Information Templates: A New Paradigm for Intelligent Active Feature Acquisition

Hung-Tien Huang*
Dept. of Computer Science
UNC Chapel Hill
hungtien@cs.unc.edu

Dzung Dinh*
Dept. of Computer Science
UNC Chapel Hill
ddinh@cs.unc.edu

Junier Oliva
Dept. of Computer Science
UNC Chapel Hill
joliva@cs.unc.edu

Abstract

Active feature acquisition (AFA) is an instance-adaptive paradigm in which, at inference time, a policy sequentially chooses which features to acquire (at a cost) before predicting. Existing approaches either train reinforcement learning policies, which deal with a difficult MDP, or greedy policies that cannot account for the joint informativeness of features or require knowledge about the underlying data distribution. To overcome this, we propose Template-based AFA (TAFA), a non-greedy framework that learns a compact library of feature templates—sets of features that are jointly informative—and uses this library of templates to guide the next feature acquisitions. Through identifying feature templates, the proposed framework not only significantly reduces the action space considered by the policy but also alleviates the need to estimate the underlying data distribution. Extensive experiments on synthetic and real-world datasets show that TAFA outperforms the existing state-of-the-art baselines while achieving lower overall acquisition cost and computation. Code is publicly available at `https://github.com/busFred/tafa-demo`.

1 Introduction

Many ML methods implicitly assume fully observed feature vectors at inference time; however, acquiring features often incurs non-trivial costs in time, money, or effort. *Active feature acquisition* (AFA) [1] addresses this mismatch by treating inference as a sequential process in which an agent adaptively selects features while balancing predictive accuracy and acquisition cost. AFA policies personalize the selected set of features from instance to instance due to their sequential nature, where future acquisitions depend on the observed values of previous acquisitions. For example, in health care applications, this implies that certain patients may have their demographic, lab, and X-ray-related features acquired for inference, while other patients may have demographic and EEG-based features acquired. Beyond healthcare, applications abound in robotics (making decisions with judicious use of sensors), education (assessing student knowledge with judicious question choices), troubleshooting (assessing the issues with judicious choices of diagnostic tests), etc.

Given the combinatorial nature of the dynamic AFA acquisition process, the potential acquisition space is exponential in instances' dimensionality. This large decision space, in turn, makes learning, computing, and *understanding* policies difficult. In this work, we turn our attention to a previously overlooked question: *what are the canonical feature subsets driving the cost/benefit performance in AFA tasks?* We answer this question and introduce template-based AFA (TAFA), a novel approach that simplifies AFA according to a learned dictionary of feature subsets (*templates*) in order to improve both the efficacy as well as inference time of AFA agents, all whilst avoiding the need for RL-based training. In addition, we leverage our template concept to learn interpretable policies that

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

succinctly and exactly characterize the subsets of features an AFA agent may acquire, as well as the types of instances that map to each subset—capturing both *what information matters* and *for whom*.

Contributions First, we introduce a template-based framework for efficient non-greedy acquisition, achieving strong accuracy-cost tradeoffs with substantially lower inference latency. Second, we develop a mutation-greedy search to construct high-quality template libraries, and an end-to-end refinement stage that tunes templates via Gumbel-Softmax relaxation [2]. Third, we show that the template-search objective has submodular structure, which motivates our greedy-style search and yields a $(1 - 1/e)$ approximation guarantee for the standard greedy algorithm in a special case (0–1 loss and uniform costs; Appx. E); we further show that our non-recursive cost/benefit criterion lower-bounds the AFA MDP value. Fourth, we present a student/teacher distillation framework that yields an interpretable decision-tree policy with a human-readable list of possible acquisition subsets and corresponding rules. Fifth, we conduct extensive experiments demonstrating that TAFA outperforms baselines in both predictive performance and computational efficiency.

2 Background and Related Works

Notation and Data Let $x = \langle x_d \rangle_{d=1}^{D} \in \mathbb{R}^D$ be a D-dimensional input feature vector and y be its corresponding output for a supervised learning task. We use $\mathbf{o} \subseteq [D] \equiv \{1, \cdots, D\}$ and $\mathbf{o} \in \mathfrak{P}([D])$ to denote a subset of feature indices, where $\mathfrak{P}(\cdot)$ is the power set. Let $x_\mathbf{o} \in \mathbb{R}^{|\mathbf{o}|}$ be the corresponding subset feature values indexed by $\mathbf{o}$. We assume standard access to a supervised learning dataset $\mathcal{D}_{\text{trn}} = \left\{(x^{(n)}, y^{(n)})\right\}_{n=1}^{N} \overset{\text{iid}}{\sim} \mathcal{P}$, where $\mathcal{P}$ is the data distribution, and a predictor $\widehat{y}(x_\mathbf{o})$ that can make an inference about the response variable using an arbitrary subset of features from the corresponding instance.

