COMPOSE THEN SCALARIZE: JOINT-STATE VALUATION FOR DATA ALLOCATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Allocating a limited data budget across domains requires estimating the value of candidate additions from limited training evidence. Independent scalar-gain valuation based on single-domain pilots predicts each domain's task-score gain independently of allocations to other domains. When the task score aggregates multiple metrics through a nonlinear utility, however, a candidate's marginal value can change with the joint allocation. We propose Compose-Then-Scalarize (CTS), a joint-state valuation method that reuses fixed single-domain pilot evidence to reassess candidate additions as the joint allocation changes. CTS fits signed metric-response curves, additively composes their predictions across domains into a joint metric state, and applies the utility to obtain state-dependent candidate values without additional pilot training. Experiments on autonomous driving and satellite image classification show that CTS achieves the highest mean task score at every evaluated allocation size. Compared with the strongest of the seven baselines at matched budgets, CTS improves the composite driving score by 0.71–1.61 points and worst-region accuracy by 0.93–3.21 percentage points. Matched ablations support the benefit of joint-state valuation, while controlled retraining shows that CTS more accurately predicts background-dependent candidate preferences than a matched control that values candidates at their own-domain states. Code is available at `https://anonymous.4open.science/r/ICLR-CTS`.

1 INTRODUCTION

The composition of training data shapes the performance of deep learning models (Xie et al., 2023; Ye et al., 2025). When data are drawn from heterogeneous sources, a central question is how to allocate a limited data budget across these sources. This question arises in research on data collection (Mahmood et al., 2025; Zha et al., 2025), domain reweighting (Xie et al., 2023; Fan et al., 2024), and mixture optimization (Liu et al., 2025; Ye et al., 2025). However, the computational cost of training a model for each candidate allocation limits the scope of direct search. This motivates estimating the value of untested joint allocations from limited training evidence.

Learning curves and proxy training provide such estimates by relating data quantities or data mixture proportions to model performance (Kang et al., 2025; Liu et al., 2025; Ostapenko et al., 2025). One practical approach uses single-domain pilots, each augmenting a shared base dataset with a small number of examples from one domain and measuring the resulting task-score gain (Dimlioglu et al., 2026). For each domain, a scalar-gain curve is fitted to these observations and then used to predict the gain from adding a given number of examples from that domain. These predicted scalar gains are composed across domains by summation and added to the base score, yielding an independent scalar-gain valuation of joint allocations (Mahmood et al., 2025; Dimlioglu et al., 2026).

In many applications, the task score aggregates multiple metrics through a nonlinear utility function, as in the composite driving score and worst-region accuracy considered here (Koh et al., 2021; Cao et al., 2025). However, the independent scalar-gain valuation does not preserve how individual metrics change. Our pilot analysis in Section 3.3 shows that domains with similar task-score gains can exhibit distinct metric-response patterns (Figure 1). These differences matter for valuing candidate additions: under a nonlinear utility, the gain from a fixed metric response can depend on the joint metric state at which it is evaluated (Roijers et al., 2013). Allocating data to other domains can change this predicted state, altering a candidate's marginal value even when its predicted metric

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.

response remains fixed. This raises a practical question: *can fixed single-domain pilot evidence be reused to predict which candidate addition is preferable as allocations to other domains change?*

We propose **joint-state valuation** for data allocation, implemented through *Compose-Then-Scalarize* (CTS). CTS fits domain-specific response curves to the signed metric changes observed in single-domain pilots. Under an additive response approximation, CTS forms a predicted joint metric state by composing the predicted domain responses with the base metric vector and then applies the utility function to estimate the value of the joint allocation. Depending on the task, the utility either reproduces the task score from the retained metrics or serves as an allocation surrogate. As the allocation changes, CTS updates the joint state and reassesses candidate marginal gains using the fixed response curves, without additional pilot training.

Controlled retraining experiments show that changing the composition of a shared background dataset can reverse the observed preference between two fixed candidate batches. Across six candidate pairs, five background compositions, and three seeds, CTS correctly predicts 72 of 90 preferences (80.0%), compared with 47 (52.2%) for a matched control that shares the same response curves and utility but values candidates at their own-domain states. The allocations produced by CTS also improve final model performance, yielding the highest mean task score at every evaluated allocation size across all three experimental settings. Compared with the strongest of the seven baselines at matched budgets, CTS improves the composite driving score by 0.71–1.61 points and worst-region accuracy by 0.93–3.21 percentage points. Our main contributions are threefold:

- We show how independent scalar-gain valuation can obscure allocation-relevant metric-response differences and characterize when joint-state valuation changes predicted candidate preferences.
- We introduce CTS, which implements joint-state valuation by composing signed metric responses before scalarization and reusing fixed single-domain pilot evidence to reassess candidate marginal values at the predicted joint metric state.
- We demonstrate improved final model performance across two tasks, with matched ablations isolating the effect of joint-state valuation and controlled retraining showing more accurate predictions of background-dependent candidate preferences.

2 Related Work

Data selection and domain allocation. Training-data composition can be optimized at both the example and domain levels. Example-selection methods use sample informativeness and diversity (Wei et al., 2015; Sener & Savarese, 2018; Ash et al., 2020), as well as gradient approximation and validation performance (Mirzasoleiman et al., 2020; Killamsetty et al., 2021). Data-pruning research examines example importance and performance scaling (Paul et al., 2021; Sorscher et al., 2022), while clustering-based sensitivity sampling supports scalable subset construction (Axiotis et al., 2024). At the domain level, allocation methods determine how much each source contributes to training. DoReMi learns mixture weights through a group distributionally robust optimization proxy (Xie et al., 2023), DoGE uses gradient-based generalization estimates (Fan et al., 2024), and Chameleon derives weights from similarities between domain representations (Xie et al., 2025). In this work, we study how many additional samples to allocate to each domain, with the domain partition and within-domain rankings held fixed.

Predictive data allocation. Learning curves relate training data quantities to model performance and support data acquisition decisions (Kaplan et al., 2020; Mohr & van Rijn, 2024). Some methods update their estimates during training: ADO refits domain learning curves (Jiang et al., 2025), while Aioli estimates dynamic mixing laws to adapt mixture proportions (Chen et al., 2025). Other methods, including RegMix and multi-fidelity mixture optimization, learn performance predictors from proxy runs on different data mixtures (Liu et al., 2025; Yen et al., 2025). AutoScale combines mixture optimization at smaller data budgets with extrapolation to larger training scales (Kang et al., 2025). For data collection, Learn-Optimize-Collect includes an additive source-contribution model (Mahmood et al., 2025), while factored scaling curves guide acquisition along individual factors and their combinations (Zha et al., 2025). Ostapenko et al. (2025) estimate source-utility scaling curves from annealing runs of varying durations. MOSAIC is most closely related to our setting, fitting scalar task-score gain curves from single-domain pilots and summing their predictions to guide allocation (Dimlioglu et al., 2026). CTS reuses the same single-domain pilot runs as MOSAIC, but

instead retains signed metric responses and reassesses candidate marginal values at the predicted joint metric state without additional pilot training.

Component-level prediction and structured utility. Mixture-optimization methods also differ in the granularity and aggregation of their predictions. Data Mixing Laws predicts validation-domain losses from mixture proportions and aggregates them into an overall loss (Ye et al., 2025), while Olmix compares per-task, per-family, and aggregate regression (Chen et al., 2026). MixMin combines per-example predictions from source-specific proxy models before evaluating target risk (Thudi et al., 2025). TiKMiX-D combines domain–task influence estimates under a nonlinear objective to optimize mixture weights (Wang et al., 2026), while GRAPE jointly adapts source-domain and target-task weights through a minimax objective (Fan et al., 2025). More broadly, composite-indicator research studies information loss under aggregation (Zhou et al., 2010), while multi-objective decision-making examines how utility functions govern the evaluation of vector outcomes (Roijers et al., 2013; Hayes et al., 2022). Our focus is on how fixed single-domain pilot evidence can be reused to value candidate additions as the joint allocation changes.

3 Data Allocation: Formulation and Motivation

3.1 Data Allocation with Single-Domain Pilots

Data allocation seeks to improve model performance by distributing a limited data budget across sources (Xie et al., 2023; He et al., 2024). Given a base dataset $\mathcal{D}_0$ and its model $f_0 = \mathrm{Train}(\mathcal{D}_0)$, we consider selecting B additional examples from a candidate pool $\mathcal{D}_{\mathrm{pool}}$. Following the allocation setup of Dimlioglu et al. (2026), we partition $\mathcal{D}_{\mathrm{pool}}$ into K domains $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ and rank the examples within each domain. With the partition and within-domain rankings held fixed, an allocation is fully specified by the sample counts $\boldsymbol{b} = (b_1, \ldots, b_K)$. Retraining on the resulting dataset yields the model $f_{\boldsymbol{b}} = \mathrm{Train}\big(\mathcal{D}_0 \cup \bigcup_{i=1}^{K} \mathcal{D}_i(b_i)\big)$, where $\mathcal{D}_i(b_i)$ contains the top b_i examples in domain i.

Let $J(f)$ denote the prespecified task score used to evaluate model f. Our allocation objective is

$$\boldsymbol{b}^\star = \operatorname*{arg\,max}_{\boldsymbol{b} \in \Pi_B} J(f_{\boldsymbol{b}}), \qquad \Pi_B = \Big\{ \boldsymbol{b} : \sum_{i=1}^{K} b_i = B,\ 0 \le b_i \le |\mathcal{D}_i| \Big\}. \tag{1}$$

Exhaustively searching Π_B by retraining a model for every allocation is costly. We instead use a fixed set of single-domain pilot experiments to estimate how additional data affect model performance (Zha et al., 2025; Dimlioglu et al., 2026). Let $\mathcal{P}$ denote a small set of pilot sample counts. For each domain i and count $n \in \mathcal{P}$, we train a pilot model $f_i(n) = \mathrm{Train}(\mathcal{D}_0 \cup \mathcal{D}_i(n))$ by augmenting the shared base dataset with the top n examples from domain i. These pilots provide shared training evidence, leaving open what information to retain and how to reuse it to value candidate additions as the joint allocation changes.

3.2 Independent Scalar-Gain Valuation

The independent scalar-gain valuation models each domain's task-score gain separately and combines the predicted gains to value a joint allocation. This follows a *scalarize-then-compose* (STC) order: each domain's effect is reduced to a scalar before the predicted gains are summed.

For each domain i, STC summarizes each pilot's effect by its task-score gain relative to the base model, $J(f_i(n)) - J(f_0)$, and fits a scalar-valued gain curve (details in Appendix A.4):

$$\widehat{s}_i = \mathrm{Fit}\big\{(n,\ J(f_i(n)) - J(f_0)) : n \in \mathcal{P}\big\}. \tag{2}$$

At count b_i, $\widehat{s}_i(b_i)$ predicts the task-score gain from augmenting the base dataset with b_i examples from domain i alone. STC adds these predicted domain gains to the base score, yielding an estimate of the task score after retraining on the joint allocation (Mahmood et al., 2025):

$$\widehat{v}_{\mathrm{STC}}(\boldsymbol{b}) = J(f_0) + \sum_{i=1}^{K} \widehat{s}_i(b_i). \tag{3}$$

Following Dimlioglu et al. (2026), we use this valuation model to allocate the budget greedily, selecting one additional example at a time according to its predicted marginal gain:

$$\Delta_i^{\mathrm{STC}}(b_i) = \widehat{s}_i(b_i + 1) - \widehat{s}_i(b_i). \tag{4}$$

Starting from $\boldsymbol{b} = \mathbf{0}$, we repeatedly increment b_j by one for $j = \arg\max_{i:b_i<|\mathcal{D}_i|} \Delta_i^{\mathrm{STC}}(b_i)$ until $\sum_i b_i = B$. We then retrain on the base dataset augmented with the selected examples to obtain $f_{\boldsymbol{b}}$ and evaluate its performance using $J(f_{\boldsymbol{b}})$.

This valuation model has two properties relevant to joint allocation. First, a domain's predicted marginal gain depends only on its own allocated count. At a fixed count for domain i, its predicted marginal value therefore remains unchanged as allocations to other domains vary. Second, when the task score aggregates multiple metrics, each scalar gain compresses their individual responses into a single value, so similar scalar gains can conceal distinct response directions and magnitudes.

3.3 Empirical Motivation: What Scalar Gains Omit

To examine whether similar scalar gains can correspond to different metric responses, we analyze single-domain pilots in an autonomous driving task. The Extended Predictive Driver Model Score (EPDMS) combines component metrics through a nonlinear aggregation and serves as the scalar task score $J(f)$. Figure 1 compares 100-example pilots across four geographic domains, showing changes in nine component metrics and EPDMS relative to each pilot's same-seed base model; the analysis protocol, seed-level heatmaps, and 200-example pilot results are provided in Appendix B.

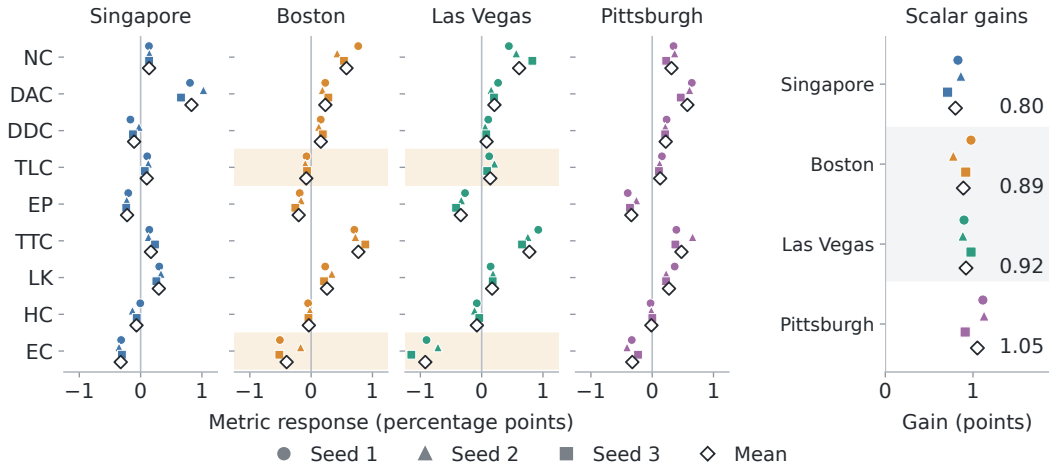


Figure 1: Metric responses and EPDMS gains at a pilot sample count of 100. Shading highlights the Boston–Las Vegas comparison: TLC and EC responses on the left; EPDMS gains on the right.

Within each domain, metric-response vectors are consistent in direction across three seeds, with all pairwise cosine similarities exceeding 0.92. Across domains, similar scalar gains can accompany distinct metric responses. Boston and Las Vegas, for example, have similar mean EPDMS gains of 0.89 and 0.92 points, yet their underlying metric responses differ markedly. Las Vegas's mean extended comfort (EC) decrease is more than twice Boston's (0.92 versus 0.40 percentage points). Their traffic-light compliance (TLC) responses also have opposite signs across all three seeds. The 200-example pilots show similar within-domain consistency and cross-domain differences.

The scalar summaries alone therefore leave unresolved whether domains with similar gains remain similarly valuable as allocations to other domains change. This motivates retaining the underlying metric responses to reassess candidate value at the evolving joint metric state.

4 Compose-Then-Scalarize for Joint-State Valuation

We propose joint-state valuation, which reassesses candidate additions at the predicted joint metric state of the current allocation. We implement this principle through *Compose-Then-Scalarize* (CTS), which composes predicted metric responses across domains before applying the utility.

4.1 Modeling Signed Metric Responses

Let $\boldsymbol{m}(f) \in \mathcal{M} \subseteq \mathbb{R}^d$ collect the d component metrics retained for allocation, with base metric vector $\boldsymbol{m}_0 = \boldsymbol{m}(f_0)$. For example, in our autonomous driving task, each entry of $\boldsymbol{m}(f)$ is the arithmetic mean of one of the nine EPDMS component metrics across driving scenes, giving $d = 9$. For each single-domain pilot $f_i(n)$, we record the signed metric response

$$\boldsymbol{r}_i(n) = \boldsymbol{m}(f_i(n)) - \boldsymbol{m}_0. \tag{5}$$

This vector retains both improvements and degradations in individual metrics relative to the base model. For each domain, we fit a vector-valued response curve to these observations:

$$\widehat{\boldsymbol{r}}_i = \mathrm{Fit}\big\{(n, \boldsymbol{r}_i(n)) : n \in \mathcal{P}\big\}. \tag{6}$$

At count b_i, $\widehat{\boldsymbol{r}}_i(b_i)$ predicts the metric response to augmenting the base dataset with b_i examples from domain i alone. STC fits one scalar-gain curve per domain, whereas CTS fits a separate response curve for each metric in each domain. In our implementation, both methods use the same exponential function family, fitted by nonlinear least squares. Full fitting details are provided in Appendix A.4. STC and CTS thus use the same single-domain pilots but retain different statistics.

4.2 Joint-State Valuation and Allocation

For a joint allocation $\boldsymbol{b}$, CTS forms a predicted joint metric state and then applies a prespecified utility function $u : \mathcal{M} \to \mathbb{R}$ to obtain a predicted allocation value:

$$\widehat{\boldsymbol{m}}(\boldsymbol{b}) = \boldsymbol{m}_0 + \sum_{k=1}^{K} \widehat{\boldsymbol{r}}_k(b_k), \qquad \widehat{v}_{\mathrm{CTS}}(\boldsymbol{b}) = u[\widehat{\boldsymbol{m}}(\boldsymbol{b})]. \tag{7}$$

Here, $\widehat{\boldsymbol{m}}(\boldsymbol{b})$ approximates the metric state of the jointly retrained model under allocation $\boldsymbol{b}$ by adding the predicted domain responses to the base metric vector.

To value an additional example from domain i at the current allocation $\boldsymbol{b}$, CTS first obtains the corresponding metric-response increment from the fitted response curve $\widehat{\boldsymbol{r}}_i$. It then adds this increment to the current predicted joint metric state and computes the utility difference:

$$\boldsymbol{d}_i(b_i) = \widehat{\boldsymbol{r}}_i(b_i + 1) - \widehat{\boldsymbol{r}}_i(b_i), \qquad \Delta_i^{\mathrm{CTS}}(\boldsymbol{b}) = u[\widehat{\boldsymbol{m}}(\boldsymbol{b}) + \boldsymbol{d}_i(b_i)] - u[\widehat{\boldsymbol{m}}(\boldsymbol{b})]. \tag{8}$$

CTS retains single-domain response predictions while valuing each candidate increment at the current predicted joint state. At a fixed b_i, allocations to other domains can change this state and hence the increment's marginal value under a nonlinear utility, without changing $\boldsymbol{d}_i(b_i)$.

Using the same greedy allocation procedure as in Section 3.2, CTS starts from $\boldsymbol{b} = \boldsymbol{0}$ and repeatedly selects $j = \arg\max_{i:b_i<|\mathcal{D}_i|} \Delta_i^{\mathrm{CTS}}(\boldsymbol{b})$, increments b_j by one, and updates $\widehat{\boldsymbol{m}}(\boldsymbol{b})$ before computing marginal gains for the next step. The process continues until $\sum_{i=1}^{K} b_i = B$.

During allocation, CTS reuses the fitted curves to reassess candidate increments at the updated predicted joint state without additional pilot training. After allocating the budget, we retrain on the base dataset augmented with the selected examples to obtain $f_{\boldsymbol{b}}$ and evaluate its performance using $J(f_{\boldsymbol{b}})$. Algorithm 1 in Appendix A.4 summarizes the STC and CTS procedures, highlighting their differences in response fitting and valuation.

