



AFABench: A Generic Framework for Benchmarking Active Feature Acquisition

Downloaded from: <https://research.chalmers.se>, 2026-10-13 07:51 UTC

Citation for the original published paper (version of record):

Schütz, V., Wu, H., Rezvan, R. et al (2026). AFABench: A Generic Framework for Benchmarking Active Feature Acquisition. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2-B: 9765-9776.
<http://dx.doi.org/10.1145/3770855.3817493>

N.B. When citing this work, cite the original published paper.

AFABench: A Generic Framework for Benchmarking Active Feature Acquisition

Valter Schütz*

valter.schutz@chalmers.se

Department of Computer Science and
Engineering

Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden

Han Wu*

hanwu@chalmers.se

Department of Computer Science and
Engineering

Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden

Reza Rezvan

rezvan@chalmers.se

Department of Computer Science and
Engineering

Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden

Linus Aronsson

linaro@chalmers.se

Department of Computer Science and
Engineering

Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden

Morteza Haghir Chehreghani

morteza.chehreghani@chalmers.se

Department of Computer Science and
Engineering

Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden

Abstract

In many real-world scenarios, acquiring all features of a data instance can be expensive or impractical due to monetary cost, latency, or privacy concerns. *Active Feature Acquisition* (AFA) addresses this challenge by dynamically selecting a subset of informative features for each data instance, trading off predictive performance against acquisition cost. Numerous methods have been proposed for AFA, ranging from myopic information-theoretic strategies to non-myopic reinforcement learning approaches. However, a fair and systematic evaluation of these methods remains difficult due to the lack of standardized benchmarks. In this paper, we introduce **AFABench**, the first benchmark framework for AFA. Our benchmark includes a diverse set of synthetic and real-world datasets, supports a wide range of acquisition policies, and provides a modular design that enables easy integration of new methods and tasks. We implement and evaluate representative algorithms from all major categories, including static, myopic, and reinforcement learning-based approaches. To test the lookahead capabilities of AFA policies, we introduce a novel synthetic dataset, *CUBE-NM*, designed to expose the limitations of myopic selection. Our results highlight key trade-offs between different AFA strategies and provide actionable insights for future research. The benchmark code is available at: <https://github.com/Linusaronsson/AFA-Benchmark>.

CCS Concepts

• **Computing methodologies** → **Cost-sensitive learning**; **Active learning settings**; **Feature selection**.

*Both authors contributed equally to this research.



This work is licensed under a Creative Commons Attribution 4.0 International License. *KDD '26, Jeju Island, Republic of Korea*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2259-2/2026/08

<https://doi.org/10.1145/3770855.3817493>

Keywords

Active Learning; Active Feature Acquisition; Cost-Sensitive Learning; Feature Selection

ACM Reference Format:

Valter Schütz, Han Wu, Reza Rezvan, Linus Aronsson, and Morteza Haghir Chehreghani. 2026. AFABench: A Generic Framework for Benchmarking Active Feature Acquisition. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3770855.3817493>

1 Introduction

In many real-world applications, acquiring feature values of data instances can be costly. In healthcare, different medical tests can have monetary cost, latency, or privacy concerns [29]. In recommender systems, querying user preferences can impose a cognitive burden or intrude on privacy [13]. Similar constraints arise in sensor systems and robotics, where measurements require power, time, or motion [15, 16]. In such cases, acquiring the full set of features for every data instance may be infeasible or inefficient. *Active Feature Acquisition* (AFA) addresses this challenge by learning to dynamically select, for each data instance, a small subset of informative features to acquire, optimizing predictive performance under a constrained acquisition budget.

Unlike static feature selection, which selects the same set of features for every example, AFA enables instance-wise decisions: the features selected for one instance can differ from those selected for another. This flexibility makes AFA more realistic and efficient in domains where the informativeness or relevance of features varies between instances. Moreover, since acquisition is sequential, AFA policies can condition future acquisition decisions on previously observed feature values, mirroring expert decision-making strategies such as in clinical diagnosis, where test results guide further examination.

Despite its relevance and importance, AFA has until recently received less attention than adjacent topics such as active learning [27]. In active learning, all features are typically available, and the goal is to selectively acquire costly labels. In contrast, AFA assumes that both labels and features are available during training, but only partial features can be acquired at test time. AFA is also different from static feature selection methods, which fix a global subset of features regardless of the test instance. For comprehensive reviews of traditional feature selection methods, we refer the reader to [4, 11, 18].

A recent survey by Aronsson et al. [2] categorizes AFA methods into several paradigms. At a high level, the literature can be divided into (i) *myopic methods*, which select features by maximizing a one-step utility without explicit long-term planning, and (ii) *non-myopic methods*, which optimize long-term utility and are often formulated within a reinforcement learning (RL) framework. By learning policies that account for future acquisitions, non-myopic methods can yield lookahead strategies that outperform myopic selection in certain regimes. Section 4.1 details the AFA methods considered in this work. A central question in this benchmark is whether the additional computational cost of non-myopic planning is justified relative to simpler myopic heuristics.

Although this body of work has grown considerably in recent years, progress is hindered by the lack of a standardized and unified evaluation framework. Most existing methods are evaluated in isolation using inconsistent datasets, model architectures, or acquisition costs. To our knowledge, **no prior work has provided a dedicated benchmark for the AFA setting**. This makes it difficult to perform fair comparisons or gain generalizable insights.

This paper introduces the first benchmark for AFA. We present a unified and extensible framework for comparing AFA methods under controlled, fair, and reproducible conditions. The benchmark supports a diverse collection of synthetic and real-world datasets and includes representative algorithms across the main methodological categories: myopic approaches (both generative and discriminative), non-myopic approaches (including model-free RL, model-based RL, and non-RL variants), as well as static baselines. Crucially, the framework is **modular and extensible**, enabling researchers to incorporate new methods, acquisition strategies, and datasets with minimal engineering effort. This makes our framework a valuable tool for both researchers developing new AFA techniques and practitioners seeking to apply AFA in domain-specific tasks.

Our main contributions are:

- We introduce **AFABench**, the *first benchmark* for Active Feature Acquisition, allowing standardized and fair comparisons across a wide range of methods and settings.
- Our framework is **modular and extensible**, making it easy to add new AFA methods, datasets, and evaluation protocols.
- We implement and evaluate representative methods from all major paradigms, including myopic methods (generative and discriminative), non-myopic methods based on RL (model-free and model-based), non-myopic methods without explicit RL, and static feature selection.
- We propose a novel synthetic dataset, *CUBE-NM*, that highlights the limitations of myopic strategies and provides a testbed for

evaluating non-myopic acquisition policies (which we justify theoretically, see Theorem 4.1).

To promote reproducibility and future development, our benchmark is open source and available at: <https://github.com/Linusaronsson/AFA-Benchmark>. It contains step-by-step instructions on how to extend the benchmark with new models and new datasets.

2 Problem Formulation

In this section, we introduce the AFA problem and the relevant notation used throughout the paper.

2.1 Notation

Let $\mathbf{x} = \{x_1, \dots, x_d\} \in \mathcal{X}$ denote a data instance with d features, and let $y \in \mathcal{Y}$ be the corresponding response variable in a supervised learning setting. We assume that $(\mathbf{x}, y)$ is distributed according to a joint data distribution $p(\mathbf{x}, y)$. Throughout, we use bold symbols (e.g., $\mathbf{x}, y$) to represent random variables, and their non-bold counterparts (e.g., x, y) to denote specific realizations sampled from the distribution, i.e., $x, y \sim p(\mathbf{x}, y)$. In practice, each x_i may correspond to a single feature (as is typical for tabular data), but it may also represent a group of features, for example a patch of pixels in an image. See Section 3 for details. For any subset $S \subseteq [d] \triangleq \{1, \dots, d\}$, we write $\mathbf{x}_S = \{x_i \mid i \in S\} \in \mathcal{X}_S$ to denote the corresponding subset of features. For an instance x , acquiring feature $i \in [d]$ reveals x_i and incurs the cost $c_i \in \mathbb{R}^+$. In addition, we denote $c(S) \triangleq \sum_{i \in S} c_i$ for $S \subseteq [d]$.