Sequential Acquisition and Policy AFA [3] formulates inference as a sequential process in which an agent acquires unobserved feature values at a cost and decides when to stop acquisition and make a prediction (see Fig. 1); i.e., for a current set of observed features $\mathbf{o}$, an AFA policy $\pi(x_\mathbf{o}) \in ([D] \setminus \mathbf{o}) \cup \{\varnothing\}$, either: 1) acquires a new feature $\mathbf{o} \leftarrow \mathbf{o} \cup \{\mathbf{a}\}$ when choosing action $a = \pi(x_\mathbf{o}) \in [D]$ (i.e., continuing the policy at $\pi(x_{\mathbf{o} \cup \{a\}})$); or 2) uses the predictor to infer $\widehat{y}(x_\mathbf{o})$ when terminating action is chosen $a = \pi(x_\mathbf{o}) = \varnothing$.

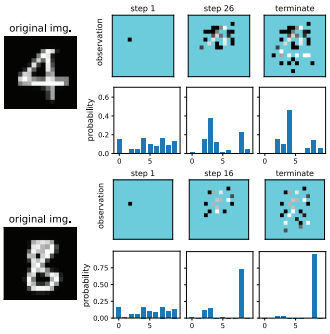


Figure 1: Rolling out an AFA policy. The AFA agent gathers features until it has enough information to make a decision about the instance at hand.

Conditional Mutual Information Several AFA methods greedily select the next feature by estimating conditional mutual information (CMI), often using generative models to approximate $\mathcal{P}$ and compute expected information gain (e.g., EDDI [4]). While effective, such approaches can incur substantial computational and optimization challenges [5]. DIME [5] offers a discriminative alternative, recovering CMI at optimality with a regression-based objective that avoids generative modeling.

Reinforcement Learning Approaches The sequential active acquisition problem can be posed as a Markov Decision Process (MDP) [6–8] where states are observed features and actions correspond to either acquiring an unobserved feature or terminating the acquisition process; the rewards can reflect negative acquisition costs and predictive performance measures (when terminating). After deciding to terminate acquisition and use a feature subset $\mathbf{b} \subseteq [D]$ to infer about the ground truth label y, the cumulative reward for an AFA episode on instance x is then $G = -e(x_\mathbf{b}, y)$; where we coin e as the *negative cumulative reward objective*:

$$e(x_\mathbf{b}, y) = l(\widehat{y}(x_\mathbf{b}), y) + \lambda \textstyle\sum_{b \in \mathbf{b}} c(b), \tag{1}$$

and l is a supervised-learning (e.g., cross-entropy) loss, $c : [D] \mapsto \mathbb{R}_+$ is a relative cost associated with each feature, and λ is an *application-specific* trade-off parameter controlling the trade-off between cost and predictive performance. The negative cumulative reward objective (eq. 1) shall play an integral role in the design of our AFA templates below.

Direct (Non-RL) Acquisition Objectives Training an RL agent can overcome limitations of CMI-based methods, which may fail to capture features that are only informative jointly with respect to the target [9]. However, learning RL agents for the AFA MDP remains challenging due to poor sample efficiency, training instability, catastrophic forgetting [10], large action spaces, and credit

assignment difficulties [8]. To avoid RL drawbacks, SEFA [11] proposes learning feature-wise stochastic latent representations and selecting features by maximizing an expected gradient-based objective in latent space. While this yields a non-RL acquisition strategy, it requires training an additional latent model and is not directly tied to the AFA MDP reward structure. In a complementary approach, Valancius et al. [9] introduced the acquisition-conditioned oracle (ACO), which avoids both RL and generative modeling by estimating likely values of unobserved features at inference time and selecting informative, cost-effective subsets. Despite its theoretical appeal, ACO is computationally prohibitive, as it evaluates objectives over exponentially many future acquisition sequences; even its approximate variant (AACO), which subsamples, remains slow at inference.