Task-specific utilities. For autonomous driving, we retain scene-averaged EPDMS component metrics as a compact response representation, keeping response fitting and repeated candidate valuation lightweight. We apply the EPDMS multiplicative and weighted aggregation to these mean metrics, yielding an allocation surrogate for the task score $J(f)$, which averages scores computed at the scene level. For satellite image classification, we retain regional accuracies and use their minimum as the utility, exactly matching the worst-region aggregation used by $J(f)$. Full metric definitions, utility formulas, and the computational tradeoff of the driving surrogate are provided in Appendix A.2.

4.3 When Joint-State Valuation Changes Candidate Preferences

When does joint-state valuation change candidate rankings? We compare valuation rules at the same allocation, holding the metric-response predictions and utility fixed. A direct comparison between

STC and CTS is insufficient for this purpose because STC fits task-score gains directly, whereas CTS fits component responses and applies the utility afterward. We therefore introduce a *matched scalarize-then-compose* (M-STC) control, which shares CTS's fitted response curves and utility but sums scalar gains computed separately for each domain:

$$\widehat{v}_{\text{M-STC}}(\boldsymbol{b}) = u[\boldsymbol{m}_0] + \sum_{i=1}^{K} \big(u[\boldsymbol{m}_0 + \widehat{\boldsymbol{r}}_i(b_i)] - u[\boldsymbol{m}_0]\big). \tag{9}$$

Its predicted marginal value for an additional example from domain i is

$$\Delta_i^{\text{M-STC}}(b_i) = u[\boldsymbol{m}_0 + \widehat{\boldsymbol{r}}_i(b_i + 1)] - u[\boldsymbol{m}_0 + \widehat{\boldsymbol{r}}_i(b_i)]. \tag{10}$$

Thus, M-STC values the same candidate increment at the own-domain state $\boldsymbol{m}_0 + \widehat{\boldsymbol{r}}_i(b_i)$, whereas CTS values it at the predicted joint state $\widehat{\boldsymbol{m}}(\boldsymbol{b})$. For an affine utility, CTS and M-STC assign identical predicted allocation values and marginal gains (proof in Appendix C). For a nonlinear utility, however, their marginal gains can differ. To quantify how the joint state changes the marginal value of a fixed candidate increment, we define the marginal adjustment as

$$\kappa_i(\boldsymbol{b}) = \Delta_i^{\text{CTS}}(\boldsymbol{b}) - \Delta_i^{\text{M-STC}}(b_i). \tag{11}$$

For a smooth utility, such as our driving surrogate, the marginal adjustment reflects how utility curvature couples a candidate increment with other domains' accumulated responses. Under the minimum utility used in our satellite image classification task, other domains' responses can shift the bottleneck region, thereby changing the value of the same candidate increment. The exact finite-difference expression for this adjustment and the corresponding derivations are provided in Appendix C.

At a fixed allocation $\boldsymbol{b}$, suppose M-STC prefers A to B, i.e., $\Delta_A^{\text{M-STC}} > \Delta_B^{\text{M-STC}}$. CTS strictly reverses this preference, preferring B to A, if and only if

$$\kappa_B - \kappa_A > \Delta_A^{\text{M-STC}} - \Delta_B^{\text{M-STC}}. \tag{12}$$

In other words, CTS reverses the preference when the joint state favors the initially lower-ranked candidate enough to outweigh the initial M-STC preference margin.

An illustrative allocation example. Consider an abstract base metric vector $(1, 1)$ and utility $u[x, y] = xy$. Domains A, B, and C each provide one block of equal size, with predicted responses $(0.12, 0)$, $(0, 0.10)$, and $(0.40, 0)$; the budget permits two blocks. Both rules first select C, moving the predicted joint state to $(1.40, 1)$. For the remaining block, M-STC selects A because its gain of 0.12 exceeds B's 0.10. CTS instead selects B, whose gain rises to 0.14 while A's remains 0.12. The adjustment difference 0.04 exceeds the initial M-STC preference margin of 0.02, reversing the preference (details in Appendix C). Thus, with fixed additive response predictions, joint-state valuation under a nonlinear utility can yield nonadditive predicted values and different allocations.

5 Experiments

5.1 Experimental Setup

Tasks and models. For autonomous driving, we use `Hydra-MDP` (Li et al., 2024) for trajectory planning under two complementary training settings: `navtrain` (Dauner et al., 2024), a curated subset of OpenScene containing nontrivial driving scenarios, and the full `OpenScene` trainval split (OpenScene Contributors, 2023), which covers a broader collection of driving logs. For satellite image classification, we use an ImageNet-pretrained `DenseNet-121` (Huang et al., 2017) on the 62-class `FMoW-WILDS` benchmark (Koh et al., 2021).

Allocation settings. Following the autonomous driving setup of Dimlioglu et al. (2026), we use four city domains: Singapore, Boston, Las Vegas, and Pittsburgh. The navtrain and OpenScene base datasets contain 460 and 1000 clips, leaving candidate pools of 4141 and 31539 clips, respectively. The additional budgets are $B \in \{100, 200, 400, 800, 1600\}$ for navtrain and $B \in \{250, 500, 1000, 2000, 4000\}$ for OpenScene. For FMoW-WILDS, we use five geographic domains: Africa, the Americas, Asia, Europe, and Oceania. The base dataset contains 7686 images (approximately 10% of the training split), with additional budgets $B \in \{2000, 4000, 6000, 8000, 10000\}$. Further details on the datasets and allocation setup are provided in Appendix A.1.

Baselines. We compare CTS with random selection (`Random`), uncertainty-based selection (`Uncertainty`), and coverage-based selection (`CoreSet`; Sener & Savarese, 2018). We also evaluate domain reweighting (`Chameleon`; Xie et al., 2025), regression-based mixture prediction (`RegMix`; Liu et al., 2025), parametric mixture modeling (Data Mixing Laws, `DML`; Ye et al., 2025), and scaling-aware allocation (`MOSAIC`; Dimlioglu et al., 2026). MOSAIC instantiates independent scalar-gain valuation through STC and shares the domain partition, within-domain rankings, and single-domain pilots with CTS. Baseline implementation details are provided in Appendix A.5.

Evaluation and reporting. We evaluate final model performance using the task score J: `EPDMS` for autonomous driving (Cao et al., 2025) and `worst-region accuracy` for FMoW-WILDS (Koh et al., 2021). Following the evaluation protocol of Dimlioglu et al. (2026), we report driving performance on `navtest`. For FMoW-WILDS, the out-of-distribution (OOD) test split is reserved for final evaluation. We report mean $\pm$ standard deviation across random seeds, with response fitting and allocation performed independently for each seed. Appendices A.2–A.4 provide the metric/utility definitions, training/pilot protocols, and response-fitting/allocation details.

5.2 Retraining Results

Table 1: Retraining results: `EPDMS` on `navtest` in (a,b) and OOD `worst-region accuracy` (%) in (c); higher is better. Subscripts give standard deviations across seeds; bold marks the highest mean. B counts additional clips in (a,b) and images in (c).

(a) `navtrain`: Base dataset of 460 clips					
Budget B	100	200	400	800	1600
`Random`	$84.66_{\pm 0.60}$	$85.45_{\pm 0.09}$	$86.69_{\pm 0.20}$	$87.41_{\pm 0.37}$	$88.62_{\pm 0.22}$
`Uncertainty`	$84.50_{\pm 0.48}$	$84.84_{\pm 0.54}$	$86.07_{\pm 0.75}$	$86.69_{\pm 0.34}$	$87.75_{\pm 0.37}$
`CoreSet`	$85.29_{\pm 0.47}$	$86.12_{\pm 0.31}$	$87.09_{\pm 0.29}$	$88.48_{\pm 0.12}$	$89.30_{\pm 0.19}$
`Chameleon`	$84.57_{\pm 0.18}$	$86.04_{\pm 0.30}$	$87.04_{\pm 0.60}$	$88.33_{\pm 0.23}$	$89.50_{\pm 0.20}$
`RegMix`	$84.91_{\pm 0.44}$	$85.96_{\pm 0.28}$	$87.31_{\pm 0.24}$	$88.27_{\pm 0.31}$	$89.68_{\pm 0.21}$
`DML`	$84.73_{\pm 0.51}$	$86.22_{\pm 0.33}$	$86.96_{\pm 0.27}$	$88.52_{\pm 0.25}$	$89.43_{\pm 0.29}$
`MOSAIC` (STC)	$86.29_{\pm 0.43}$	$87.04_{\pm 0.37}$	$88.21_{\pm 0.03}$	$89.10_{\pm 0.12}$	$90.18_{\pm 0.25}$
CTS (Ours)	$\mathbf{87.27}_{\pm 0.09}$	$\mathbf{88.06}_{\pm 0.14}$	$\mathbf{89.15}_{\pm 0.15}$	$\mathbf{89.83}_{\pm 0.18}$	$\mathbf{90.89}_{\pm 0.18}$
(b) `OpenScene`: Base dataset of 1000 clips					
Budget B	250	500	1000	2000	4000
`Random`	$72.84_{\pm 1.14}$	$74.19_{\pm 1.05}$	$75.84_{\pm 0.90}$	$78.39_{\pm 0.12}$	$80.38_{\pm 0.55}$
`Uncertainty`	$70.78_{\pm 0.59}$	$69.77_{\pm 0.48}$	$71.12_{\pm 0.38}$	$69.94_{\pm 1.40}$	$73.46_{\pm 0.19}$
`CoreSet`	$76.26_{\pm 0.48}$	$78.12_{\pm 0.87}$	$80.46_{\pm 0.02}$	$81.37_{\pm 0.13}$	$83.63_{\pm 0.36}$
`Chameleon`	$72.97_{\pm 1.72}$	$75.98_{\pm 0.06}$	$79.08_{\pm 0.74}$	$81.35_{\pm 0.39}$	$82.92_{\pm 0.13}$
`RegMix`	$74.18_{\pm 1.36}$	$76.41_{\pm 0.74}$	$78.86_{\pm 0.61}$	$81.62_{\pm 0.35}$	$83.46_{\pm 0.31}$
`DML`	$73.62_{\pm 1.49}$	$76.16_{\pm 0.82}$	$79.27_{\pm 0.55}$	$81.24_{\pm 0.42}$	$83.18_{\pm 0.37}$
`MOSAIC` (STC)	$77.38_{\pm 1.58}$	$79.38_{\pm 1.05}$	$81.68_{\pm 0.52}$	$82.78_{\pm 0.41}$	$84.25_{\pm 0.14}$
CTS (Ours)	$\mathbf{78.76}_{\pm 0.24}$	$\mathbf{80.99}_{\pm 0.38}$	$\mathbf{82.78}_{\pm 0.36}$	$\mathbf{84.04}_{\pm 0.35}$	$\mathbf{85.27}_{\pm 0.17}$
(c) `FMoW-WILDS`: Base dataset of 7686 images					
Budget B	2000	4000	6000	8000	10000
`Random`	$22.69_{\pm 1.77}$	$23.71_{\pm 0.96}$	$23.11_{\pm 0.63}$	$24.99_{\pm 2.09}$	$26.98_{\pm 0.46}$
`Uncertainty`	$22.37_{\pm 1.26}$	$24.71_{\pm 0.93}$	$23.86_{\pm 0.51}$	$27.32_{\pm 1.46}$	$26.87_{\pm 1.21}$
`CoreSet`	$22.78_{\pm 0.51}$	$23.73_{\pm 0.57}$	$24.54_{\pm 1.71}$	$25.44_{\pm 1.43}$	$27.30_{\pm 1.99}$
`Chameleon`	$22.97_{\pm 1.73}$	$23.50_{\pm 0.48}$	$23.78_{\pm 0.43}$	$26.11_{\pm 0.74}$	$26.24_{\pm 0.29}$
`RegMix`	$22.84_{\pm 1.46}$	$23.91_{\pm 0.72}$	$24.28_{\pm 1.18}$	$25.73_{\pm 1.51}$	$27.41_{\pm 1.16}$
`DML`	$23.12_{\pm 1.62}$	$23.86_{\pm 0.81}$	$24.67_{\pm 1.35}$	$25.39_{\pm 1.67}$	$27.16_{\pm 1.28}$
`MOSAIC` (STC)	$21.67_{\pm 1.01}$	$24.10_{\pm 1.69}$	$23.41_{\pm 1.00}$	$25.06_{\pm 1.26}$	$24.95_{\pm 1.47}$
CTS (Ours)	$\mathbf{24.05}_{\pm 1.05}$	$\mathbf{26.15}_{\pm 1.30}$	$\mathbf{27.88}_{\pm 0.89}$	$\mathbf{29.02}_{\pm 1.06}$	$\mathbf{29.92}_{\pm 1.43}$

Autonomous driving. Table 1(a,b) shows that CTS achieves the highest mean EPDMS at every evaluated budget on both navtrain and OpenScene, outperforming all seven baselines. Compared with MOSAIC (STC), the strongest baseline in all ten driving configurations, CTS improves mean EPDMS by 0.71–1.02 points on navtrain and 1.02–1.61 points on OpenScene. CTS also achieves

higher mean EPDMS with fewer additional clips. On navtrain, CTS reaches 87.27 with 100 additional clips, exceeding Random with 400 clips (86.69) and MOSAIC with 200 clips (87.04). On OpenScene, CTS reaches 80.99 with 500 additional clips, exceeding Random with 4000 clips (80.38) and CoreSet with 1000 clips (80.46). Relative to Random, CTS uses only one-quarter as many additional clips on navtrain and one-eighth as many on OpenScene.

Satellite image classification. Table 1(c) shows that CTS achieves the highest mean OOD worst-region accuracy at all five evaluated budgets, exceeding the strongest baseline at each budget by 0.93–3.21 percentage points. This advantage also translates into greater data efficiency: with 6000 additional images, CTS reaches 27.88%, surpassing every baseline trained with 10000 additional images (best: 27.41% from RegMix), while using 40% fewer additional images.

Detailed component and regional results are reported in Appendix D.1 (Tables 10–12). The task-score gains are accompanied by broader improvements in driving component metrics and regional accuracies. Together, these findings support the effectiveness of CTS across distinct tasks and model architectures. Its allocation decisions translate into higher final model performance at matched budgets and, in the comparisons above, higher scores with fewer additional training examples.

5.3 Valuation Ablations

Table 2 compares CTS's joint-state valuation with two matched valuation ablations, M-STC and Static-vector. All three share the fitted response curves, utility, greedy allocation framework, and retraining protocol, differing only in the state at which each candidate increment is valued.

Table 2: Ablations of joint-state valuation with shared response predictions and utility.

(a) `navtrain`: `EPDMS`; base dataset of 460 clips

Variant	$B = 100$	$B = 200$	$B = 400$	$B = 800$	$B = 1600$
M-STC	$86.81_{\pm 0.34}$	$87.68_{\pm 0.26}$	$88.26_{\pm 0.17}$	$89.56_{\pm 0.21}$	$90.67_{\pm 0.23}$
Static-vector	$86.65_{\pm 0.38}$	$86.98_{\pm 0.32}$	$88.49_{\pm 0.20}$	$89.11_{\pm 0.19}$	$90.58_{\pm 0.27}$
CTS (Ours)	$\mathbf{87.27}_{\pm 0.09}$	$\mathbf{88.06}_{\pm 0.14}$	$\mathbf{89.15}_{\pm 0.15}$	$\mathbf{89.83}_{\pm 0.18}$	$\mathbf{90.89}_{\pm 0.18}$

(b) `FMoW-WILDS`: OOD `worst-region accuracy` (%); base dataset of 7686 images

Variant	$B = 2000$	$B = 4000$	$B = 6000$	$B = 8000$	$B = 10000$
M-STC	$22.98_{\pm 1.81}$	$24.86_{\pm 1.57}$	$25.71_{\pm 1.06}$	$27.68_{\pm 1.74}$	$28.85_{\pm 1.61}$
Static-vector	$22.42_{\pm 1.18}$	$24.21_{\pm 1.84}$	$25.87_{\pm 1.24}$	$26.96_{\pm 1.91}$	$28.13_{\pm 1.76}$
CTS (Ours)	$\mathbf{24.05}_{\pm 1.05}$	$\mathbf{26.15}_{\pm 1.30}$	$\mathbf{27.88}_{\pm 0.89}$	$\mathbf{29.02}_{\pm 1.06}$	$\mathbf{29.92}_{\pm 1.43}$

M-STC. M-STC is the matched own-domain-state control: it values each candidate increment at the base state plus its own domain's accumulated response, whereas CTS uses the predicted joint state incorporating all domains' responses. This corresponds to scalarizing each domain's response before composition in M-STC and composing responses before scalarization in CTS. CTS achieves higher mean final model performance at every evaluated budget on both tasks, exceeding M-STC by 0.22–0.89 EPDMS points on navtrain and 1.07–2.17 percentage points in OOD worst-region accuracy on FMoW-WILDS. These results support valuing candidate increments at the predicted joint state to improve data allocation and final model performance.

Static-vector. Static-vector fixes the valuation state at the base metric vector while retaining the count-dependent candidate increment: $\Delta_i^{\mathrm{SV}}(b_i) = u[\boldsymbol{m}_0 + \boldsymbol{d}_i(b_i)] - u[\boldsymbol{m}_0]$. CTS consistently outperforms the Static-vector ablation across all evaluated budgets on both tasks, by 0.31–1.08 EPDMS points on navtrain and 1.63–2.06 percentage points in OOD worst-region accuracy on FMoW-WILDS. The advantage over Static-vector highlights the benefit of updating the predicted joint state as allocation proceeds, beyond retaining count-dependent metric responses alone.

5.4 Allocation Changes and Conditional Data Value

Impact on data allocation. To examine how CTS changes data allocation, Figure 2(a) compares its domain allocations with those of MOSAIC on navtrain at matched budgets. The two methods produce different allocations across all five budgets and three seeds. For example, at $B = 800$,

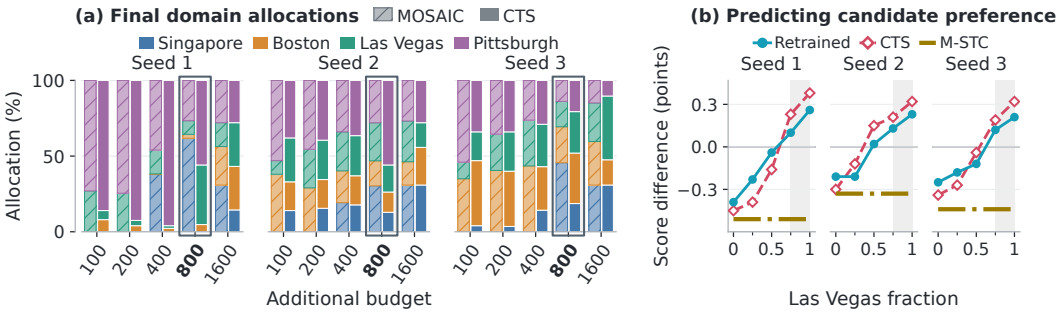


Figure 2: Allocation changes and conditional data value. (a) Domain allocations on navtrain across five budgets and three seeds. (b) Predicted surrogate-utility differences (CTS and M-STC) versus observed EPDMS differences after retraining on OpenScene with fixed 400-clip `Singapore` and `Boston` candidate batches. The `Las Vegas` share increases in a shared 400-clip `Las Vegas`–`Pittsburgh` background. Positive differences favor `Singapore`; negative ones favor `Boston`.

