SOFT STRATEGY SELECTION FOR BATCH-MODE ACTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world deployment of active learning (AL) typically forces practitioners to choose an acquisition strategy before any data is labeled. This is a daunting task: strategy performance varies widely across settings (e.g., datasets and surrogate model architectures) and cannot be assessed without deployment. Existing strategy selection methods explore one strategy from a portfolio at each round and identify the optimal one using bandit feedback or model retraining. Many acquisition rounds are therefore spent exploring strategies rather than collecting the most informative data. Such overhead is a major barrier to AL-driven design of high-throughput experiments, such as genetic perturbation screens and directed evolution, where AL runs consist of only a few rounds with large batch sizes. This regime permits a natural alternative: acquiring data using multiple AL strategies within a single batch. We refer to this as *soft strategy selection* and introduce FractAL, a method specifically designed for this task. FractAL infers a per-strategy reward using influence-function-based data attribution, which requires no additional retraining, and then computes budget shares for each strategy in the portfolio using online mirror descent. We benchmark FractAL across seven active learning setups spanning classification, regression, and genetic perturbation effect prediction. The results highlight that strategy selection is a hard problem: every existing method performs worse than random sampling on at least one setup. FractAL, however, matches or outperforms every baseline, including random sampling, on all seven setups. Its allocations concentrate budget on the strongest strategies in the portfolio while pruning the weakest. FractAL is therefore a reliable choice for real-world deployments, where the optimal strategy is unknown in advance, an important step towards making active learning practical for high-throughput experiments and modern scientific discovery.

1 INTRODUCTION

Designing informative scientific experiments is a central challenge across the life and chemical sciences, arising in directed evolution of proteins (Tran et al., 2026), antibody engineering (Frey et al., 2025), perturbation biology (Rood et al., 2024), and molecular design (Smith et al., 2018). In these fields, researchers decide which genes, proteins, antibodies, or compounds to include in each experiment. Active Learning (AL) automates this process. At each experimental round $t \in [T]$, an acquisition strategy $\mathcal{A}$ selects a batch $\mathcal{B}^t$ of size B from the input pool $\mathcal{D}_{\text{pool}}$. Then, outcomes are measured at selected inputs, and the surrogate model f_θ is retrained on the augmented training set $\mathcal{D}_{\text{train}}$ (Settles, 2009; Ren et al., 2021). The goal of a strategy is to maximize model performance under experimental budget constraints.

Many AL strategies have been proposed, targeting model uncertainty (Houlsby et al., 2011), batch diversity (Sener & Savarese, 2018), pool representativeness (Yehuda et al., 2022; Hacohen et al., 2022), or combinations of these. However, the number of strategies keeps growing, and no single one performs well across domains, datasets, and model architectures (Munjal et al., 2022; Hacohen et al., 2022; Lüth et al., 2023; Werner et al., 2024).

Adaptive strategy selection methods address this problem by choosing a single strategy $\mathcal{A}^t$ at each round t out of a portfolio $\mathcal{P} = \{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$. This may be naturally framed as a multi-armed bandit (MAB) problem (Baram et al., 2004; Hsu & Lin, 2015; Pang et al., 2018) in which each strategy

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.

Table 1: Comparison with related work on three axes: support for batch sampling, fractional budget allocation across the strategy portfolio, and the ability to operate without additional model training.

Related work	Batch Sampling	Fractional Budget Allocation	No Retraining / Aux. Models
ALBL (Hsu & Lin, 2015)	✗	✗	✓
AutoAL (Wang et al., 2025)	✓	✗	✗
SelectAL (Hacohen & Weinshall, 2023)	✓	✗	✗
FractAL *(this work)*	✓	✓	✓

is one arm of the MAB, and at each round the learner observes a reward from the previous round (e.g., accuracy on a validation dataset) and selects the next arm using a bandit solver. Strategy-wise performance may also be estimated by retraining the model after removing a subset of training points selected according to that strategy's own criterion (Hacohen & Weinshall, 2023).

Both approaches are, however, poorly suited to the design of high-throughput biological experiments (Rood et al., 2024; Huang et al., 2024; Yang et al., 2025; DeMeo et al., 2025), where assays efficiently provide results for thousands of inputs at once, but experimental time remains dominated by biological timescales. In these settings, with few rounds (small T) and large batch sizes (large B), relying on a single strategy at each round is unnecessarily risky.

We instead introduce the problem of *soft strategy selection*. In this setting, we allocate fractions p_k^t of each round's budget B to all strategies in the portfolio $\mathcal{P}$, where the vector $\mathbf{p}^t \in \Delta^{K-1}$ lies in the $(K-1)$-dimensional probability simplex. This results in a batch $\mathcal{B}^t = \bigcup_{k=1}^{K} \mathcal{B}_k^t$, composed of strategy-specific sub-batches. We propose FractAL, the first method designed to address this problem. FractAL estimates a strategy-wise reward vector $\mathbf{r}^t$ by attributing changes in validation risk to individual strategies via influence functions (Koh & Liang, 2017), and then updates the allocations using online mirror descent on the probability simplex.

After introducing the necessary background (Section 2), we present FractAL in detail (Section 3). We demonstrate that FractAL surpasses bandit- and retraining-based baselines on multiple classification and regression problems, including recent high-throughput genetic perturbation screening datasets (Figure 1, Section 4). These experiments also highlight that FractAL concentrates budget on the strongest strategy in each portfolio while pruning the weakest. Moreover, it is the only strategy selection method that outperforms random selection across all datasets (Figure 1), suggesting that fractional budget allocation is key to improving AL performance.

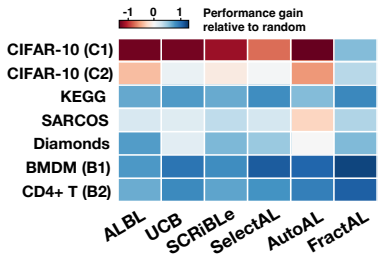


Figure 1: Performance gain over random sampling for the six adaptive strategy selection methods, across the seven setups of Section 4 (Appendix G).

2 Background

FractAL builds on two distinct lines of work: adaptive strategy selection for active learning and influence functions for data attribution. A comparison with existing adaptive strategy selection methods appears in Table 1.

2.1 Adaptive Strategy Selection for Active Learning

Adaptive strategy selection has been well studied, and existing work falls broadly into three paradigms: bandit-based, retraining-based, and differentiable search methods.

Bandit-Based Strategy Selection. Early approaches formulate strategy selection as a multi-armed bandit (MAB) problem where AL strategies act as experts (Hsu & Lin, 2015; Baram et al., 2004; Pang et al., 2018). For example, ALBL (Hsu & Lin, 2015) relies on the Exp4.P solver (Beygelzimer et al., 2011) with importance-weighted accuracy rewards. All these methods select a single point per iteration; in the batch setting, this amounts to committing each round to a single strategy, preventing strategy-level reward attribution within a batch.

Retraining-Based Strategy Selection. SelectAL (Hacohen & Weinshall, 2023) uses a derivative-based surrogate test to switch between predefined *low-budget* and *high-budget* strategy families. At each round, each strategy selects points to remove from the training set, and the strategy yielding the largest risk increase after retraining is selected for the next round. The original specification of SelectAL is limited to $K = 2$ strategies, but its main selection principle generalizes to any portfolio size, so we adopt it as a baseline. Evaluating K strategies over T AL rounds requires $O(KT)$ model retrainings, which becomes expensive when the surrogate model f_θ is costly to train.

Differentiable Strategy Search. AutoAL (Wang et al., 2025) makes strategy selection differentiable via bi-level optimization with two co-trained auxiliary networks (SearchNet and FitNet). The bi-level optimization, however, introduces a substantial number of hyperparameters and adds computational overhead per iteration.

2.2 Influence Functions

Influence functions (Hampel, 1974) quantify the impact of individual datapoints on a model's performance. A datapoint's influence estimates how the validation loss would change if that point were removed from the training set. This is done by first approximating the resulting perturbation to the model parameters, and then propagating it to the validation loss via a first-order Taylor expansion around the converged parameters (Koh & Liang, 2017).

Influence functions have previously been used as forward-looking AL acquisition heuristics, by estimating the influence at unmeasured inputs $x \in \mathcal{D}_{\text{pool}}$ and selecting the highest-ranked candidates (Liu et al., 2021; Xia & Henao, 2023). This approach, however, requires the construction of a pseudo-label for each unmeasured input (e.g., enumerating all possible labels, or training a different surrogate model) (Xu & Kazantsev, 2019), which makes it impractical for single- and multi-task regression settings. Here, we instead estimate per-strategy rewards using *retrospective* influence of training points $x \in \mathcal{D}_{\text{train}}$ whose outputs have already been measured.

3 FractAL: Fractional Active Learning

FractAL has two main components. We first define a strategy-wise reward signal, computed via influence-based attribution instead of retraining (Section 3.1). Based on this signal, we then update the allocation $\mathbf{p}^{t+1}$ via online mirror descent on the probability simplex Δ^{K-1} (Section 3.2).

3.1 Strategy-wise Influence Attribution

At round $t \in [T]$, our training data $\mathcal{D}^t_{\text{train}} = \bigcup_{t'=0}^{t} (x_i, y_i)_{i \in \mathcal{B}^{t'}}$ is composed of the inputs and outputs for the union of all batches up to round t, where each batch $\mathcal{B}^t = \bigcup_{k=1}^{K} \mathcal{B}^t_k$ is composed of sub-batches $\mathcal{B}^t_k$ of size $n^t_k = |\mathcal{B}^t_k|$ selected by strategy $\mathcal{A}_k$. After updating the model parameters $\hat{\theta}^t$ by training the model f_θ on $\mathcal{D}^t_{\text{train}}$, we assess the contribution of each strategy's sub-batch $\mathcal{B}^t_k$ to the validation risk. For $\boldsymbol{\epsilon} \in \mathbb{R}^K$, we define the perturbed training risk:

$$\mathcal{F}_t(\boldsymbol{\epsilon}, \theta) := \mathcal{R}(\mathcal{D}^t_{\text{train}}; \theta) + \sum_{k=1}^{K} \frac{\epsilon_k}{n^t_k} \sum_{(x,y) \in \mathcal{B}^t_k} \mathcal{L}\left((x, y); \theta\right), \tag{1}$$

in which each coefficient ϵ_k reweights every point acquired by strategy $\mathcal{A}_k$. We denote the minimizer of the perturbed risk as $\hat{\theta}^t_{\boldsymbol{\epsilon}} := \arg\min_\theta \mathcal{F}_t(\boldsymbol{\epsilon}, \theta)$ so that $\hat{\theta}^t_{\mathbf{0}} = \hat{\theta}^t$ recovers the unperturbed parameters. The *strategy-wise influence vector* $\mathcal{I}(\hat{\theta}^t) \in \mathbb{R}^K$ is defined as the gradient with respect to $\boldsymbol{\epsilon}$ of the validation risk evaluated at the perturbed parameters, taken at $\boldsymbol{\epsilon} = \mathbf{0}$. Its k-th component $\mathcal{I}_k(\hat{\theta}^t)$ thus quantifies the first-order change in validation risk under upweighting of $\mathcal{A}_k$'s sub-batch. Under standard regularity assumptions on the loss function and the optimality of $\hat{\theta}^t$, it admits a closed-form expression:

Proposition 1 (Strategy-wise Influence Decomposition)**.** *Assume the loss $\mathcal{L}$ is twice continuously differentiable in θ near $\hat{\theta}^t$, that $\hat{\theta}^t$ minimizes the training risk, and that the Hessian*

$H_{\hat{\theta}^t} = \nabla^2_\theta \mathcal{R}(\mathcal{D}^t_{train}; \hat{\theta}^t)$ is positive definite. Then the k-th component of the influence vector is given by:

$$\mathcal{I}_k(\hat{\theta}^t) = -\nabla_\theta \mathcal{R}(\mathcal{D}_{val}; \hat{\theta}^t)^\top H^{-1}_{\hat{\theta}^t} \left(\frac{1}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t\right) \right). \tag{2}$$

The proof appears in Appendix A. We define the *reward* of strategy k at round t as $r^t_k \triangleq -\mathcal{I}_k(\hat{\theta}^t)$. The sign of r^t_k indicates whether $\mathcal{A}_k$'s sub-batch reduced ($r^t_k > 0$) or increased ($r^t_k < 0$) validation risk, while the magnitude reflects the size of the effect.

Implementation details. The inverse Hessian-vector product $\mathbf{v}^t = \nabla_\theta \mathcal{R}(\mathcal{D}_{\text{val}}; \hat{\theta}^t)^\top H^{-1}_{\hat{\theta}^t}$ from Eq. 2 is independent of k, so we compute it once per round via the conjugate gradient method (Martens, 2010) and then obtain each r^t_k via a single inner product. When f_θ is parameterized as a deep neural network, we compute the Hessian with respect to the last layer parameters only (Koh & Liang, 2017) and regularize it using a damping term ($\lambda = 0.01$). Both measures help ensure positive definiteness (see Appendix B.1).

3.2 Fractional Portfolio Allocation

Raw reward values $\mathbf{r}^t$ can be noisy and lead to unstable allocation updates, particularly for strategies $\mathcal{A}_k$ where the sub-batch $\mathcal{B}^t_k$ contained few samples. We therefore use the smoothed reward vector $\bar{\mathbf{r}}^t$, defined as an exponentially weighted moving average over rounds:

$$\bar{\mathbf{r}}^t = (1-\alpha)\bar{\mathbf{r}}^{t-1} + \alpha\, \mathbf{r}^t, \tag{3}$$

with $\bar{\mathbf{r}}^0 = \mathbf{0}$, and $\alpha = 0.5$ throughout our experiments. An intermediate allocation $\mathbf{q}^{t+1}$ is defined as the minimizer of the KL-regularized negative reward:

$$\mathbf{q}^{t+1} = \arg\min_{\mathbf{q}\in\Delta^{K-1}} \left[-\eta^t \langle \bar{\mathbf{r}}^t, \mathbf{q}\rangle + D_{\mathrm{KL}}(\mathbf{q} \,\|\, \mathbf{p}^t) \right], \tag{4}$$

where $\eta^t = \eta_0 / \max_k |\bar{r}^t_k|$ is an adaptive step size, with $\eta_0 = 3$ across all experiments. This proximal-point form, standard in online convex optimization, prevents the allocation from changing too abruptly between rounds. Solving Eq. 4 yields the exponentiated gradient step:

$$q^{t+1}_k = \frac{p^t_k \exp(\eta^t \bar{r}^t_k)}{\sum_{k'=1}^K p^t_{k'} \exp(\eta^t \bar{r}^t_{k'})}. \tag{5}$$

The final allocation $\mathbf{p}^{t+1}$ mixes $\mathbf{q}^{t+1}$ with the uniform distribution over the K strategies, using mixing weight $Kp_{\min}$:

$$p^{t+1}_k = (1 - Kp_{\min})\, q^{t+1}_k + p_{\min}, \tag{6}$$

where $p_{\min} = 1/B$ to guarantee that every strategy has a nonzero allocation. The allocation $\mathbf{p}^{t+1}$ then determines sub-batch sizes for the next round. Finally, strategies select their sub-batches sequentially in decreasing order of allocation, each excluding points already selected, so that the resulting sub-batches are disjoint (Algorithm 1, Appendix B.2).