Our TAFA approach yields a policy with a decision space that is constrained to a learned collection of templates and avoids planning that infers over all possible feature combinations, improving efficiency while staying aligned with AFA MDP rewards (see Thm. 3.2 below).

Interpretable AFA Despite their effectiveness, most AFA policies provide limited justification for why a particular instance is routed to a specific acquisition set. A recent interpretable AFA approach [12] applies post-hoc explanations (e.g., SHAP [13]) to obtain instance-wise feature importance rankings and then learns a sequential policy that selects the next unacquired feature with the highest attribution given the current partial information. While intuitive, this strategy trains the policy to follow the predictor's attributions rather than yielding transparent, rule-based justifications for acquisition decisions. In parallel, works on interpretable RL have built structured policies such as decision-tree and differentiable decision-tree models [14, 15]. Although these approaches can be adapted to AFA by treating feature indices as actions, their single-feature action formulation makes capturing set-level interactions difficult, often requiring large trees to approach black-box performance.

In contrast, *our TAFA approach* compresses the acquisition decision space by acting over learned feature-set templates rather than individual raw features, enabling distillation into compact step-wise trees with explicit ACQUIRE IF rules and yielding an intrinsically interpretable policy.

Submodularity and Feature Selection Optimal feature selection often exhibits submodular structure with diminishing marginal utility as subset size grows, enabling greedy algorithms with provable guarantees [16]. Early work leveraged these properties for *dataset-level*, static feature selection that identifies a *single subset* intended to perform well on average across instances [17, 18].

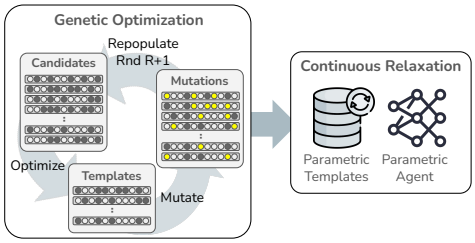


Figure 2: Overview of template optimization.

Our TAFA approach departs from the static paradigm (and is the first to exploit submodularity in AFA, to our knowledge) by learning *multiple* feature subsets (templates) that enable instance-adaptive inference-time acquisition, allowing different instances to be routed to different informative subsets. The negated TAFA objective is submodular (Thm. 3.1), motivating our greedy-style template search.

3 Methodology

Our proposed template-based approach searches for an idealized collection of feature subsets, the *templates*, which enable a near-optimal cost/benefit objective (eq. 1) under oracle selection on training instances (§ 3.1). This leads to a submodular set optimization problem, for which a greedy approach remains intractable; thus, we derive a greedy+genetic algorithm (§ 3.3; see Fig. 2). Moreover, we develop a discrete Gumbel relaxation technique to further refine templates (§ 3.4). Finally, utilizing the templates, we propose an intrinsically interpretable policy that explicitly shows the policy's decision process (§ 3.5). Our template-based policies enable faster inference with strong teachers, enabling the efficient training of an interpretable student.

3.1 Template Search as Set Optimization

We propose to find a collection $\mathcal{B}^\star = \{\mathbf{b}^{(i)}\}_{i=1}^{B}$ of B feature templates, where each template $\mathbf{b}^{(i)} \subseteq [D]$ is an informative combination of features. We formulate the problem of finding an op-

timal collection of templates as a set optimization problem[2]. As discussed earlier, an effective AFA policy must balance acquiring informative features against the associated costs; therefore, we define the objective function as:

$$g(\mathcal{B}) = \mathbb{E}_{x,y\sim\mathcal{P}}\left[\min_{\mathbf{b}\in\mathcal{B}} l(\widehat{y}(x_{\mathbf{b}}), y) + \lambda\sum_{b\in\mathbf{b}} c(b)\right] \tag{2}$$

i.e., $g(\mathcal{B}) = \mathbb{E}_{x,y\sim\mathcal{P}}\left[\min_{\mathbf{b}\in\mathcal{B}} e(x_{\mathbf{b}}, y)\right]$, the expected oracle cost/benefit objective (eq. 1) for data instances with templates in $\mathcal{B}$. Thus, our template optimization problem is:

$$\mathcal{B}^\star = \arg\min_{\mathcal{B}\in\mathfrak{S}_B} g(\mathcal{B}), \tag{3}$$

where $\mathfrak{S}_B$ represents the search space of all possible template collections, $|\mathfrak{S}_B| = \mathcal{O}\left(\binom{2^D}{B}\right)$. An important property of our objective function is that its negation is a submodular set function.

