ↑	STS-B min MSE ↓
Ours	89.030 ± 0.450	92.948 ± 0.550	0.403 ± 0.025
Similarity Top-K	85.626 ± 0.215	92.848 ± 0.674	0.397 ± 0.030

The NLP oracle Utility and classification coverage slightly favor our identification, whereas STS-B minimum MSE favors cosine Top-K.