CTS allocates a smaller share to Singapore than MOSAIC in every seed and a larger combined share to Las Vegas and Pittsburgh. Because the within-domain rankings are fixed, these shifts in domain allocation translate directly into different selected training subsets. The corresponding CTS selection sequences are provided in Appendix D.2.

Conditional value prediction. On OpenScene, we test whether CTS's joint-state valuation predicts which of two fixed candidate batches yields higher EPDMS under different background compositions. Each 400-clip background mixes Las Vegas and Pittsburgh clips, with the Las Vegas fraction varied over $\{0, 0.25, 0.5, 0.75, 1\}$. For each background, we retrain two models on the same 1000-clip base dataset augmented with that background, adding the Singapore batch for f_A and the Boston batch for f_B. Their EPDMS difference, $J(f_A) - J(f_B)$, measures relative candidate value under that background. Figure 2(b) compares CTS and M-STC predictions with the observed differences.

As the Las Vegas fraction increases, the observed preference reverses from `Boston` to `Singapore` in all three seeds. CTS captures this reversal, correctly predicting the preferred candidate in all 15 comparisons, versus 8 for M-STC, which always favors `Boston`. At Las Vegas fractions of 0.75 and 1, CTS favors `Singapore` while M-STC favors `Boston`; after retraining, the `Singapore`-augmented models outperform their `Boston`-augmented counterparts by 0.10–0.26 EPDMS points across these six conditions. Across all six city pairs, CTS achieves 80.00% preference accuracy (72/90), compared with 52.22% (47/90) for M-STC; additional pairwise and seed-level results are provided in Appendix D.3. Together, these results show that CTS more accurately predicts background-dependent candidate preferences than M-STC.

Additional analyses. CTS remains effective with a caption-based domain partition (Appendix D.4). Its advantage over M-STC persists across alternative pilot configurations (Appendix D.5) and all three tested response-curve families (Appendix D.6). At the measured navtrain budgets $B \in \{100, 200, 400\}$, fitting and allocation together take less than 76 ms, a negligible fraction of the measured workflow time. The allocator's traced peak memory remains below 419 KiB (Appendix D.7).

6 Conclusion

We introduced joint-state valuation as a principle for reusing fixed single-domain pilot evidence during data allocation. CTS implements this principle by composing signed metric-response predictions before applying the utility at the evolving joint state. This enables state-dependent candidate valuation under nonlinear utilities without additional pilot training. Experiments on autonomous driving and satellite image classification demonstrate improved final model performance, while matched ablations and controlled retraining support the practical benefits of joint-state valuation.

Limitations and future work. Our greedy allocator does not guarantee global optimality. The response model treats each domain's metric response as a function of its own added sample count, without learning how that response changes with allocations to other domains. Future work could use a small number of mixed-domain pilots to estimate allocation-conditioned responses.

AI USE STATEMENT

Generative AI tools were used to assist with language editing for readability and clarity; manuscript organization, including restructuring the appendices; LaTeX layout adjustments for figures and tables; and consistency checks such as standardizing cross-references. They were not used to generate synthetic data, develop theoretical models, formulate mathematical claims, propose hypotheses, design experiments, implement methods, interpret results, or perform any other tasks requiring disclosure. All AI-assisted edits were reviewed and verified by the authors. The authors are responsible for the research, verification, and final content of this work.

ETHICS STATEMENT

This work studies data allocation using existing autonomous-driving and satellite image classification benchmarks. Reuse of these datasets should respect their original licenses and applicable privacy constraints. Allocation decisions depend on the chosen metrics and domain partition, which may not capture all relevant populations or operating conditions. We report component-level and regional performance to make these outcomes more transparent. The driving experiments provide offline benchmark evidence and do not establish safety for real-world deployment.

REPRODUCIBILITY STATEMENT

Appendix A documents dataset preparation and splits, evaluation metrics and allocation utilities, training schedules and random seeds, pilot construction, response fitting, and baseline implementations. Algorithm 1 summarizes the allocation procedures, and Section 5.3 specifies the matched controls. Appendix A.4 distinguishes pilot-feedback and final-evaluation protocols and reports the independent-feedback check on navtrain. The assumptions and derivations underlying the joint-state valuation analysis are provided in Appendix C, while Appendix D details the supplementary evaluations and computational-cost measurements. Source code and experimental configurations are available at `https://anonymous.4open.science/r/ICLR-CTS`.

REFERENCES

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

Kyriakos Axiotis, Vincent Cohen-Addad, Monika Henzinger, Sammy Jerome, Vahab Mirrokni, David Saulpic, David Woodruff, and Michael Wunder. Data-efficient learning via clustering-based sensitivity sampling: Foundation models and beyond. In *International Conference on Machine Learning (ICML)*, pp. 2086–2107, 2024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923, 2025. URL `https://arxiv.org/abs/2502.13923`.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Conference on Robot Learning (CoRL)*, pp. 4709–4722, 2025.

Mayee F. Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. In *International Conference on Learning Representations (ICLR)*, pp. 46089–46132, 2025.

Mayee F. Chen, Tyler Murray, David Heineman, Matt Jordan, Hannaneh Hajishirzi, Christopher Ré, Luca Soldaini, and Kyle Lo. Olmix: A framework for data mixing throughout LM development. In *International Conference on Machine Learning (ICML)*, 2026.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 28706–28719, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Tolga Dimlioglu, Nadine Chang, Maying Shen, Rafid Mahmood, and Jose M. Alvarez. Scaling-aware data selection for end-to-end autonomous driving systems. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17798–17808, 2026.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. DoGE: Domain reweighting with generalization estimation. In *International Conference on Machine Learning (ICML)*, pp. 12895–12915, 2024.

Simin Fan, Maria Ios Glarou, and Martin Jaggi. GRAPE: Optimize data mixture for group robust multi-target adaptive pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 158023–158073, 2025.

Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

Yue He, Dongbai Li, Pengfei Tian, Han Yu, Jiashuo Liu, Hao Zou, and Peng Cui. Domain-wise data acquisition to improve performance under distribution shift. In *International Conference on Machine Learning (ICML)*, pp. 17934–17945, 2024.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J. Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. In *International Conference on Learning Representations (ICLR)*, pp. 58014–58037, 2025.

Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. AutoScale: Scale-aware data mixing for pre-training LLMs. In *Conference on Language Modeling (COLM)*, 2025.

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

Napat Karnchanachari, Dimitris Geromichalos, Kok Seang Tan, Nanxiang Li, Christopher Eriksen, Shakiba Yaghoubi, Noushin Mehdipour, Gianmarco Bernasconi, Whye Kit Fong, Yiluan Guo, and Holger Caesar. Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 629–636, 2024.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. GLISTER: Generalization based data subset selection for efficient and robust learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 8110–8118, 2021.

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

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M. Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, pp. 5637–5664, 2021.

Youngwan Lee and Jongyoul Park. CenterMask: Real-time anchor-free instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13903–13912, 2020.

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, Yu-Gang Jiang, and Jose M. Alvarez. Hydra-MDP: End-to-end multimodal planning with multi-target Hydra-distillation. arXiv preprint arXiv:2406.06978, 2024. URL https://arxiv.org/abs/2406.06978.

Zhenxin Li, Wenhao Yao, Zi Wang, Xinglong Sun, Joshua Chen, Nadine Chang, Maying Shen, Zuxuan Wu, Shiyi Lan, and Jose M. Alvarez. Generalized trajectory scoring for end-to-end multimodal planning. arXiv preprint arXiv:2506.06664, 2025. URL https://arxiv.org/abs/2506.06664.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. RegMix: Data mixture as regression for language model pre-training. In *International Conference on Learning Representations (ICLR)*, pp. 38305–38339, 2025.

Rafid Mahmood, James Lucas, Jose M. Alvarez, Sanja Fidler, and Marc T. Law. Optimizing data collection for machine learning. *Journal of Machine Learning Research (JMLR)*, 26(38):1–52, 2025.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning (ICML)*, pp. 6950–6960, 2020.

Felix Mohr and Jan N. van Rijn. Learning curves for decision making in supervised machine learning: A survey. *Machine Learning*, 113(11):8371–8425, 2024.

OpenScene Contributors. OpenScene: The largest up-to-date 3D occupancy prediction benchmark in autonomous driving. https://github.com/OpenDriveLab/OpenScene, 2023.

Oleksiy Ostapenko, Charles Guille-Escuret, Luke Kumar, Max Tian, Denis Kocetkov, Gopeshh Subbaraj, Raymond Li, Joel Lamy-Poirier, Sebastien Paquet, and Torsten Scholak. Using scaling laws for data source utility estimation in domain-specific pre-training. arXiv preprint arXiv:2507.22250, 2025. URL https://arxiv.org/abs/2507.22250.

Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3D object detection? In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3122–3132, 2021.

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 (NeurIPS)*, pp. 20596–20607, 2021.

Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research (JAIR)*, 48:67–113, 2013.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. Beyond neural scaling laws: Beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 19523–19536, 2022.

Anvith Thudi, Evianne Rovers, Yangjun Ruan, Tristan Thrush, and Chris J. Maddison. MixMin: Finding data mixtures via convex minimization. In *International Conference on Machine Learning (ICML)*, pp. 59492–59508, 2025.

Yifan Wang, Binbin Liu, Fengze Liu, Yuanfan Guo, Jiyao Deng, Xuecheng Wu, Weidong Zhou, Xiaohuan Zhou, and Taifeng Wang. TiKMiX: Efficient semi-dynamic data mixture via data influence for LLM pre-training. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5777–5793, 2026.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning (ICML)*, pp. 1954–1963, 2015.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 69798–69818, 2023.

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning. In *International Conference on Machine Learning (ICML)*, pp. 68725–68745, 2025.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *International Conference on Learning Representations (ICLR)*, pp. 82263–82287, 2025.

Thomson Yen, Andrew Wei Tung Siah, Haozhe Chen, Tianyi Peng, Daniel Guetta, and Hongseok Namkoong. Data mixture optimization: A multi-fidelity multi-scale Bayesian framework. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 98818–98842, 2025.

Lihan Zha, Apurva Badithela, Michael Zhang, Justin Lidard, Jeremy Bao, Emily Zhou, David Snyder, Allen Z. Ren, Dhruv Shah, and Anirudha Majumdar. Guiding data collection via factored scaling curves. arXiv preprint arXiv:2505.07728, 2025. URL `https://arxiv.org/abs/2505.07728`.

Peng Zhou, Li-Wei Fan, and De-Qun Zhou. Data aggregation in constructing composite indicators: A perspective of information loss. *Expert Systems with Applications*, 37(1):360–365, 2010.

A Experimental Settings and Implementation Details

A.1 Datasets and Data Preparation

A.1.1 Autonomous Driving

Data preparation and domains. `OpenScene` is a compact redistribution of nuPlan (Karnchanachari et al., 2024), providing more than 120 hours of driving data sampled at 2 Hz (OpenScene Contributors, 2023). NAVSIM curates `navtrain` from the OpenScene trainval split by filtering out trivial driving scenarios, and `navtest` from the OpenScene test split (Dauner et al., 2024). We use `navtrain` and the OpenScene trainval split as two training configurations, covering curated driving scenarios and the broader set of driving logs, respectively. Following Dimlioglu et al. (2026), we divide each training log into non-overlapping 10-second clips of 20 frames, discarding any shorter remainder. Each clip serves as one allocation unit and inherits its log's city label: Singapore, Boston, Las Vegas, or Pittsburgh. Both driving settings are evaluated on `navtest`.

Base datasets and candidate pools. The clip construction yields 4601 clips for `navtrain` and 32539 for `OpenScene`. Within each setting, we randomly select the base dataset $\mathcal{D}_0$ and use the remaining clips as a candidate pool $\mathcal{D}_{\text{pool}}$, with sizes specified in Table 3. The base dataset is included in every final training set, while the allocator determines how the additional budget is distributed across the four city domains. For each seed, all methods share the base dataset and candidate pool. Each budget B counts newly selected clips, giving a final dataset of $|\mathcal{D}_0| + B$ clips.

Table 3: Autonomous-driving data configurations. Counts and budgets are in 10-second clips.

Dataset	Total	Base	Pool	Additional budgets B
`navtrain`	4601	460	4141	$\{100, 200, 400, 800, 1600\}$
`OpenScene`	32539	1000	31539	$\{250, 500, 1000, 2000, 4000\}$

A.1.2 FMoW-WILDS

Temporal splits and geographic domains. Each example in `FMoW-WILDS` is a 224×224 RGB satellite image labeled with one of 62 building or land-use categories (Koh et al., 2021). We use the official training, OOD validation, and OOD test splits to evaluate generalization over time. The training split contains images taken before 2013, the OOD validation split contains images from 2013–2015, and the OOD test split contains images from 2016 onward (Table 4). Using the dataset's country-to-region mapping, we group images into Africa, the Americas, Asia, Europe, and Oceania. These five regions serve as allocation domains in the training split and evaluation groups in the later-year splits. Only images from the training split enter the base dataset or candidate pool.

Table 4: FMoW-WILDS splits used in this study. Only the training split supplies allocation data.

Split	Image years	Images	Role
Training	Before 2013	76863	Base dataset and candidate pool
OOD validation	2013–2015	19915	Out-of-distribution (OOD) validation
OOD test	2016 onward	22108	Out-of-distribution (OOD) evaluation

Base dataset and candidate pool. From the 76863 images in the official training split, we uniformly sample 7686 images (approximately 10%) without replacement to form the base dataset $\mathcal{D}_0$. We use a fixed sampling seed of 0, without stratification by region or class. The remaining 69177 images form the candidate pool $\mathcal{D}_{\text{pool}}$, partitioned by geographic region. This base/pool partition is held fixed across all methods and all three experimental seeds. We consider additional budgets $B \in \{2000, 4000, 6000, 8000, 10000\}$. Each budget counts newly selected images that augment the entire base dataset, giving a final training size of $|\mathcal{D}_0| + B$ images, as listed in Table 5.

Table 5: FMoW-WILDS data configurations with a base dataset of 7686 images (approximately 10% of the training split). All counts are in images.

Additional budget B	2000	4000	6000	8000	10000
Final training size	9686	11686	13686	15686	17686

A.2 Evaluation Metrics and Allocation Utilities

During allocation, the task-specific utility u maps the predicted metric vector to an estimated allocation value. Model performance is evaluated using the prespecified task score J.

A.2.1 Autonomous Driving: Metrics and Scores

The metric vector contains the nine EPDMS components defined in Table 6:

$$\boldsymbol{m} = (\mathrm{NC}, \mathrm{DAC}, \mathrm{DDC}, \mathrm{TLC}, \mathrm{EP}, \mathrm{TTC}, \mathrm{LK}, \mathrm{HC}, \mathrm{EC})^\top. \tag{13}$$

For allocation, each component is the arithmetic mean of the corresponding metric across driving scenes, expressed on the $[0, 1]$ scale. The component definitions follow Cao et al. (2025).

Table 6: Component metrics used in the driving metric vector (Cao et al., 2025). Higher values indicate better performance for every component.

Metric	Meaning
NC	No at-fault collisions
DAC	Drivable area compliance
DDC	Driving direction compliance
TLC	Traffic light compliance
EP	Ego progress along the route
TTC	Time-to-collision safety
LK	Lane keeping
HC	History comfort
EC	Extended comfort across consecutive predictions

The official EPDMS and the allocation surrogate share the scoring form

$$\phi[\boldsymbol{m}; \boldsymbol{w}] = \mathrm{NC} \cdot \mathrm{DAC} \cdot \mathrm{DDC} \cdot \mathrm{TLC} \cdot \frac{\sum_{c \in \mathcal{W}} w_c m_c}{\sum_{c \in \mathcal{W}} w_c}, \tag{14}$$

where $\mathcal{W} = \{5, 6, 7, 8, 9\}$ indexes EP, TTC, LK, HC, and EC in Equation 13. The weight vector is $\boldsymbol{w} = (w_5, w_6, w_7, w_8, w_9)^\top$, with default $\boldsymbol{w}_0 = (5, 5, 2, 2, 2)^\top$. This form combines four multiplicative penalties with a weighted average of progress, safety, lane-keeping, and comfort metrics.

We evaluate driving models on `navtest` using the unmodified single-stage EPDMS implementation provided by GTRS (Li et al., 2025).[1] The scorer applies human-reference filtering and its prescribed treatment of LK at intersections. When the adjacent-frame information required for EC is unavailable, EC is excluded from that scene's weighted aggregation, and the remaining weights are renormalized. Let $\widetilde{\boldsymbol{m}}_j(f)$ denote the component vector after the scorer's prescribed processing for scene j, and let $\boldsymbol{w}_j$ denote its effective weights; zero weights omit unavailable components from both sums. The official task score averages the resulting scores over N evaluated scenes:

$$J(f) = \frac{100}{N} \sum_{j=1}^{N} \phi[\widetilde{\boldsymbol{m}}_j(f); \boldsymbol{w}_j]. \tag{15}$$

For allocation, we apply the same scoring form to the scene-averaged component vector with fixed default weights:

$$u_{\mathrm{AD}}[\boldsymbol{m}] = 100 \cdot \phi[\boldsymbol{m}; \boldsymbol{w}_0]. \tag{16}$$

The surrogate therefore uses a fixed denominator of 16. The factor of 100 places both J and u_{AD} on the reported 0–100 score scale and does not change allocation preferences. Thus, J averages scene-level scores, whereas u_{AD} scores the scene-averaged component vector. These operations generally differ because scoring is nonlinear; the surrogate also uses fixed component weights rather than scene-specific availability adjustments.

[1] GTRS commit `92a740d`, using `run_pdm_score_one_stage.py`.

A.2.2 Driving Surrogate: Approximation and Cost

CTS values candidate additions at a predicted joint state of scene-averaged metrics using u_{AD}. This requires fitting only domain-level mean responses, keeping fitting and allocation costs low.

A scene-level alternative would predict the joint metric state of each evaluation scene and value candidate additions by the change in the average predicted scene score. For K training domains, N evaluation scenes, and d metrics, independent fitting requires one curve per domain–scene–metric combination (KNd curves), compared with one per domain–metric pair for CTS (Kd curves). Each candidate addition changes only one domain's count, requiring Nd response increments and N updated scene scores; CTS instead updates d mean metrics and computes the surrogate once. Table 7 summarizes these counts for a fixed evaluation set with unchanged scoring masks. Both representations reuse fitted curves without model retraining or evaluation during allocation.

Table 7: Computational requirements of a scene-level representation following the computation order of J and the mean-metric representation used by CTS.

Requirement	Scene-level (J order)	Mean-metric (CTS)
Fitted response curves	KNd	Kd
Response increments per candidate	Nd	d
Score computations per candidate	N	1

A.2.3 FMoW-WILDS

The metric vector in our satellite image classification task contains $d = 5$ regional accuracies for Africa, the Americas, Asia, Europe, and Oceania, in that order:

$$\boldsymbol{m} = (m_1, m_2, m_3, m_4, m_5)^\top, \qquad u_{\mathrm{FMoW}}[\boldsymbol{m}] = \min\{m_1, m_2, m_3, m_4, m_5\}. \tag{17}$$

Each regional accuracy is the fraction of correctly classified images within that region, expressed on the $[0, 1]$ scale. Regional OOD-validation accuracies provide the feedback for response fitting and allocation, while OOD-test accuracies evaluate the retrained models. The task score is:

$$J(f) = \min_{c \in \{1,\ldots,5\}} m_c(f) = u_{\mathrm{FMoW}}[\boldsymbol{m}(f)]. \tag{18}$$

So allocation and model evaluation use the same aggregation on predicted and measured regional accuracies, respectively. Reported accuracies are expressed as percentages.