4 Experiments

We evaluate FractAL against bandit-based and retraining-based strategy selection methods across three task types: classification (CIFAR-10), single-task regression (KEGG metabolic reaction network, SARCOS inverse dynamics, and diamond pricing), and multi-task regression (high-throughput genetic perturbation screens). The screening experiments cover two immune cell types: CD4+ T cells and bone-marrow-derived myeloid cells. Throughout these experiments, we report the test performance of the surrogate model. We also assess whether each selection method concentrates budget on the strongest strategies, ranked by their performance when run alone for the entire AL run, which we refer to as their *pure-strategy performance*. Unless mentioned otherwise, metrics are averaged over 5 data splits and 10 random seeds per split, with standard error reported over the seeds after averaging across the splits.

Algorithm 1 FractAL: Fractional Allocation for Active Learning Strategy Portfolios

Require: Strategy portfolio $\mathcal{P} = \{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$; pool $\mathcal{D}_{\text{pool}}$; initial labeled set $\mathcal{D}_{\text{init}}$; validation set $\mathcal{D}_{\text{val}}$; batch size B; rounds T.

Initialize $\mathcal{D}^0_{\text{train}} \leftarrow \mathcal{D}_{\text{init}}$, $\mathbf{p}^1 \leftarrow (1/K, \ldots, 1/K) \in \Delta^{K-1}$, $\bar{\mathbf{r}}^0 \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots, T-1$ **do**
  Allocate integer budgets n^t_k summing to B
  Form disjoint sub-batches $\mathcal{B}^t_k$ using $\mathcal{A}_k$ ▷ App. B.2
  Form composite batch $\mathcal{B}^t = \bigsqcup_{k=1}^{K} \mathcal{B}^t_k$ and update $\mathcal{D}^t_{\text{train}} \leftarrow \mathcal{D}^{t-1}_{\text{train}} \cup \mathcal{B}^t$
  Train f_θ on $\mathcal{D}^t_{\text{train}}$ to obtain $\hat{\theta}^t$
  Compute reward $r^t_k = -\mathcal{I}_k(\hat{\theta}^t)$ for $k = 1, \ldots, K$ ▷ Eq. 2
  Smooth: $\bar{r}^t_k \leftarrow (1-\alpha)\bar{r}^{t-1}_k + \alpha\, r^t_k$
  Exponentiated gradient step: $q^{t+1}_k \leftarrow \frac{p^t_k \exp(\eta^t \bar{r}^t_k)}{\sum_{k'=1}^{K} p^t_{k'} \exp(\eta^t \bar{r}^t_{k'})}$
  Enforce exploration floor: $p^{t+1}_k \leftarrow (1 - K p_{\min}) q^{t+1}_k + p_{\min}$
end for

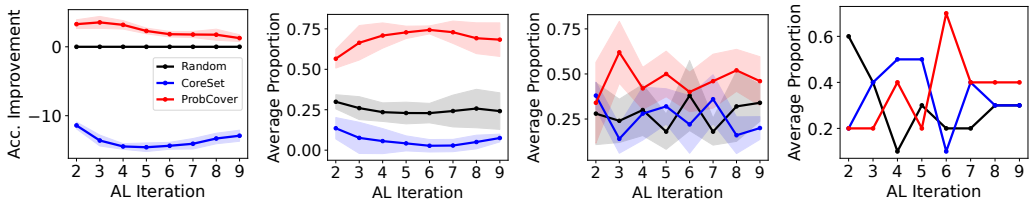
Strategy Portfolio. Within each task type, all strategy selection methods share a common portfolio. For regression tasks, we use random selection (Random), CoreSet (Sener & Savarese, 2018), ProbCover (Yehuda et al., 2022), and TypiClust (Hacohen et al., 2022) ($K = 4$). For classification, we additionally include BADGE (Ash et al., 2020) and three uncertainty-based strategies (Settles, 2009): least-confidence (LeastConf), maximum-entropy (MaxEnt), and minimum-margin (MinMargin), for a total of $K = 8$ strategies. Further details appear in Appendix C.

Baselines. We compare FractAL against five adaptive strategy selection methods and one non-adaptive control. The first is ALBL (Hsu & Lin, 2015), a bandit-based strategy selection method which uses the adversarial Exp4.P solver with the validation metric as feedback. The second uses the same bandit formulation but replaces the Exp4.P solver by the upper confidence bound algorithm (UCB). Third, we cast soft strategy allocation as a linear bandit problem and solve it with SCRiBLe, a self-concordant regularized linear bandit solver (Abernethy et al., 2009). Like FractAL, SCRiBLe produces soft allocations, but it observes only bandit feedback. It therefore isolates the contribution of influence-based attribution. Finally, we include the retraining-based method SelectAL (Hacohen & Weinshall, 2023) and the differentiable strategy selection method AutoAL (Wang et al., 2025). As a non-adaptive control, we also benchmark a uniform allocation across the portfolio (Equal Split). Specifications of all the baselines appear in Appendix D. We also report the performance of the best and worst individual strategies for each setup (in terms of pure-strategy performance). Since those strategies are unknown in advance, these results serve only as a reference range.

Allocation diagnostics. Qualitatively different strategy allocations can yield similar final test performance, so test metrics alone do not reveal whether a method allocates budget sensibly. A natural desideratum for a soft strategy selection method is to concentrate budget on strategies that perform well and prune those that do not. We quantify this for each setup by first identifying the sets of best- and worst-performing strategies from their pure-strategy test performance (independently of any adaptive method). We then compute the overlap between each method's top-ranked strategies and the best-performing set (Best-Set Overlap), and between its bottom-ranked strategies and the worst-performing set (Worst-Set Overlap), where the number of top- (or bottom-) ranked strategies considered equals the size of the Best- (or Worst-)Set (see Appendix E). We report these diagnostics for the three methods that assign an internal score to every strategy at each round (SCRiBLe, SelectAL, and FractAL).

4.1 Classification Benchmark: CIFAR-10

We evaluate FractAL on CIFAR-10 (Krizhevsky et al., 2009) using a ResNet-18 surrogate model (He et al., 2016) trained from scratch over $T = 10$ rounds, with test accuracy as the performance metric. We follow the pre-processing and training protocol of Hacohen & Weinshall (2023) and consider two portfolio configurations. Configuration 1 (C1) is a three-strategy portfolio — CoreSet, Random, and ProbCover — chosen to span weak, neutral, and strong strategies based on their pure-



(a) Test performance of single strategies (b) FractAL allocations (c) SelectAL allocations (d) ALBL allocations

Figure 2: Setup (C1): CIFAR-10. (a) Test accuracy of the three individual strategies. (b–d) Per-round average budget allocation under FractAL, SelectAL, and ALBL. Solid lines denote the mean and shaded bands indicate one standard deviation.

Table 2: Test accuracy (↑) on CIFAR-10.

Method	(C1)	(C2)
Pure strategies (best/worst in hindsight)		
Best pure	*71.2*±0.1 *ProbCover*	*71.2*±0.1 *ProbCover*
Worst pure	*58.7*±0.2 *CoreSet*	*58.7*±0.2 *CoreSet*
Non-adaptive controls		
Random	69.4±0.1	69.4±0.1
Equal Split	67.8±0.1	69.1±0.1
Adaptive strategy selection methods		
ALBL	67.5±0.5	68.6±0.3
UCB	67.5±0.1	69.5±0.1
SCRiBLe	67.7±0.2	69.2±0.1
SelectAL	68.3±0.1	69.4±0.1
AutoAL	67.5±0.1	68.3±0.1
FractAL	**70.2**±0.2	**69.8**±0.2

Table 3: Test MSE ($\times 10^{-1}$; ↓) on regression datasets.

Method	KEGG	SARCOS	Diamonds
Pure strategies (best/worst in hindsight)			
Best pure	*0.49*±0.00 *ProbCover*	*0.72*±0.01 *ProbCover*	*0.08*±0.01 *CoreSet*
Worst pure	*0.97*±0.01 *TypiClust*	*0.79*±0.01 *Random*	*0.12*±0.01 *ProbCover*
Non-adaptive controls			
Random	0.79±0.02	0.79±0.01	0.11±0.01
Equal Split	0.62±0.01	0.76±0.01	**0.08**±0.01
Adaptive strategy selection methods			
ALBL	0.63±0.02	0.77±0.01	0.09±0.01
UCB	0.61±0.01	0.77±0.01	0.10±0.01
SCRiBLe	0.63±0.01	0.75±0.01	0.09±0.01
SelectAL	0.59±0.01	0.76±0.01	0.09±0.01
AutoAL	0.65±0.01	0.82±0.01	0.10±0.01
FractAL	**0.58**±0.01	**0.74**±0.01	**0.08**±0.01

strategy performance (Table 9). This setting tests whether FractAL concentrates budget on the strong strategy and prunes the weak one. Configuration 2 (C2) deploys all eight strategies simultaneously. Because most of the eight strategies perform within one standard error of each other and of Random (Table 9), the selection problem in this setup is considerably harder. Across both configurations, we use $|\mathcal{D}_{\text{init}}| = B = 500$ and $|\mathcal{D}_{\text{val}}| = 1024$. Implementation details, including hyperparameter configurations and optimization protocols, appear in Appendix F.1.

FractAL achieves the highest test accuracy (70.2 ± 0.2) in C1 among selection methods and is the only one to surpass the Random baseline (Table 2). The allocation dynamics in Figure 2b-d explain this gap: FractAL down-weights CoreSet's share toward zero and concentrates budget on ProbCover, while ALBL oscillates between strategies and SelectAL allocates roughly equal shares to CoreSet and Random, despite the substantial gap in their pure-strategy performance. FractAL leads both allocation diagnostics by a wide margin (Table 4, Welch's t-test against the next best baseline, $p < 0.05$ for both). SCRiBLe, which shares FractAL's soft allocation but observes only validation feedback, does no better than a uniform allocation. Neither scalar validation feedback nor deletion-based scoring separates the strategies as cleanly as influence-based attribution.

In the harder C2 setup, the strategy selection methods are separated by much smaller performance margins. FractAL is still highest among adaptive methods (69.8 ± 0.2) and is the only one to significantly exceed Random performance (Welch's t-test, $p < 0.05$). The allocation diagnostics are more informative here. FractAL remains well above chance on both (one-sample t-test against chance, $p < 0.05$), with the larger gap on Worst-Set Overlap reflecting its consistent pruning of CoreSet to zero (Table 4, Figure 6b), while the closest baseline barely clears chance on either. This setup shows that FractAL remains effective with a larger portfolio.

Table 4: Best-Set Overlap (↑) and Worst-Set Overlap (↑) across all seven setups. *Chance* is the expected overlap under a uniform allocation, $|\text{set}|/K$ (set cardinalities in Table 6).

Setup	*Chance*		SCRiBLe		SelectAL		FractAL	
	Best-Set Overlap	Worst-Set Overlap	Best-Set Overlap	Worst-Set Overlap	Best-Set Overlap	Worst-Set Overlap	Best-Set Overlap	Worst-Set Overlap
CIFAR-10 (C1)	0.33	0.33	$0.34_{\pm 0.03}$	$0.35_{\pm 0.05}$	$0.46_{\pm 0.02}$	$0.41_{\pm 0.03}$	$\mathbf{0.79}_{\pm 0.02}$	$\mathbf{0.89}_{\pm 0.02}$
CIFAR-10 (C2)	0.25	0.13	$0.24_{\pm 0.01}$	$0.10_{\pm 0.02}$	$0.29_{\pm 0.02}$	$0.17_{\pm 0.02}$	$\mathbf{0.46}_{\pm 0.01}$	$\mathbf{0.61}_{\pm 0.01}$
KEGG	0.50	0.25	$0.51_{\pm 0.01}$	$0.29_{\pm 0.03}$	$0.50_{\pm 0.01}$	$0.24_{\pm 0.01}$	$\mathbf{0.58}_{\pm 0.01}$	$\mathbf{0.31}_{\pm 0.02}$
SARCOS	0.50	0.75	$0.51_{\pm 0.01}$	$0.75_{\pm 0.01}$	$0.53_{\pm 0.02}$	$0.77_{\pm 0.01}$	$\mathbf{0.54}_{\pm 0.01}$	$\mathbf{0.79}_{\pm 0.01}$
Diamonds	0.50	0.50	$\mathbf{0.52}_{\pm 0.02}$	$\mathbf{0.52}_{\pm 0.01}$	$0.51_{\pm 0.01}$	$0.51_{\pm 0.01}$	$\mathbf{0.52}_{\pm 0.02}$	$\mathbf{0.52}_{\pm 0.02}$
BMDMs (B1)	0.25	0.50	$0.26_{\pm 0.02}$	$0.50_{\pm 0.01}$	$0.59_{\pm 0.03}$	$0.56_{\pm 0.01}$	$\mathbf{0.70}_{\pm 0.10}$	$\mathbf{0.62}_{\pm 0.01}$
T Cells (B2)	0.25	0.50	$0.20_{\pm 0.03}$	$0.50_{\pm 0.01}$	$0.25_{\pm 0.01}$	$0.49_{\pm 0.00}$	$\mathbf{0.57}_{\pm 0.12}$	$\mathbf{0.61}_{\pm 0.05}$

4.2 Regression Benchmarks

We evaluate FractAL on three regression datasets: KEGG metabolic reaction network (Naeem & Asghar, 2011), SARCOS inverse dynamics (Vijayakumar & Schaal, 2000), and diamond pricing (Diamonds; Vanschoren et al., 2014), with input dimensions 27, 21, and 29 respectively. We follow the pre-processing of Holzmüller et al. (2023) and use a three-layer multi-layer perceptron (MLP) as the surrogate model. We report mean squared error (MSE) on the held-out test set. All methods use $|\mathcal{D}_{\text{init}}| = B = 500$ and $|\mathcal{D}_{\text{val}}| = 1024$ over $T = 10$ rounds. Implementation details appear in Appendix F.2.

FractAL achieves the lowest test MSE among selection methods on all three regression datasets (Table 3). FractAL also leads both allocation diagnostics on KEGG and SARCOS, with a tie on Diamonds (Table 4). On Diamonds, the margins are narrow because the pure strategies perform similarly (Table 10), leaving little room to improve from uniform allocation. Allocations from bandit baselines either oscillate (KEGG, SARCOS) or remain near uniform (Diamonds), never converging on a stable allocation (Figures 7–9). This suggests that their feedback is too noisy to effectively discriminate between strategies in this setup.