Theorem 3.1. *The negated objective $-g(\mathcal{B})$ of eq. 2 is submodular in $\mathcal{B}$ (empty-collection convention in Appx. D); equivalently, g is supermodular.*

Proof. See the "facility location" problem discussed in Krause and Golovin [19] and Frieze [20], and Appx. D in the supplementary for details of the proof. □

This submodularity ensures diminishing returns: adding a template to a larger collection provides no more benefit than adding it to a smaller one. When per-instance rewards $-e$ are nonnegative (e.g., a shifted 0-1 loss with uniform costs), plain greedy maximization of $-g$ enjoys the classic $(1 - 1/e)$ guarantee (for general losses, only w.r.t. a row-shifted objective; Appx. E). This guarantee does not carry over to our deployed setting (cross-entropy loss and the genetic search below); rather, submodularity motivates our design and is consistent with the diminishing returns observed in B (§ C.3). **However, even a greedy algorithm shall be $\mathcal{O}(B2^D)$ and thus intractable**. Hence, below (§ 3.3), we further develop evolution-based and relaxation strategies.

3.2 Template-Based Policy

Here, we discuss how one may construct AFA policies through an optimized template collection $\mathcal{B}$. Intuitively, a straightforward policy operates in two stages: template selection and feature acquisition. Given current observations $x_{\mathbf{o}}$, we first select the most promising template (eq. 1):

$$\mathbf{b}^\star = \arg\min_{\mathbf{b}\in\mathcal{B}} \left\{\widehat{l}_{\mathbf{b}}(x_{\mathbf{o}}) + \lambda\sum_{j\in\mathbf{b}\setminus\mathbf{o}} c(j)\right\} \tag{4}$$

where $\widehat{l}_{\mathbf{b}}(x_{\mathbf{o}})$ estimates the prediction loss when using template $\mathbf{b}$ given the current observations, $\mathbb{E}_{y,x_{\mathbf{b}}|x_{\mathbf{o}}}[l\left(\widehat{y}\left(x_{\mathbf{o}\cup\mathbf{b}}\right), y\right)]$, and $\lambda\sum_{j\in\mathbf{b}\setminus\mathbf{o}} c(j)$ represents the remaining acquisition cost for unobserved features in template $\mathbf{b}$. In our `tafa-knn` variant, $\widehat{l}_{\mathbf{b}}(x_{\mathbf{o}})$ is estimated in a supervised fashion: we retrieve the k training instances nearest to $x_{\mathbf{o}}$ on the observed coordinates ($\|x_{\mathbf{o}} - x'_{\mathbf{o}}\|_2$) and average their cached losses $l(\widehat{y}(x'_{\mathbf{o}\cup\mathbf{b}}), y')$; we acquire a cheapest unobserved feature once a template is chosen:

$$\pi(x_{\mathbf{o}}) \in \arg\min_{a\in\mathbf{b}^\star\setminus\mathbf{o}} c(a). \tag{5}$$

If $\mathbf{b}^\star\setminus\mathbf{o} = \varnothing$, we set $\pi(x_{\mathbf{o}}) = \varnothing$ and terminate. Let $V(x_{\mathbf{o}})$ denote the optimal state value of the AFA MDP with reward $-e$ (eq. 1; see Appx. F). Although eq. 4 requires no Bellman recursion, the negative of its minimized criterion lower-bounds V when conditional losses are exact:

Theorem 3.2. *The optimal AFA MDP value function is lower bounded by the negative of the minimized TAFA cost criterion, evaluated using exact conditional expected losses:*

$$V(x_{\mathbf{o}}) \geq \max_{\mathbf{b}\in\mathcal{B}} \left\{-\mathbb{E}_{y,x_{\mathbf{b}\setminus\mathbf{o}}|x_{\mathbf{o}}}\left[l\left(\widehat{y}(x_{\mathbf{o}\cup\mathbf{b}}), y\right)\right] - \lambda\sum_{j\in\mathbf{b}\setminus\mathbf{o}} c(j)\right\}.$$

Proof. See Appx. F for details. □

Hence, optimizing the exact TAFA criterion maximizes a lower bound on the optimal AFA value without requiring Bellman recursion, and can be viewed as a structured approximation to more general AFA approaches. While methods like ACO [9] consider all possible feature acquisition sequences, our approach restricts choices to a learned set of templates $\mathcal{B}$, trading optimality for computational efficiency and interpretability. Importantly, the policy may switch templates dynamically as new features are observed, rather than committing to a single template per instance (§ C.5).