A.3 Training and Retraining Details

Table 8 summarizes the base-training and final-retraining settings.

Autonomous driving. We adopt the training recipe of Dimlioglu et al. (2026), using Adam (Kingma & Ba, 2015) without weight decay, a constant learning rate, and a per-GPU batch size of 20 on eight A100 GPUs. For `navtrain`, the learning rate is 10^{-4}. The base model and allocations with up to 800 additional clips are trained for 60 epochs, while the 1600-clip allocation is trained for 50 epochs. All `navtrain` configurations use three random seeds: 0, 2025, and 424242. For `OpenScene`, training uses a learning rate of 2×10^{-4} for 40 epochs. Budgets of 250, 500, and 1000 clips use the same three seeds, while budgets of 2000 and 4000 clips use seeds 0 and 2025. We report means and standard deviations across the corresponding runs. Final performance is evaluated using the checkpoint at the end of the prescribed training schedule for each setting and budget.

Hydra-MDP initializes its encoder with a pretrained VoVNetV2-99 backbone (Lee & Park, 2020; Park et al., 2021) and uses a trajectory vocabulary of 16384. In addition to trajectory imitation, it learns from rule-compliance scores assigned to candidate trajectories through distillation (Li et al., 2024). Rule-compliance distillation is enabled for `navtrain` but disabled for `OpenScene`, where computing the required compliance scores incurs substantial preprocessing cost.

FMoW-WILDS. We use DenseNet-121 for `FMoW-WILDS` (Koh et al., 2021) with ImageNet-pretrained weights (Deng et al., 2009) and a 62-class classification head. The 224×224 RGB images are converted to tensors and normalized using the ImageNet channel means $(0.485, 0.456, 0.406)$

Table 8: Training configurations across the three dataset settings.

	Autonomous driving		Satellite classification
Setting	`navtrain`	`OpenScene`	`FMoW-WILDS`
Model	Hydra-MDP		DenseNet-121
Pretrained initialization	VoVNetV2-99 encoder		ImageNet
Trajectory vocabulary	16384		—
Rule-compliance distillation	Enabled	Disabled	—
Optimizer	Adam, no weight decay		
Batch size	20 per GPU		64 total
Hardware	8×A100		RTX 4090
Initial learning rate	10^{-4}	2×10^{-4}	10^{-4}
Learning-rate schedule	Constant		$\times 0.96$ per epoch
Epoch limit	60 / 50	40	50
Early stopping	Disabled		Patience 5
Number of seeds	3	3 / 2	3
Reported statistics	Mean ± standard deviation		

Epoch limits and early-stopping settings refer to base training and final retraining; pilot continuation schedules are specified in Section A.4. For `navtrain`, the epoch limit is 50 at $B = 1600$ and 60 otherwise. For `OpenScene`, three seeds are used at $B \leq 1000$ and two at larger budgets.

and standard deviations $(0.229, 0.224, 0.225)$. We optimize the cross-entropy loss using Adam with a batch size of 64, an initial learning rate of 10^{-4}, and no weight decay. Training runs for at most 50 epochs, with the learning rate multiplied by 0.96 after each epoch. After each epoch, we evaluate worst-region accuracy on the OOD validation split and retain the checkpoint with the highest value. Training stops early if this validation metric does not improve for five consecutive epochs. Final retrained models are evaluated on the OOD test split using their retained checkpoints. Results are reported as mean ± standard deviation over three random seeds: 0, 1, and 2.

A.4 Pilot, Response-Fitting, and Allocation Details

STC and CTS use the same single-domain pilot runs, with response fitting and allocation performed independently for each seed. The task models and base-training recipes are specified in Section A.3.

Pilot initialization. Following the driving pilot protocol of Dimlioglu et al. (2026), each driving pilot starts from the same-seed base model's final checkpoint, which defines f_0. For FMoW-WILDS, the same-seed base checkpoint with the highest OOD-validation worst-region accuracy defines f_0 and initializes every pilot. Each pilot continues training on the base dataset augmented with selected examples from a single domain.

Within-domain ranking. Driving clips are ranked by ascending EPDMS obtained from f_0 within each city. FMoW-WILDS images are ranked by ascending true-class confidence from f_0 within each region. These rankings remain fixed and are shared by pilot construction and final allocation.

Pilot counts and continuation. For navtrain, we add 100 or 200 clips per domain and train each pilot for 10 epochs; for OpenScene, we add 200 or 400 clips per domain and train each pilot for 5 epochs. For FMoW-WILDS, each pilot adds 500 or 1000 images from a single source region and trains for 5 additional epochs.

Pilot feedback. Given the constraints on reserving separate validation data in the `navtrain` and `OpenScene` settings, we follow the MOSAIC protocol (Dimlioglu et al., 2026) and evaluate both base and pilot models on `navtest`. Driving pilot feedback is collected from the final checkpoint after the prescribed continuation schedule and used for response fitting and allocation. The same split is used for final performance reporting, which limits the independence of the driving evaluation. For FMoW-WILDS, we record the five regional OOD-validation accuracies at the final pilot epoch and use these observations for response fitting and allocation. The OOD test split is reserved for final evaluation, as described in Section A.3.

Driving feedback-split check. To assess sensitivity to this reuse, we reran MOSAIC and CTS on `navtrain` with 10% of the training data held out at the log level for validation and excluded from model training and allocation. Pilot feedback for response fitting and allocation came exclusively from this held-out set. We used `navtest` only to evaluate the final retrained models.

Across the five budgets in Table 9, each method's mean EPDMS changes by at most 0.38 points relative to the original protocol. CTS also retains higher mean EPDMS than MOSAIC at every budget, with its advantage changing by at most 0.08 points between the two protocols. We therefore retain the original-protocol results in the main tables for direct comparison with prior work. The FMoW-WILDS experiments provide complementary evidence using separate validation and test splits. CTS achieves higher mean worst-region test accuracy than all evaluated baselines after retraining at every evaluated budget, confirming its consistent advantage under this evaluation protocol.

Table 9: Sensitivity to the pilot-feedback split on `navtrain`. Entries report mean EPDMS (0–100; higher is better), with standard deviations as subscripts across seeds 0, 2025, and 424242. Validation denotes the 10% log-level holdout excluded from training and allocation; both protocols use `navtest` for final evaluation. B counts additional clips.

Pilot feedback	Method	$B = 100$	$B = 200$	$B = 400$	$B = 800$	$B = 1600$
`navtest`	MOSAIC	$86.29_{0.43}$	$87.04_{0.37}$	$88.21_{0.03}$	$89.10_{0.12}$	$90.18_{0.25}$
`navtest`	CTS	$87.27_{0.09}$	$88.06_{0.14}$	$89.15_{0.15}$	$89.83_{0.18}$	$90.89_{0.18}$
Validation	MOSAIC	$86.01_{0.39}$	$86.68_{0.42}$	$87.87_{0.05}$	$88.81_{0.16}$	$89.84_{0.22}$
Validation	CTS	$86.92_{0.12}$	$87.77_{0.11}$	$88.77_{0.19}$	$89.53_{0.15}$	$90.63_{0.21}$

Response construction. STC fits task-score gains $J(f_i(n)) - J(f_0)$ as scalar summaries and values candidate additions based only on the allocation to their own domain. CTS fits component responses $r_{ic}(n) = m_c(f_i(n)) - m_c(f_0)$, for $c \in \{1, \ldots, d\}$, retaining metric-level predictions to value candidate additions at the evolving predicted joint metric state. Both use the corresponding same-seed base checkpoint f_0 and the same feedback split as the pilot. Driving component percentages are divided by 100 before subtracting the base values; FMoW regional accuracies are expressed on the $[0, 1]$ scale. Each fit includes the base observation at $n = 0$, where both component responses and scalar gains are zero. The fitting-count sets are therefore $\{0, 100, 200\}$ for `navtrain`, $\{0, 200, 400\}$ for `OpenScene`, and $\{0, 500, 1000\}$ for FMoW-WILDS. The four driving domains and nine metrics yield 36 component-response curves per seed; the five FMoW source regions and five evaluation regions yield 25. M-STC and Static-vector reuse CTS's fitted component-response curves and do not fit separate scalar-gain curves.

Response fitting. We use the same exponential family for scalar gains and component responses, following Dimlioglu et al. (2026). Let $\mathcal{P}$ denote the nonzero pilot counts specified above. For a target $y(n)$, the fitted function and least-squares objective are

$$g(n; a, \tau) = a(1 - e^{-n/\tau}), \qquad \mathcal{L}(a, \tau) = \sum_{n \in \mathcal{P} \cup \{0\}} \big(g(n; a, \tau) - y(n)\big)^2. \tag{19}$$

The implementation includes the base observation $(0, 0)$. For finite parameters with $\tau \neq 0$, $g(0; a, \tau) = 0$ by construction, so this observation contributes zero residual and imposes no additional parameter constraint. The fitted curves are $\widehat{s}_i(n) = g(n; a_i, \tau_i)$ for STC and $\widehat{r}_{ic}(n) = g(n; a_{ic}, \tau_{ic})$ for CTS. Each parameter pair is estimated independently, without sharing parameters across domains or metric components. Both tasks use SciPy's `curve_fit` with unweighted residuals and `maxfev=100000`.

Prediction and allocation. We follow Algorithm 1, allocating one example per step according to the corresponding marginal-value rule, subject to domain capacities, until the additional budget B is exhausted. When feasible domains have equal predicted marginal gains, we select the domain whose name is lexicographically greatest. The fitted curves remain fixed throughout allocation. They are evaluated between observed counts, including between zero and the smallest nonzero pilot count, and extrapolated beyond the largest pilot count when needed. During allocation with budget B, all queried counts for domain i satisfy $0 \leq n \leq \min(B, |\mathcal{D}_i|)$. Predicted responses vary with the allocated counts; CTS also updates the joint metric state at which candidate increments are valued.

Allocating b_i examples to domain i selects the first b_i in its fixed within-domain ranking. Ties in within-domain ranking scores are broken by ascending lexicographic order of sample identifiers.

Algorithm 1: STC and CTS from shared single-domain pilots

Inputs. Base dataset $\mathcal{D}_0$; ranked candidate domains $\{\mathcal{D}_i\}_{i=1}^{K}$; budget B; pilot counts $\mathcal{P}$; task score J; metric vector $\boldsymbol{m}(\cdot)$ and utility u for CTS.

Shared pilot evidence. Train the base model f_0 on $\mathcal{D}_0$. For each domain i and count $n \in \mathcal{P}$, train a pilot model $f_i(n)$ using the base dataset plus the top n ranked examples from domain i. Evaluate J and record $\boldsymbol{m}$ for the base and pilot models, with $\boldsymbol{m}_0 = \boldsymbol{m}(f_0)$.

Response fitting.

Independent scalar-gain valuation (STC)	Joint-state valuation (CTS)
Fit $\widehat{s}_i$ to the scalar gains $J(f_i(n)) - J(f_0)$.	Fit $\widehat{\boldsymbol{r}}_i$ componentwise to $\boldsymbol{m}(f_i(n)) - \boldsymbol{m}_0$.
Marginal-value rules at allocation $\boldsymbol{b}$.	
	$\widehat{\boldsymbol{m}}(\boldsymbol{b}) = \boldsymbol{m}_0 + \sum_{k=1}^{K} \widehat{\boldsymbol{r}}_k(b_k)$.
	$\boldsymbol{d}_i(b_i) = \widehat{\boldsymbol{r}}_i(b_i + 1) - \widehat{\boldsymbol{r}}_i(b_i)$.
$\Delta_i = \widehat{s}_i(b_i + 1) - \widehat{s}_i(b_i)$.	$\Delta_i = u[\widehat{\boldsymbol{m}}(\boldsymbol{b}) + \boldsymbol{d}_i(b_i)] - u[\widehat{\boldsymbol{m}}(\boldsymbol{b})]$.

Allocation. For each method, initialize $\boldsymbol{b} = \boldsymbol{0}$. While $\sum_i b_i < B$, compute the marginal gains for all feasible domains using the corresponding rule above at the current allocation $\boldsymbol{b}$, select $j = \arg\max_{i:b_i<|\mathcal{D}_i|} \Delta_i$ using the same tie-breaking rule, and increment b_j by one.

Final retraining. For each method, augment the base dataset with the examples selected by its final allocation $\boldsymbol{b}$, retrain the model on the resulting dataset to obtain $f_{\boldsymbol{b}}$, and evaluate $J(f_{\boldsymbol{b}})$.

A.5 Baseline Details

All seven baselines use the same base dataset, candidate pool, and additional data budgets as CTS. Final models follow the task-specific training and evaluation protocols described in Section A.3.

Random. Random samples additional examples uniformly from the candidate pool without replacement. For both tasks, a single shuffled ordering defines nested subsets across budgets: the first B examples form the selected subset.

Uncertainty. Uncertainty ranks candidates by predictive entropy, $H(x) = -\sum_{\xi} p_\xi(x) \log p_\xi(x)$, and selects the B highest-scoring examples. For driving, $p_\xi(x)$ is the predicted probability of trajectory candidate ξ, and frame-level entropies are averaged within each clip (Dimlioglu et al., 2026). For FMoW-WILDS, the probabilities correspond to the 62 image classes.

CoreSet. CoreSet greedily expands coverage in the base model's feature space (Sener & Savarese, 2018). Starting from the base dataset, each step selects the unselected pool example with the largest Euclidean distance to its nearest already included example, until B examples have been added. For driving, we follow the geometric selection procedure of Dimlioglu et al. (2026). For FMoW-WILDS, distances are computed between the pooled features immediately before the classification layer.

Chameleon. Chameleon assigns domain sampling weights based on representation similarities (Xie et al., 2025). For both tasks, we follow the adaptation of Dimlioglu et al. (2026): sample features are averaged within each domain, a domain-affinity matrix is constructed, and kernel-based scores determine the domain sampling weights. The additional subset is sampled according to these weights.

RegMix. RegMix learns a performance predictor from proxy training runs on different data mixtures and uses its predictions to select a mixture (Liu et al., 2025). We use the ridge-regression variant, with domain proportions as inputs and official EPDMS for driving or OOD-validation worst-region accuracy for FMoW-WILDS as the target.

Data Mixing Laws (DML). DML models domain-specific validation losses as parametric functions of training-mixture proportions and combines these predictions to optimize the mixture (Ye et al., 2025). We adapt its component-wise prediction approach to the metric representation used by CTS, fitting each component by nonlinear least squares. For driving, DML predicts the nine scene-

averaged component metrics and applies the same EPDMS surrogate utility as CTS. For FMoW-WILDS, it predicts the five regional OOD-validation accuracies and takes their minimum.

Mixture training and search. For RegMix and DML, each method uses 16 additional mixture-training runs per dataset setting and seed, with domain proportions sampled from a symmetric Dirichlet distribution with concentration parameter one. Each run augments the base dataset with 200 clips on `navtrain`, 400 clips on `OpenScene`, or 1000 images on `FMoW-WILDS`. These runs use the same task model, base checkpoint, and training recipe as the single-domain pilots, continuing training for 10, 5, and 5 epochs, respectively. RegMix and DML search the same sampled candidate proportions and each selects the mixture with the highest predicted objective. Each method reuses its selected proportions across all reported additional budgets, followed by final retraining.

MOSAIC. MOSAIC fits domain-specific scalar curves to task-score gains observed in single-domain pilots and greedily allocates examples using their predicted marginal gains (Dimlioglu et al., 2026). It instantiates independent scalar-gain valuation (STC), where each candidate's predicted marginal value depends only on its own domain's allocated count and does not change with allocations to other domains. CTS and MOSAIC share the domain partition, within-domain rankings, and single-domain pilots. Response fitting and greedy allocation follow the protocol in Section A.4.

B METRIC-RESPONSE ANALYSIS

B.1 ANALYSIS PROTOCOL

Task and pilot evidence. The motivational analyses use single-domain pilots with Hydra-MDP on the NAVSIM `navtrain` split (Dauner et al., 2024; Li et al., 2024). We analyze pilot counts of 100 and 200 using three random seeds, 0, 2025, and 424242, displayed as Seed 1, Seed 2, and Seed 3, respectively. Metric definitions and pilot training protocols are provided in Section A.2 and Section A.4. The following comparisons support the analysis in Section 3.3.

Paired metric responses and scalar gains. For random seed ℓ, the metric-response vector is

$$\boldsymbol{r}_i^{(\ell)}(n) = \boldsymbol{m}\Big(f_i^{(\ell)}(n)\Big) - \boldsymbol{m}\Big(f_0^{(\ell)}\Big), \tag{20}$$

where each component of $\boldsymbol{m}$ is expressed on the $[0, 1]$ scale. For figures and descriptive reporting, we multiply component responses by 100 to express changes in percentage points. Positive and negative responses denote improvement and degradation, respectively. Scalar gains are the differences in EPDMS between each pilot and its same-seed base model, $J(f_i^{(\ell)}(n)) - J(f_0^{(\ell)})$, expressed in points on the reported 0–100 scale. At each pilot count, we average scalar gains and signed component responses across seeds to compare domains. These averages are used only for descriptive analysis; response fitting and allocation are performed independently for each seed.

Within-domain consistency. At each pilot count $n \in \{100, 200\}$, we compute the cosine similarity between every pair of seed-specific response vectors within each domain:

$$\cos(\boldsymbol{a}, \boldsymbol{a}') = \frac{\boldsymbol{a}^\top \boldsymbol{a}'}{\|\boldsymbol{a}\|_2 \|\boldsymbol{a}'\|_2}.$$

Here, $\boldsymbol{a} = \boldsymbol{r}_i^{(\ell)}(n)$ and $\boldsymbol{a}' = \boldsymbol{r}_i^{(\ell')}(n)$ are response vectors from two distinct seeds $\ell \neq \ell'$ for the same domain i and pilot count n. Three seeds yield three pairs per domain, giving 12 cosine similarities across the four domains at each count. We report the minimum of these 12 values.

B.2 ADDITIONAL METRIC-RESPONSE ANALYSIS

The supplementary point plot and heatmaps use the same underlying pilot observations and same-seed base subtraction as Figure 1. The heatmaps display individual seed responses.

Responses at a pilot sample count of 200. The minimum within-domain pairwise cosine similarity is 0.89, indicating consistent response directions across seeds at this pilot count. Boston and Las Vegas have similar mean EPDMS gains of 1.56 and 1.63 points, respectively, but retain distinct component-response patterns. Their TLC responses have opposite signs, and their mean EC decreases are 0.61 and 1.40 percentage points, respectively. Together, these results show consistent metric-response directions across seeds within each domain, while demonstrating that similar scalar gains can conceal distinct metric-level trade-offs across domains.

		NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC
Singapore	Seed 1	+0.14	+0.81	-0.17	+0.11	-0.20	+0.14	+0.30	-0.01	-0.32
	Seed 2	+0.14	+1.02	-0.03	+0.12	-0.23	+0.13	+0.33	-0.13	-0.35
	Seed 3	+0.14	+0.66	-0.13	+0.07	-0.23	+0.23	+0.26	-0.06	-0.30
Boston	Seed 1	+0.77	+0.23	+0.16	-0.08	-0.19	+0.70	+0.23	-0.05	-0.51
	Seed 2	+0.43	+0.18	+0.12	-0.10	-0.16	+0.72	+0.34	-0.02	-0.17
	Seed 3	+0.54	+0.28	+0.19	-0.07	-0.26	+0.88	+0.21	-0.04	-0.52
Las Vegas	Seed 1	+0.44	+0.27	+0.11	+0.12	-0.27	+0.92	+0.15	-0.08	-0.90
	Seed 2	+0.56	+0.15	+0.06	+0.21	-0.33	+0.75	+0.19	-0.12	-0.71
	Seed 3	+0.83	+0.20	+0.08	+0.09	-0.42	+0.66	+0.18	-0.04	-1.15
Pittsburgh	Seed 1	+0.35	+0.65	+0.23	+0.16	-0.40	+0.39	+0.37	-0.03	-0.33
	Seed 2	+0.37	+0.61	+0.21	+0.12	-0.26	+0.66	+0.23	-0.02	-0.41
	Seed 3	+0.23	+0.47	+0.21	+0.11	-0.36	+0.38	+0.23	+0.00	-0.23

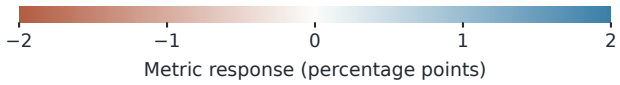


Figure 3: Seed-level metric responses at $n = 100$ (Figure 1).

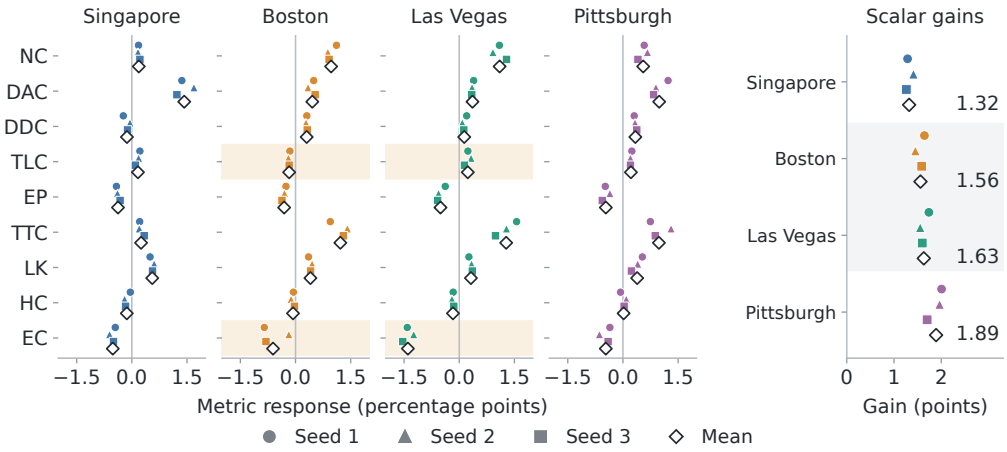


Figure 4: Metric responses and EPDMS gains at a pilot sample count of $n = 200$.

		NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC
Singapore	Seed 1	+0.18	+1.36	-0.23	+0.22	-0.42	+0.21	+0.50	-0.04	-0.45
	Seed 2	+0.17	+1.69	-0.05	+0.18	-0.39	+0.20	+0.61	-0.20	-0.60
	Seed 3	+0.22	+1.23	-0.12	+0.10	-0.32	+0.34	+0.56	-0.17	-0.50
Boston	Seed 1	+1.11	+0.49	+0.31	-0.15	-0.26	+0.95	+0.35	-0.06	-0.85
	Seed 2	+0.88	+0.34	+0.28	-0.19	-0.30	+1.41	+0.45	-0.12	-0.18
	Seed 3	+0.91	+0.53	+0.32	-0.17	-0.37	+1.30	+0.41	-0.03	-0.80
Las Vegas	Seed 1	+1.09	+0.39	+0.20	+0.24	-0.38	+1.56	+0.26	-0.17	-1.41
	Seed 2	+0.92	+0.35	+0.08	+0.32	-0.56	+1.28	+0.33	-0.20	-1.24
	Seed 3	+1.29	+0.34	+0.12	+0.14	-0.59	+0.99	+0.36	-0.15	-1.54
Pittsburgh	Seed 1	+0.58	+1.23	+0.31	+0.24	-0.48	+0.75	+0.53	-0.07	-0.36
	Seed 2	+0.67	+0.90	+0.33	+0.20	-0.36	+1.31	+0.40	+0.09	-0.64
	Seed 3	+0.41	+0.83	+0.38	+0.20	-0.56	+0.88	+0.23	+0.03	-0.40

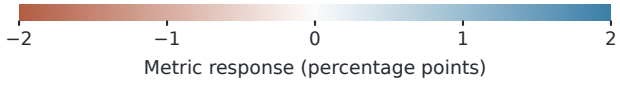


Figure 5: Seed-level metric responses at $n = 200$ (Figure 4).

C Analysis of Joint-State Valuation

Comparison setting. This appendix provides the derivations for Section 4.3. We compare CTS's joint-state valuation with M-STC's own-domain-state valuation at the same allocation $\boldsymbol{b}$, using the same base metric vector $\boldsymbol{m}_0$, fitted domain responses $\widehat{\boldsymbol{r}}_i$, and utility u. Their predicted value functions are

$$\widehat{v}_{\mathrm{CTS}}(\boldsymbol{b}) = u\Big[\boldsymbol{m}_0 + \sum_i \widehat{\boldsymbol{r}}_i(b_i)\Big],$$

$$\widehat{v}_{\text{M-STC}}(\boldsymbol{b}) = u[\boldsymbol{m}_0] + \sum_i \big(u[\boldsymbol{m}_0 + \widehat{\boldsymbol{r}}_i(b_i)] - u[\boldsymbol{m}_0]\big).$$

All evaluated states must lie in the domain of u. The analysis concerns the additive prediction model, without assuming that actual joint-training responses are additive. Independently fitting scalar-gain curves need not produce the M-STC value function, even with the same curve family and fitting procedure. Any state transformation must be included in the mapping being analyzed.[2]

Proposition (Affine equivalence). With shared response predictions and an affine utility, CTS and M-STC assign identical predicted allocation values and marginal gains. Their greedy allocation sequences coincide when initialized at the same allocation and using identical feasibility and tie-breaking rules.

Proof. For $u[\boldsymbol{m}] = \boldsymbol{w}^\top \boldsymbol{m} + c$, both value functions reduce to

$$\widehat{v}_{\mathrm{CTS}}(\boldsymbol{b}) = \widehat{v}_{\text{M-STC}}(\boldsymbol{b}) = u[\boldsymbol{m}_0] + \sum_i \boldsymbol{w}^\top \widehat{\boldsymbol{r}}_i(b_i).$$

Taking finite differences gives

$$\Delta_i^{\mathrm{CTS}}(\boldsymbol{b}) = \Delta_i^{\text{M-STC}}(b_i) = \boldsymbol{w}^\top \boldsymbol{d}_i(b_i).$$

Starting from the same allocation, identical marginal gains, feasible candidate sets, and tie-breaking rules yield the same next selection. Induction therefore gives identical allocation sequences. These statements hold in exact arithmetic; numerical implementations must also treat ties consistently.

Marginal adjustment. We suppress the dependence on the current allocation $\boldsymbol{b}$ where unambiguous. For a feasible candidate domain i, define

$$\boldsymbol{x}_i = \boldsymbol{m}_0 + \widehat{\boldsymbol{r}}_i(b_i), \qquad \boldsymbol{c}_i = \sum_{j \neq i} \widehat{\boldsymbol{r}}_j(b_j), \qquad \boldsymbol{d}_i = \widehat{\boldsymbol{r}}_i(b_i + 1) - \widehat{\boldsymbol{r}}_i(b_i).$$

Thus, $\widehat{\boldsymbol{m}}(\boldsymbol{b}) = \boldsymbol{x}_i + \boldsymbol{c}_i$, and $\boldsymbol{d}_i$ is the candidate increment $\boldsymbol{d}_i(b_i)$ defined in the main text.

The two marginal values are

$$\Delta_i^{\text{M-STC}} = u[\boldsymbol{x}_i + \boldsymbol{d}_i] - u[\boldsymbol{x}_i], \qquad \Delta_i^{\mathrm{CTS}} = u[\boldsymbol{x}_i + \boldsymbol{c}_i + \boldsymbol{d}_i] - u[\boldsymbol{x}_i + \boldsymbol{c}_i].$$

The marginal adjustment measures the change in predicted marginal gain when the same candidate increment is valued at the joint state rather than its own-domain state:

$$\begin{aligned}\kappa_i &= \Delta_i^{\mathrm{CTS}} - \Delta_i^{\text{M-STC}} \\ &= u[\boldsymbol{x}_i + \boldsymbol{c}_i + \boldsymbol{d}_i] - u[\boldsymbol{x}_i + \boldsymbol{c}_i] - u[\boldsymbol{x}_i + \boldsymbol{d}_i] + u[\boldsymbol{x}_i].\end{aligned}$$

This exact finite-difference identity requires neither differentiability nor infinitesimal increments.

Suppose u is twice continuously differentiable on an open neighborhood of $\{\boldsymbol{x}_i + s\boldsymbol{c}_i + t\boldsymbol{d}_i : s, t \in [0,1]\}$, and let $\boldsymbol{H}_u$ denote its Hessian. With $F(s,t) = u[\boldsymbol{x}_i + s\boldsymbol{c}_i + t\boldsymbol{d}_i]$, the fundamental theorem of calculus gives

$$\begin{aligned}\kappa_i &= F(1,1) - F(1,0) - F(0,1) + F(0,0) \\ &= \int_0^1 \int_0^1 \frac{\partial^2 F}{\partial s \partial t}(s,t)\, dt\, ds \\ &= \int_0^1 \int_0^1 \boldsymbol{c}_i^\top \boldsymbol{H}_u(\boldsymbol{x}_i + s\boldsymbol{c}_i + t\boldsymbol{d}_i)\boldsymbol{d}_i\, dt\, ds.\end{aligned}$$

[2] For example, applying nonlinear clipping before an affine utility does not generally preserve the affine equivalence established below.

The adjustment therefore depends on utility curvature coupling other domains' accumulated responses with the candidate increment. Nonlinearity alone does not ensure a nonzero adjustment: the relevant directional curvature may vanish or its contributions may cancel.

If $\|\boldsymbol{H}_u(\boldsymbol{m})\|_{\mathrm{op}} \le L$ throughout this set, then

$$|\kappa_i| \le L\|\boldsymbol{c}_i\|_2\|\boldsymbol{d}_i\|_2,$$

where $\|\cdot\|_{\mathrm{op}}$ is the operator norm induced by the Euclidean norm. This bound concerns the adjustment in predicted marginal value, not prediction error or retraining performance.

Preference reversal. For two feasible candidates A and B at the same allocation, suppose M-STC prefers A, with margin $\gamma = \Delta_A^{\text{M-STC}} - \Delta_B^{\text{M-STC}} > 0$. Then

$$\Delta_B^{\mathrm{CTS}} - \Delta_A^{\mathrm{CTS}} = \kappa_B - \kappa_A - \gamma.$$

CTS strictly reverses this preference if and only if $\kappa_B - \kappa_A > \gamma$; equality gives a tie.

This condition characterizes pairwise preference reversal and changes the next greedy selection when A and B are the only remaining feasible candidates.

Worked example. Consider the main-text example with abstract base metrics $(1, 1)$, utility $u[x, y] = xy$, and one selectable block of equal size per domain. Domains A, B, and C have predicted responses $(0.12, 0)$, $(0, 0.10)$, and $(0.40, 0)$, respectively, and the budget permits two blocks. Both rules first select C and exhaust its block, leaving the joint state $(1.40, 1)$ and candidates A and B. Their own-domain states remain $(1, 1)$.

For this utility,

$$\boldsymbol{H}_u = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Since the Hessian is constant, the adjustment reduces to $\kappa_i = \boldsymbol{c}_i^\top \boldsymbol{H}_u \boldsymbol{d}_i$. With $\boldsymbol{c}_A = \boldsymbol{c}_B = (0.40, 0)^\top$, this gives

$$\kappa_A = 0, \qquad \kappa_B = 0.40 \times 0.10 = 0.04.$$

The M-STC margin is $\gamma = 0.12 - 0.10 = 0.02$, so the adjustment difference satisfies the strict reversal condition. M-STC selects A, whereas CTS selects B, whose marginal value increases from 0.10 to 0.14. Evaluating the predicted joint metric states of the two selected allocations gives utility values of 1.52 for $\{C, A\}$ and 1.54 for $\{C, B\}$. Thus, CTS's state-dependent valuation selects the allocation with higher predicted joint-state utility among these two alternatives.

Connection to the driving surrogate. The driving utility in Appendix A.2 is a product of NC, DAC, DDC, and TLC, multiplied by a weighted sum of EP, TTC, LK, HC, and EC, with an overall scale factor. Its mixed curvature couples the multiplicative components with one another and with the weighted components. CTS incorporates these interactions by valuing candidate responses at the joint metric state. For example, with other components fixed and positive, higher predicted TLC increases the marginal utility of an EP improvement. When responses affect only the weighted components while the multiplicative components remain fixed, the utility is affine on the relevant states and both rules agree. Thus, CTS captures the state-dependent metric interactions encoded by the surrogate while preserving additive valuation where it applies.

Connection to the FMoW utility. For $u[\boldsymbol{m}] = \min_c m_c$, CTS values each candidate by its predicted improvement in the worst-region accuracy of the joint state. Other domains' responses can shift the limiting region, making different candidate improvements valuable. CTS reuses the fitted regional responses to reassess candidate value at the evolving predicted joint state, using the same minimum-over-regions objective as model evaluation. The exact finite-difference expression for κ_i applies directly, including where the minimizing coordinate changes. If one coordinate c attains the minimum at all four states $\boldsymbol{x}_i$, $\boldsymbol{x}_i + \boldsymbol{d}_i$, $\boldsymbol{x}_i + \boldsymbol{c}_i$, and $\boldsymbol{x}_i + \boldsymbol{c}_i + \boldsymbol{d}_i$, both marginal values equal the c-th component of $\boldsymbol{d}_i$ and $\kappa_i = 0$. Otherwise, the finite difference quantifies the adjustment, and the ranking condition determines when it changes candidate preferences. This gives CTS a direct mechanism for adapting allocation to the predicted regional bottleneck.

D Additional Experimental Results

D.1 Additional Retraining Results

Metric definitions are given in Section A.2.

Table 10: Driving metrics on navtest for the `navtrain` training setting. Entries show means with standard deviations as subscripts across three seeds; Base values are reported as means.