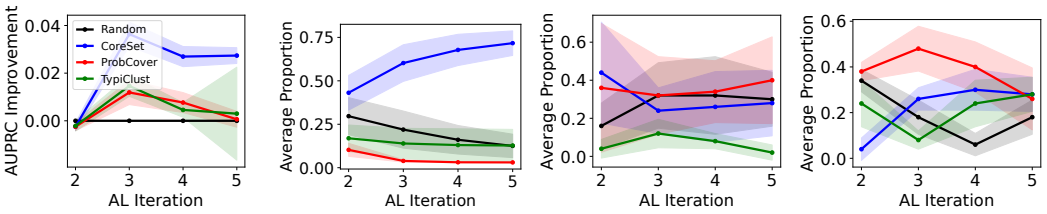
These datasets also illustrate the failure mode that motivates adaptive strategy selection. CoreSet, which performed the worst on CIFAR-10 (Table 9), is among the strongest on all three regression setups (Table 10). Conversely, ProbCover is strongest on KEGG and SARCOS but weakest on Diamonds. No fixed strategy is safe across these setups, which emphasizes the need for adaptive selection.

4.3 Design of High-Throughput Genetic Perturbation Screens

Pooled genetic screens such as Perturb-seq, with a single-cell transcriptomics readout (Dixit et al., 2016), and Optical Pooled Screens (OPS), with an imaging readout (Feldman et al., 2019), make it possible to alter the function of thousands of genes in a single experiment and measure their effects on cells. These data are key to training predictive models of cellular responses to perturbations across biological contexts, which is central to inferring gene regulatory circuits, identifying therapeutic targets, and engineering cellular phenotypes (Rood et al., 2024; Roohani et al., 2024). A recurring difficulty is that perturbation effects measured under one set of conditions (e.g., a different modality, timepoint, or cellular context) may not generalize well to another (Dibaeinia et al., 2026).

This motivates an experimental design problem where one chooses which perturbations to profile in order to maximize the predictive accuracy of the resulting model. This is a well-defined active learning task (Huang et al., 2024). Because each round of wet-lab experimentation takes weeks but provides measurements for hundreds of perturbations in parallel, this setting also matches FractAL's operating regime (DeMeo et al., 2025). We evaluate FractAL on two practical instances of perturbation prediction, each testing a different aspect of generalization: cross-modal prediction from imaging to transcriptomic data (B1), and cross-temporal prediction within a single cell type (B2).

Datasets. For the cross-modal task (B1), we rely on matched Perturb-seq and OPS datasets of primary mouse bone-marrow-derived myeloid cells (BMDMs) under immune stimulation (Anony-



(a) Test performance of single strategies (b) FractAL allocations (c) SelectAL allocations (d) ALBL allocations

Figure 3: Setup (B2): T cell perturbation prediction. (a) Test AUPRC of the four pure strategies. (b–d) Per-round average budget allocation under FractAL, SelectAL and ALBL. Solid lines denote the mean and shaded bands indicate one standard deviation.

mous, 2026), retaining the 514 perturbations with significant effects in both modalities. The goal is to predict each perturbation's effect on gene expression (quantified as the log-fold change between perturbed and control cells; Boyeau et al., 2023) from embeddings of the imaging data. For the cross-temporal task (B2), we rely on a genome-scale Perturb-seq screen of primary human CD4+ T cells profiled at multiple time points after T cell receptor stimulation (Zhu et al., 2026), retaining the 709 perturbations with significant effects at both time points. The goal is to predict the late-stage transcriptional response (48 hours post-stimulation) of each perturbation from observations of its early-stage response (8 hours post-stimulation), testing the model's ability to extrapolate across the temporal dynamics of T cell activation. Data processing details for both tasks appear in Appendix F.3.

Experimental setup. Across both tasks, we use an initial labeled set of size $|\mathcal{D}_{\text{init}}| = 35$, batch size $B = 35$, and a validation set of size $|\mathcal{D}_{\text{val}}| = 40$. The held-out test sets are of size $|\mathcal{D}_{\text{test}}| = 103$ (B1) and 141 (B2), corresponding to roughly 20% of each dataset. Batch sizes are small because paired measurements of this kind are recent, and the resulting datasets are still limited in size. We run active learning for $T = 6$ rounds (measuring roughly 40–50% of the total pool) using a single-layer MLP as the surrogate model. Classical regression metrics such as MSE are known to be misaligned with the downstream task of identifying which genes respond to a perturbation (Zhu et al., 2025; Littman et al., 2025). We therefore report the area under the precision-recall curve (AUPRC) for detecting differentially expressed genes. The portfolio comprises Random, CoreSet, ProbCover, and TypiClust.

Table 5: AUPRC (↑) for perturbation prediction tasks.

Method	(B1): BMDMs	(B2): T cells
Pure strategies (best/worst in hindsight)		
Best pure	*0.4377*$\pm$0.0005 *CoreSet*	*0.3696*$\pm$0.0002 *CoreSet*
Worst pure	*0.4165*$\pm$0.0011 *Random*	*0.3403*$\pm$0.0045 *Random*
Non-adaptive controls		
Random	0.4165$\pm$0.0011	0.3403$\pm$0.0045
Equal Split	0.4279$\pm$0.0006	0.3442$\pm$0.0065
Adaptive strategy selection methods		
ALBL	0.4289$\pm$0.0018	0.3548$\pm$0.0021
UCB	0.4323$\pm$0.0013	0.3590$\pm$0.0015
SCRiBLe	0.4299$\pm$0.0006	0.3557$\pm$0.0006
SelectAL	0.4346$\pm$0.0010	0.3579$\pm$0.0012
AutoAL	0.4336$\pm$0.0005	0.3603$\pm$0.0008
FractAL	**0.4363**$\pm$0.0006	**0.3642**$\pm$0.0013

Results. FractAL achieves the highest test AUPRC among adaptive methods on both perturbation prediction tasks (Table 5). It also leads every allocation diagnostic (Table 4). FractAL steadily concentrates budget on the strongest strategy (here, CoreSet) on both tasks, whereas the bandit baselines (UCB, SCRiBLe, ALBL) never settle on a stable allocation (Figures 3, 10 and 11). This behavior is consistent with the bandit feedback signal being too sparse to confidently commit budget within the few rounds available in this setting. FractAL also downweights the weakest strategy (Random), whereas the closest baseline (SelectAL) still retains substantial allocation to it. On B2, FractAL is the only method to significantly outperform chance on Worst-Set Overlap (one-sample t-test against chance, $p < 0.05$). This suggests the influence-based reward is a more reliable signal for pruning than SelectAL's deletion-based test.

4.4 Aggregate Results, Ablations, and Runtime

Aggregate performance. Metrics within a single experiment separate methods only by narrow margins. We therefore summarize performance across all seven setups, normalizing the metric so that

0 corresponds to matching random performance and 1 to matching the best strategy in the portfolio for that setup. FractAL attains the highest mean gain (0.689) and is the only method with a positive worst-case gain (0.222): every other method falls below Random on at least one setup. It is also the top-ranked method on average (mean rank 1.14 of 7, median 1), ahead of the next closest method, SelectAL (3.14, median 3). FractAL's advantage reflects consistent competitiveness across every task we evaluate (Appendix G).

Ablation Studies. We conduct a series of ablation studies to isolate the contribution of each component of FractAL to its performance across the three task domains (Appendix I). FractAL's performance is robust to the choice of the smoothing parameter α. Mirror descent outperforms an additive update with clipping and renormalization. Soft fractional allocation improves over hard allocation (i.e., picking one strategy per round) on five out of seven setups while tying on the remaining two. Finally, the influence-based reward matches the performance of a retraining-based signal, and the two correlate well (Spearman's $\rho = 0.7$).

Runtime Analysis. We report the runtime for an entire active learning run on the CIFAR-10 dataset (C2, $K = 8$) on a single NVIDIA L40S GPU (Figure 4, exact numbers in Appendix J). For FractAL and each of the baselines, we evaluate the time spent querying strategies in the portfolio and assembling the batch (Acquisition) versus the time spent on computing the reward signal and the allocation (Allocation Update). FractAL and SCRiBLe query every strategy each round, so their acquisition time is longer than that of the methods that commit to one at every round (ALBL, UCB, SelectAL). Allocation-update time varies much more across methods: 0.5 min for FractAL, compared with 19.8 min for SelectAL and 28.5 min for AutoAL, which represents at least a 40-fold reduction compared to methods that involve additional model training. The runtime advantage of our method grows with both the portfolio size K and the size of the surrogate model, since FractAL computes a single inverse-Hessian-vector product per round regardless of K, while retraining-based attribution scales linearly in both. The bandit baselines have a faster allocation update than FractAL, but they perform poorly in this study (Table 4). Overall, FractAL completes an entire AL run $2.5\times$ faster than the second-best-performing method in the study (SelectAL), and $5\times$ faster than AutoAL.

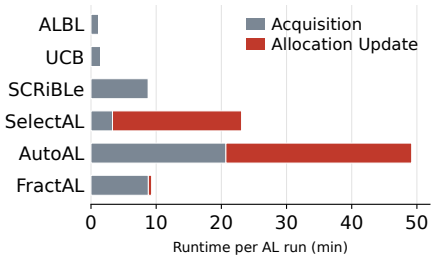


Figure 4: Runtime for an AL run on (C2).

5 Discussion

We addressed the problem of strategy selection in active learning, where strategies perform unpredictably across tasks, datasets and model architectures. We proposed FractAL, a fractional allocation framework that acquires data from multiple strategies in parallel and updates their budget shares at each round using influence-based reward attribution. Aggregated across the seven setups, FractAL is the only method whose worst-case performance exceeds that of random sampling; every baseline falls below it on at least one setup (Appendix G). Across classification, regression, and high-throughput genetic perturbation screens, FractAL consistently up-weights the strongest strategy, and down-weights the ones with poor performance. In contrast, bandit-based baselines oscillate between strategies, and retraining-based methods underperform in this study. The allocation advantage is clearest in the iteration-scarce, large-batch regime that motivates the framework, where each round of feedback must carry substantial signal to be useful.

Several directions warrant further investigation. Influence functions are known to be fragile in deep networks (Basu et al., 2021), and alternative data valuation frameworks — Shapley values (Ghorbani & Zou, 2019), gradient similarity (Pruthi et al., 2020), or Banzhaf values (Wang & Jia, 2023) — may stabilize the reward signal under more complex surrogates. Beyond strategy selection, the fractional allocation framework extends to allocate budget across competing *priors* (e.g., protein–protein interaction networks (Szklarczyk et al., 2025) or gene ontology annotations (Aleksander et al., 2025)). Finally, we also see this problem as a promising direction for active experimental design more broadly.

AI USE STATEMENT

In this work, we used generative AI tools for editing assistance (grammar, clarity, and LaTeX formatting) on the manuscript text, and for code suggestions while implementing the numerical experiments. Specifically, all such AI-assisted code suggestions, which were limited to auxiliary code such as boilerplate plotting and data-loading scaffolding, were reviewed and tested for correctness by the authors. We did not use generative AI tools to generate research ideas, experimental results, or to implement the proposed methodology. The authors take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses publicly available or de-identified genomic screening data (Section 4.3, Data Availability Statement in the Appendix); no human-subjects data collection was performed by the authors. We do not foresee direct harmful applications of a strategy-selection method for active learning.

REPRODUCIBILITY STATEMENT

We provide full derivations and assumptions underlying Proposition 1 in Appendix A, including the practical approximations (last-layer Hessian, damping) used in our implementation. Pseudocode for FractAL appears as Algorithm 1 in the main text, and for all baselines in Appendix D. Hyperparameters, optimizer settings, and architectural details for every setup are listed in Appendix F (summarized in Table 7), and dataset sources and preprocessing steps appear alongside each setup's description and in the Data Availability Statement. We include a supplementary zip file containing our full codebase and the datasets required to reproduce all reported results.

REFERENCES

Jacob D Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Conference on Learning Theory*, 2009.

Suzi A Aleksander, James P Balhoff, Seth Carbon, J Michael Cherry, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, David P Hill, Patrick Kalita, et al. The Gene Ontology knowledge-base in 2026. *Nucleic Acids Research*, 54(D1):gkaf1292, 2025.

Anonymous. Companion manuscript in preparation. Data available to reviewers at `https://osf.io/rpjqc/overview?view_only=31aa1bcbaa8b4963b48831715b768c70`; will be made public for the camera-ready version, 2026.

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

Yoram Baram, Ran El Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *International Conference on Learning Representations*, 2021.

Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandit algorithms with supervised learning guarantees. In *International Conference on Artificial Intelligence and Statistics*, pp. 19–26, 2011.

Pierre Boyeau, Jeffrey Regier, Adam Gayoso, Michael I Jordan, Romain Lopez, and Nir Yosef. An empirical Bayes method for differential expression analysis of single cells with deep generative models. *Proceedings of the National Academy of Sciences*, 120(21):e2209124120, 2023.

Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A Guertin, Joo Han Chang, Robert A Lindquist, Jason Moffat, et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology*, 7(10):R100, 2006.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 2020.

Benjamin DeMeo, Charlotte Nesbitt, Samuel A Miller, Daniel B Burkhardt, Inna Lipchina, Doris Fu, Peter Holderrieth, David Kim, Sergey Kolchenko, Artur Szalata, et al. Active learning framework leveraging transcriptomics identifies modulators of disease phenotypes. *Science*, 390(6776):8577, 2025.

Payam Dibaeinia, Sudarshan Babu, Mei Knudson, Ali ElSheikh, Yibo Wen, Han Liu, Jason Perera, and Aly A Khan. Virtual cells need context, not just scale. *International Conference on Machine Learning*, 2026.

Atray Dixit, Oren Parnas, Biyu Li, Jenny Chen, Charles P Fulco, Livnat Jerby-Arnon, Nemanja D Marjanovic, Danielle Dionne, Tyler Burks, Raktima Raychowdhury, et al. Perturb-Seq: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens. *Cell*, 167(7): 1853–1866, 2016.

David Feldman, Avtar Singh, Jonathan L Schmid-Burgk, Rebecca J Carlson, Anja Mezger, Anthony J Garrity, Feng Zhang, and Paul C Blainey. Optical pooled screens in human cells. *Cell*, 179(3): 787–799, 2019.

Nathan C Frey, Isidro Hötzel, Samuel D Stanton, Ryan Kelly, Robert G Alberstein, Emily K Makowski, Karolis Martinkus, Daniel Berenberg, Jack Bevers III, Tyler Bryson, et al. Lab-in-the-loop therapeutic antibody design with deep learning. *bioRxiv*, pp. 2025–02, 2025.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251, 2019.