2.2 Active Feature Acquisition

As discussed in Section 1, the AFA problem appears in many variants. In this section, we describe a common formulation adopted for our benchmark [2]. In AFA, one learns (possibly jointly) two functions: (i) a predictor $f(S, \mathbf{x}_S)$ that produces an output given any observed subset $S \subseteq [d]$ (typically implemented via feature masking, see, e.g., [2]), and (ii) an acquisition policy $\pi(S, \mathbf{x}_S) \in \mathcal{A}_B(S)$, where

$$\mathcal{A}_B(S) \triangleq \{a \in U : c(S) + c_a \leq B\} \cup \{\text{STOP}\}, \quad (1)$$

is the set of allowed actions after acquiring S . Here, $U = [d] \setminus S$ denotes the set of unobserved features, and B is a per-instance budget on cumulative feature cost. In practice, the policy often outputs a distribution over actions. If π outputs a feature acquisition action $a \in U$, we incur cost c_a and observe x_a . We then update S and U as follows: $S \leftarrow S \cup \{a\}$ and $U \leftarrow U \setminus \{a\}$. If instead π selects STOP, we output $\hat{y} = f(S, \mathbf{x}_S)$ and terminate. Let $\pi[x] \subseteq [d]$ denote the final set of acquired feature indices when the policy stops on instance x . Given this, the optimization objective for AFA is shown below.

$$\begin{aligned} \min_{f, \pi} \mathbb{E}_{\mathbf{x}, y} \mathbb{E}_{\pi} [\ell(f(\mathbf{x}_{\pi[x]}), y) + \alpha c(\pi[x])] \\ \text{s.t. } c(\pi[x]) \leq B \text{ for all } x \in \mathcal{X}. \end{aligned} \quad (2)$$

where $\alpha \geq 0$ controls the cost–accuracy trade-off and ℓ is a loss function. The term $\alpha c(\pi[x])$ allows the acquisition cost to vary across instances, enabling the policy to spend more resources on instances that are harder to predict.

Hard vs. soft budget. From the above, α induces a *soft* budget and is therefore referred to as a *soft-budget parameter*. By contrast, B

imposes a *hard* budget. A soft budget is beneficial because it adapts the cost spent per instance. However, selecting an appropriate value of α can be unintuitive in many applications (as discussed by, e.g., [12]). A hard budget B is more natural when resources are strictly limited per instance and is often easier to interpret. Its drawback is that some instances may require acquiring features whose total cost exceeds B to achieve accurate predictions.

A note on terminology. Throughout the paper, the term *AFA method* refers to the predictor f and the policy π induced by the method. Some methods learn f and π jointly, for instance by training f on the states (S, x_S) encountered under π during learning. Other methods instead pretrain f on randomly sampled states (S, x_S) and then use this predictor when training π (optionally continuing to update f using states encountered under π). This distinction is discussed further in Section 3.2.

2.3 Scope of the Benchmark

Most prior work on AFA considers two special cases of (2): (i) the *hard-budget* setting, where $\alpha = 0$ (only a hard budget), and (ii) the *soft-budget* setting, where $B = c(\lfloor d \rfloor)$ (all features are, in principle, acquirable for every instance, so only the soft-budget term matters). Following prior work, we also focus on these special cases in our benchmark.

We restrict the scope of our benchmark to methods developed for the common *offline* setting, where the goal is to deliver accurate but cost-effective predictions at *test time*. In this setting, one assumes access to a fully observed training dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ of N i.i.d. samples $(x^{(i)}, y^{(i)}) \sim p(\mathbf{x}, \mathbf{y})$. A predictor f and an acquisition policy π are learned on $\mathcal{D}$. Then, for each test instance, the policy π is used to acquire features, and the predictor f produces a prediction based solely on the acquired subset. Several methods exist for learning f and π based on $\mathcal{D}$.

Finally, we narrow the scope to classification, as this is the main focus of previous work [2].

2.4 MDP Formulation of AFA

The AFA problem can be formulated as a *partially observable Markov decision process* (POMDP) [2]. Equivalently, it can be expressed as a fully observable MDP by defining the state space as $\mathcal{S} = \{(S, x_S) : S \in 2^{[d]}, x_S \in X_S\}$, where a state comprises the set S of acquired features together with their observed values x_S [2]. The action space and transition dynamics are exactly as described in Section 2.2. In the standard MDP formulation of AFA, the reward is $r((S, x_S), a) = -\alpha c_a$ for $a \in \mathcal{A}_B(S) \setminus \{\text{STOP}\}$ and $r((S, x_S), \text{STOP}) = -\ell(f(S, x_S), y)$. Given this, it is straightforward to show that maximizing expected return in this MDP is equivalent to (2). This connection was first established by [8], where model-free reinforcement learning (RL) was used to learn non-myopic policies for solving (2).

Some RL methods in this benchmark adopt alternative reward functions, for example by shaping intermediate rewards to reflect the information gained about the label y , in order to mitigate sparse-reward issues [14, 32]. Such shaping can, however, bias learning toward more myopic acquisition strategies. The (PO)MDP underlying AFA is known to be highly intractable (especially for large d), motivating the development of myopic heuristics. We refer to [2] for details. The benchmark in this paper investigates whether the

additional computational cost of non-myopic planning is justified relative to simpler myopic heuristics.

3 A General Framework for Implementing AFA Methods

We now describe how we implement the AFA problem introduced in Section 2.2 in our benchmark. The framework described here applies primarily to evaluation, since AFA methods may be trained in arbitrary ways, but consistent evaluation is essential for fair benchmarking.

To benchmark AFA methods in a standardized manner, we must make explicit assumptions about their inputs and outputs. During evaluation, the components of the AFA problem in our benchmark interact episodically for each instance $x \in \mathcal{X}$ in the dataset. An example episode is illustrated in Figure 1, where the instance $x \in \mathcal{X}$ is a 4×4 feature matrix representing an image.

An *Initializer* determines which features are observed at the start of an episode, using either deterministic or stochastic rules. The resulting masked instance is then passed in parallel to a *Predictor* and a *Policy*. The *Predictor* produces an intermediate prediction $\hat{y}_t$, which is stored for subsequent performance evaluation but does not otherwise affect the episode dynamics. In parallel, the *Policy* outputs an action a_t , interpreted either as a stop decision or as a request to reveal additional feature(s) (see (1)).

Our framework does not require a one-to-one correspondence between actions and the features to be revealed. Instead, this mapping is handled by a separate component, the *Unmasker*, which translates actions into the next set of features to reveal. In Figure 1, at time step $t = 0$, the *Policy* outputs $a_0 = 2$, which, given the image setting and the current *Unmasker*, corresponds to the upper-right 2×2 patch. After the *Unmasker* reveals the selected features, the time index is incremented and the episode continues until the policy selects the stop action or the feature budget is exceeded.

Acquiring features in groups and doing so sequentially (one group per step), rather than selecting individual features, is a common strategy for mitigating the computational intractability of the AFA-MDP in high-dimensional settings [2, 9]. Consequently, when a user adds a new dataset with a different acquisition scheme, they only need to implement an *Unmasker* that specifies the corresponding masking and unmasking rules.

3.1 Cost Accumulation

An episode may terminate when the cumulative cost $c(\pi[x])$ exceeds the budget B (see (2)). Each action selected by the *Policy* incurs a cost that depends on both the dataset and the *Unmasker*, as illustrated by the image-based example in Figure 2. We assume that each dataset specifies a fixed cost per feature. Since a single action may reveal multiple features, the *Unmasker* is responsible for determining the set of features revealed by an action and the corresponding action cost. For image patches, this cost is computed as the average cost over the features revealed by the patch. Costs are accumulated over time steps. If the *Policy* outputs an action that would cause the budget to be **exceeded**, the action is overridden and treated as a stop action.