	Multiplicative components				Weighted components					Task score
Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	**EPDMS**
Base	95.30	95.94	99.09	99.60	88.09	94.55	94.49	98.25	82.39	83.97
$B = 100$										
Random	$95.43_{0.84}$	$96.41_{0.20}$	$98.98_{0.07}$	$99.54_{0.15}$	$88.68_{0.63}$	$94.69_{0.89}$	$94.82_{0.34}$	$98.27_{0.04}$	$82.81_{0.64}$	$84.66_{0.60}$
Uncertainty	$95.68_{0.33}$	$96.23_{0.38}$	$98.91_{0.12}$	$99.51_{0.06}$	$88.21_{0.22}$	$94.77_{0.36}$	$94.88_{0.13}$	$98.27_{0.04}$	$83.50_{0.32}$	$84.50_{0.48}$
CoreSet	$95.63_{0.41}$	$96.88_{0.33}$	$99.13_{0.09}$	$99.56_{0.03}$	$88.39_{0.65}$	$94.75_{0.51}$	$94.97_{0.34}$	$98.25_{0.03}$	$82.94_{0.20}$	$85.29_{0.47}$
Chameleon	$95.14_{0.20}$	$96.50_{0.19}$	$99.17_{0.02}$	$99.53_{0.02}$	$88.80_{0.18}$	$94.35_{0.11}$	$95.10_{0.12}$	$98.25_{0.04}$	$82.93_{0.75}$	$84.57_{0.18}$
RegMix	$95.70_{0.36}$	$96.55_{0.25}$	$99.08_{0.08}$	$99.56_{0.04}$	$88.52_{0.39}$	$94.91_{0.42}$	$95.01_{0.26}$	$98.28_{0.03}$	$83.18_{0.34}$	$84.91_{0.44}$
DML	$95.61_{0.42}$	$96.42_{0.31}$	$99.12_{0.07}$	$99.54_{0.05}$	$88.66_{0.44}$	$94.83_{0.37}$	$94.94_{0.29}$	$98.27_{0.03}$	$83.05_{0.39}$	$84.73_{0.51}$
MOSAIC	$96.75_{0.28}$	$97.06_{0.09}$	$99.03_{0.03}$	$99.60_{0.03}$	$87.74_{0.28}$	$96.09_{0.35}$	$94.92_{0.32}$	$98.27_{0.02}$	$82.80_{0.62}$	$86.29_{0.43}$
CTS	$96.96_{0.13}$	$97.08_{0.22}$	$99.34_{0.15}$	$99.63_{0.01}$	$88.35_{0.21}$	$96.21_{0.17}$	$95.18_{0.06}$	$98.30_{0.04}$	$84.12_{0.67}$	$\mathbf{87.27}_{0.09}$
$B = 200$										
Random	$95.90_{0.35}$	$96.58_{0.23}$	$99.11_{0.08}$	$99.65_{0.02}$	$88.75_{0.19}$	$95.08_{0.33}$	$95.14_{0.29}$	$98.27_{0.04}$	$83.14_{0.11}$	$85.45_{0.09}$
Uncertainty	$95.61_{0.66}$	$96.53_{0.38}$	$98.96_{0.18}$	$99.57_{0.09}$	$88.51_{0.11}$	$94.74_{0.59}$	$94.88_{0.28}$	$98.28_{0.03}$	$83.34_{0.34}$	$84.84_{0.54}$
CoreSet	$96.19_{0.49}$	$97.05_{0.11}$	$99.13_{0.06}$	$99.60_{0.04}$	$88.68_{0.13}$	$95.39_{0.44}$	$95.17_{0.11}$	$98.29_{0.01}$	$83.45_{0.55}$	$86.12_{0.31}$
Chameleon	$96.12_{0.52}$	$96.76_{0.34}$	$99.33_{0.18}$	$99.60_{0.10}$	$88.74_{0.54}$	$95.38_{0.71}$	$95.41_{0.19}$	$98.30_{0.02}$	$83.73_{0.13}$	$86.04_{0.30}$
RegMix	$96.18_{0.39}$	$96.94_{0.27}$	$99.20_{0.09}$	$99.61_{0.05}$	$88.61_{0.36}$	$95.42_{0.40}$	$95.35_{0.23}$	$98.29_{0.02}$	$83.62_{0.28}$	$85.96_{0.28}$
DML	$96.27_{0.35}$	$97.08_{0.29}$	$99.25_{0.07}$	$99.63_{0.04}$	$88.54_{0.41}$	$95.51_{0.35}$	$95.28_{0.25}$	$98.29_{0.02}$	$83.81_{0.32}$	$86.22_{0.33}$
MOSAIC	$96.83_{0.31}$	$97.51_{0.18}$	$99.24_{0.06}$	$99.61_{0.01}$	$88.20_{0.13}$	$96.16_{0.29}$	$95.36_{0.18}$	$98.26_{0.02}$	$82.63_{0.35}$	$87.04_{0.37}$
CTS	$97.13_{0.29}$	$97.64_{0.24}$	$99.32_{0.03}$	$99.69_{0.01}$	$88.42_{0.53}$	$96.53_{0.28}$	$95.24_{0.23}$	$98.30_{0.01}$	$84.24_{0.46}$	$\mathbf{88.06}_{0.14}$
$B = 400$										
Random	$96.71_{0.25}$	$96.91_{0.20}$	$99.18_{0.09}$	$99.71_{0.01}$	$88.75_{0.15}$	$96.02_{0.23}$	$95.76_{0.16}$	$98.30_{0.01}$	$82.96_{0.10}$	$86.69_{0.20}$
Uncertainty	$96.39_{0.60}$	$96.97_{0.38}$	$99.00_{0.08}$	$99.65_{0.01}$	$88.22_{0.42}$	$95.55_{0.66}$	$94.98_{0.22}$	$98.25_{0.02}$	$83.64_{0.16}$	$86.07_{0.75}$
CoreSet	$96.73_{0.24}$	$97.27_{0.17}$	$99.36_{0.02}$	$99.64_{0.02}$	$88.80_{0.11}$	$95.95_{0.26}$	$95.81_{0.23}$	$98.29_{0.03}$	$83.48_{0.46}$	$87.09_{0.29}$
Chameleon	$96.33_{0.36}$	$97.55_{0.20}$	$99.37_{0.07}$	$99.63_{0.03}$	$88.97_{0.42}$	$95.59_{0.38}$	$95.87_{0.20}$	$98.30_{0.01}$	$83.10_{0.35}$	$87.04_{0.60}$
RegMix	$96.82_{0.27}$	$97.38_{0.19}$	$99.38_{0.05}$	$99.67_{0.03}$	$88.88_{0.24}$	$96.14_{0.25}$	$95.88_{0.17}$	$98.30_{0.02}$	$83.67_{0.29}$	$87.31_{0.24}$
DML	$96.68_{0.30}$	$97.21_{0.23}$	$99.34_{0.06}$	$99.65_{0.04}$	$88.95_{0.29}$	$96.03_{0.28}$	$95.94_{0.19}$	$98.30_{0.02}$	$83.54_{0.31}$	$86.96_{0.27}$
MOSAIC	$97.75_{0.08}$	$97.79_{0.11}$	$99.42_{0.06}$	$99.72_{0.04}$	$87.62_{0.11}$	$97.17_{0.09}$	$95.54_{0.08}$	$98.24_{0.01}$	$82.81_{0.27}$	$88.21_{0.03}$
CTS	$97.85_{0.26}$	$97.76_{0.27}$	$99.41_{0.04}$	$99.74_{0.02}$	$88.85_{0.15}$	$96.99_{0.23}$	$96.13_{0.17}$	$98.32_{0.02}$	$85.65_{0.46}$	$\mathbf{89.15}_{0.15}$
$B = 800$										
Random	$96.94_{0.35}$	$97.15_{0.36}$	$99.35_{0.12}$	$99.69_{0.05}$	$89.16_{0.06}$	$96.22_{0.41}$	$96.28_{0.45}$	$98.29_{0.03}$	$83.63_{0.02}$	$87.41_{0.37}$
Uncertainty	$96.98_{0.40}$	$96.88_{0.16}$	$99.13_{0.11}$	$99.69_{0.07}$	$88.31_{0.45}$	$96.22_{0.32}$	$95.42_{0.15}$	$98.28_{0.03}$	$82.95_{0.31}$	$86.69_{0.34}$
CoreSet	$97.21_{0.12}$	$98.06_{0.23}$	$99.49_{0.06}$	$99.67_{0.05}$	$88.84_{0.08}$	$96.62_{0.13}$	$96.22_{0.19}$	$98.30_{0.03}$	$83.67_{0.27}$	$88.48_{0.12}$
Chameleon	$97.07_{0.17}$	$97.97_{0.18}$	$99.48_{0.08}$	$99.68_{0.03}$	$88.99_{0.50}$	$96.57_{0.19}$	$96.38_{0.18}$	$98.29_{0.03}$	$83.28_{0.22}$	$88.33_{0.23}$
RegMix	$97.28_{0.18}$	$98.01_{0.15}$	$99.49_{0.05}$	$99.69_{0.03}$	$88.91_{0.31}$	$96.69_{0.22}$	$96.39_{0.15}$	$98.30_{0.02}$	$83.74_{0.26}$	$88.27_{0.31}$
DML	$97.36_{0.16}$	$98.08_{0.17}$	$99.51_{0.04}$	$99.71_{0.03}$	$88.78_{0.35}$	$96.77_{0.20}$	$96.31_{0.16}$	$98.31_{0.02}$	$83.89_{0.29}$	$88.52_{0.25}$
MOSAIC	$97.65_{0.14}$	$98.33_{0.06}$	$99.54_{0.05}$	$99.73_{0.05}$	$88.68_{0.44}$	$97.03_{0.16}$	$96.19_{0.14}$	$98.26_{0.02}$	$82.93_{0.48}$	$89.10_{0.12}$
CTS	$97.88_{0.12}$	$98.83_{0.07}$	$99.61_{0.08}$	$99.73_{0.04}$	$88.98_{0.17}$	$97.37_{0.09}$	$96.49_{0.22}$	$98.29_{0.01}$	$83.59_{0.32}$	$\mathbf{89.83}_{0.18}$
$B = 1600$										
Random	$97.17_{0.07}$	$98.19_{0.43}$	$99.42_{0.05}$	$99.69_{0.02}$	$89.36_{0.12}$	$96.50_{0.14}$	$96.45_{0.25}$	$98.31_{0.03}$	$83.17_{0.76}$	$88.62_{0.22}$
Uncertainty	$96.92_{0.38}$	$97.66_{0.08}$	$99.22_{0.10}$	$99.77_{0.02}$	$89.02_{0.28}$	$96.24_{0.40}$	$96.10_{0.07}$	$98.30_{0.01}$	$82.92_{0.38}$	$87.75_{0.37}$
CoreSet	$97.50_{0.10}$	$98.31_{0.34}$	$99.59_{0.03}$	$99.72_{0.05}$	$89.27_{0.21}$	$96.86_{0.07}$	$96.75_{0.22}$	$98.30_{0.03}$	$83.88_{0.50}$	$89.30_{0.19}$
Chameleon	$97.43_{0.22}$	$98.46_{0.17}$	$99.60_{0.05}$	$99.75_{0.03}$	$89.60_{0.19}$	$96.83_{0.30}$	$96.89_{0.07}$	$98.30_{0.03}$	$83.87_{0.34}$	$89.50_{0.20}$
RegMix	$97.62_{0.16}$	$98.42_{0.19}$	$99.59_{0.04}$	$99.75_{0.02}$	$89.48_{0.24}$	$96.94_{0.25}$	$96.91_{0.09}$	$98.30_{0.02}$	$83.96_{0.30}$	$89.68_{0.21}$
DML	$97.54_{0.19}$	$98.36_{0.21}$	$99.61_{0.04}$	$99.76_{0.02}$	$89.56_{0.21}$	$96.88_{0.28}$	$96.96_{0.10}$	$98.30_{0.02}$	$83.85_{0.32}$	$89.43_{0.29}$
MOSAIC	$98.04_{0.24}$	$98.61_{0.32}$	$99.63_{0.06}$	$99.73_{0.02}$	$89.28_{0.19}$	$97.50_{0.32}$	$97.07_{0.06}$	$98.28_{0.04}$	$83.70_{0.41}$	$90.18_{0.25}$
CTS	$98.08_{0.26}$	$98.70_{0.28}$	$99.65_{0.03}$	$99.78_{0.01}$	$89.39_{0.21}$	$97.61_{0.39}$	$97.24_{0.08}$	$98.30_{0.04}$	$83.81_{0.29}$	$\mathbf{90.89}_{0.18}$

Component improvements on navtrain. CTS achieves the highest EPDMS at all five budgets, exceeding the strongest baseline by 0.71–1.02 points. Compared with MOSAIC, it improves NC, EP, and EC at every budget, and TTC and LK at four of the five budgets. At $B = 400$, the 0.94-point EPDMS gain accompanies improvements of 1.23, 0.59, and 2.84 percentage points in EP, LK, and EC, respectively. The component results therefore show a consistent overall advantage supported by improvements across collision avoidance, progress, lane keeping, and comfort.

Table 11: Driving metrics on navtest for the `OpenScene` training setting. Entries show means with standard deviations as subscripts across two or three seeds; Base values are reported as means.

	Multiplicative components				Weighted components					Task score
Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	**EPDMS**
Base	94.05	83.90	96.28	99.60	85.96	92.95	93.26	98.25	81.88	72.00
$B = 250$										
Random	$94.27_{0.60}$	$84.63_{1.46}$	$97.38_{0.23}$	$99.66_{0.04}$	$85.18_{1.02}$	$93.23_{0.64}$	$93.33_{0.56}$	$98.26_{0.01}$	$82.66_{0.76}$	$72.84_{1.14}$
Uncertainty	$93.97_{0.44}$	$82.49_{0.30}$	$96.78_{0.44}$	$99.66_{0.02}$	$85.18_{0.81}$	$92.98_{0.42}$	$93.18_{0.66}$	$98.23_{0.08}$	$82.15_{0.27}$	$70.78_{0.59}$
CoreSet	$95.11_{0.47}$	$87.66_{0.61}$	$98.38_{0.21}$	$99.67_{0.04}$	$86.09_{1.13}$	$94.08_{0.84}$	$94.47_{0.20}$	$98.31_{0.05}$	$83.38_{0.74}$	$76.26_{0.48}$
Chameleon	$94.02_{1.25}$	$84.30_{1.18}$	$97.48_{0.71}$	$99.58_{0.06}$	$87.48_{1.41}$	$92.69_{1.23}$	$93.43_{0.04}$	$98.26_{0.01}$	$83.15_{1.80}$	$72.97_{1.72}$
RegMix	$94.62_{0.71}$	$85.37_{0.92}$	$97.82_{0.24}$	$99.64_{0.04}$	$86.02_{0.91}$	$93.41_{0.67}$	$93.76_{0.41}$	$98.27_{0.02}$	$83.12_{0.58}$	$74.18_{1.36}$
DML	$94.38_{0.76}$	$84.91_{1.03}$	$97.66_{0.28}$	$99.62_{0.05}$	$86.35_{0.96}$	$93.18_{0.71}$	$93.91_{0.45}$	$98.26_{0.03}$	$82.94_{0.63}$	$73.62_{1.49}$
MOSAIC	$94.89_{0.74}$	$88.76_{1.17}$	$98.54_{0.43}$	$99.61_{0.04}$	$86.50_{1.03}$	$93.93_{0.88}$	$94.88_{0.14}$	$98.26_{0.03}$	$83.77_{0.67}$	$77.38_{1.58}$
CTS	$95.41_{0.93}$	$89.37_{0.87}$	$98.79_{0.45}$	$99.70_{0.06}$	$87.38_{1.13}$	$94.63_{1.07}$	$95.36_{0.15}$	$98.32_{0.05}$	$84.96_{0.39}$	$\mathbf{78.76}_{0.24}$
$B = 500$										
Random	$94.65_{0.21}$	$85.72_{0.88}$	$97.87_{0.44}$	$99.64_{0.06}$	$85.53_{0.22}$	$93.51_{0.32}$	$93.73_{0.24}$	$98.27_{0.05}$	$83.26_{0.14}$	$74.19_{1.05}$
Uncertainty	$93.32_{0.47}$	$82.26_{0.40}$	$96.09_{0.52}$	$99.60_{0.08}$	$84.51_{0.43}$	$92.23_{0.73}$	$92.38_{0.56}$	$98.30_{0.01}$	$82.85_{1.07}$	$69.77_{0.48}$
CoreSet	$95.56_{0.78}$	$88.96_{0.57}$	$98.95_{0.09}$	$99.71_{0.07}$	$86.21_{0.99}$	$94.69_{0.79}$	$95.14_{0.13}$	$98.31_{0.03}$	$84.24_{0.34}$	$78.12_{0.87}$
Chameleon	$95.00_{0.58}$	$87.11_{0.09}$	$98.16_{0.02}$	$99.67_{0.16}$	$86.67_{2.25}$	$94.22_{0.45}$	$94.20_{0.44}$	$98.30_{0.01}$	$83.69_{0.24}$	$75.98_{0.06}$
RegMix	$95.16_{0.52}$	$87.62_{0.54}$	$98.32_{0.18}$	$99.67_{0.05}$	$86.34_{0.61}$	$94.38_{0.46}$	$94.87_{0.31}$	$98.30_{0.02}$	$84.02_{0.37}$	$76.41_{0.74}$
DML	$95.29_{0.49}$	$87.39_{0.62}$	$98.41_{0.16}$	$99.66_{0.06}$	$86.21_{0.68}$	$94.51_{0.42}$	$94.72_{0.35}$	$98.29_{0.03}$	$83.91_{0.42}$	$76.16_{0.82}$
MOSAIC	$95.57_{1.05}$	$90.54_{0.45}$	$98.83_{0.29}$	$99.67_{0.09}$	$86.08_{1.79}$	$94.85_{1.23}$	$95.68_{0.27}$	$98.25_{0.04}$	$83.80_{0.16}$	$79.38_{1.05}$
CTS	$95.88_{0.47}$	$91.03_{0.37}$	$99.04_{0.15}$	$99.69_{0.06}$	$86.86_{1.08}$	$95.43_{0.59}$	$96.14_{0.34}$	$98.27_{0.04}$	$85.57_{0.25}$	$\mathbf{80.99}_{0.38}$
$B = 1000$										
Random	$95.21_{0.58}$	$87.15_{1.44}$	$98.26_{0.39}$	$99.72_{0.07}$	$85.56_{0.96}$	$94.35_{0.60}$	$94.50_{0.66}$	$98.31_{0.03}$	$82.50_{0.52}$	$75.84_{0.90}$
Uncertainty	$94.04_{0.70}$	$83.77_{0.02}$	$96.96_{0.08}$	$99.70_{0.08}$	$83.11_{0.66}$	$93.21_{1.00}$	$92.87_{0.14}$	$98.32_{0.02}$	$81.91_{0.75}$	$71.12_{0.38}$
CoreSet	$95.93_{0.24}$	$91.05_{0.26}$	$99.28_{0.11}$	$99.71_{0.04}$	$86.39_{0.48}$	$95.01_{0.21}$	$95.75_{0.08}$	$98.28_{0.03}$	$84.58_{0.42}$	$80.46_{0.02}$
Chameleon	$95.89_{0.19}$	$89.57_{0.68}$	$98.94_{0.16}$	$99.71_{0.07}$	$86.39_{0.51}$	$95.06_{0.26}$	$95.44_{0.27}$	$98.29_{0.01}$	$84.23_{0.74}$	$79.08_{0.74}$
RegMix	$95.78_{0.35}$	$90.48_{0.43}$	$99.17_{0.09}$	$99.70_{0.03}$	$86.51_{0.42}$	$94.93_{0.31}$	$95.68_{0.18}$	$98.29_{0.02}$	$84.41_{0.31}$	$78.86_{0.61}$
DML	$95.91_{0.32}$	$90.72_{0.39}$	$99.22_{0.08}$	$99.71_{0.03}$	$86.67_{0.39}$	$95.07_{0.28}$	$95.61_{0.20}$	$98.30_{0.02}$	$84.56_{0.28}$	$79.27_{0.55}$
MOSAIC	$96.00_{0.22}$	$92.20_{0.48}$	$99.33_{0.07}$	$99.67_{0.05}$	$86.63_{0.41}$	$95.24_{0.24}$	$96.17_{0.22}$	$98.28_{0.03}$	$84.33_{0.30}$	$81.68_{0.52}$
CTS	$96.85_{0.28}$	$92.27_{0.42}$	$99.49_{0.07}$	$99.71_{0.03}$	$86.74_{0.56}$	$96.03_{0.37}$	$96.24_{0.24}$	$98.27_{0.04}$	$85.69_{0.17}$	$\mathbf{82.78}_{0.36}$
$B = 2000$										
Random	$95.58_{0.54}$	$89.26_{0.64}$	$98.67_{0.18}$	$99.70_{0.12}$	$86.44_{0.42}$	$94.88_{0.61}$	$95.26_{0.18}$	$98.30_{0.00}$	$83.96_{0.96}$	$78.39_{0.12}$
Uncertainty	$93.14_{0.79}$	$82.66_{1.12}$	$96.64_{0.64}$	$99.53_{0.09}$	$84.52_{1.23}$	$92.19_{1.22}$	$93.22_{0.29}$	$98.28_{0.03}$	$80.98_{1.30}$	$69.94_{1.40}$
CoreSet	$95.89_{0.22}$	$91.77_{0.14}$	$99.44_{0.06}$	$99.66_{0.04}$	$87.39_{0.05}$	$94.98_{0.18}$	$95.99_{0.47}$	$98.29_{0.00}$	$85.55_{0.19}$	$81.37_{0.13}$
Chameleon	$96.38_{0.25}$	$91.31_{0.26}$	$99.15_{0.03}$	$99.71_{0.05}$	$86.55_{0.40}$	$95.60_{0.29}$	$95.99_{0.15}$	$98.34_{0.01}$	$85.04_{0.15}$	$81.35_{0.39}$
RegMix	$96.21_{0.29}$	$92.18_{0.31}$	$99.31_{0.06}$	$99.72_{0.02}$	$86.83_{0.34}$	$95.72_{0.27}$	$96.08_{0.14}$	$98.31_{0.02}$	$85.31_{0.22}$	$81.62_{0.35}$
DML	$96.08_{0.33}$	$91.96_{0.35}$	$99.27_{0.07}$	$99.70_{0.03}$	$86.94_{0.37}$	$95.58_{0.30}$	$96.17_{0.16}$	$98.30_{0.02}$	$85.14_{0.26}$	$81.24_{0.42}$
MOSAIC	$96.90_{0.38}$	$92.29_{0.36}$	$99.48_{0.05}$	$99.73_{0.01}$	$86.61_{0.73}$	$96.16_{0.26}$	$96.34_{0.06}$	$98.28_{0.05}$	$84.69_{0.01}$	$82.78_{0.41}$
CTS	$96.96_{0.26}$	$93.04_{0.28}$	$99.52_{0.06}$	$99.78_{0.02}$	$87.03_{0.36}$	$96.25_{0.30}$	$96.53_{0.14}$	$98.30_{0.02}$	$86.21_{0.19}$	$\mathbf{84.04}_{0.35}$
$B = 4000$										
Random	$96.32_{0.59}$	$90.53_{0.06}$	$99.06_{0.07}$	$99.79_{0.05}$	$86.36_{0.48}$	$95.66_{0.52}$	$95.68_{0.09}$	$98.30_{0.01}$	$84.46_{0.14}$	$80.38_{0.55}$
Uncertainty	$94.67_{0.28}$	$85.11_{0.51}$	$97.15_{0.54}$	$99.71_{0.04}$	$84.26_{0.69}$	$93.72_{0.40}$	$93.26_{0.09}$	$98.28_{0.02}$	$81.34_{1.06}$	$73.46_{0.19}$
CoreSet	$97.11_{0.18}$	$92.93_{0.60}$	$99.44_{0.06}$	$99.82_{0.02}$	$86.65_{0.55}$	$96.42_{0.19}$	$96.66_{0.30}$	$98.16_{0.12}$	$85.10_{0.06}$	$83.63_{0.36}$
Chameleon	$96.76_{0.24}$	$92.32_{0.02}$	$99.51_{0.01}$	$99.77_{0.01}$	$86.98_{0.17}$	$95.91_{0.31}$	$96.49_{0.12}$	$98.32_{0.01}$	$85.51_{0.11}$	$82.92_{0.13}$
RegMix	$96.82_{0.27}$	$92.91_{0.19}$	$99.55_{0.04}$	$99.79_{0.02}$	$86.87_{0.29}$	$96.04_{0.22}$	$96.54_{0.11}$	$98.31_{0.02}$	$85.68_{0.18}$	$83.46_{0.31}$
DML	$96.71_{0.30}$	$92.74_{0.23}$	$99.48_{0.05}$	$99.78_{0.02}$	$87.02_{0.26}$	$95.97_{0.25}$	$96.61_{0.12}$	$98.32_{0.02}$	$85.53_{0.21}$	$83.18_{0.37}$
MOSAIC	$96.97_{0.32}$	$93.59_{0.11}$	$99.59_{0.04}$	$99.80_{0.01}$	$87.14_{0.98}$	$96.18_{0.45}$	$96.62_{0.08}$	$98.28_{0.01}$	$85.06_{0.34}$	$84.25_{0.14}$
CTS	$97.32_{0.25}$	$93.71_{0.24}$	$99.59_{0.07}$	$99.82_{0.02}$	$87.16_{0.67}$	$96.36_{0.38}$	$96.98_{0.14}$	$98.32_{0.02}$	$86.69_{0.67}$	$\mathbf{85.27}_{0.17}$

Component improvements on OpenScene. CTS leads in EPDMS at every budget by 1.02–1.61 points over the strongest baseline. Relative to MOSAIC, seven of the nine components improve at all five budgets: NC, DAC, TLC, EP, TTC, LK, and EC. The EC gains range from 1.19 to 1.77 percentage points, accompanied by consistent improvements in progress, safety, and compliance metrics. At $B = 500$, all nine components improve together, with a 1.61-point EPDMS gain. These results extend CTS's advantage to the broader driving logs and show that its higher composite scores accompany broad, persistent component improvements.