Guy Hacohen and Daphna Weinshall. How to select which active learning strategy is best suited for your specific problem and budget. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. *International Conference on Machine Learning*, 2022.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

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

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A framework and benchmark for deep batch active learning for regression. *Journal of Machine Learning Research*, 24(164):1–81, 2023.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv*, 2011.

Wei-Ning Hsu and Hsuan-Tien Lin. Active learning by learning. In *AAAI Conference on Artificial Intelligence*, pp. 2659–2665, 2015.

Kexin Huang, Romain Lopez, Jan-Christian Hütter, Takamasa Kudo, Antonio Rios, and Aviv Regev. Sequential optimal experimental design of perturbation screens guided by multi-modal priors. In *International Conference on Research in Computational Molecular Biology*, pp. 17–37, 2024.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pp. 1885–1894, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

Russell Littman, Jacob Levine, Sepideh Maleki, Yongju Lee, Vladimir Ermakov, Lin Qiu, Alexander Wu, Kexin Huang, Romain Lopez, Gabriele Scalia, et al. Gene-embedding-based prediction and functional evaluation of perturbation expression responses with presage. *bioRxiv*, pp. 2025–06, 2025.

Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *IEEE/CVF International Conference on Computer Vision*, pp. 9274–9283, 2021.

Carsten Lüth, Till Bungert, Lukas Klein, and Paul Jaeger. Navigating the pitfalls of active learning evaluation: A systematic framework for meaningful performance assessment. *Advances in Neural Information Processing Systems*, 36:9789–9836, 2023.

James Martens. Deep learning via Hessian-free optimization. In *International Conference on Machine Learning*, volume 27, pp. 735–742, 2010.

Prateek Munjal, Nasir Hayat, Munawar Hayat, Jamshid Sourati, and Shadab Khan. Towards robust and reproducible active learning using neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 223–232, 2022.

M. Naeem and S. Asghar. KEGG metabolic reaction network (undirected), 2011.

Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy M Hospedales. Dynamic ensemble active learning: A non-stationary bandit with expert advice. In *International Conference on Pattern Recognition*, pp. 2269–2276, 2018.

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

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys*, 54(9):1–40, 2021.

Jennifer E Rood, Anna Hupalowska, and Aviv Regev. Toward a foundation model of causal cell and tissue biology with a perturbation cell and tissue atlas. *Cell*, 187(17):4520–4545, 2024.

Yusuf Roohani, Kexin Huang, and Jure Leskovec. Predicting transcriptional outcomes of novel multigene perturbations with GEARS. *Nature Biotechnology*, 42(6):927–935, 2024.

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

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2009.

Justin S Smith, Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of Chemical Physics*, 148(24), 2018.

Damian Szklarczyk, Katerina Nastou, Mikaela Koutrouli, Rebecca Kirsch, Farrokh Mehryary, Radja Hachilif, Dewei Hu, Matteo E Peluso, Qingyao Huang, Tao Fang, et al. The STRING database in 2025: protein networks with directionality of regulation. *Nucleic Acids Research*, 53(D1): D730–D737, 2025.

Vincent Q Tran, Matthew Nemeth, Liam J Bartie, Sita S Chandrasekaran, Alison Fanton, Hyungseok C Moon, Brian L Hie, Silvana Konermann, and Patrick D Hsu. Rapid directed evolution guided by protein language models and epistatic interactions. *Science*, pp. 1820, 2026.

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

Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *International Conference on Machine Learning*, volume 1, pp. 288–293, 2000.

Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 6388–6421, 2023.

Yifeng Wang, Xueying Zhan, and Siyu Huang. AutoAL: Automated active learning with differentiable query strategy search. In *International Conference on Machine Learning*, 2025.

Ethan Weinberger, Chris Lin, and Su-In Lee. Isolating salient variations of interest in single-cell data with contrastiveVI. *Nature Methods*, 20(9):1336–1345, 2023.

Thorben Werner, Johannes Burchert, Maximilian Stubbemann, and Lars Schmidt-Thieme. A cross-domain benchmark for active learning. *Advances in Neural Information Processing Systems*, 37: 62875–62911, 2024.

Meng Xia and Ricardo Henao. Reliable active learning via influence functions. *Transactions on Machine Learning Research*, 2023:1364, 2023.

Minjie Xu and Gary Kazantsev. Understanding goal-oriented active learning via influence functions. *Workshop on ML with Guarantees, Advances in Neural Information Processing Systems*, 2019.

Jason Yang, Ravi G Lal, James C Bowden, Raul Astudillo, Mikhail A Hameedi, Sukhvinder Kaur, Matthew Hill, Yisong Yue, and Frances H Arnold. Active learning-assisted directed evolution. *Nature Communications*, 16(1):714, 2025.

Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active learning through a covering lens. *Advances in Neural Information Processing Systems*, 35:22354–22367, 2022.

Hongxu Zhu, Amir Asiaee, Leila Azinfar, Jun Li, Han Liang, Ehsan Irajizad, Kim-Anh Do, and James P Long. AUPRC: a metric for evaluating the performance of in-silico perturbation methods in identifying differentially expressed genes. *Briefings in bioinformatics*, 26(5):bbaf426, 2025.

Ronghui Zhu, Emma Dann, Jun Yan, Justine Reyes Retana, Ryunosuke Goto, Reese C Guitche, Lillian K Petersen, Mineto Ota, Jonathan K Pritchard, and Alexander Marson. Genome-scale perturb-seq in primary human CD4+ T cells maps context-specific regulators of T cell programs and human immune traits. *Cell*, 2026.

APPENDICES

The following list gives an overview of the structure of the Appendix:

- Data availability statement for the datasets used in our experiments.
- Discussion of the broader societal impacts of our work.
- Appendix A: derivation of the strategy-wise influence score.
- Appendix B: practical estimation of influence scores and construction of the composite batch.
- Appendix C: the acquisition strategies in our portfolio.
- Appendix D: details and pseudocode for each strategy selection baseline.
- Appendix E: definition of the allocation diagnostics.
- Appendix F: datasets, surrogate models, and hyperparameters for each setup.
- Appendix G: aggregate gain and rank of each method across the seven setups.
- Appendix H: per-setup results with the pure strategies added.
- Appendix I: ablations on the smoothing parameter, allocation update, reward attribution, and soft allocation.
- Appendix J: wall-clock runtime by phase for each method.
- Appendix K: budget allocation dynamics for all methods and setups.

DATA AVAILABILITY STATEMENT

All the classification datasets (CIFAR-10)[1], standard regression datasets (KEGG, SARCOS and Diamonds)[2][3][4] and the CD4+ T cell genome-wide Perturb-seq data[5] are publicly accessible. The OPS images and the corresponding Perturb-Seq data for the BMDMs experiment (Anonymous, 2026) will be released when the original manuscript is published.

BROADER SOCIETAL IMPACTS

The primary societal impact of FractAL lies in its potential to accelerate the pace of scientific discovery while significantly reducing resource waste. By optimizing the allocation of experimental budgets in high-throughput settings, such as genetic perturbation screens and chemical property prediction, our framework enables researchers to achieve critical insights with fewer physical experiments. This efficiency not only lowers the financial barriers to complex biological research but also reduces the environmental footprint and logistical overhead associated with large-scale wet-lab operations. Furthermore, the generalizable nature of our active learning approach could extend these benefits to other resource-constrained fields, such as climate modeling or material science. While the increased efficiency of genetic research necessitates continued adherence to established biosafety and ethical guidelines, FractAL itself is a methodological contribution intended to democratize and streamline the process of data-driven discovery.

[1] Dataset Source: `https://www.cs.toronto.edu/~kriz/cifar.html`

[2] KEGG: `https://archive.ics.uci.edu/ml/datasets/KEGG+Metabolic+Reaction+Network+%28Undirected%29`

[3] SARCOS Source: `http://www.gaussianprocess.org/gpml/data/`

[4] Diamonds Source: `https://www.openml.org/d/42225`

[5] Dataset Source: `https://virtualcellmodels.cziscience.com/dataset/genome-scale-tcell-perturb-seq`

A Proof of Proposition 1

Proposition 1 (Strategy-wise Influence Decomposition)**.** *Assume the loss $\mathcal{L}$ is twice continuously differentiable in θ near $\hat{\theta}^t$, that $\hat{\theta}^t$ minimizes the training risk, and that the Hessian $H_{\hat{\theta}^t} = \nabla^2_\theta \mathcal{R}(\mathcal{D}^t_{train}; \hat{\theta}^t)$ is positive definite. Then the k-th component of the influence vector is given by:*

$$\mathcal{I}_k(\hat{\theta}^t) = -\nabla_\theta \mathcal{R}(\mathcal{D}_{val}; \hat{\theta}^t)^\top H^{-1}_{\hat{\theta}^t} \left(\frac{1}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t\right) \right). \tag{2}$$

Assumptions. These assumptions are standard in the influence-function literature (Koh & Liang, 2017). They do not hold exactly for deep networks trained by stochastic gradient descent using early-stopping. We describe those, as well as practical workarounds in Appendix B.

Proof. We adapt the standard influence-function derivation (Koh & Liang, 2017; Xu & Kazantsev, 2019) to our portfolio setting, where the perturbation vector $\boldsymbol{\epsilon} \in \mathbb{R}^K$ has one component per strategy rather than one per training point. The proof proceeds in two steps. First, we derive the parameter sensitivity $\frac{d\hat{\theta}^t_{\boldsymbol{\epsilon}}}{d\boldsymbol{\epsilon}}$ at $\boldsymbol{\epsilon} = \mathbf{0}$ via the implicit function theorem. Second, we apply the chain rule to obtain the closed form for $\mathcal{I}_k(\hat{\theta}^t)$.

Step 1: Parameter Sensitivity We first recall the perturbed risk from Eq. 1: for a perturbation vector $\boldsymbol{\epsilon} \in \mathbb{R}^K$,

$$\mathcal{F}_t(\boldsymbol{\epsilon}, \theta) = \mathcal{R}(\mathcal{D}^t_{\text{train}}; \theta) + \sum_{k=1}^{K} \frac{\epsilon_k}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \mathcal{L}\left((x,y); \theta\right). \tag{7}$$

The perturbed parameter $\hat{\theta}^t_{\boldsymbol{\epsilon}}$ is its minimizer:

$$\hat{\theta}^t_{\boldsymbol{\epsilon}} := \arg\min_\theta \mathcal{F}_t(\boldsymbol{\epsilon}, \theta), \tag{8}$$

with $\hat{\theta}^t_{\mathbf{0}} = \hat{\theta}^t$.

By optimality, it satisfies the first-order condition:

$$\nabla_\theta \mathcal{R}\left(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t_{\boldsymbol{\epsilon}}\right) + \sum_{k=1}^{K} \frac{\epsilon_k}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t_{\boldsymbol{\epsilon}}\right) = 0. \tag{9}$$

Let $\Delta_{\boldsymbol{\epsilon}} := \hat{\theta}^t_{\boldsymbol{\epsilon}} - \hat{\theta}^t$. As $\boldsymbol{\epsilon} \to \mathbf{0}$, we have $\Delta_{\boldsymbol{\epsilon}} \to \mathbf{0}$.

Since $\mathcal{L}$ is twice continuously differentiable and $H_{\hat{\theta}^t}$ is positive definite, the implicit function theorem applied to Eq. 9 implies that, for $\boldsymbol{\epsilon}$ near $\mathbf{0}$, the minimizer $\hat{\theta}^t_{\boldsymbol{\epsilon}}$ near $\hat{\theta}^t$ is a continuously differentiable function of $\boldsymbol{\epsilon}$. In particular, $\Delta_{\boldsymbol{\epsilon}} = O(\|\boldsymbol{\epsilon}\|)$ as $\boldsymbol{\epsilon} \to \mathbf{0}$. Taylor-expanding Eq. 9 around $\hat{\theta}^t$ and discarding terms of order $\boldsymbol{\epsilon}\Delta_{\boldsymbol{\epsilon}}$ and higher:

$$0 = \nabla_\theta \mathcal{R}(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t) + \nabla^2_\theta \mathcal{R}(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t)\, \Delta_{\boldsymbol{\epsilon}} + \sum_{k=1}^{K} \frac{\epsilon_k}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t\right) + O(\|\boldsymbol{\epsilon}\|^2). \tag{10}$$

Since $\hat{\theta}^t$ minimizes $\mathcal{R}(\mathcal{D}^t_{\text{train}}; \cdot)$, the first term vanishes: $\nabla_\theta \mathcal{R}(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t) = 0$. Recognizing $\nabla^2_\theta \mathcal{R}(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t) = H_{\hat{\theta}^t}$ and using the regularity assumption to invert it, we solve for $\Delta_{\boldsymbol{\epsilon}}$:

$$\Delta_{\boldsymbol{\epsilon}} = -H^{-1}_{\hat{\theta}^t} \left[\sum_{k=1}^{K} \frac{\epsilon_k}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t\right) + O(\|\boldsymbol{\epsilon}\|^2) \right] \tag{11}$$

This expresses the parameter shift $\Delta_{\boldsymbol{\epsilon}}$ as a function of $\boldsymbol{\epsilon}$, with each ϵ_k scaling its strategy's gradient contribution.

Step 2: Chain Rule for the Influence Vector By definition, the k-th component of the influence vector is the partial derivative of the validation risk at the perturbed parameter, evaluated at $\boldsymbol{\epsilon} = \mathbf{0}$:

$$\mathcal{I}_k(\hat{\theta}^t) = \frac{\partial}{\partial \epsilon_k} \mathcal{R}\left(\mathcal{D}_{\text{val}}; \hat{\theta}^t_{\boldsymbol{\epsilon}}\right)\Big|_{\boldsymbol{\epsilon}=\mathbf{0}}. \tag{12}$$

Applying the chain rule:

$$\mathcal{I}_k(\hat{\theta}^t) = \nabla_\theta \mathcal{R}(\mathcal{D}_{\text{val}}; \hat{\theta}^t)^\top \frac{\partial \hat{\theta}^t_{\boldsymbol{\epsilon}}}{\partial \epsilon_k}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}}. \tag{13}$$

The Jacobian column on the right is read off from Eq. 11 by differentiating with respect to ϵ_k at $\boldsymbol{\epsilon} = \mathbf{0}$ (higher order terms of $\boldsymbol{\epsilon}$ vanish here at $\boldsymbol{\epsilon} = \mathbf{0}$):

$$\frac{\partial \hat{\theta}^t_{\boldsymbol{\epsilon}}}{\partial \epsilon_k}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} = -H^{-1}_{\hat{\theta}^t} \left(\frac{1}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t\right) \right). \tag{14}$$

Substituting into the chain rule expression:

$$\mathcal{I}_k(\hat{\theta}^t) = -\nabla_\theta \mathcal{R}(\mathcal{D}_{\text{val}}; \hat{\theta}^t)^\top H^{-1}_{\hat{\theta}^t} \left(\frac{1}{n^t_k} \sum_{(x,y)\in\mathcal{B}^t_k} \nabla_\theta \mathcal{L}\left((x,y); \hat{\theta}^t\right) \right), \tag{15}$$

which matches Eq. 2. □

B Implementation Details for FractAL

B.1 Practical Estimation of Influence Scores

Proposition 1 assumes an exact empirical risk minimizer and a positive-definite Hessian, and is stated for the full parameter vector. None of these holds exactly in our setting. We describe below how each departs from the idealized statement, the approximation we use in response, and empirical evidence that the resulting estimator remains reliable.