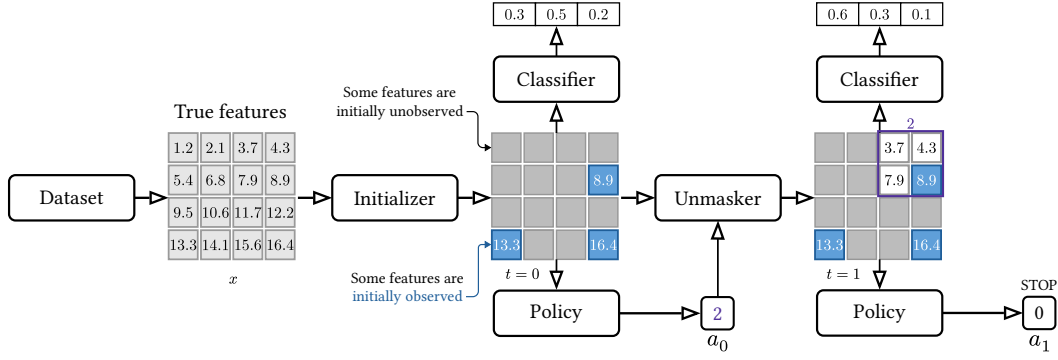


Figure 1: An AFA episode where the instance $x \in \mathcal{X}$ is an image. The Initializer creates the initial mask, the Policy selects actions, the Unmasker reveals features, and the Predictor produces predictions from each partially observed state.

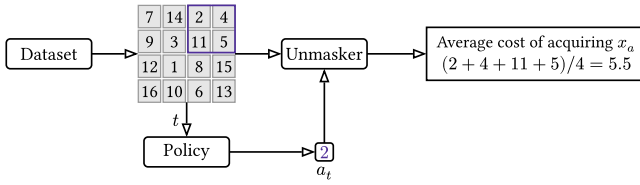


Figure 2: An example of how an image-based Unmasker calculates the cost of an action by averaging the feature costs within the selected patch.

3.2 External vs. Internal Predictors

Although Figure 1 depicts the Predictor and Policy as separate components, some approaches train them jointly [2]. While joint training can improve performance, it conflates gains due to improved action selection with gains due to improved prediction. To disentangle these effects, we compute two predictions during evaluation: one produced by an external classifier and one produced by the method’s internal predictor (when available). We focus on the results based on the external classifier — since it is available to all methods and enables fairer comparison — but provide results for the internal predictors in Appendix A.

3.3 Pipeline Stages

The whole pipeline can be executed with a single command, lowering the barrier to adoption. To achieve this, we assume that the construction of every policy can be split into a pretraining and training stage (both optional), followed by a mandatory evaluation stage, which uses the same script for all methods and implements the framework described above. For example, consider a method that relies on a pretrained external classifier. The pipeline is then as follows. First, pretrain the external classifier. Next, train the policy π using this classifier. Finally, evaluate the resulting policy and the corresponding predictor on test instances. In addition, some methods (e.g., AACO [30]) do not require an explicit training stage, since all computations are performed at inference time. The pipeline stages are illustrated in Figure 3.

The hard budget B and the soft-budget parameter α are specified at both training and evaluation time, and may differ between the

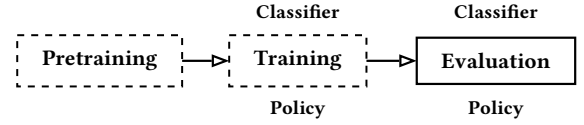


Figure 3: The pipeline consists of three stages. Pretraining and training are optional, while evaluation is mandatory and shared across methods.

two. This is necessary because some methods use only the soft-budget parameter during training (for example, RL-based methods), whereas others use it only at evaluation time (for example, myopic methods based on (3)). More generally, one can combine an arbitrary choice of soft and/or hard budgets at training with an arbitrary choice of soft and/or hard budgets at evaluation, to accommodate different AFA methods.

4 AFABench: An AFA Benchmark

AFABench is a benchmark of AFA methods that leverages the implementation framework described in Section 3. It is written in Python and PyTorch [1] for training and comparing AFA methods under different conditions. It supports multiple datasets, initializers, unmaskers, predictors, and policies. Furthermore, the whole pipeline is executable with a single command, using Snakemake [22].

As the main focus of this work is to benchmark feature acquisition policies, neither the Initializer nor the Unmasker is investigated in-depth. More specifically, only a *cold start* Initializer is used, which always sets the initial feature mask to all zeros. A *direct* Unmasker is used for almost every dataset, which maps each action to a single feature. For *CUBE-NM*, where the context features are represented in a one-hot format, we additionally use a *context* Unmasker with a dedicated action that acquires all context features at once, since the context features are conceptually a single feature. *Imagenette* also uses a different, patch-based image Unmasker, like the one in Figure 1. Similarly, since benchmarking different types of classifiers is not the goal, a simple MLP classifier is used for all datasets except *Imagenette*, where a Vision Transformer (ViT) is used [7].

4.1 Included AFA Methods

As described in the recent survey by Aronsson et al. [2], AFA methods can be broadly categorized as follows: (i) *myopic* heuristics, which perform no explicit planning beyond a one-step lookahead and are often further classified as *discriminative* or *generative*, (ii) *non-myopic* methods based on reinforcement learning (RL), which learn acquisition policies by exploiting the MDP structure underlying AFA (see Section 2.4) and can be divided into *model-free* and *model-based* approaches, and (iii) *non-myopic* methods that do not rely on explicit RL.

All myopic methods included in this benchmark acquire the next feature $a \in \mathcal{A}_B(S) \setminus \{\text{STOP}\}$ by maximizing the expected information gain about the label (relative to its cost):

$$\pi_{\text{CMI}}(S, x_S) = \underset{a \in \mathcal{A}_B(S) \setminus \{\text{STOP}\}}{\operatorname{argmax}} I(\mathbf{y}; \mathbf{x}_a \mid x_S) / c_a. \quad (3)$$

Here, $I(\mathbf{y}; \mathbf{x}_a \mid x_S) = H(\mathbf{y} \mid x_S) - \mathbb{E}_{\mathbf{x}_a \mid x_S} [H(\mathbf{y} \mid x_S, \mathbf{x}_a)]$ denotes the conditional mutual information (CMI) between $\mathbf{y}$ and $\mathbf{x}_a$, where H is the entropy. The myopic methods differ only in how this CMI term is estimated. Their myopic nature is evident in that they ignore interactions between $\mathbf{x}_a$ and the remaining unobserved features in U , that is, they perform no planning beyond a one-step lookahead.

The benchmark includes at least one representative state-of-the-art method from each category to provide a broad comparison of AFA strategies. Among the myopic methods, EDDI [19] estimates the CMI using a generative model, whereas GDFS [5] and DIME [9] estimate CMI using discriminative predictors. The model-free RL-based methods include Jafa [28] and OL [14]. ODIN [32] uses model-based RL by utilizing an explicit model for planning. AACO [30] represents a non-myopic approach that does not explicitly rely on RL, instead using oracle-based lookahead to guide acquisitions. Finally, CAE [3] is a static feature selection baseline based on global feature importance. All included methods are summarized in Table 2 in Appendix D.

4.2 Included Datasets

The benchmark covers both synthetic and real-world tasks across several domains: *CUBE*, a widely used synthetic dataset for AFA, and *CUBE-NM*, a new synthetic dataset introduced in this work (Section 4.3); *MNIST*, handwritten digit recognition [17], and *FashionMNIST*, grayscale clothing-item classification [31], both treated as tabular problems by viewing each pixel as a feature; *PhysioNet*, ICU mortality prediction based on EHR data from the PhysioNet Challenge 2012 [10]; *MiniBooNE*, a particle identification dataset from the MiniBooNE experiment at Fermilab [23]; *ACTG175*, a clinical trial dataset from the UCI repository [20]; *CKD* (chronic kidney disease) and *BankMarketing*, both standard UCI tabular datasets [21, 24]; and *Imagenette*, a 10-class subset of ImageNet used here to evaluate patch-based image acquisition [6].

Additionally, we include results for a noiseless variant of *CUBE-NM* (*CUBE-NM-noiseless*) and a variant of *CUBE* with non-uniform costs (*CUBE-NUC*), where the 10 informative features are repeated in two blocks, but features in the second block cost twice as much as in the first block. All included datasets are summarized in Table 1 in Appendix D.