Table 12: Regional and worst-region OOD-test accuracy on FMoW-WILDS. Entries show means with standard deviations as subscripts across three seeds; B counts additional images; the shared base contains 7686 images.

B	Method	Africa	Americas	Asia	Europe	Oceania	**Worst region**
2000	Random	$22.69_{1.77}$	$39.01_{0.21}$	$38.07_{0.66}$	$36.90_{0.38}$	$42.14_{3.09}$	$22.69_{1.77}$
	Uncertainty	$22.37_{1.26}$	$39.17_{0.59}$	$38.78_{1.37}$	$36.95_{1.67}$	$42.54_{1.53}$	$22.37_{1.26}$
	CoreSet	$22.78_{0.51}$	$39.17_{0.15}$	$37.11_{0.51}$	$37.25_{1.73}$	$42.34_{1.05}$	$22.78_{0.51}$
	Chameleon	$22.97_{1.73}$	$39.87_{0.44}$	$38.83_{2.08}$	$36.87_{1.18}$	$45.85_{1.09}$	$22.97_{1.73}$
	RegMix	$22.84_{1.46}$	$39.42_{0.37}$	$38.16_{0.92}$	$37.08_{1.24}$	$43.57_{1.88}$	$22.84_{1.46}$
	DML	$23.12_{1.62}$	$39.18_{0.42}$	$38.47_{1.06}$	$36.71_{1.35}$	$44.03_{2.11}$	$23.12_{1.62}$
	MOSAIC	$21.67_{1.01}$	$36.53_{1.18}$	$35.20_{1.21}$	$35.03_{0.97}$	$40.49_{2.31}$	$21.67_{1.01}$
	CTS	$24.05_{1.05}$	$37.23_{1.20}$	$35.30_{0.94}$	$35.60_{1.19}$	$45.15_{2.38}$	$\mathbf{24.05}_{1.05}$
4000	Random	$23.71_{0.96}$	$40.87_{0.75}$	$38.81_{0.59}$	$38.24_{0.86}$	$44.34_{1.50}$	$23.71_{0.96}$
	Uncertainty	$24.71_{0.93}$	$40.45_{0.91}$	$39.70_{0.29}$	$39.54_{0.64}$	$44.90_{1.23}$	$24.71_{0.93}$
	CoreSet	$23.73_{0.57}$	$40.37_{0.65}$	$40.59_{0.50}$	$38.93_{0.96}$	$43.54_{4.21}$	$23.73_{0.57}$
	Chameleon	$23.50_{0.48}$	$42.18_{0.69}$	$39.73_{0.66}$	$37.35_{0.78}$	$46.95_{1.52}$	$23.50_{0.48}$
	RegMix	$23.91_{0.72}$	$41.36_{0.68}$	$39.28_{0.77}$	$38.76_{0.81}$	$45.16_{1.91}$	$23.91_{0.72}$
	DML	$23.86_{0.81}$	$41.71_{0.73}$	$39.61_{0.69}$	$38.49_{0.92}$	$44.82_{2.07}$	$23.86_{0.81}$
	MOSAIC	$24.10_{1.69}$	$35.86_{0.97}$	$35.55_{1.56}$	$34.91_{1.19}$	$39.24_{0.83}$	$24.10_{1.69}$
	CTS	$26.15_{1.30}$	$37.31_{2.03}$	$36.34_{1.37}$	$34.79_{1.37}$	$45.60_{4.85}$	$\mathbf{26.15}_{1.30}$
6000	Random	$23.11_{0.63}$	$42.42_{0.93}$	$41.88_{1.85}$	$40.62_{1.07}$	$48.20_{0.52}$	$23.11_{0.63}$
	Uncertainty	$23.86_{0.51}$	$42.58_{0.65}$	$41.95_{1.12}$	$40.35_{0.74}$	$47.30_{1.48}$	$23.86_{0.51}$
	CoreSet	$24.54_{1.71}$	$42.19_{0.39}$	$41.47_{0.86}$	$41.10_{0.58}$	$43.84_{2.27}$	$24.54_{1.71}$
	Chameleon	$23.78_{0.43}$	$44.30_{0.86}$	$42.47_{1.67}$	$38.57_{1.20}$	$48.15_{2.69}$	$23.78_{0.43}$
	RegMix	$24.28_{1.18}$	$43.18_{0.72}$	$41.67_{1.31}$	$40.04_{1.09}$	$47.26_{1.83}$	$24.28_{1.18}$
	DML	$24.67_{1.35}$	$42.86_{0.81}$	$41.93_{1.42}$	$39.72_{1.18}$	$47.81_{2.04}$	$24.67_{1.35}$
	MOSAIC	$23.41_{1.00}$	$37.25_{2.00}$	$36.56_{4.30}$	$35.80_{1.47}$	$41.89_{2.40}$	$23.41_{1.00}$
	CTS	$27.88_{0.89}$	$38.41_{0.47}$	$39.01_{1.28}$	$35.96_{1.53}$	$45.55_{3.19}$	$\mathbf{27.88}_{0.89}$
8000	Random	$24.99_{2.09}$	$43.10_{0.79}$	$43.28_{1.50}$	$41.70_{0.86}$	$45.60_{0.76}$	$24.99_{2.09}$
	Uncertainty	$27.32_{1.46}$	$43.60_{0.52}$	$43.10_{0.37}$	$42.44_{0.33}$	$47.65_{0.88}$	$27.32_{1.46}$
	CoreSet	$25.44_{1.43}$	$42.94_{0.28}$	$41.94_{0.43}$	$42.14_{0.74}$	$46.85_{0.91}$	$25.44_{1.43}$
	Chameleon	$26.11_{0.74}$	$45.64_{1.14}$	$42.96_{0.53}$	$40.25_{1.44}$	$51.20_{0.78}$	$26.11_{0.74}$
	RegMix	$25.73_{1.51}$	$44.07_{0.84}$	$42.62_{0.73}$	$41.46_{1.03}$	$48.34_{1.42}$	$25.73_{1.51}$
	DML	$25.39_{1.67}$	$44.46_{0.91}$	$42.31_{0.86}$	$41.83_{1.12}$	$48.92_{1.58}$	$25.39_{1.67}$
	MOSAIC	$25.06_{1.26}$	$38.34_{2.64}$	$38.25_{5.46}$	$37.24_{2.96}$	$41.84_{0.91}$	$25.06_{1.26}$
	CTS	$29.02_{1.06}$	$38.44_{1.12}$	$41.08_{2.47}$	$35.69_{0.80}$	$44.94_{5.77}$	$\mathbf{29.02}_{1.06}$
10000	Random	$26.98_{0.46}$	$46.02_{0.10}$	$43.95_{0.91}$	$44.40_{1.13}$	$47.50_{0.35}$	$26.98_{0.46}$
	Uncertainty	$26.87_{1.21}$	$45.37_{1.29}$	$44.89_{0.69}$	$44.51_{0.78}$	$47.60_{0.90}$	$26.87_{1.21}$
	CoreSet	$27.30_{1.99}$	$45.22_{0.94}$	$45.17_{1.15}$	$44.36_{1.03}$	$46.40_{1.88}$	$27.30_{1.99}$
	Chameleon	$26.24_{0.29}$	$46.65_{0.58}$	$44.48_{1.30}$	$42.64_{2.22}$	$54.60_{1.40}$	$26.24_{0.29}$
	RegMix	$27.41_{1.16}$	$45.72_{0.51}$	$44.36_{1.12}$	$43.28_{1.31}$	$49.16_{1.07}$	$27.41_{1.16}$
	DML	$27.16_{1.28}$	$46.08_{0.57}$	$44.71_{1.24}$	$42.91_{1.43}$	$49.63_{1.19}$	$27.16_{1.28}$
	MOSAIC	$24.95_{1.47}$	$40.02_{3.63}$	$39.63_{5.96}$	$38.30_{1.80}$	$45.95_{2.96}$	$24.95_{1.47}$
	CTS	$29.92_{1.43}$	$39.85_{1.35}$	$42.59_{4.00}$	$36.62_{0.65}$	$45.95_{4.63}$	$\mathbf{29.92}_{1.43}$

Regional performance on FMoW-WILDS. CTS achieves the highest worst-region accuracy at every budget, exceeding the strongest baseline by 0.93–3.21 percentage points. Its worst-region accuracy increases steadily from 24.05 to 29.92 as the budget grows. Relative to MOSAIC, CTS also improves Asia accuracy at every budget and Oceania accuracy at four budgets, matching it at the largest budget. At $B = 6000$, all five regional means improve, including gains of 4.47 points in Africa and 2.45 points in Asia. At $B = 10000$, CTS exceeds RegMix in worst-region accuracy (29.92 vs. 27.41), although RegMix performs better in the Americas (45.72 vs. 39.85). Together, the results demonstrate consistent gains in the worst-region objective alongside improvements across multiple regions, supporting CTS for allocation under a geographic robustness objective.

D.2 CTS Selection Order

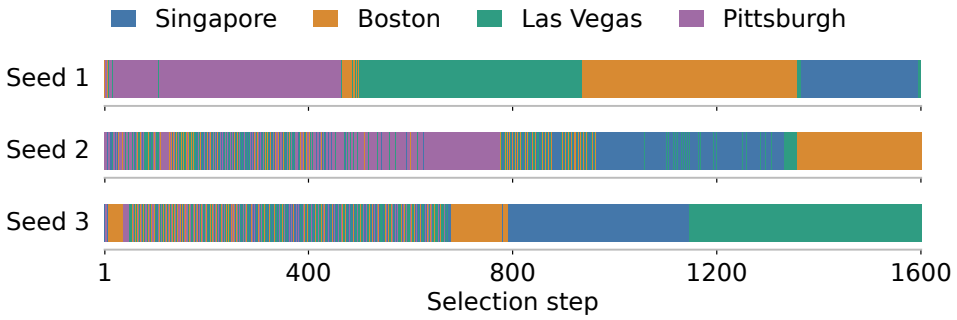


Figure 6: CTS selection order on navtrain at an additional budget of 1600 clips. Each colored segment identifies the source domain of the selected clips.

Figure 6 shows how CTS constructs the allocations in Figure 2(a), with sustained domain selection and frequent switching. The three seeds produce distinct trajectories under the same budget and allocation rule. Seed 1 concentrates early selections in Pittsburgh, followed by long runs from Las Vegas and Boston and a later Singapore block. Seeds 2 and 3 interleave domains more frequently early on, then develop longer runs: Seed 2 ends with 243 consecutive Boston clips, while Seed 3 selects 353 consecutive Singapore clips followed by 455 Las Vegas clips.

These patterns make the sequential nature of CTS allocation explicit. At each step, CTS compares candidate increments at the current joint metric state, subject to domain capacities, using the same fixed fitted response curves throughout selection. A domain can receive consecutive selections while it remains the highest-valued feasible candidate, and subsequent budget can move to another domain as relative marginal values or feasibility change. The resulting trajectories show how one allocation rule accommodates both sustained domain investment and fine-grained interleaving, producing distinct training subsets from the response estimates fitted independently for each seed.

D.3 Conditional Preference Prediction across Domain Pairs

Comparison setup. We extend the fixed-candidate comparison in Section 5.4 to all six pairs of the four city domains on `OpenScene`: Singapore, Boston, Las Vegas, and Pittsburgh. This analysis tests whether CTS's joint-state valuation improves conditional preference prediction across candidate pairs. For each seed, every domain supplies a fixed 400-clip candidate batch from its ranked pool. For a candidate pair (A, B), the shared 400-clip background is drawn from the other two domains (C, D), with fraction $q \in \mathcal{Q} = \{0, 0.25, 0.5, 0.75, 1\}$ from domain C and the remainder from domain D. The candidate batches remain fixed as q changes. For each condition, we retrain two models on the same 1000-clip base dataset and background, adding either the batch from A or the batch from B. Both models receive 800 additional clips and follow the 40-epoch OpenScene retraining protocol in Section A.3. The six pairs, five background fractions, and three seeds $\mathcal{S} = \{0, 2025, 424242\}$ yield 90 comparisons, including the 15 Singapore–Boston conditions reported in the main text.

Predicted and observed gaps. CTS and M-STC share the per-seed response curves fitted from the original OpenScene observations at counts $\{0, 200, 400\}$ following Section A.4. Both use the surrogate utility u_{AD} defined in Section A.2. For seed ℓ, the predicted background metric state is

$$\widehat{\boldsymbol{m}}^{(\ell)}_{CD}(q) = \boldsymbol{m}^{(\ell)}_0 + \widehat{\boldsymbol{r}}^{(\ell)}_C(400q) + \widehat{\boldsymbol{r}}^{(\ell)}_D(400(1-q)). \tag{21}$$

CTS predicts the relative value of the two candidate batches by composing each response with this background state:

$$\widehat{G}^{(\ell)}_{AB,\mathrm{CTS}}(q) = u_{\mathrm{AD}}\left[\widehat{\boldsymbol{m}}^{(\ell)}_{CD}(q) + \widehat{\boldsymbol{r}}^{(\ell)}_A(400)\right] - u_{\mathrm{AD}}\left[\widehat{\boldsymbol{m}}^{(\ell)}_{CD}(q) + \widehat{\boldsymbol{r}}^{(\ell)}_B(400)\right]. \tag{22}$$

For M-STC, the shared background contributions cancel when subtracting the two predicted allocation values:

$$\widehat{G}^{(\ell)}_{AB,\text{M-STC}}(q) = u_{\mathrm{AD}}\left[\boldsymbol{m}^{(\ell)}_0 + \widehat{\boldsymbol{r}}^{(\ell)}_A(400)\right] - u_{\mathrm{AD}}\left[\boldsymbol{m}^{(\ell)}_0 + \widehat{\boldsymbol{r}}^{(\ell)}_B(400)\right]. \tag{23}$$

Thus, CTS values the fixed candidate batches at the shared background state, while M-STC's prediction is independent of q for a fixed seed and candidate pair.

Let $f_A^{(\ell)}(q)$ and $f_B^{(\ell)}(q)$ denote the models retrained with the respective candidate batches under the same seed and background. The observed difference is

$$G_{AB}^{(\ell)}(q) = J\Big(f_A^{(\ell)}(q)\Big) - J\Big(f_B^{(\ell)}(q)\Big), \tag{24}$$

where J is the official EPDMS. Predicted gaps use the surrogate utility u_{AD}, and observed gaps use EPDMS after retraining. We use the same ordering of A and B in both differences: positive gaps favor A, and negative gaps favor B.

Evaluation. For method $\mathbb{M} \in \{\text{M-STC}, \text{CTS}\}$, preference accuracy for pair (A, B) is

$$\mathrm{Acc}_{AB}^{(\mathbb{M})} = \frac{1}{15} \sum_{\ell \in \mathcal{S}} \sum_{q \in \mathcal{Q}} \mathbb{I}\Big[\widehat{G}_{AB,\mathbb{M}}^{(\ell)}(q)\, G_{AB}^{(\ell)}(q) > 0\Big]. \tag{25}$$

The mean absolute error in predicting the observed EPDMS gap is

$$\mathrm{MAE}_{AB}^{(\mathbb{M})} = \frac{1}{15} \sum_{\ell \in \mathcal{S}} \sum_{q \in \mathcal{Q}} \Big|\widehat{G}_{AB,\mathbb{M}}^{(\ell)}(q) - G_{AB}^{(\ell)}(q)\Big|. \tag{26}$$

For each candidate pair, we compute preference accuracy and gap MAE over five background fractions and three seeds (15 conditions). We also pool all six pairs to obtain these metrics over 90 conditions. Figure 7 breaks down correct prediction counts by candidate pair and seed.

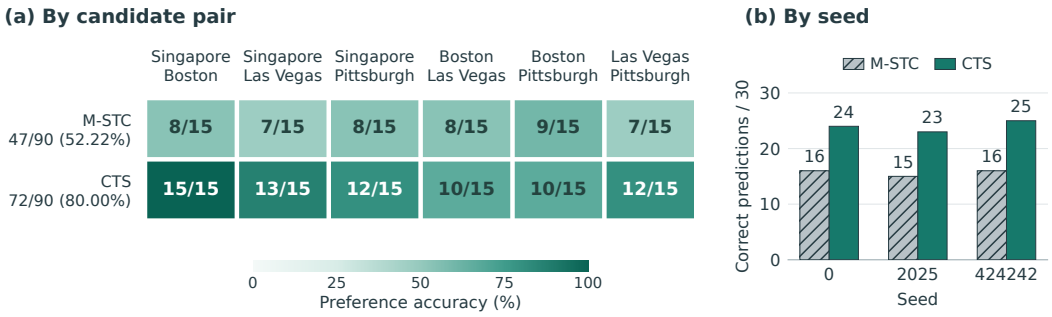


Figure 7: Conditional preference prediction on `OpenScene`. (a) Correct predictions per candidate pair out of 15 comparisons (five background fractions and three seeds), with a shared accuracy color scale and pooled accuracy in the row labels. (b) Correct predictions per seed out of 30 comparisons (six candidate pairs and five background fractions). Both panels summarize the same 90 conditions; the 400-clip background is drawn from the two domains outside each candidate pair.

Results. CTS achieves higher preference accuracy for every candidate pair and each of the three seeds (Figure 7). Its predictions also more closely match the observed EPDMS gaps: the MAE over all 90 conditions is 0.23 points, compared with 0.67 for M-STC. For the 15 Singapore–Boston conditions shown in the main text, the corresponding MAEs are 0.10 and 0.39 points, respectively. With response predictions and surrogate utility fixed, these improvements in both preference accuracy and gap prediction support joint-state valuation for predicting how candidate value changes with the shared background.

D.4 Robustness to Domain Partitioning

We directly reuse the six caption-based domains of Dimlioglu et al. (2026), replacing the four geographic domains with a partition based on driving and scene content. Their construction generates descriptions of 10-second clips using Qwen2.5-VL-32B-Instruct (Bai et al., 2025) and the prompt provided in MOSAIC's supplementary material (Dimlioglu et al., 2026). The descriptions are represented using TF-IDF features over the top 1024 unigrams and bigrams after removing common English stop words, and grouped into six semantic domains.