[2]A function $g : \mathfrak{P}(\mathcal{V}) \to \mathbb{R}$ is said to be a set function if it assigns each subset $\mathcal{S} \subseteq \mathcal{V}$ of a finite set $\mathcal{V}$ to a value $g(\mathcal{S})$.

3.3 Mutation-Guided Greedy Template Search

As aforementioned, the large $\mathcal{O}(2^D)$ space of potential templates makes even greedy routines intractable, notwithstanding submodularity. Thus, here we develop a genetic search procedure that conducts a greedy search on a smaller, mutated candidate set of templates, as expounded below.

Given a candidate set, $\mathcal{C}$, to select templates from, a training dataset, $\mathcal{D}_{\text{trn}}$, and previously selected templates, $\widehat{\mathcal{B}}_{t-1}$, it is straightforward to show that the next template selection problem optimizes:

$$\mathbf{b}_t^\star = \arg\min_{\widetilde{\mathbf{b}} \in \mathcal{C} \setminus \widehat{\mathcal{B}}_{t-1}} \frac{1}{|\mathcal{D}_{\text{trn}}|} \sum_{\mathcal{D}_{\text{trn}}} \min \left(e\left(x_{\widetilde{\mathbf{b}}}, y\right), \min_{\mathbf{b} \in \widehat{\mathcal{B}}_{t-1}} e\left(x_{\mathbf{b}}, y\right) \right) \tag{6}$$

Intuitively, eq. 6 asks: *if we add candidate $\widetilde{\boldsymbol{b}}$ to $\mathcal{B}_{t-1}$, how much does the best-achievable performance improve on average.* As an unrestricted candidate set, $\mathcal{C}$, is exponential in size, we propose a genetic [21–23], mutation-guided optimization to evolve a tractable $\mathcal{C}$, where, following the submodularity intuition, at multiple rounds the template collection is fit greedily from candidates.

At each round r, we first run a greedy search over the current candidate template set $\mathcal{C}^{(r)}$, yielding a selected template collection $\widehat{\mathcal{B}}^{(r)}$. The candidate set for the next round is then constructed by 1) mutating each template in $\widehat{\mathcal{B}}^{(r)}$, 2) carrying forward the previously optimized templates $\widehat{\mathcal{B}}^{(r)}$, and 3) injecting an additional set of randomly sampled templates to encourage exploration. Mutations randomly drop features from existing templates with probability 0.5. This design balances exploitation—by refining high-utility templates—with exploration through stochastic mutations and fresh random candidates. The initial candidate set $\mathcal{C}^{(0)}$ is populated entirely with random templates. The search is not sensitive to initialization, as confirmed by a 5-seed ablation (§ C.1). An initial feature, o_{init}, is identified *post-hoc* after the first round, then pinned into every template (see Appx. B for the identification procedures and § C.6 for an ablation showing robustness across strategies). The MUTATE routine ensures that every candidate contains o_{init}, including mutated, retained, and randomly generated candidates. The repository size B is a tunable capacity parameter; our ablations show performance is not sensitive to B beyond a modest threshold (§ C.3). See Alg. 1 for the pseudocode and Fig. 2 for a visual.

Algorithm 1 ITERATIVEMUTATESEARCH

Require: training dataset $\mathcal{D}_{\text{trn}}$, predictor $\widehat{y}$, task objective function l, cost function c, trade-off parameter λ, template set size B, candidate set size S, mutative rounds R.
Ensure: $\widehat{\mathcal{B}}$, o_{init}: template collection and identified initial feature.