4.3 CUBE-NM: A Novel Synthetic Dataset for AFA

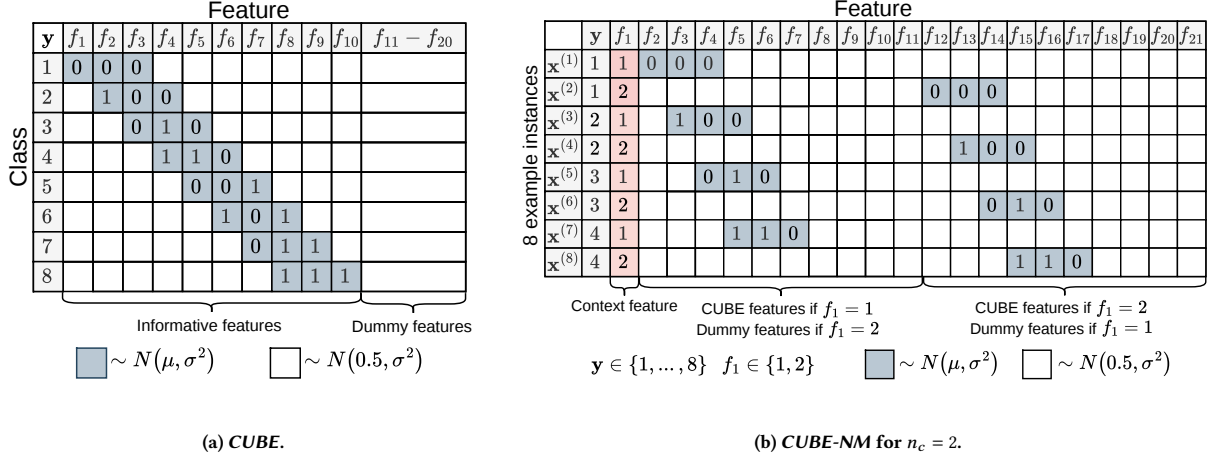
In this section, we introduce *CUBE-NM*, a dataset for benchmarking non-myopic methods for AFA. We first recall the widely used *CUBE* dataset [25, 28, 32], illustrated in Figure 4a. *CUBE* consists of 20-dimensional real-valued vectors from 8 classes. For each class, three informative features are drawn from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with class-specific mean μ , while all remaining features are sampled as noise from $\mathcal{N}(0.5, \sigma^2)$. The locations of the informative features (and their mean) depend on the class (colored cells in Figure 4a, where the numbers indicate the corresponding means). This structure makes *CUBE* well suited for evaluating instance-specific (dynamic) acquisition, but it offers limited separation between myopic and non-myopic methods (shown formally in Theorem 4.1).

CUBE-NM extends *CUBE* by adding a context feature that selects which block of features is informative, while the context value itself is not predictive under myopic criteria. Concretely, each instance has a single integer-valued context feature $f_1 \in \{1, \dots, n_c\}$. For each context $j \in [n_c]$, we associate a block of 10 real-valued features $B_j = \{f_{1+10(j-1)+r} : r = 1, \dots, 10\}$. Conditional on $f_1 = j$, the features in B_j follow the same informative structure as *CUBE* (with $\mathcal{N}(\mu, \sigma^2)$ for informative coordinates and $\mathcal{N}(0.5, \sigma^2)$ for noise coordinates). The total dimensionality is $d = 1 + 10n_c$. Figure 4b shows example instances for the case $n_c = 2$. An optimal policy should therefore acquire f_1 to identify the active block, and then query within that block.

This design creates a setting where myopic methods struggle: the key first step is to acquire f_1 , but f_1 provides no immediate label information, since $I(\mathbf{y}; \mathbf{x}_{f_1}) = 0$ (see (3)). As long as there exist other candidate features with $I(\mathbf{y}; \mathbf{x}_a) > 0$, myopic methods will keep selecting within blocks without first identifying the correct one. Non-myopic methods can instead pay a small upfront cost to observe f_1 , after which they can focus queries in the informative block. Theorem 4.1 formalizes the benefit of a non-myopic approach on this dataset.

Theorem 4.1 (Informal). *Consider the CUBE-NM dataset with n_c contexts in the noiseless regime $\sigma = 0$ (such that 100% prediction accuracy is possible). Assume uniform unit acquisition costs, i.e., $c_i = 1$ for all $i \in [d]$. Then the myopic CMI policy in (3) requires, in expectation across instances, $13(2n_c + 1)/16$ feature acquisitions to achieve 100% accuracy. In contrast, there exists an optimal (non-myopic) policy that achieves 100% accuracy after acquiring only $1\{n_c \geq 2\} + 9/4$ features in expectation.*

A formal statement and proof of Theorem 4.1 is included in the extended version of this paper [26]. For $n_c = 1$ and $\sigma = 0$, *CUBE-NM* reduces to *CUBE*, since f_1 is constant and can be ignored by any method (and the noise features $f_{11} - f_{20}$ in *CUBE* have a negligible effect when σ is small). In this case, Theorem 4.1 shows that the myopic method is nearly optimal: the optimal and myopic policies require 2.25 and 2.4375 feature acquisitions in expectation, respectively. As n_c increases, the gap grows linearly. For example, for $n_c = 3$ we obtain expected costs 3.25 and 5.6875 for the optimal and myopic policy, respectively. Thus, n_c tunes the benefit of non-myopic acquisition. A useful property is that increasing n_c also

Figure 4: Visualization of (a) *CUBE*, and (b) our proposed synthetic dataset *CUBE-NM*.

increases the dimensionality $d = 10n_c + 1$, making it harder to learn a policy that reliably exploits the larger gap. Finally, while Theorem 4.1 assumes $\sigma = 0$, the benefit of non-myopic selection on this dataset remains for $\sigma > 0$, since this does not change the fundamental structure of the dataset.

4.4 Ensuring a Fair Benchmark

In this section, we summarize the key assumptions needed for a fair and objective benchmark. Notably, nearly all prior AFA studies violate at least one of the assumptions below, most often the first. In the extended version of this paper, we discuss these assumptions in more detail [26].

Hard vs. soft budget. We evaluate soft-budget and hard-budget settings separately, since existing AFA methods often target only one and mixing them can confound comparisons. Where possible, methods are adapted to match each objective. Most existing AFA work does not make this distinction, which can confound results by comparing methods that optimize two different objectives.

Common components. All methods are implemented in a unified framework with shared components kept consistent across methods. For example, EDDI and ODIN use the same pretrained partial VAE. In addition, in our experiments using a shared external predictor (see next point), all methods use the same pretrained predictor.

Internal vs. external predictor. Because some methods jointly learn f with the policy π while others assume a fixed f , we report two settings: (i) evaluation with a shared external predictor for all methods, and (ii) evaluation with the method’s internal predictor when available.

Consistent RL framework. RL-based methods share the same overall training loop, differing only in the RL algorithm and reward definition.

Excluded methods. There exist additional approaches to AFA beyond the included methods in our benchmark. We exclude these methods because they are either (i) tied to a specific predictor class (e.g., decision trees), or (ii) depend on strong assumptions about the data distribution (e.g., Naïve Bayes). In contrast, the benchmark

focuses on general, model-agnostic methods that are compatible with a broad range of domains and predictors.

Randomness. In our experiments, we report the average performance over multiple data splits (i.e., training, validation, and test). We also include the standard deviation across runs, visualized as error bars.

Adherence to original architecture. Although the fairest approach would be to perform extensive hyperparameter sweeps for each combination of methods/datasets/budgets, this is not realistic in practice. Therefore, we have opted to use the hyperparameters presented in the original implementation for each method, although some exceptions to this rule do exist. For example, when the parameters of a method have been tuned to datasets with a small number of features and we find that it underfits to (*Fashion*)*MNIST*, we have increased the complexity of the model. Some other changes have been made for consistency reasons.

5 Experiments

As discussed in Section 4, we analyze the soft-budget and hard-budget settings separately. Here, consistent with most prior work on AFA [2], we focus primarily on uniform cost, that is, $c_i = 1$ for all $i \in [d]$, except on the *CUBE-NUC* dataset. We report accuracy for class-balanced datasets and F1-score for class-imbalanced datasets. For the (noisy) *CUBE-NM* dataset, we use $n_c = 5$ and $\sigma = 0.3$. For the *CUBE* variants we also use $\sigma = 0.3$. All results are averaged over five train/validation/test splits.