Optimality of the trained parameters. The derivation uses the first-order optimality condition $\nabla_\theta R(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t) = 0$ to eliminate the leading term of the Taylor expansion. In practice however, $\hat{\theta}^t$ is obtained by solving a stochastic optimization problem with a first-order method, using an early stopping criterion on the validation set. Therefore, this condition holds only approximately.

Positive definiteness of the Hessian. The Hessian of the training risk at the trained parameters, $H_{\hat{\theta}^t} = \nabla^2_\theta R(\mathcal{D}^t_{\text{train}}; \hat{\theta}^t)$, need not be positive definite. For the multiclass softmax head used on CIFAR-10, it is provably singular since a common shift of all logits leaves the loss unchanged. We therefore add an explicit damping term before inversion in every setup, so that the system we invert is $H_{\hat{\theta}^t} + \lambda_{\text{damp}} I$ with $\lambda_{\text{damp}} = 0.01$.

Last-layer restriction. Exact computation of the influence scores may become intractable for surrogate models with a large number of parameters. We therefore restrict the Hessian to the last-layer parameters θ_L (Koh & Liang, 2017). In the case of ResNet-18, this corresponds to the final linear classification layer, and for the MLPs it is the final fully connected layer.

Reliability of the approximation. We validate the resulting scores by comparing them against the retraining-based signal they approximate. Across rounds on CIFAR-10 (C1), the two rewards rank strategies consistently, with a Spearman correlation of 0.7 - 0.8 (Table 15). Additionally, substituting one for the other leaves downstream performance comparable on all seven setups (Table 14).

B.2 Forming the Composite Batch

At each round t, the composite batch $\mathcal{B}^t$ is formed by combining sub-batches independently proposed by each strategy in the portfolio. We describe here the procedure used to guarantee that the realized batch contains exactly B unique points with each attributed to a single strategy (Algorithm 1).

Budget rounding. Since Bp^t_k is generally not an integer, we allocate $n^t_k = \lfloor Bp^t_k \rfloor$ points to each strategy $\mathcal{A}_k$ and assign the remainder $B - \sum_k \lfloor Bp^t_k \rfloor$ to the strategy with the largest allocation, $\mathcal{A}_j$ with $j = \arg\max_k p^t_k$ (ties broken by strategy index). The realized sub-batch sizes therefore sum to exactly B.

Overlap resolution. All strategies select candidates exclusively from the currently unlabeled pool $\mathcal{D}_{\text{pool}} \setminus \mathcal{D}^{t-1}_{\text{train}}$, so no previously acquired point can be re-selected. Within a round, however, two strategies may independently propose the same unlabeled point. We resolve this by having strategies select sequentially, in decreasing order of their allocated budget p^t_k (ties broken by strategy index): each strategy's candidate set excludes points already claimed by a higher-allocated strategy earlier in the same round. This guarantees the realized sub-batches $\mathcal{B}^t_1, \ldots, \mathcal{B}^t_K$ are disjoint, and that every acquired point is attributed to exactly one strategy in the reward computation.

Empirical overlap rate. We measured how often this overlap resolution rule is actually invoked by logging, prior to resolution, how frequently two strategies' independently-ranked candidate sets would have proposed the same point. On CIFAR-10 (C2, K=8, B=500), an average of 3 of the 500 points acquired per round were claimed by more than one strategy (0.6% of the batch). The rule therefore affects well under 1% of any batch, and the choice of tie-breaking order therefore has a negligible effect on the reported results.

C ACTIVE LEARNING STRATEGIES

Below we provide details about each of the data acquisition strategies used in our pool. The first four are suitable for all experiments:

- **Random** is a strategy that randomly chooses a given number of points from the unlabeled pool
- **ProbCover** (Yehuda et al., 2022) selects unlabeled points that maximize coverage of the data distribution under a probabilistic similarity graph, ensuring chosen samples "cover" as many other points as possible.
- **TypiClust** (Hacohen et al., 2022) performs k-means clustering in an embedding space and samples typical points from the clusters ordered by their respective total size and labeled set size.
- **CoreSet** (Sener & Savarese, 2018) iteratively picks the most distant points from the already labeled points, given an embedding of all the unlabeled inputs. We use the Euclidean distance as the distance metric in our experiments.

The last four apply specifically to classification tasks in our work:

- **BADGE.** BADGE (Ash et al., 2020) represents each unlabeled sample as the gradient of the cross-entropy loss it induces on the last layer of the surrogate model and then samples diverse points in the gradient embedding space.
- **LeastConf** strategy picks the samples that have the lowest maximum predicted probability.
- **MaxEnt** picks samples with highest entropy in the predicted probabilities.
- **MinMargin** picks the samples which have minimum difference between the probability of top two predicted classes.

D STRATEGY SELECTION BASELINES

D.1 ALBL: ACTIVE LEARNING BY LEARNING

ALBL (Hsu & Lin, 2015) utilizes the Exp4.P bandit solver to treat the strategies as experts giving advice over the unlabeled data points. At iteration t, the solver maintains weights w_k^t and determines the selection probability of experts s_k^t as:

$$s_k^t = (1 - Kp_{\min}) \frac{w_k^t}{\sum_{k'=1}^{K} w_{k'}^t} + p_{\min} \tag{16}$$

where we set $p_{\min} = 1/B$. The strategy for the next iteration is then sampled with the distribution s^t. Let's assume the chosen strategy is $\mathcal{A}_k$. Thus, the allocation vector $\mathbf{p}^t \in \mathbb{R}^K$ is given by

$$p_{k'}^t = \begin{cases} 1 & \text{if } k' = k, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall k' \in [K], \tag{17}$$

Conceptually, ALBL acquires the whole batch of points using $\mathcal{A}_k$ and receives a partial information reward $\mathbf{r}^t \in [0,1]^K$ based on validation performance, defined as follows:

$$r_k^t = \begin{cases} \text{Accuracy on } \mathcal{D}_{\text{val}} & \text{Classification Task} \\ \frac{1}{1+\text{Loss}(\mathcal{D}_{\text{val}})} & \text{Regression Task} \end{cases} \tag{18}$$

and $r_{k'}^t = 0$ for all $k' \neq k$. This reward is then utilized to update strategy selection weights using an exponential rule:

$$w_k^{t+1} = w_k^t \exp\left(\frac{p_{\min}}{2}(\hat{y}_k(t) + \hat{v}_k(t) p_{\min})\right) \tag{19}$$

where $\hat{y}_k(t)$ and $\hat{v}_k(t)$ are the reward and variance estimators. The whole algorithm is explained in Algorithm 2.

Algorithm 2 ALBL: Active Learning by Learning

Require: Strategy portfolio $\mathcal{P} = \{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$; pool $\mathcal{D}_{\text{pool}}$; initial labeled set $\mathcal{D}_{\text{init}}$; validation set $\mathcal{D}_{\text{val}}$; batch size B; rounds T

Initialization: $w_k^1 = 1, \forall k \in [K]$; $\mathcal{D}_{\text{train}}^0 = \mathcal{D}_{\text{init}}$
for $t = 1, 2, \ldots T-1$ **do**
  Receive advice vectors $\xi^1(t), \ldots, \xi^K(t)$, with $\xi^k(t) = \mathbf{e}_k \in \{0,1\}^K$
  $W^t = \sum_{k=1}^{K} w_k^t$
  for $k = 1, \ldots, K$ **do**
    $s_k^t = (1 - Kp_{\min}) \sum_{k'=1}^{K} \frac{w_{k'}^t \xi_k^{k'}(t)}{W^t} + p_{\min} = (1 - Kp_{\min}) \frac{w_k^t}{W^t} + p_{\min}$
  end for
  Draw action $k^t \sim (s_1^t, \ldots, s_K^t)$
  Define allocation vector $\mathbf{p}^t$ with $p_{k^t}^t = 1$ and $p_{k'}^t = 0, \forall k' \in [K], k' \neq k^t$
  Sample batch $\mathcal{B}^t$:

$$\mathcal{B}^t \leftarrow \text{Sample}(\mathcal{A}_{k^t}, B, \mathcal{D}_{\text{pool}} \setminus \mathcal{D}_{\text{train}}^{t-1})$$

  Query labels for $\mathcal{B}^t$ and update the training dataset:

$$\mathcal{D}_{\text{train}}^t \leftarrow \mathcal{D}_{\text{train}}^{t-1} \cup \{(x,y) : x \in \mathcal{B}^t\}$$

  Retrain model on $\mathcal{D}_{\text{train}}^t$
  Receive reward $\mathbf{r}^t \in [0,1]^K$ as in Equation 18
  Set $\hat{r}_{k^t}^t \leftarrow \frac{r_{k^t}^t}{s_{k^t}^t}$ and $\hat{r}_{k'}^t = 0, \forall k' \in [K], k' \neq k^t$
  for $k = 1, \ldots, K$ **do**
    $\hat{y}_k^t = \xi^k(t) \cdot \hat{\mathbf{r}}^{\mathbf{t}}$
    $\hat{v}_k^t = 1/s_k^t$
    $w_k^{t+1} = w_k^t \exp\left(\frac{p_{\min}}{2}(\hat{y}_k^t + \hat{v}_k^t * p_{\min})\right)$
  end for
end for

D.2 Upper Confidence Bound (UCB)

UCB is a classic multi-armed bandit algorithm that we apply for active learning. Each arm is a strategy $\mathcal{A}_k$ from the portfolio. At any iteration t, based on the confidence bound calculated from the mean reward and associated uncertainty, an arm k^t is decided to be played. Assume τ is the performance metric that we use to evaluate the quality of model parameters (typically negative validation loss for regression tasks and validation accuracy for classification tasks). Therefore, the partial information reward $\mathbf{r}^t \in \mathbb{R}^K$ obtained as consequence of the action in iteration t is defined as:

$$r_k^t = \begin{cases} \tau(\hat{\theta}^t) - \tau(\hat{\theta}^{t-1}) & \text{if } k = k^t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall k \in [K], \tag{20}$$

where $\hat{\theta}^t$ denotes the parameters of the surrogate model obtained after training on $\mathcal{D}_{\text{train}}^t$ (see Algorithm 3).

Algorithm 3 UCB: Upper Confidence Bound

Require: Strategy portfolio $\mathcal{P} = \{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$; pool $\mathcal{D}_{\text{pool}}$; initial labeled set $\mathcal{D}_{\text{init}}$; validation set $\mathcal{D}_{\text{val}}$; batch size B; rounds T, exploration parameter c, pseudo-count γ, model performance metric τ

Initialization: $D_{\text{train}}^0 = \mathcal{D}_{\text{init}}$; $c \leftarrow 1$

Initialization: counts $n_k \leftarrow \gamma$, values $\hat{\mu}_k \leftarrow 0$ for all $k \in [K]$

Train model on $\mathcal{D}_{\text{train}}^0$ and obtain $\hat{\theta}^0$

for $t = 1, 2, \ldots, T-1$ **do**

Select strategy:

$$k^t = \arg\max_k \left(\hat{\mu}_k + c\sqrt{\frac{\log t}{n_k}} \right)$$

Define allocation vector $\mathbf{p}^t$ with $p_{k^t}^t = 1$ and $p_{k'}^t = 0, \forall k' \neq k^t$

Sample batch $\mathcal{B}^t$:

$$\mathcal{B}^t \leftarrow \text{Sample}(\mathcal{A}_{k^t}, B, \mathcal{D}_{\text{pool}} \setminus \mathcal{D}_{\text{train}}^{t-1})$$

Query labels for $\mathcal{B}^t$ and update the training dataset:

$$\mathcal{D}_{\text{train}}^t \leftarrow \mathcal{D}_{\text{train}}^{t-1} \cup \{(x, y) : x \in \mathcal{B}^t\}$$

Retrain model on $\mathcal{D}_{\text{train}}^t$ and obtain optimal parameters $\hat{\theta}^t$

Receive reward $\mathbf{r}^t \in \mathbb{R}^K$ as in Eq. 20

Update UCB statistics. For all $k \in [K]$:

$$n_k \leftarrow n_k + \mathbb{1}_{k=k^t}$$

$$\hat{\mu}_k \leftarrow \hat{\mu}_k + \mathbb{1}_{k=k^t} \left[\frac{1}{n_k} (r_k^t - \hat{\mu}_k) \right]$$

end for

D.3 SCRiBLe Algorithm

SCRiBLe (Abernethy et al., 2009) is a linear bandit solver we apply for active learning in this work. Specifically, SCRiBLe operates directly on the allocation simplex Δ^{K-1}. At each round it proposes an allocation vector and observes the resulting validation loss, which it treats as the cost to be minimized. Consequently, the action space for this algorithm is the convex compact simplex Δ^{K-1}. For the (K-1)-dimensional simplex $\Delta^{K-1} = \{x \in \mathbb{R}^K : x \geq 0, \sum_k x_k = 1\}$, the log-barrier is:

$$\mathcal{C}(x) = -\sum_{k=1}^{K-1} \log x_k - \log(1 - \sum_{k=1}^{K-1} x_k) \tag{21}$$

where $x_K = 1 - \sum_{k=1}^{K-1} x_k$. The algorithm starts close to the center of the simplex, i.e. a uniform distribution and explores the domain while still being inside the simplex. At any iteration t, the algorithm perturbs the proportion vector in a randomly chosen direction of eigenvector of $\nabla^2\mathcal{C}(\mathbf{p}^t)$ and obtains a batch according to this updated proportion. The whole algorithm is described in Algorithm 4.