for $r \leftarrow 0, \ldots, R$ **do**
  if $r = 0$ **then**
    $\mathcal{C}^{(r)} \leftarrow$ MAKERANDOMCANDIDATES(D, S)
  else
    $\mathcal{C}^{(r)} \leftarrow$ MUTATE$(\widehat{\mathcal{B}}^{(r-1)}, S, o_{\text{init}})$
  end if
  Evaluate $e(x_{\widetilde{\mathbf{b}}}, y) \quad \forall \left((x, y), \widetilde{\mathbf{b}}\right) \in \mathcal{D}_{\text{trn}} \times \mathcal{C}^{(r)}$ {eq. 1}
  Initialize $\widehat{\mathcal{B}}_1^{(r)} \leftarrow \left\{ \arg\min_{\mathbf{b} \in \mathcal{C}^{(r)}} \frac{1}{|\mathcal{D}_{\text{trn}}|} \sum_{(x,y) \in \mathcal{D}_{\text{trn}}} e(x_{\mathbf{b}}, y) \right\}$
  for $t = 2, \ldots, B$ **do**
    $\mathbf{b}_t^\star \leftarrow \arg\min_{\widetilde{\mathbf{b}}} \sum_{(x,y) \in \mathcal{D}_{\text{trn}}} \min \left(e(x_{\widetilde{\mathbf{b}}}, y), \min_{\mathbf{b}} e(x_{\mathbf{b}}, y) \right)$ {eq. 6}
    $\widehat{\mathcal{B}}_t^{(r)} \leftarrow \widehat{\mathcal{B}}_{t-1}^{(r)} \cup \{\mathbf{b}_t^\star\}$
  end for
  $\widehat{\mathcal{B}}^{(r)} \leftarrow \widehat{\mathcal{B}}_B^{(r)}$
  if $r = 0$ **then**
    $o_{\text{init}} \leftarrow$ IDENTIFYINITFEATURE$(\widehat{\mathcal{B}}^{(r)}, \mathcal{D}_{\text{trn}})$ {see Appx. B}
    $\widehat{\mathcal{B}}^{(r)} \leftarrow \{\mathbf{b} \cup \{o_{\text{init}}\} : \mathbf{b} \in \widehat{\mathcal{B}}^{(r)}\}$
  end if
end for
return $\widehat{\mathcal{B}}^{(R)}$, o_{init}

3.4 Template Search Through Continuous Relaxation

To further refine the template sets, we provide an additional *non-greedy* tuning routine through continuous relaxation [2] to optimize templates w.r.t. *a learned actor* (rather than an oracle selection, eq. 2), yielding an updated set of templates for deployment.

Parameterized Templates We formulate a relaxed version of our template bank $\mathcal{B} = \{\mathbf{b}^{(i)}\}_{i=1}^B$ as follows. Let $\widetilde{\mathcal{M}} = \langle \widetilde{m}^{(i)} \rangle_{i=1}^B \in \mathbb{R}^{B \times D}$ denote the relaxed feature template bank, where each of the $\widetilde{m}^{(i)}$'s is defined as $\text{Sigmoid}(\mu_i)$ and $\mu \in \mathbb{R}^{B \times D}$ collects the template parameters, with $\mu_i \in \mathbb{R}^D$ denoting its i-th row. Note that one can use $\mathcal{B} \equiv \widehat{\mathcal{B}}^{(R)}$ in Alg. 1 to warm start the continuous

optimization. We fix $\widetilde{m}^{(i)}_{o_{\text{init}}} = 1$ throughout optimization to preserve the initial feature o_{init}, where o_{init} is identified as above. Additionally, we initialize the μ_i's from the template bank $\mathcal{B} = \{\mathbf{b}^{(i)}\}_{i=1}^{B}$, $\mu_{i,j} = \kappa\,(2\,\mathbb{I}\{j \in \mathbf{b}^{(i)}\} - 1)$ for all $i \in [B]$ and $j \in [D]$, where $\kappa \in \mathbb{R}_{>0}$.

Differentiable Actor over Templates Given the current observation $(x_{\mathbf{o}}, \mathbf{o})$, we train an actor π_θ to output a categorical distribution over B templates[3]:

$$h_{x_{\mathbf{o}}} = f_\theta\left([x_{\mathbf{o}},\ \mathbf{o}]\right) \in \mathbb{R}^B$$
$$\pi_\theta\left(x_{\mathbf{o}}, \mathbf{o}\right) = \text{softmax}(h_{x_o}) \in \Delta^{B-1}$$

To select a template while remaining differentiable, we employ the Gumbel-Softmax relaxation [2]. We draw a relaxed template selection $\widetilde{z} \sim \text{GumbelSoftmax}(h_{x_{\mathbf{o}}}; \tau) \in \Delta^{B-1}$ during back-propagation for end-to-end training, where $\tau > 0$ is a hyperparameter. To make a hard discrete choice during forward propagation, we select the template with the highest probability $z = \text{OneHot}\big(\arg\max_{i\in[B]} \widetilde{z}_i\big)$. We use an annealing schedule that decays τ from 2.0 to 0.1 during training, and has been confirmed to be beneficial with the ablation done in § C.2.