In Section 5.1, we study the soft-budget setting, in which policies are allowed to stop early and no hard budget is enforced at evaluation time. Instead, each method’s stopping behavior is governed by its soft-budget parameter. For each method–dataset pair, we tune this parameter so that the average number of selections spans a representative range. If the soft-budget parameter cannot be tuned reliably for a given dataset, we exclude the corresponding method from that dataset.

In Section 5.2, we study the hard-budget setting, in which policies have no incentive to stop early and therefore always select up to the fixed budget B . During evaluation, all methods are constrained

by the same hard budget and are not provided with any soft-budget parameter α . Section 5.3 visualizes the actions selected by different methods on the *CUBE-NM* dataset, introduced in Section 4.3. Finally, in Section 5.4 we report the runtime of each method, split into the pretraining, training, and evaluation stages as in Figure 3.

We only report performance achieved at the end of each episode, in both the hard-budget and the soft-budget settings, in accordance with (2). As discussed in Section 4, we evaluate each method using both (i) a shared external predictor and (ii) a method-specific internal predictor. For brevity, in the main text we only present results for the external predictor, which was trained on the first split of each dataset. We defer the internal predictor results to Appendix A. We report variance only for the final performance metric (variance on the y -axis). In the soft-budget setting, there is also variance in the number of selected features across instances, and it is common to visualize this variance as well (i.e., variance on the x -axis). These results can be found in the extended paper [26].

5.1 Soft-budget

First, OL is not included in the soft-budget setting, since adapting it to this setting is not straightforward and would require substantial deviations from its original formulation [14]. Second, the static method CAE is also omitted, because static methods do not perform instance-wise selection and are therefore not applicable in the soft-budget setting.

Figure 5 shows results for the soft-budget setting using a shared external classifier. For each method, the soft-budget parameter α induces a policy π whose feature acquisition decisions may correspond to different cumulative costs across instances. We therefore map each value of α to the resulting cumulative cost averaged across instances and over train, validation, and test splits. This average cost is shown on the x -axis.

We observe that the myopic approaches EDDI, GDFS, and DIME are often among the best-performing methods. Interestingly, within this group, the generative approach EDDI is not substantially worse than the discriminative methods GDFS and DIME, which contrasts with the conclusions reported in [5, 9]. This discrepancy may be explained by differences in the evaluated datasets, or by our tuning of the PVAE model used by EDDI to estimate (3).

However, for several datasets, including *ACTG175*, *FashionMNIST*, and *CUBE-NM*, we find a clear advantage for non-myopic selection. In particular, the non-myopic method AACO, which does not rely on RL, performs well on all three of these datasets. In contrast, the RL-based methods, while in principle well-suited to settings with non-myopic structure, often do not match the performance of AACO. We also find that many RL-based methods exhibit substantial variance, most notably ODIN, suggesting unstable training dynamics. A plausible explanation is the intractability of the underlying MDP for AFA, which many of these methods attempt to solve directly (see [2] and Section 2.4).

5.2 Hard-budget

Figure 6 shows the results for the hard-budget setting using a shared external classifier. For each method, each hard budget B (shown on the x -axis) induces a policy π . We then use this policy to acquire

features until the budget B is exhausted, and report the prediction performance at the end of the episode.

We find that the benefit of non-myopic methods over myopic methods is less pronounced here than in the soft-budget setting. A likely reason is that the hard budget constrains every instance to the same total cost, which limits instance-wise adaptivity. If an optimal non-myopic strategy requires acquiring more than B features for some instances to achieve high accuracy, this behavior is not feasible under a hard budget. Conversely, increasing B to accommodate such instances tends to reduce the gap, since myopic methods also perform well once the budget is sufficiently large. Finally, static methods perform well on some datasets. For instance, CAE ranks among the top performers on *BankMarketing* and *MiniBooNE*, suggesting that a single fixed subset of features is effective for these tasks. In contrast, on datasets such as *CKD* and *CUBE*, dynamic feature acquisition is more beneficial.

5.3 Feature Selections on CUBE-NM

By Theorem 4.1, the optimal policy on *CUBE-NM* with $\sigma = 0$ achieves 100% accuracy at an expected cost of 3.25. In contrast, Figures 5 and 6 show that all evaluated methods fall well short of this performance. This indicates that none of the methods is able to recover the optimal policy on any of the *CUBE-NM* datasets, including the noiseless case with $\sigma = 0$. These results highlight an important direction for future work on AFA: developing non-myopic methods that can effectively identify such policies, and better understanding when the additional planning effort required by non-myopic policies is justified for a given dataset. Moreover, since the theoretically optimal performance on *CUBE-NM* is known by Theorem 4.1, this dataset provides a useful benchmark for evaluating whether non-myopic methods can recover the optimal policy.

In Figure 7, we visualize the features acquired by five methods on noisy *CUBE-NM* with $\sigma = 0.3$ and $n_c = 5$ at different time steps in the hard-budget setting. The context feature f_1 corresponds to row 1 on the y -axis. In Figure 11 in Appendix B, we provide the same visualization for *CUBE-NM-noiseless*. We observe that the RL-based method OL learns to select the context feature consistently. However, Figure 6 shows that it does not find the optimal follow-up selections (due to suboptimal performance). The same figure also shows that OL achieves the best early-selection performance on *CUBE-NM-noiseless*, likely because it uses the context feature to some extent. In contrast, the RL-based method JAFSA never selects the context feature and consequently performs worse. Overall, this suggests that RL-based methods can learn non-myopic policies, but may fail to do so reliably due to training instability.

At the same time, the non-myopic method AACO performs strongly, improving over the myopic baselines, despite not selecting the context feature. This suggests that *CUBE-NM* may admit effective non-myopic strategies that do not acquire the context feature, although the results alone do not establish this conclusively. Moreover, while AACO captures non-myopic behavior, it can be shown that it will never acquire the context feature as its first action on *CUBE-NM*. Thus, AACO should be viewed as an approximation of the MDP underlying AFA that goes beyond a purely myopic approximation, but is not guaranteed to recover the optimal non-myopic policy.

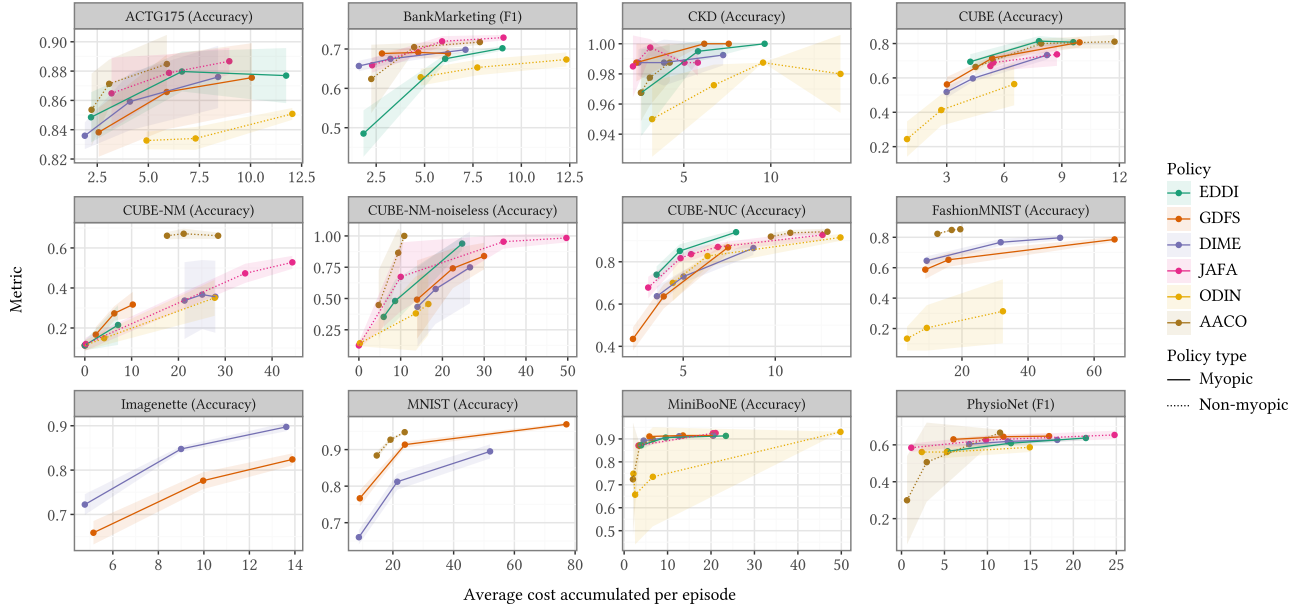


Figure 5: Results for the soft-budget setting using a shared external classifier across all methods. Not all methods are applicable in the soft-budget setting (see main text).