Algorithm 4 SCRiBLe

Require: Strategy portfolio $\mathcal{P} = \{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$; pool $\mathcal{D}_{\text{pool}}$; initial labeled set $\mathcal{D}_{\text{init}}$; validation set $\mathcal{D}_{\text{val}}$; batch size B; rounds T, $\eta > 0$, ϑ-self-concordant barrier $\mathcal{C}$, $\mathcal{K} = \Delta^{K-1}$

Initialization: $D^0_{\text{train}} = \mathcal{D}_{\text{init}}$; $\mathbf{p}^1 \leftarrow \arg\min_{\mathbf{p}\in\mathcal{K}} \mathcal{C}(\mathbf{p}) = \{\frac{1}{K}, \cdots, \frac{1}{K}\}$
for $t = 1, \ldots, T-1$ **do**
  Compute eigenpairs $\{(\mathbf{e}_k, \lambda_k)\}_{k=1}^{K-1}$ of $\nabla^2\mathcal{C}(\mathbf{p}^t)$
  Sample $k^t \sim \text{Unif}(\{1, \ldots, K-1\})$, $\varepsilon^t \sim \{-1, +1\}$ uniformly
  $\mathbf{p}^t \leftarrow \mathbf{p}^t + \varepsilon^t \lambda_{k^t}^{-1/2} \mathbf{e}_{k^t}$
  Sample strategy sub-batches: $\mathcal{B}_i^t \leftarrow \text{Sample}(\mathcal{A}_i, \lfloor p_i^t * B \rfloor, \mathcal{D}_{\text{pool}} \setminus \mathcal{D}^{t-1}_{\text{train}})$
  Aggregate batch $\mathcal{B}^t = \bigcup_{k=1}^{K} \mathcal{B}_k^t$;
  Query labels for $\mathcal{B}^t$ and update the training dataset:

$$\mathcal{D}^t_{\text{train}} \leftarrow \mathcal{D}^{t-1}_{\text{train}} \cup \{(x, y) : x \in \mathcal{B}^t\}$$

  Retrain model on $\mathcal{D}^t_{\text{train}}$ and obtain optimal parameters $\hat{\theta}^t$
  Compute cost $c^t \stackrel{\text{def}}{=} \mathcal{R}(\mathcal{D}_{\text{val}}; \hat{\theta}^t)$
  $\tilde{\mathbf{c}}^t \leftarrow (K-1)\, c^t\, \varepsilon^t \lambda_{k^t}^{1/2} \mathbf{e}_{k^t}$
  $\mathbf{p}^{t+1} \leftarrow \arg\min_{\mathbf{p}\in\mathcal{K}} \left(\eta \sum_{s=1}^{t} \tilde{\mathbf{c}}^{s\top} \mathbf{p} + \mathcal{C}(\mathbf{p})\right)$
end for

D.4 SELECTAL

SelectAL (Hacohen & Weinshall, 2023) was originally introduced as a retraining-based active learning framework that treats strategy selection as a choice between two regimes: a "low-budget" regime focused on representativeness and a "high-budget" regime focused on uncertainty and atypicality. To determine the optimal regime at each iteration, SelectAL assesses the utility of a strategy as follows. At any iteration t, it treats the labeled set $\mathcal{D}^t_{\text{train}}$ as if it were unlabeled and removes a small subset of the points, specifically $\max(\lfloor 0.05 \cdot |\mathcal{D}^t_{\text{train}}| \rfloor, 1)$ (the 5% value follows the original publication), chosen according to a candidate strategy's logic, and measures the resulting change in model performance. The strategy whose deletion induces the largest drop in performance is judged most informative and is selected for the subsequent forward acquisition step.

While the original work suggests using one fixed representative strategy for deletion per budget category (which may differ from the corresponding category's acquisition strategy), such a constraint limits the framework's applicability in complex portfolios like our setup (C2) (Section 4.1). Therefore, in our implementation, we allow SelectAL to evaluate all K strategies in the portfolio by setting the forward-selection and deletion-based strategies to be identical for each candidate. This adaptation ensures a fair comparison by allowing SelectAL to operate over the same strategy pool as FractAL.

D.5 AUTOAL

AutoAL (Wang et al., 2025) frames strategy selection as a differentiable search problem via two co-trained networks. First, FitNet estimates the informativeness of each unlabeled sample. Second, SearchNet learns a continuous weighting over the strategy portfolio. Each candidate strategy contributes a score to every unlabeled point, and SearchNet combines these into a single per-point score via gradient-based optimization on the labeled set (relaxing what would otherwise be a discrete strategy search into a continuous one, in the spirit of differentiable architecture search); the points with the highest combined scores are then selected for labeling. Unlike FractAL, which allocates

budget fractionally across strategies to form disjoint sub-batches, AutoAL blends strategies at the level of individual point scores, and its bilevel optimization introduces additional hyperparameters and per-round computational overhead from jointly training the two auxiliary networks.

E ALLOCATION DIAGNOSTICS

We seek to establish a quantitative notion of quality for a strategy portfolio allocation that can be compared across multiple adaptive strategy selection methods. One potential avenue for this diagnostic is to consider "pure-strategy" test performance (i.e., running each strategy alone across the full AL run), to decide on which strategies are empirically best- and worst-performing at each round. Based on this evaluation, a natural desideratum for a soft strategy selection method is to concentrate budget on the strategies that perform well and withhold it from those that do not.

This criterion is not valid in general, because samples from different strategies may be complementary. In those cases, combining samples from a seemingly poor strategy with a strong one could create synergies and boost downstream performance. Our experiments, however, suggest this does not happen here. No strategy selection method in our study outperforms the best pure strategy in the portfolio (Tables 9, 10, 11). Remarkably, this holds even for AutoAL, which blends strategies at the level of individual point scores rather than allocating budget across them, and can therefore in principle select points that exploit complementarity between strategies. It nonetheless underperforms both random and a uniform allocation on every aggregate measure (Table 8). We therefore take the pure-strategy ranking as our reference, while noting that portfolios with genuinely complementary strategies remain untested here.

On some tasks, several strategies perform comparably well, so identifying a single best (or worst) strategy is too strict a standard. A method that correctly favors any strategy within this comparable set should not be penalized for not isolating the single top performer. We therefore evaluate against the best and worst *sets* of pure strategies, of cardinality k (reported per task in Table 6). At each AL round, Best-Set Overlap is the fraction of the best set that is also among the top-k strategies by the method's internal allocation, and Worst-Set Overlap is the fraction of the worst set that is also among the bottom-k strategies.

To determine k for each task, we take the mean $\pm$ one standard deviation at each round of each pure strategy's test performance (across seeds, after averaging within splits) and count how many strategies' bands overlap the best-performing (or, for Worst-Set Overlap, the worst-performing) strategy's band; the cardinality of the set is one plus this count. We take the mode of this per-round value across rounds $t{=}2$ to $T{-}1$ as the fixed set size reported for that task. This is a practical heuristic rather than a formal hypothesis test, and portfolios whose strategies are more tightly clustered in performance naturally have larger cardinality sets.

Table 6: Cardinality of the Best and Worst Sets for the allocation quality assessment for all the datasets

Dataset	|Best Set|	|Worst Set|
CIFAR 10 (C1)	1	1
CIFAR 10 (C2)	2	1
KEGG	2	1
SARCOS	2	3
Diamonds	2	2
BMDMs (B1)	1	2
T cells (B2)	1	2

F EXPERIMENTAL SETUPS

We now discuss the setups used in each experiment in more detail. A summary of different active learning configurations is presented in Table 7.

Table 7: Active learning configurations for each setup: surrogate model, sizes of the initial labeled, validation, test and pool sets, and batch size B. All strategy selection methods and pure strategies share the same configuration within a setup.

Dataset	Surrogate Model	$\lvert\mathcal{D}_{\text{init}}\rvert$	$\lvert\mathcal{D}_{\text{val}}\rvert$	$\lvert\mathcal{D}_{\text{test}}\rvert$	$\lvert\mathcal{D}_{\text{pool}}\rvert$	B
CIFAR 10	ResNet-18	500	1024	10000	48976	500
KEGG	3 layer MLP	500	1024	12921	50663	500
SARCOS	3 layer MLP	500	1024	8896	34564	500
Diamonds	3 layer MLP	500	1024	10788	42128	500
BMDMs	1 layer MLP	35	40	103	371	35
CD4+ T cells	1 layer MLP	35	40	141	528	35

F.1 CLASSIFICATION DATASET

Data processing. We follow the data processing recipe from Hacohen & Weinshall (2023). Starting from the raw CIFAR 10 images, we first apply random cropping of the same size 32×32 with a padding of 4, followed by random horizontal flips. Post these augmentations, the pixel values are scaled to $[0, 1]$ and are then standardized using the channel-wise mean and standard deviation of $[0.4914, 0.4822, 0.4465]$ and $[0.247, 0.2435, 0.2616]$ respectively.

Surrogate Model. We use a ResNet-18 (He et al., 2016) as the surrogate model for CIFAR 10. The ResNet is trained for 50 epochs in each active learning iteration. We used an SGD optimizer, with 0.9 Nesterov momentum, 0.0003 weight decay, cosine learning rate scheduling with a base learning rate of 0.025, dropout with probability 0.5 and batch size of 100 examples. We borrow all the architectural and hyperparameter details from Hacohen & Weinshall (2023). We select the best checkpoint based on validation accuracy. For ProbCover, we use $\delta = 0.6$ as outlined in Yehuda et al. (2022).

Embeddings. We used the 512 dimensional $\text{L2}-$normalized SimCLR (Chen et al., 2020) embeddings[6] for strategies like CoreSet, ProbCover and TypiClust, that operate in the embedding space. Note that the embeddings remain fixed throughout the AL run, as prescribed in Hacohen & Weinshall (2023).

F.2 REGRESSION DATASETS

Data processing. All three regression datasets, KEGG, SARCOS and Diamonds use the same data processing pipeline. We closely follow Holzmüller et al. (2023) for the data processing recipe. Specifically, we process the input features as follows:

$$x_j^{\text{processed}} = 5\text{tanh}\left(\frac{1}{5}\frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j}\right) \tag{22}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the feature wise mean and standard deviation of the dataset $\mathcal{D}_{\text{pool}}$. This is done to make sure all input features lie in the range $[-5, 5]$ and simplify the learning process for the MLP.

Surrogate Model. We take inspiration from Holzmüller et al. (2023) for the surrogate model and use a 3 layer MLP with ReLU activations and 2 hidden layers of 512 dimension each. We use a weight decay of 0.0003 for regularization and optimize the network using Adam optimizer with a base learning rate of 0.01. At every AL iteration, the surrogate model is trained for 256 epochs with a batch size of 64 and a learning rate schedule that decays linearly with the epochs. We select the best checkpoint based on validation loss.

[6]Embeddings obtained from: https://github.com/fbickfordsmith/ssl-embeddings

Embeddings. We use the processed input features concatenated together as the embedding for each input sample for strategies like CoreSet, ProbCover and TypiClust.

Portfolio. For our portfolio, we choose only Random, CoreSet, ProbCover and TypiClust as they naturally apply to these setups. MaxEnt, MinMargin and LeastConf are more suited for classification tasks as they operate with logits and hence are removed from the portfolio for regression setups.

F.3 High-Throughput Perturbation Screens

B1 (BMDMs) data processing. Starting from the matched Perturb-seq and OPS count data (Anonymous, 2026), we processed the transcriptomic readouts with `scanpy` for quality control and normalization, used ContrastiveVI (Weinberger et al., 2023) to isolate perturbation-induced variation from background biological variation, and computed per-perturbation log-fold changes (against non-targeting controls) with lvm-DE (Boyeau et al., 2023) across 4,000 highly variable genes. Imaging embeddings were obtained by averaging CellProfiler features (Carpenter et al., 2006) per perturbation and retaining the top 70 principal components.

B2 (CD4+ T cells) data processing. We used the publicly available genome-scale Perturb-seq dataset of primary human CD4+ T cells (Zhu et al., 2026), downloaded from the CZI Virtual Cell Models repository. Per-perturbation log-fold changes against non-targeting controls were computed at both the 8-hour and 48-hour time points across 4,000 highly variable genes. Input embeddings were constructed by applying PCA to the 8-hour log-fold changes and retaining the top 70 principal components; the 48-hour log-fold changes serve as the prediction target.

Surrogate model. We use an MLP with a single hidden layer and ReLU activation, followed by a linear output layer that maps the 70-dimensional input embedding to a 4,000-dimensional log-fold change vector. The model is trained with SGD (batch size 64) for 600 epochs using a learning rate of 1.0 with a linear scheduler lower-bounded at 0.1. We select the best checkpoint based on validation loss. Ground-truth labels for each of the 4,000 highly variable genes are obtained by thresholding absolute log-fold changes, $|\mathrm{LFC}| > 0.5$.

Portfolio. Following similar logic as in Appendix F.2, we choose only Random, CoreSet, ProbCover and TypiClust for our portfolio in these tasks.

G PERFORMANCE ACROSS SETUPS

The seven setups report different performance metrics (accuracy, MSE, and AUPRC) which are not directly comparable, making averaging raw performance differences across setups meaningless. For a method with terminal performance p on a given setup, we therefore define its relative gain over Random as

$$g = \frac{p - p_{\text{random}}}{p_{\text{best}} - p_{\text{random}}}, \tag{23}$$

where p_{random} and p_{best} denote the terminal performance of Random and of the best pure strategy on that setup. By construction, $g = 0$ corresponds to matching Random and $g = 1$ to matching the best pure strategy in the portfolio. The gap from g to 1 measures the cost of not knowing in advance which strategy is best, and $g < 0$ indicates performance worse than random sampling.

We report the mean, worst-case, and rank summaries across the seven setups in Table 8. Ranks are computed per setup among the seven methods and then averaged, with ties resolved from unrounded performance values.

Table 8: Relative gain and rank among the six adaptive methods and the equal split control, averaged across the seven setups. Mean Rank and Worst Rank report the average and maximum rank among the 7 methods (rank 1 = best, computed per setup then averaged/maximized across setups). Median Rank corresponds to the median value of rank across the setups. Mean Gain and Rank values are mean $\pm$ SE across setups.

Method	Mean Gain (↑)	Worst Gain (↑)	Mean Rank (↓)	Median Rank (↓)	Worst Rank (↓)
Equal Split	0.230 $\pm$0.214	-0.888	4.29 $\pm$0.77	4	7
ALBL	0.150 $\pm$0.227	-1.055	5.43 $\pm$0.27	6	6
UCB	0.228 $\pm$0.214	-1.055	4.14 $\pm$0.55	4	6
SCRiBLe	0.266 $\pm$0.208	-0.944	4.14 $\pm$0.47	4	6
SelectAL	0.370 $\pm$0.178	-0.611	3.14 $\pm$0.43	3	5
AutoAL	0.027 $\pm$0.251	-1.055	5.71 $\pm$0.77	7	7
FractAL	**0.689** $\pm$**0.096**	**0.222**	**1.14** $\pm$**0.13**	**1**	**2**

FractAL has the highest mean gain (0.689) and the only positive *worst-case* gain (0.222): every other method, including the non-adaptive Equal Split control, has at least one setup where its terminal performance falls below Random's, with AutoAL, ALBL, and UCB falling short by more than the entire Random-to-best span in their weakest setup. FractAL is also the top-ranked method on average (mean rank 1.14 of 7, median 1, against 3.14 and median 3 for the next closest method, SelectAL). We also note that no method attains $g > 1$ on any setup (see also Appendix E). These results indicate that FractAL's advantage reflects consistent competitiveness across the full range of setups we evaluate.