Table 13: Driving metrics on navtest for the navtrain training setting with six caption-based domains. Entries show means with standard deviations as subscripts across three seeds; B counts additional clips. Base values are reported as means.

	Multiplicative components				Weighted components					Task score
Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	**EPDMS**
Base	95.30	95.94	99.09	99.60	88.09	94.55	94.49	98.25	82.39	83.97
$B = 100$										
Chameleon	$95.43_{0.61}$	$96.14_{0.02}$	$98.94_{0.12}$	$99.56_{0.06}$	$88.45_{0.26}$	$94.52_{0.59}$	$94.82_{0.07}$	$98.28_{0.02}$	$83.27_{0.49}$	$84.35_{0.47}$
RegMix	$95.83_{0.25}$	$96.19_{0.20}$	$98.87_{0.07}$	$99.58_{0.04}$	$88.18_{0.36}$	$94.91_{0.40}$	$94.88_{0.24}$	$98.30_{0.04}$	$83.36_{0.47}$	$84.60_{0.31}$
DML	$95.80_{0.33}$	$96.09_{0.31}$	$98.95_{0.09}$	$99.56_{0.05}$	$88.45_{0.65}$	$94.98_{0.43}$	$94.71_{0.27}$	$98.27_{0.04}$	$83.22_{0.57}$	$84.38_{0.53}$
MOSAIC	$96.53_{0.31}$	$96.91_{0.30}$	$99.03_{0.12}$	$99.54_{0.06}$	$87.62_{0.21}$	$95.80_{0.32}$	$94.85_{0.34}$	$98.24_{0.02}$	$82.66_{0.66}$	$85.85_{0.41}$
CTS	$96.82_{0.24}$	$97.15_{0.18}$	$99.31_{0.07}$	$99.62_{0.03}$	$88.20_{0.31}$	$96.10_{0.27}$	$95.12_{0.19}$	$98.30_{0.02}$	$83.75_{0.41}$	$\mathbf{86.72}_{0.25}$
$B = 200$										
Chameleon	$96.20_{0.11}$	$96.58_{0.25}$	$98.97_{0.21}$	$99.63_{0.01}$	$88.02_{0.19}$	$95.42_{0.24}$	$94.88_{0.23}$	$98.30_{0.00}$	$82.88_{1.50}$	$85.39_{0.02}$
RegMix	$96.20_{0.48}$	$96.86_{0.37}$	$98.90_{0.13}$	$99.64_{0.04}$	$87.87_{0.30}$	$95.50_{0.39}$	$94.80_{0.20}$	$98.29_{0.03}$	$83.19_{0.24}$	$85.43_{0.35}$
DML	$96.50_{0.47}$	$96.92_{0.18}$	$98.99_{0.11}$	$99.65_{0.06}$	$87.94_{0.61}$	$95.57_{0.46}$	$94.91_{0.23}$	$98.29_{0.03}$	$83.33_{0.41}$	$85.64_{0.40}$
MOSAIC	$97.07_{0.30}$	$97.19_{0.25}$	$99.08_{0.06}$	$99.64_{0.03}$	$87.79_{0.46}$	$96.28_{0.36}$	$94.99_{0.29}$	$98.25_{0.01}$	$82.92_{0.55}$	$86.75_{0.17}$
CTS	$97.35_{0.21}$	$97.58_{0.26}$	$99.35_{0.05}$	$99.69_{0.02}$	$88.40_{0.37}$	$96.65_{0.31}$	$95.35_{0.24}$	$98.31_{0.01}$	$84.05_{0.36}$	$\mathbf{87.75}_{0.19}$
$B = 400$										
Chameleon	$95.61_{0.32}$	$96.50_{0.25}$	$99.04_{0.11}$	$99.59_{0.06}$	$88.64_{0.56}$	$94.84_{0.42}$	$94.90_{0.13}$	$98.29_{0.00}$	$82.73_{0.34}$	$84.95_{0.45}$
RegMix	$96.14_{0.23}$	$96.67_{0.18}$	$99.18_{0.09}$	$99.64_{0.05}$	$88.78_{0.30}$	$95.39_{0.35}$	$95.14_{0.24}$	$98.30_{0.03}$	$83.12_{0.25}$	$85.85_{0.29}$
DML	$96.08_{0.27}$	$96.51_{0.21}$	$99.14_{0.09}$	$99.59_{0.04}$	$88.72_{0.30}$	$95.41_{0.23}$	$95.27_{0.15}$	$98.31_{0.03}$	$83.06_{0.44}$	$85.55_{0.24}$
MOSAIC	$97.36_{0.10}$	$97.91_{0.05}$	$99.33_{0.10}$	$99.66_{0.02}$	$88.37_{0.41}$	$96.68_{0.11}$	$95.43_{0.34}$	$98.27_{0.03}$	$83.00_{1.38}$	$88.11_{0.05}$
CTS	$97.72_{0.18}$	$98.10_{0.22}$	$99.45_{0.06}$	$99.75_{0.03}$	$88.70_{0.28}$	$97.05_{0.24}$	$96.05_{0.17}$	$98.32_{0.02}$	$84.10_{0.33}$	$\mathbf{89.05}_{0.17}$
$B = 800$										
Chameleon	$96.26_{0.49}$	$96.83_{0.47}$	$99.10_{0.05}$	$99.71_{0.03}$	$88.89_{0.48}$	$95.53_{0.46}$	$95.64_{0.21}$	$98.28_{0.01}$	$82.97_{0.19}$	$86.10_{0.55}$
RegMix	$96.86_{0.24}$	$97.06_{0.13}$	$99.22_{0.07}$	$99.71_{0.04}$	$88.87_{0.33}$	$96.10_{0.24}$	$95.86_{0.18}$	$98.31_{0.03}$	$83.51_{0.31}$	$86.70_{0.53}$
DML	$96.90_{0.21}$	$97.19_{0.15}$	$99.25_{0.07}$	$99.73_{0.04}$	$88.72_{0.31}$	$96.18_{0.46}$	$95.67_{0.24}$	$98.31_{0.02}$	$83.82_{0.27}$	$87.07_{0.34}$
MOSAIC	$97.92_{0.09}$	$98.08_{0.17}$	$99.50_{0.05}$	$99.73_{0.01}$	$88.20_{0.25}$	$97.35_{0.14}$	$96.12_{0.22}$	$98.25_{0.04}$	$83.00_{0.50}$	$88.99_{0.09}$
CTS	$98.00_{0.14}$	$98.55_{0.17}$	$99.62_{0.04}$	$99.79_{0.02}$	$88.75_{0.24}$	$97.55_{0.20}$	$96.65_{0.15}$	$98.33_{0.01}$	$84.25_{0.29}$	$\mathbf{90.05}_{0.15}$
$B = 1600$										
Chameleon	$96.61_{0.30}$	$97.22_{0.26}$	$99.25_{0.11}$	$99.72_{0.06}$	$89.05_{0.24}$	$96.02_{0.39}$	$95.90_{0.18}$	$98.32_{0.01}$	$82.35_{0.27}$	$86.99_{0.57}$
RegMix	$97.04_{0.17}$	$97.53_{0.29}$	$99.36_{0.06}$	$99.75_{0.03}$	$89.16_{0.33}$	$96.36_{0.25}$	$96.24_{0.08}$	$98.31_{0.03}$	$83.09_{0.43}$	$87.83_{0.36}$
DML	$97.01_{0.14}$	$97.52_{0.34}$	$99.33_{0.05}$	$99.76_{0.03}$	$89.28_{0.26}$	$96.19_{0.43}$	$96.15_{0.12}$	$98.32_{0.03}$	$82.69_{0.26}$	$87.62_{0.36}$
MOSAIC	$97.98_{0.05}$	$98.59_{0.12}$	$99.60_{0.03}$	$99.76_{0.01}$	$89.03_{0.24}$	$97.49_{0.11}$	$97.02_{0.25}$	$98.27_{0.03}$	$83.61_{0.31}$	$89.98_{0.13}$
CTS	$98.20_{0.12}$	$98.90_{0.15}$	$99.72_{0.03}$	$99.83_{0.01}$	$89.45_{0.20}$	$97.88_{0.18}$	$97.35_{0.11}$	$98.34_{0.01}$	$84.45_{0.25}$	$\mathbf{91.05}_{0.13}$

We evaluate this partition on navtrain at additional budgets $B \in \{100, 200, 400, 800, 1600\}$ across three random seeds. The base dataset, candidate pool, evaluation split, and additional budgets remain unchanged from the geographic-partition setting. Within-domain rankings are recomputed using the same scoring rule, and MOSAIC and CTS share the single-domain pilots constructed for the six-domain partition. Chameleon computes sampling weights for the six domains. RegMix and DML follow the mixture-training, fitting, and search protocol in Section A.5, applied to the caption-based domains. Table 13 reports the caption-partition results.

CTS achieves the highest mean EPDMS among the five compared methods at all five budgets, exceeding MOSAIC by 0.87–1.07 points. It also achieves higher mean values than MOSAIC on all nine component metrics at every budget, with improvements spanning both the multiplicative and weighted components. Together with the geographic-partition results, these findings show that CTS remains effective when both domain semantics and domain count change, supporting its robustness across geographic and caption-based partitions.

D.5 Sensitivity to Pilot Evidence

We test whether CTS's advantage over M-STC in final retraining performance persists when the number and range of single-domain pilot observations change. On navtrain, we use Hydra-MDP with a 460-clip base dataset, four geographic domains, additional budgets $B \in \{100, 400\}$, and seeds $\{0, 2025, 424242\}$. The default nonzero pilot counts $\{100, 200\}$ follow MOSAIC (Dimlioglu et al., 2026). We compare this configuration with three alternatives: $\{50, 100, 200\}$ adds a smaller-

count observation without changing the maximum count; $\{200, 400\}$ retains two observations but shifts them to larger counts; and $\{50, 100, 200, 400\}$ increases both the number of observations and the count range. All fits also include the shared base observation $(0, 0)$.

For each seed and domain, pilot datasets are nested prefixes of the same fixed within-domain ranking. Each pilot starts independently from the same-seed base-model checkpoint and trains for 10 epochs on the base dataset augmented with the selected single-domain clips. Response curves are fitted separately for each pilot configuration and random seed, using the exponential family and fitting procedure in Section A.4. Within each configuration and seed, CTS and M-STC share the fitted curves, utility function, and greedy allocation framework, differing only in their valuation rules. Each method selects its own subset at each budget, followed by final retraining for 60 epochs and evaluation using the official EPDMS J.

Table 14 reports final retraining EPDMS as mean and standard deviation across seeds. Pilot cost is the total training GPU-hours for all nonzero-count pilots across the four domains, averaged over seeds and excluding base training and final retraining. The four configurations require 8, 12, 8, and 16 pilot runs per seed, respectively. Within each configuration, the pilot evidence is reused by both methods and across both allocation budgets.

Table 14: Sensitivity to pilot evidence on `navtrain`. Final EPDMS means are shown with standard deviations across three seeds as subscripts; higher is better. Pilot cost is mean training GPU-hours per seed, shared by both methods and across budgets; lower is better.

Pilot counts	Pilot cost	$B = 100$		$B = 400$	
	(GPU-hours)	M-STC	**CTS**	M-STC	**CTS**
$\{100, 200\}$	2.31	$86.81_{0.34}$	$\mathbf{87.27}_{0.09}$	$88.26_{0.17}$	$\mathbf{89.15}_{0.15}$
$\{50, 100, 200\}$	3.26	$86.82_{0.24}$	$\mathbf{87.29}_{0.21}$	$88.17_{0.13}$	$\mathbf{89.13}_{0.18}$
$\{200, 400\}$	2.87	$86.84_{0.38}$	$\mathbf{87.32}_{0.32}$	$88.30_{0.25}$	$\mathbf{89.18}_{0.16}$
$\{50, 100, 200, 400\}$	4.88	$86.86_{0.41}$	$\mathbf{87.35}_{0.32}$	$88.34_{0.26}$	$\mathbf{89.21}_{0.21}$

Results. CTS achieves higher mean final EPDMS than M-STC under all four pilot configurations at both budgets, with gains of 0.46–0.49 points at $B = 100$ and 0.87–0.96 points at $B = 400$. Its advantage therefore persists when observations are added or shifted to larger pilot counts. At either budget, CTS's mean score spans only 0.08 points across the four configurations. The richest configuration costs 2.11 times as much as the default (4.88 versus 2.31 GPU-hours per seed), while increasing CTS's mean EPDMS by only 0.08 and 0.06 points at $B = 100$ and $B = 400$, respectively. These results show that CTS maintains its advantage across the tested pilot configurations and support retaining the default MOSAIC setting, which achieves nearly the same final performance at the lowest measured pilot-training cost.

D.6 Sensitivity to the Response-Curve Family

We examine whether CTS's advantage over M-STC persists across response-curve families and how the choice of family affects response prediction and final retraining performance. We retain the `navtrain` setup from Section D.5, including budgets $\{100, 400\}$ and seeds $\{0, 2025, 424242\}$, and fix the nonzero fitting counts to $\{100, 200\}$. For each domain–metric pair, we compare the original exponential response with an anchored power-law response and a logarithmic response:

$$g_{\exp}(n) = a(1 - e^{-n/\tau}), \tag{27}$$

$$g_{\text{power}}(n) = a\big(1 - (1 + n/N_0)^{-\alpha}\big), \tag{28}$$

$$g_{\log}(n) = a \log(1 + n/\tau). \tag{29}$$

Here, $N_0 = 460$ is fixed to the base dataset size. Each family has two fitted parameters and satisfies $g(0) = 0$. All curves are fitted by unweighted least squares on signed responses, including the base observation $(0, 0)$, without log-transforming the targets.

Response curves are fitted separately for each family and random seed. Within each family and seed, CTS and M-STC share the fitted curves, utility function, greedy allocation framework, and

retraining protocol, differing only in their valuation rules. Each method selects its own subset at each budget, followed by final retraining for 60 epochs and evaluation using official EPDMS.

We evaluate held-out extrapolation error at $n = 400$, beyond the largest fitting count of 200. The 400-example pilot observations are excluded from fitting and tuning in this experiment. For seed ℓ, the response mean absolute error (MAE) in percentage points is

$$E_\ell = \frac{100}{Kd} \sum_{i=1}^{K} \sum_{c=1}^{d} \left| \widehat{r}_{ic}^{(\ell)}(400) - r_{ic}^{(\ell)}(400) \right|, \qquad K = 4, \quad d = 9, \tag{30}$$

where responses are changes computed from metrics on the $[0, 1]$ scale. We report the mean of E_ℓ across seeds. Table 15 summarizes this extrapolation error and final retraining performance.

Table 15: Sensitivity to the response-curve family on `navtrain`. All curves use fitting counts $\{0, 100, 200\}$; $n = 400$ is held out from fitting and tuning. Response MAE measures extrapolation error at $n = 400$ in percentage points (lower is better); final EPDMS uses the 0–100 scale (higher is better). MAE is averaged across seeds; score subscripts give standard deviations across three seeds.

Response family	Response MAE	$B = 100$		$B = 400$	
	(pp)	M-STC	**CTS**	M-STC	**CTS**
Exponential	0.12	$86.81_{0.34}$	$\mathbf{87.27}_{0.09}$	$88.26_{0.17}$	$\mathbf{89.15}_{0.15}$
Anchored power law	0.15	$85.47_{0.42}$	$\mathbf{86.34}_{0.31}$	$86.55_{0.23}$	$\mathbf{88.20}_{0.24}$
Logarithmic	0.21	$83.56_{0.53}$	$\mathbf{85.12}_{0.55}$	$84.38_{0.36}$	$\mathbf{87.39}_{0.35}$

Results. CTS achieves higher mean final EPDMS than M-STC with all three response families, with gains of 0.46–1.56 points at $B = 100$ and 0.89–3.01 points at $B = 400$. Its advantage therefore extends beyond the original exponential parameterization. Among the tested families, the exponential family achieves the lowest held-out response MAE at $n = 400$ (0.12 percentage points, compared with 0.15 for the power law and 0.21 for the logarithmic family) and the highest mean final EPDMS for both allocation rules at both budgets. Together, these results support retaining the exponential family used by MOSAIC (Dimlioglu et al., 2026) as the default, while demonstrating that CTS's advantage persists across the tested response families.

D.7 Computational Cost and Measurement Protocol

CTS shares STC's single-domain pilots and final retraining protocol. Its computational overhead comes from fitting component responses and valuing candidate increments at the predicted joint state; allocation requires no additional model training.

Fitting, allocation, and storage. For K domains and d metrics, original STC (MOSAIC) fits K scalar-gain curves, whereas CTS and M-STC share Kd component-response curves. Each fit uses the function family and fitting procedure in Section A.4, with cost depending on solver convergence. The fitted curves are reused across allocation budgets. With constant-time scalar-curve evaluation and $O(d)$ utility computation, the full-scan procedure in Algorithm 1 costs $O(Kd)$ per step for CTS and $O(BKd)$ for budget B. Under the same full-scan procedure, M-STC also takes $O(BKd)$, while original STC takes $O(BK)$. Curve parameters and allocation state require $O(Kd)$ storage for CTS and M-STC, versus $O(K)$ for original STC; retaining the selected-example sequence adds $O(B)$.

Measurement protocol. We independently measure CTS on `navtrain` for $B \in \{100, 200, 400\}$ using a workstation with one RTX 4090 GPU and an Intel Core i9-14900KF CPU. Fitting and allocation run on the CPU with numerical-library thread counts set to one. For each seed, we perform three warm-up calls and 20 timed calls using `time.perf_counter`, then report the mean of the per-seed medians. We measure allocator memory using `tracemalloc`. Tracing starts after inputs and fitted parameters are prepared and includes the selected-example sequence and per-step allocation trace. We report the maximum traced peak across seeds, which measures allocator allocations.

For comparison with the training workload, we report seed-0 wall-clock durations of the complete workflow on the same workstation: base and pilot training, fitting and allocation, final retraining,

and evaluation. Table 16 reports CTS's own costs; its final row divides the reported combined fitting and allocation time by the corresponding workflow duration.

Table 16: Computational cost of CTS on `navtrain`, measured on a single-RTX-4090 workstation. Fitting and allocation run on the CPU; workflow times are elapsed wall-clock durations.

Metric	$B = 100$	$B = 200$	$B = 400$
Response-curve fitting (ms)	9.69	9.69	9.69
Greedy allocation (ms)	16.49	32.77	65.44
Fitting + allocation (ms)	26.18	42.45	75.13
Allocator peak (KiB)	115.24	214.28	418.59
Full workflow wall time (h)	126.44	137.08	147.33
(Fitting + allocation) / workflow (%)	5.75×10^{-6}	8.60×10^{-6}	14.16×10^{-6}

Results. Response fitting takes 9.69 ms and is performed once, with fitted curves reused across budgets. Increasing the budget fourfold from 100 to 400 clips raises allocation time from 16.49 to 65.44 ms, consistent with the linear dependence on B. Thus, updating the joint metric state and reassessing candidate values remains inexpensive across the tested budgets. Together, fitting and allocation take less than 76 ms, compared with complete-workflow wall times of 126.44–147.33 h. The measured allocator peak stays below 419 KiB, including the selected-example sequence and per-step allocation trace. These measurements show that CTS supports state-dependent allocation with negligible computation relative to the measured workflow and a small allocator memory footprint.