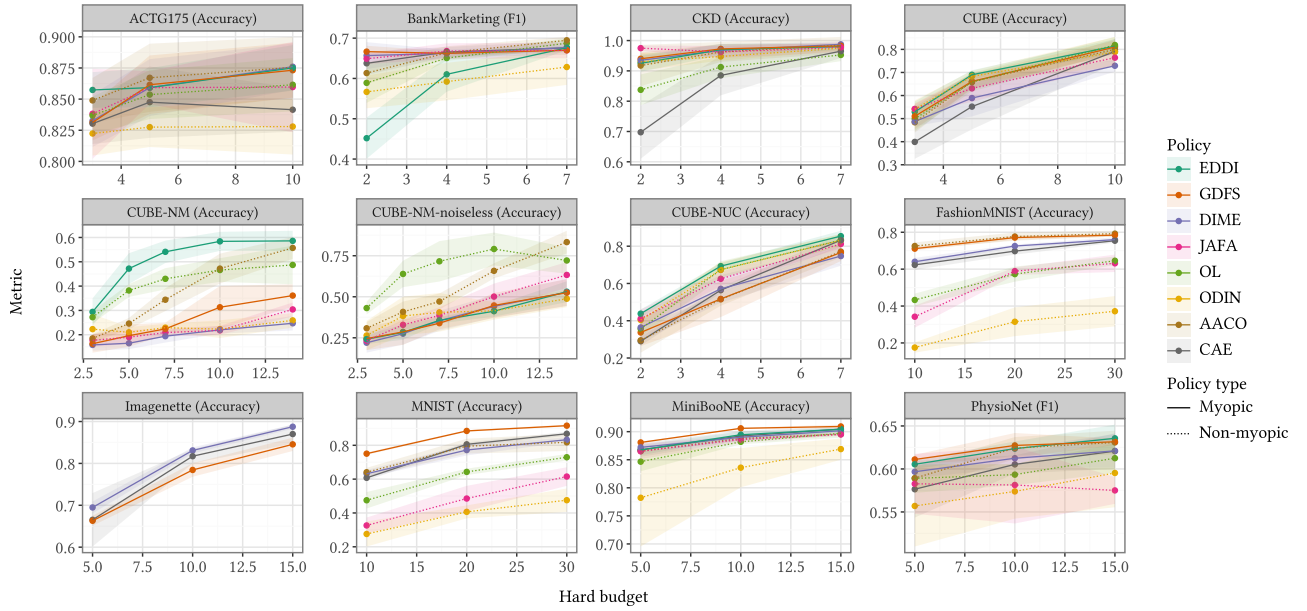


Figure 6: Results for the hard-budget setting using a shared external classifier across all methods.

Finally, we observe that the myopic methods EDDI and GDFS sometimes acquire the context feature. In principle, this should not occur: by construction, $I(y; \mathbf{x}_{\hat{f}_1}) = 0$ for the context feature f_1 , and both EDDI and GDFS converge exactly to the myopic conditional mutual information rule in (3). We therefore attribute these selections to training instability and randomness.

5.4 Compute Comparison

Figure 8 shows the time required to pretrain, train, and evaluate each policy, averaged only over datasets on which all methods were evaluated. All methods used identical hardware, except for AACO, which was trained and evaluated on CPU instead of GPU. Nevertheless, Jafa was the method that required the most compute, likely due to its use of recurrent neural networks.

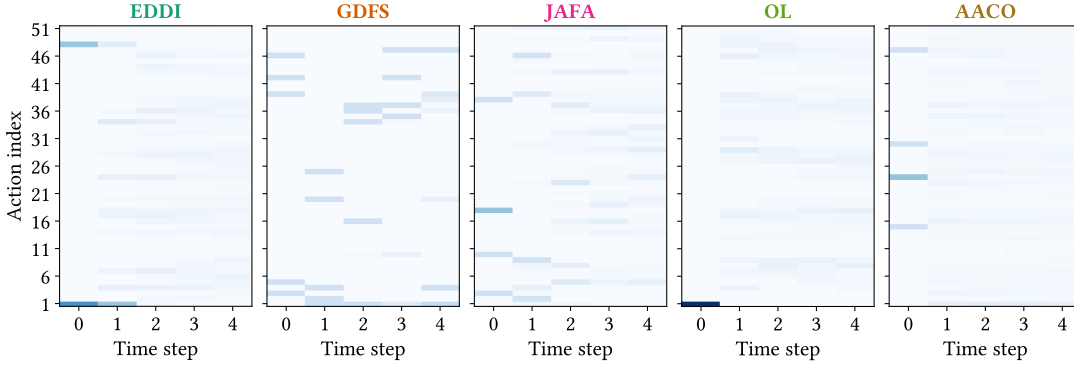


Figure 7: Heatmap of features acquired at different time steps by five methods on the noisy *CUBE-NM* dataset with $\sigma = 0.3$ and $n_c = 5$, using a hard budget of $B = 5$. Action index 1 corresponds to the context feature f_1 (see Section 4.3).

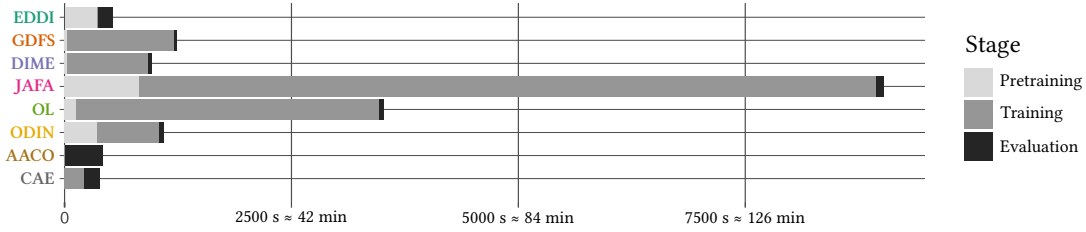


Figure 8: The average time required to pretrain, train, and evaluate each policy, averaged over common datasets and five seeds.

The decomposition into pretraining, training, and evaluation also highlights that methods have different runtime bottlenecks. RL-based methods are generally training-heavy, while oracle-style methods shift more cost to evaluation through repeated candidate scoring. This distinction is important in practice: depending on whether one prioritizes faster development cycles or lower online inference latency, different methods may be preferable even at similar predictive performance.

6 Conclusion

In this work, we study Active Feature Acquisition (AFA), where the goal is to balance predictive accuracy against the cost of acquiring features. To enable fair and comprehensive evaluation of AFA methods, we introduce **AFABench**, the first standardized benchmark framework for this task. **AFABench** includes a diverse set of synthetic and real-world datasets, supports a wide range of AFA methods, and provides a modular design to promote extensibility.

Using **AFABench**, we implement and evaluate representative algorithms from the major AFA paradigms, including (i) myopic methods, (ii) non-myopic methods based on RL, and (iii) non-myopic methods that do not rely on explicit RL. We additionally include static feature selection methods as simple baselines. Finally, we introduce a new synthetic dataset, *CUBE-NM*, to assess the lookahead capabilities of different methods, and we provide a theoretical characterization of the performance gap between optimal non-myopic and myopic policies on this dataset.

Our empirical analysis highlights key trade-offs among AFA strategies: (i) Although non-myopic methods are capable of learning non-myopic policies that can outperform myopic selection, this

is not guaranteed, largely due to the increased complexity and potential instability of these approaches. Moreover, as our results show, many real-world datasets do not exhibit strong non-myopic structure. In such cases, myopic methods offer a strong alternative with significantly shorter training time. Consequently, future work on AFA should aim to characterize when, for a given dataset, the benefits of non-myopic planning justify its additional computational cost. (ii) Dynamic, instance-wise feature selection is often advantageous, especially for datasets with high variability across samples. However, for some datasets, static selection performs comparably well. In addition to their competitive performance, static methods are typically more efficient to train. (iii) We find smaller performance differences between methods than reported in previous work, likely due to our fair benchmarking setup (see Section 4.4). We hope that this benchmark will serve as a basis for future research in cost-sensitive learning and more effective feature acquisition strategies.