H PER-SETUP RESULTS WITH PURE STRATEGIES PERFORMANCE

We report here the performance of every pure strategy in each setup's portfolio, alongside the one from all of the adaptive methods (Tables 9, 10 and 11). Note that no adaptive method has prior knowledge of which strategies perform well, and each strategy selection method must infer it during the run from its reward signal. The best and worst pure strategies therefore serve as a reference range rather than as competing methods. In each table we bold the best pure strategy and the best strategy selection method, marking ties where they occur. Across all setups, FractAL achieves the highest test performance among the strategy selection methods, and either matches the best pure strategy or ranks second in the portfolio.

Table 9: Accuracy (↑) on CIFAR-10 at the end of $T = 10$ rounds.

Method	(C1): CIFAR-10 ($K = 3$)	(C2): CIFAR-10 ($K = 8$)
Individual Strategies		
Random	69.4 ±0.1	69.4 ±0.1
CoreSet	58.7 ±0.2	58.7 ±0.2
ProbCover	**71.2 ±0.1**	**71.2 ±0.1**
BADGE	69.6 ±0.2	69.6 ±0.2
LeastConf	69.6 ±0.1	69.6 ±0.1
MaxEnt	69.5 ±0.1	69.5 ±0.1
MinMargin	69.7 ±0.1	69.7 ±0.1
TypiClust	68.8 ±0.1	68.8 ±0.1
Strategy Selection Methods		
Equal Split	67.8 ±0.1	69.1 ±0.1
UCB	67.5 ±0.1	69.5 ±0.1
ALBL	67.5 ±0.5	68.6 ±0.3
SCRiBLe	67.7 ±0.2	69.2 ±0.1
SelectAL	68.3 ±0.1	69.4 ±0.1
AutoAL	67.5 ±0.1	68.3 ±0.1
FractAL	**70.2 ±0.2**	**69.8 ±0.2**

Table 10: Test MSE ($\times 10^{-1}$, ↓) on tabular regression datasets at the end of $T = 10$ rounds.

Method	KEGG	SARCOS	Diamonds
Individual Strategies			
Random	0.79 ±0.02	0.79 ±0.01	0.11 ±0.01
CoreSet	0.53 ±0.01	0.73 ±0.01	**0.08 ±0.01**
ProbCover	**0.49 ±0.00**	**0.72 ±0.01**	0.12 ±0.01
TypiClust	0.97 ±0.01	0.77 ±0.01	0.10 ±0.01
Strategy Selection Methods			
Equal Split	0.62 ±0.01	0.76 ±0.01	**0.08 ±0.01**
UCB	0.61 ±0.01	0.77 ±0.01	0.10 ±0.01
ALBL	0.63 ±0.02	0.77 ±0.01	0.09 ±0.01
SCRiBLe	0.63 ±0.01	0.75 ±0.01	0.09 ±0.01
SelectAL	0.59 ±0.01	0.76 ±0.01	0.09 ±0.01
AutoAL	0.65 ±0.01	0.82 ±0.01	0.10 ±0.01
FractAL	**0.58 ±0.01**	**0.74 ±0.01**	**0.08 ±0.01**

Table 11: AUPRC (↑) for high-throughput perturbation datasets after $T = 6$ rounds.

Method	**B1:** BMDM	**B2:** CD4+ T Cell
Individual Strategies		
Random	0.4165 $\pm$0.0011	0.3403 $\pm$0.0045
CoreSet	**0.4377 $\pm$0.0005**	**0.3696 $\pm$0.0002**
ProbCover	0.4305 $\pm$0.0004	0.3441 $\pm$0.0003
TypiClust	0.4301 $\pm$0.0006	0.3413 $\pm$0.0061
Strategy Selection Methods		
Equal Split	0.4279 $\pm$0.0006	0.3442 $\pm$0.0065
UCB	0.4323 $\pm$0.0013	0.3590 $\pm$0.0015
ALBL	0.4289 $\pm$0.0018	0.3548 $\pm$0.0021
SCRiBLe	0.4299 $\pm$0.0006	0.3557 $\pm$0.0006
SelectAL	0.4346 $\pm$0.0010	0.3579 $\pm$0.0012
AutoAL	0.4336 $\pm$0.0005	0.3603 $\pm$0.0008
FractAL	**0.4363 $\pm$0.0006**	**0.3642 $\pm$0.0013**

I Ablation Studies

We perform the ablation experiments for our proposed FractAL method. In particular, we test the sensitivity of the method to the EWMA smoothing parameter, the contribution of the mirror descent update, the utility of the influence-based rewards as compared to retraining based rewards and the importance of soft fractional allocation among portfolio strategies. We detail the results of each of these experiments below.

I.1 Smoothing Parameter

We sweep over α values for across all setups and report the terminal accuracy. The numbers are averaged over 5 splits and 10 runs per split. Table 12 contains the accuracy numbers for different values of the smoothing parameter α. The similar performance across different α values highlights the robustness of the method to this hyperparameter. No Smoothing (i.e. $\alpha = 1.0$) performs particularly worse on two out of seven setups, CD4+ T cells and Diamonds.

Table 12: Terminal Performance of FractAL for different values of the EWMA smoothing parameter α on all seven setups. MSE numbers represent (MSE $\times 10^{-1}$)

Dataset	Metric	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 1.0$
CIFAR-10 (C1)	Acc (↑)	70.0 ±0.2	70.1 ±0.2	70.2 ±0.2	70.0 ±0.2	70.2 ±0.1
CIFAR-10 (C2)	Acc (↑)	70.0 ±0.2	69.9 ±0.1	69.8 ±0.2	70.1 ±0.2	69.9 ±0.1
KEGG	MSE (↓)	0.60 ±0.01	0.60 ±0.01	0.58 ±0.01	0.58 ±0.01	0.59 ±0.01
SARCOS	MSE (↓)	0.73 ±0.01	0.76 ±0.01	0.74 ±0.01	0.74 ±0.01	0.74 ±0.01
Diamonds	MSE (↓)	0.09 ±0.01	0.09 ±0.01	0.08 ±0.01	0.09 ±0.01	0.10 ±0.01
BMDMs	AUPRC (↑)	0.4348 ±0.0008	0.4356 ±0.0006	0.4363 ±0.0006	0.4360 ±0.0008	0.4365 ±0.0008
CD4+ T cells	AUPRC (↑)	0.3629 ±0.0013	0.3630 ±0.0013	0.3642 ±0.0013	0.3644 ±0.0013	0.3593 ±0.0050

I.2 Allocation Optimization Algorithm

We replace the mirror descent allocation update with the following additive update with clipping and renormalization:

$$f_k^t = \max\big(q_k^t + \lambda\, \eta^t\, \bar{r}_k^t,\ p_{\min}\big)\,,$$
$$q_k^{t+1} = \frac{f_k^t}{\sum_{j=1}^K f_j^t}, \tag{24}$$

where we use a fixed $\lambda = 0.15$. p_k^{t+1} is then computed as in Equation 6. We benchmark this approach against FractAL across all setups (Table 13). The mirror-descent update is never worse, and better on five of seven datasets, supporting its use for navigating the allocation simplex over an additive update with clipping and renormalization.

Table 13: Performance comparison of mirror descent vs additive update with clipping and renormalization across all setups.

Dataset	Metric	Mirror Desc.	Additive (clip + renorm.)
CIFAR-10 (C1)	Acc (↑)	**70.2 ±0.2**	69.8 ±0.2
CIFAR-10 (C2)	Acc (↑)	**69.8 ±0.2**	**69.8 ±0.1**
KEGG	MSE ($\times 10^{-1}$, ↓)	**0.58 ±0.01**	0.61 ±0.01
SARCOS	MSE ($\times 10^{-1}$, ↓)	**0.74 ±0.01**	0.76 ±0.01
Diamonds	MSE ($\times 10^{-1}$, ↓)	**0.08 ±0.01**	0.09 ±0.01
BMDMs	AUPRC (↑)	**0.4363 ±0.0006**	0.4299 ±0.0005
CD4+ T cells	AUPRC (↑)	**0.3642 ±0.0013**	0.3533 ±0.0051

I.3 Influence Reward Attribution

We test the utility of the influence rewards by swapping them with changes in performance metrics on the validation set after retraining the model (keeping the fractional allocation framework unchanged).

More specifically, the retraining reward for a strategy is the difference in performance on validation set with and without the strategy's latest sub-batch of chosen points. We perform the experiment across all our setups and report the numbers averaged over 5 splits and 10 seeds per split (Table 14).

Table 14: Performance comparison with influence vs retraining based reward across all setups.

Dataset	Metric	Influence	Retraining
CIFAR-10 (C1)	Acc (↑)	70.2 ±0.2	**70.5 ±0.1**
CIFAR-10 (C2)	Acc (↑)	69.8 ±0.2	**70.2 ±0.2**
KEGG	MSE ($\times 10^{-1}$, ↓)	**0.58 ±0.01**	0.59 ±0.01
SARCOS	MSE ($\times 10^{-1}$, ↓)	**0.74 ±0.01**	**0.74 ±0.01**
Diamonds	MSE ($\times 10^{-1}$, ↓)	**0.08 ±0.01**	**0.08 ±0.01**
BMDMs	AUPRC (↑)	0.4363 ±0.0006	**0.4383 ±0.0006**
CD4+ T cells	AUPRC (↑)	**0.3642 ±0.0013**	0.3514 ±0.0006

The influence reward is on par with the retraining-based reward: it trails by at most 0.4 accuracy points on CIFAR-10 and matches or outperforms on four of the other five tasks. However, in runtime cost, influence-based AL run is overall $2.5\times$ faster than the retraining based AL run.

To further establish influence rewards as a sound substitute for retraining reward, we also study the correlation between the two rewards across rounds for the CIFAR-10 (C1) setup. Specifically, we report the Spearman rank correlation between the two reward rankings as well as the agreement (in percentage) between the two rewards for the best and worst performing strategies. Table 15 reports the aforementioned numbers. The rank correlation ($\pm$SE) ranges between 0.7 and 0.8, while the best and worst agreements are generally $> 70\%$ indicating that the two rewards rank strategies consistently. Therefore, the influence estimator is a reliable proxy for strategy utility and hence a sound substitute for the more expensive retraining-based attribution within the proposed fractional allocation framework.

Table 15: Correspondence between the retraining based reward and corresponding influence estimates across different active learning rounds for the CIFAR-10 (C1) setup.

Round	Spearman Rank Correlation	Best Agreement (%)	Worst Agreement (%)
2	0.70 ±0.08	70.0	80.0
3	0.80 ±0.06	80.0	83.3
4	0.78 ±0.04	70.0	86.6
5	0.73 ±0.08	80.0	76.6
6	0.73 ±0.07	76.6	76.6

I.4 Soft Fractional Allocation

While preserving the influence-based reward attribution, we replace the proposed *soft* fractional allocation with a hard allocation mechanism in which the strategy with the highest reward receives the entire budget for the round. We call this variant FractAL (Hard). Soft allocation improves on five of seven setups and is comparable on the remaining two (Table 16). Distributing the budget across strategies therefore outperforms committing it to the single highest-reward strategy, even when both use the same reward signal.

Table 16: Performance comparison of hard versus soft allocation across all setups.

Dataset	Metric	FractAL	FractAL (Hard)
CIFAR-10 (C1)	Acc (↑)	**70.2** ±**0.2**	69.3 ±0.1
CIFAR-10 (C2)	Acc (↑)	69.8 ±0.2	**70.0** ±**0.1**
KEGG	MSE ($\times 10^{-1}$, ↓)	**0.58** ±**0.01**	0.60 ±0.01
SARCOS	MSE ($\times 10^{-1}$, ↓)	**0.74** ±**0.01**	**0.74** ±**0.01**
Diamonds	MSE ($\times 10^{-1}$, ↓)	**0.08** ±**0.01**	0.09 ±0.01
BMDMs	AUPRC (↑)	**0.4363** ±**0.0006**	0.4309 ±0.0010
CD4+ T cells	AUPRC (↑)	**0.3642** ±**0.0013**	0.3592 ±0.0017

J Detailed Runtime Analysis

Table 17 reports wall-clock time for a single complete AL run on CIFAR-10 (C2, $K = 8$) on one NVIDIA L40S GPU, decomposed into two phases: Acquisition, the time to query strategies in the portfolio and assemble the batch, and Allocation Update, the time to compute the reward signal and update the allocation for the next round.

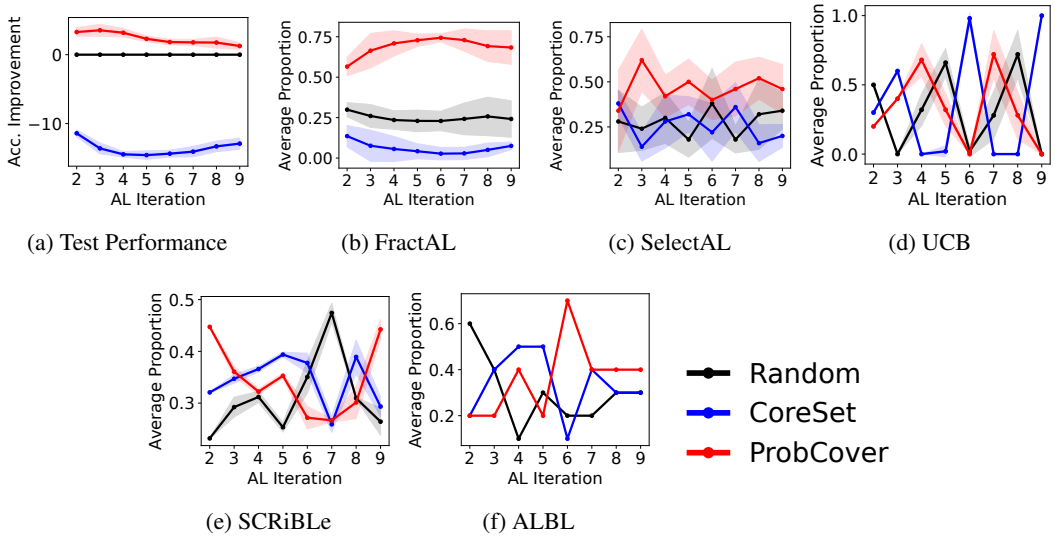
Table 17: Runtime (in minutes) of different components of a single AL run for different methods. We use the CIFAR10 (C2, $K = 8$) setting for these measurements.