Acknowledgments

The work of Linus Aronsson and Morteza Haghir Chehreghani was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The work of Valter Schütz and Morteza Haghir Chehreghani was partially supported by the Swedish Research Council VR (grant number 2023-04809). Finally, the computations and data handling were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

- [1] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Choudhri, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C.K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM. doi:10.1145/3620665.3640366
- [2] Linus Aronsson, Arman Rahbar, and Morteza Haghir Chehreghani. 2026. A Survey on Active Feature Acquisition Strategies. *arXiv preprint arXiv:2502.11067* (2026).
- [3] Muhammed Fatih Balin, Abubakar Abid, and James Y. Zou. 2019. Concrete Autoencoders: Differentiable Feature Selection and Reconstruction. In *Proceedings of the 36th International Conference on Machine Learning*.
- [4] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. 2018. Feature selection in machine learning: A new perspective. *Neurocomputing* 300 (2018), 70–79. doi:10.1016/J.NEUCOM.2017.11.077
- [5] Ian Connick Covert, Wei Qiu, Mingyu Lu, Na Yoon Kim, Nathan J White, and Su-In Lee. 2023. Learning to maximize mutual information for dynamic feature selection. In *International Conference on Machine Learning*.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. doi:10.1109/CVPR.2009.5206848
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=YichFdnTTY>
- [8] Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari. 2011. Datum-wise classification: a sequential approach to sparsity. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5–9, 2011. Proceedings, Part I* 11. Springer, 375–390.
- [9] Soham Gadgil, Ian Connick Covert, and Su-In Lee. 2024. Estimating Conditional Mutual Information for Dynamic Feature Selection. In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=Oju2Qu9jvn>
- [10] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung Kang Peng, and H. Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101, 23 (June 2000), e215–e220. doi:10.1161/01.CIR.101.23.E215
- [11] Isabelle Guyon and André Elisseeff. 2003. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* 3 (2003), 1157–1182. <https://jmlr.org/papers/v3/guyon03a.html>
- [12] Jaromír Janisch, Tomáš Pevný, and Viliam Lisý. 2020. Classification with costly features as a sequential decision-making problem. *Mach. Learn.* 109, 8 (2020), 1587–1615. doi:10.1007/S10994-020-05874-8
- [13] Arjan J. P. Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, Reginald L. Lagendijk, and Qiang Tang. 2013. *Privacy in Recommender Systems*. Springer London.
- [14] Mohammad Kachuee, Orpaz Goldstein, Kimmo Kärkkäinen, and Majid Sarrafzadeh. 2019. Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams. In *International Conference on Learning Representations*.
- [15] Andreas Krause and Carlos Guestrin. 2009. Optimal value of information in graphical models. *J. Artif. Int. Res.* 35 (2009), 557–591.
- [16] Mikko Lauri, David Hsu, and Joni Pajarinen. 2023. Partially Observable Markov Decision Processes in Robotics: A Survey. *IEEE Transactions on Robotics* 39, 1 (2023), 21–40.
- [17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. doi:10.1109/5.726791
- [18] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2018. Feature Selection: A Data Perspective. *ACM Comput. Surv.* 50, 6 (2018), 94:1–94:45. doi:10.1145/3136625
- [19] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. 2019. EDDI: Efficient dynamic discovery of high-value information with partial VAE. In *International Conference on Machine Learning*.
- [20] Julio S. G. Montaner, Peter Reiss, David Cooper, Stefano Vella, Marianne Harris, Brian Conway, Mark A. Wainberg, D. Smith, Patrick Robinson, David Hall, Maureen Myers, Joep M. A. Lange, and for the INCAS Study Group. 1998. A Randomized, Double-blind Trial Comparing Combinations of Nevirapine, Didanosine, and Zidovudine for HIV-Infected Patients: The INCAS Trial. *JAMA* 279, 12 (03 1998), 930–937. doi:10.1001/jama.279.12.930
- [21] S. Moro, P. Rita, and P. Cortez. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5K306>.
- [22] F. Mölder, K. P. Jablonski, B. Letcher, et al. 2021. Sustainable data analysis with Snakemake. *F1000Research* 10 (2021), 33. doi:10.12688/f1000research.29032.1
- [23] Byron Roe. 2005. MiniBooNE particle identification. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5QC87>.
- [24] L. Rubini, P. Soundarapandian, and P. Eswaran. 2015. Chronic Kidney Disease. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5G020>.
- [25] T. Rückstieß, C. Osendorfer, and P. van der Smagt. 2013. Minimizing data consumption with sequential online feature selection. *International Journal of Machine Learning and Cybernetics* 4 (2013), 235–243. doi:10.1007/s13042-012-0092-x
- [26] Valter Schütz, Han Wu, Reza Rezvan, Linus Aronsson, and Morteza Haghir Chehreghani. 2026. AFABench: A Generic Framework for Benchmarking Active Feature Acquisition. *arXiv preprint arXiv:2508.14734* (2026).
- [27] Burr Settles. 2009. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison.
- [28] Hajin Shim, Sung Ju Hwang, and Eunho Yang. 2018. Joint Active Feature Acquisition and Classification with Variable-Size Set Encoding. In *Advances in Neural Information Processing Systems*.
- [29] Peter D. Turney. 1995. Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. *J. Artif. Int. Res.* (1995).
- [30] Michael Valancius, Maxwell Lennon, and Junier Oliva. 2024. Acquisition Conditioned Oracle for Nongreedy Active Feature Acquisition. In *Proceedings of the 41st International Conference on Machine Learning*.
- [31] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017).
- [32] Sara Zannone, Jose Miguel Hernandez Lobato, Cheng Zhang, and Konstantina Palla. 2019. ODIN: Optimal Discovery of High-value INformation Using Model-based Deep Reinforcement Learning. In *Real-world Sequential Decision Making Workshop, ICML*.

A Internal Classifier Results

As mentioned in Section 3.2, some AFA methods train an internal predictor jointly with the policy. Figure 9 shows the soft-budget prediction performance of these internal predictors, while Figure 10 shows the hard-budget performance.

B Action Heatmap on Noiseless CUBE-NM

For comparison with the noisy setting in Figure 7, Figure 11 presents the corresponding action heatmap for the **noiseless CUBE-NM** dataset. In the noiseless case, EDDI no longer selects the context feature, which helps explain its weaker performance relative to the noisy variant. By contrast, OL is the only method that consistently learns to select the context feature across both versions of **CUBE-NM**.

C Hardware

- Myopic methods on the *Imagenette* dataset used NVIDIA A40 GPUs during the training stage.
- AACO used 4 AMD EPYC 9354 Zen4 CPU cores during both training and evaluation.
- The remaining methods used one NVIDIA T4 GPU during both (pre)training and evaluation.

D Included Datasets and Methods

The full list of included datasets in **AFABench** is given in Table 1, and the list of methods in Table 2. Due to space constraints and for readability, the static PT method and the dataset *Diabetes* were excluded from the main paper, along with several variations of