Method	Acquisition	Allocation Update	Total
ALBL	1.2	7.3×10^{-5}	1.2
UCB	1.5	7.2×10^{-5}	1.5
SCRiBLe	8.8	4.6×10^{-4}	8.8
SelectAL	3.3	19.8	23.1
AutoAL	20.7	28.5	49.2
FractAL	8.8	0.5	9.3

K Additional Budget Allocation Plots

For completeness, we provide the performance plots for pure strategies in the portfolio and budget allocation evolution plots for each of our methods and baselines on all our setups. We often notice that FractAL increases the allocation to the empirically best strategies across all setups, while reducing the share of the suboptimal ones in the portfolio. Detailed analysis for each plot is included in its respective caption.

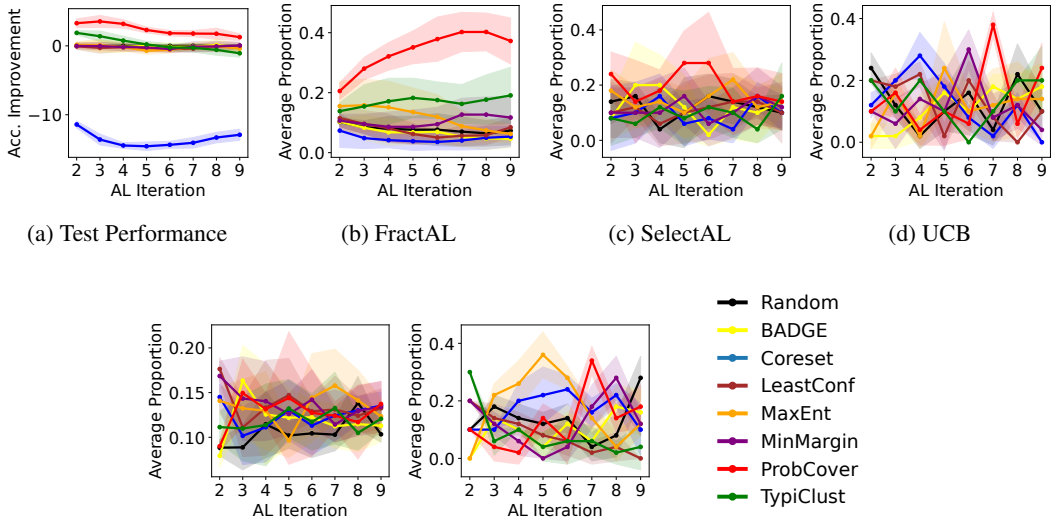
All allocation and performance plots (Figures 5–11) show rounds $t = 2$ to $t = T - 1$. We omit $t = 0$, which corresponds to acquiring $\mathcal{D}_{\text{init}}$, and $t = 1$, at which every method, including FractAL, starts from an equal split.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

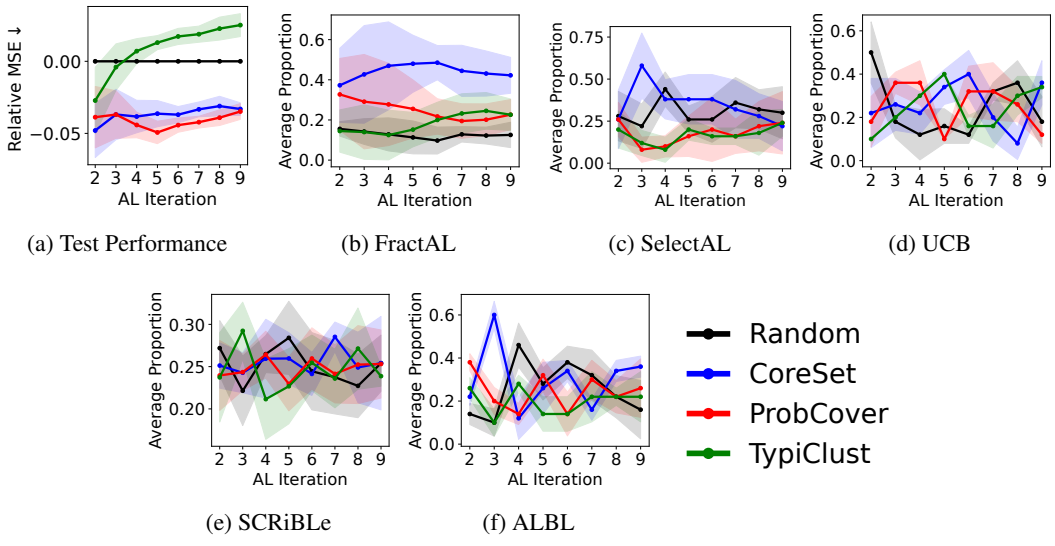
Figure 5: Setup (C1): CIFAR-10 with $K = 3$. (a) Test performance of pure strategies. (b–f) Average per-round budget allocation to Random, CoreSet, and ProbCover under each method. FractAL drives CoreSet to zero and concentrates on ProbCover. SelectAL allocates roughly equal shares to Random and CoreSet, while bandit baselines (d–f) oscillate between strategies without converging. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

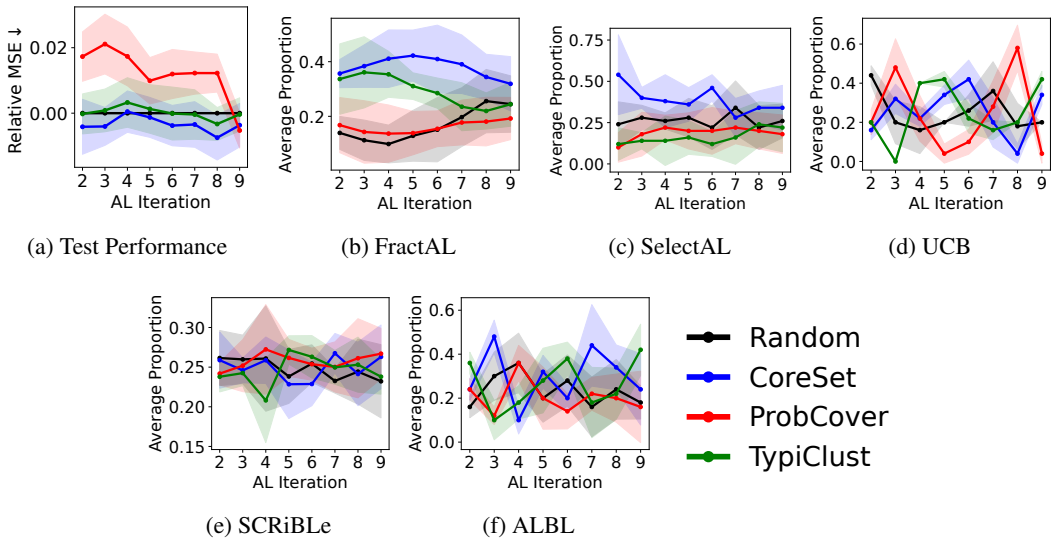
Figure 6: Setup (C2): CIFAR-10 with $K = 8$. (a) Test performance of the 8 pure strategies. (b–f) Average per-round budget allocation to all 8 strategies under each method. FractAL increases the share of ProbCover and decreases CoreSet over rounds. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

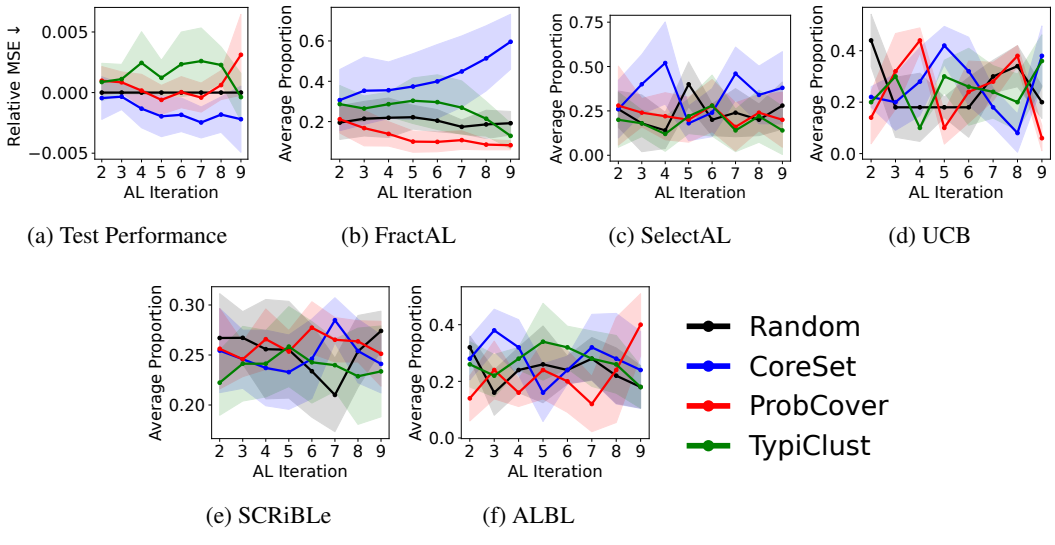
Figure 7: KEGG metabolic reaction network experiment. (a) Test performance of the four pure strategies. (b–f) Average per-round budget allocation under each method. FractAL concentrates the budget on CoreSet (the second strongest strategy) while suppressing Random and TypiClust. SelectAL also upweights CoreSet but assigns ProbCover a lower share than Random consistently. The bandit baselines (d–f) oscillate between strategies without converging. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

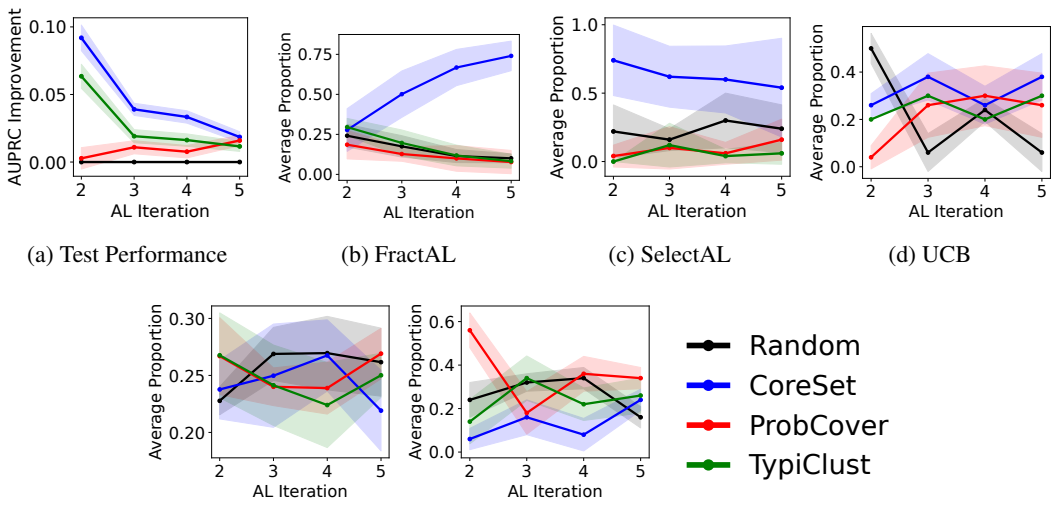
Figure 8: SARCOS experiment. (a) Test performance of the four pure strategies. (b–f) Average per-round budget allocation under each method. FractAL and SelectAL upweight CoreSet (one of the strongest strategies) and suppress Random and ProbCover. SelectAL ranks TypiClust at the lowest allocation, despite it not being the weakest strategy; FractAL avoids this mistake. The bandit baselines (d–f) oscillate without converging. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

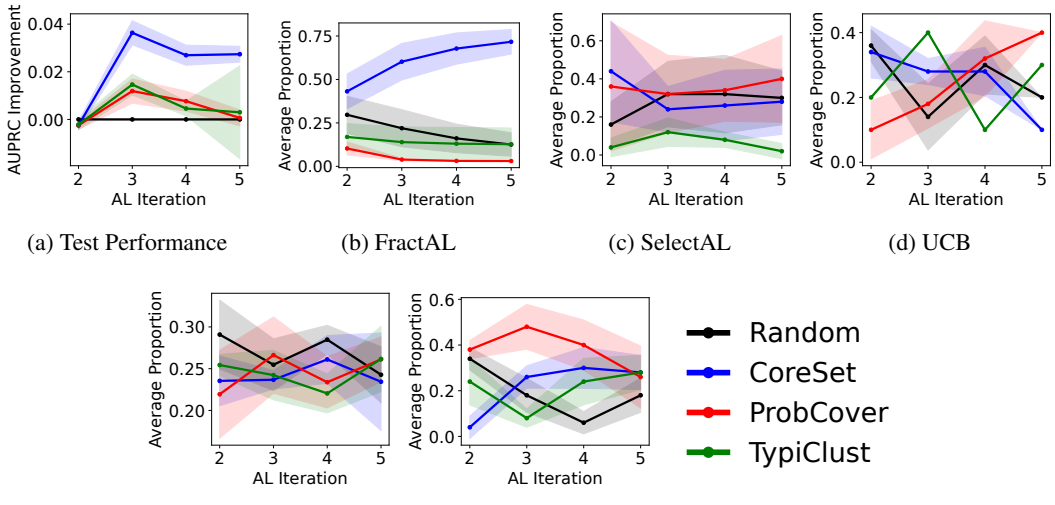
Figure 9: Diamonds experiment. (a) Test performance of the four pure strategies, which cluster within a narrow MSE range. (b–f) Average per-round budget allocation under each method. FractAL upweights CoreSet (the strongest individual strategy) and reduces the others, with the allocations aligned with the empirical ranking by the final rounds. The remaining methods (c–f) maintain roughly uniform allocations. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

Figure 10: Setup (B1): BMDM perturbation prediction. (a) Test AUPRC of the four pure strategies. (b–f) Average per-round budget allocation under each method. FractAL and SelectAL both upweight CoreSet (the strongest strategy) over rounds, but FractAL drives the other three strategies close to zero, whereas SelectAL retains substantial allocation to Random. The bandit baselines (d–f) oscillate between strategies without converging. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.



(a) Test Performance (b) FractAL (c) SelectAL (d) UCB

(e) SCRiBLe (f) ALBL

Figure 11: Setup (B2): CD4+ T cell perturbation prediction. (a) Test AUPRC of the four pure strategies. (b–f) Average per-round budget allocation under each method. FractAL consistently upweights CoreSet (the strongest strategy) and reduces the other three to similar low levels, consistent with their similar pure-strategy performance. SelectAL temporarily reduces CoreSet's allocation in rounds 3–5, and keeps ProbCover proportions high. The bandit baselines (d–f) oscillate between strategies without converging. Lines show the mean and shaded bands show $\pm$ 1 standard deviation. For test accuracy, standard deviation is computed across seeds after averaging over splits; for allocations, across splits after averaging over seeds.