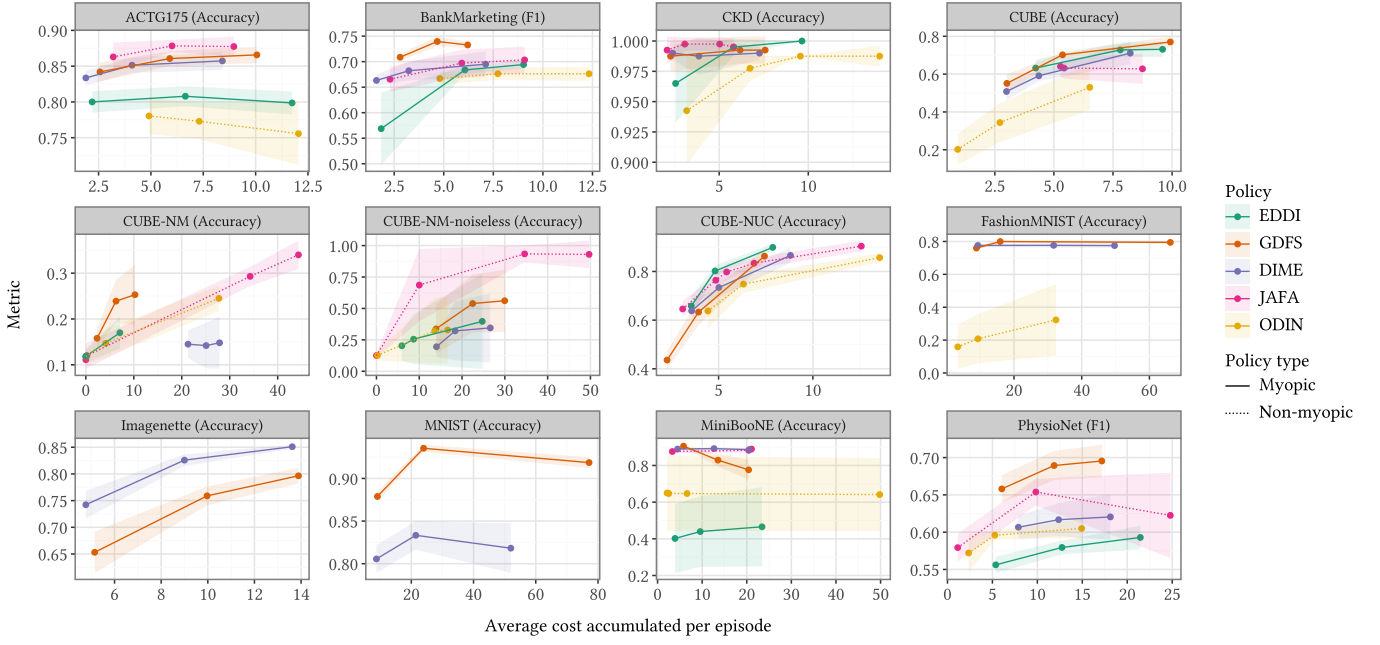


Figure 9: Results for the soft-budget setting using each method's internal predictor (if applicable). Not all methods are applicable in the soft-budget setting (see main text).

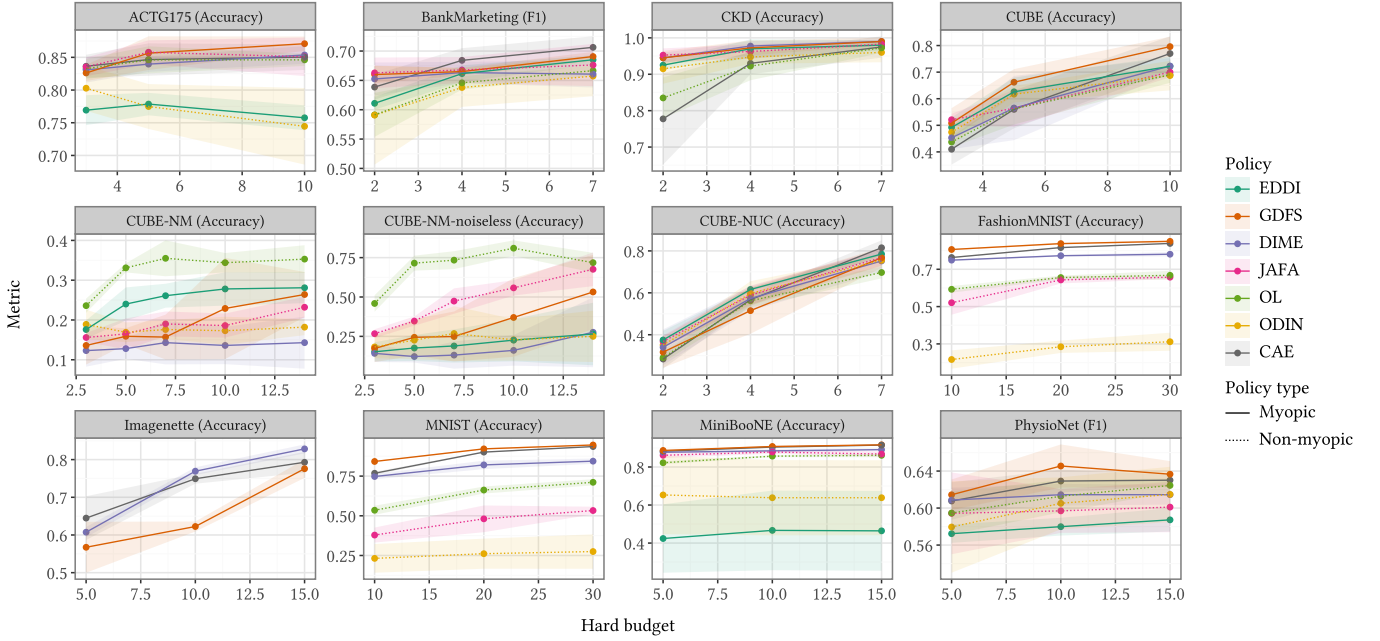


Figure 10: Results for the hard-budget setting using each method's internal predictor (if applicable).

the AFA methods, but the full results and additional details can be found in the extended version [26].

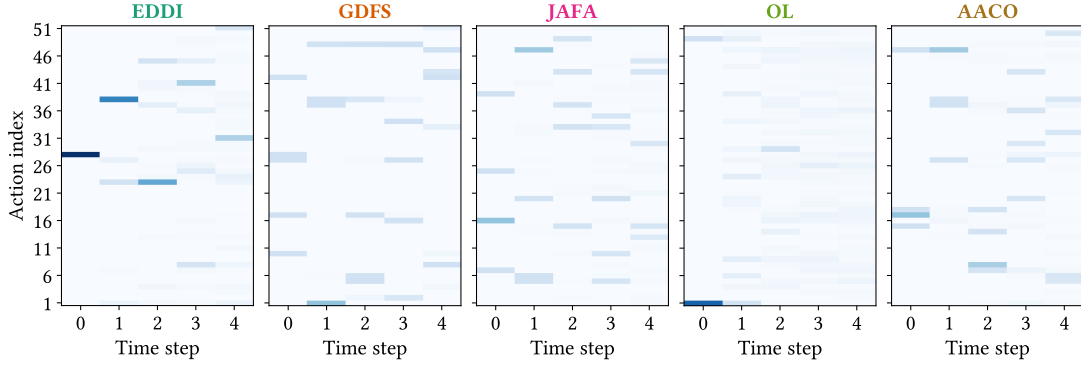


Figure 11: Heatmap of features acquired at different time steps by five methods on the noiseless *CUBE-NM* dataset with $\sigma = 0$ and $n_c = 5$, using a hard budget of $B = 5$. Action index 1 corresponds to the context feature f_1 (see Section 4.3).

Table 1: Included datasets in AFABench

Dataset	Type	Modality	Train Size	Val Size	Test Size	# Features	# Groups	# Classes
CUBE	Synthetic	Tabular	600	200	200	20	20	8
CUBE-NUC	Synthetic	Tabular	600	200	200	20	20	8
CUBE-NM	Synthetic	Tabular	600	200	200	55	51	8
MNIST	Real World	Image (tabularized)	36,000	12,000	12,000	784	784	10
FashionMNIST	Real World	Image (tabularized)	36,000	12,000	12,000	784	784	10
PhysioNet	Real World	Tabular	7,200	2,400	2,400	41	41	2
MiniBooNE	Real World	Tabular	78,038	26,012	26,014	50	50	2
ACTG175	Real World	Tabular	1,283	427	429	23	23	2
CKD	Real World	Tabular	240	80	80	24	24	2
BankMarketing	Real World	Tabular	27,126	9,042	9,043	16	16	2
Imagenette	Real World	Image	5,681	3,788	3,925	150,528	196	10

Table 2: Included methods in AFABench

Paradigm	Strategy	Myopic?	Representative Method(s)
AFA (Dynamic Feature Selection)	Generative estimation of CMI	Myopic	EDDI [19]
	Discriminative estimation of CMI	Myopic	GDFS [5]; DIME [9]
	Reinforcement learning	Non-myopic	JAFA [28]; OL [14]; ODIN [32]
	Oracle-based	Non-myopic	AACO [30]
Static Feature Selection	Global feature importance	N/A	CAE [3]