Algorithm 2 CONTINUOUSTEMPLATEREFINEMENT

Require: training data $\mathcal{D}_{\text{trn}}$, actor π_θ, template params μ (warm-started from Alg. 1), initial feature o_{init} (from Alg. 1), trade-off parameter λ, temperature τ, learning rate η, batch size U, number of features D.
Ensure: Refined (θ, μ) and $\widetilde{\mathcal{B}} = \{\widetilde{\mathbf{b}}^{(i)}\}_{i=1}^{B}$

while training **do**
  $(X, Y) \sim \mathcal{D}_{\text{trn}}$
  Initialize O_0 with o_{init} for each instance
  for $t = 0, \ldots, D-1$ **do**
    $(Z_t, \widetilde{Z}_t) \sim$ GUMBELSOFTMAX$(f_\theta([X_{O_t},\ O_t])\,; \tau)$
    $(\theta, \mu) \leftarrow (\theta, \mu) - \eta \nabla_{\theta,\mu} \frac{1}{U} \sum_{n=1}^{U} \mathcal{L}_{\text{actor}}(\theta, \mu; x^{(n)}, y^{(n)}, o_t^{(n)})$ {eq. 7}
    $\mathcal{T}_t \leftarrow$ HARDSELECTTEMPLATE(Z_t, μ)
    if $\mathcal{T}_t \setminus O_t = \varnothing$ **then**
      break {stop if all batch instances are empty}
    end if
    $A_t \leftarrow$ SELECTFEATURE$(\mathcal{T}_t \setminus O_t)$
    $O_{t+1} \leftarrow O_t \vee \text{OneHot}(A_t)$
  end for
end while
for $i = 1$ to B **do**
  $\widetilde{\mathbf{b}}^{(i)} \leftarrow \{j \in [D] : \mu_{i,j} \geq 0\} \cup \{o_{\text{init}}\}$
end for
return $(\theta, \mu), \widetilde{\mathcal{B}}$

Training Objective The relaxed feature-selection mask is computed as $\overline{s} \triangleq \widetilde{\mathcal{M}}^\top \widetilde{z}$ (with hard counterpart $s = M^\top z$ utilizing hard templates M, where $M_{i,j} = \mathbb{I}\{\mu_{i,j} \geq 0 \text{ or } j = o_{\text{init}}\}$). Thus, we optimize a continuous relaxation of the negative cumulative reward objective (eq. 1) in terms of *both the actor and the parameterized templates*:

$$\mathcal{L}_{\text{actor}}(\theta, \mu; x, y, \mathbf{o}) = \ell\left(\widehat{y}_\phi\left(x \odot \mathbf{o}^+, \mathbf{o}^+\right), y\right) + \lambda \textstyle\sum_{j=1}^{D} c(j)\left[\overline{s}_j - \mathbf{o}_j\right]_+ \tag{7}$$

where $\widehat{y}_\phi$ is a (differentiable w.r.t. inputs) predictor, $\mathbf{o}^+ = \mathbf{o} + \overline{s} \odot (1 - \mathbf{o})$ represents the updated observed features after following the selected template, and $[\overline{s}_j - \mathbf{o}_j]_+ \triangleq \max(\overline{s}_j - \mathbf{o}_j, 0)$ is introduced to capture the cost of newly acquired features.

Training Procedure Let U be the batch size. We denote the batch tensors $X = [x^{(u)}]_{u=1}^{U}$, $Y = [y^{(u)}]_{u=1}^{U}$, and the observations at step t be $O_t = [o_t^{(u)}]_{u=1}^{U} \in \{0,1\}^{U \times D}$. We initialize O_0 to all zeros except for the initial feature, i.e., the o_{init}-th column of O_0 is set to one. See Alg. 2 for pseudocode. We perform a rollout of at most D steps: at each time step t, we sample hard/soft templates $(Z_t, \widetilde{Z}_t)$ via Gumbel-Softmax, update (θ, μ) using $\mathcal{L}_{\text{actor}}$, and advance the observation masks O_{t+1} by acquiring the selected features A_t until all instances terminate. Note that stopping is handled independently for each instance, and terminated instances remain inactive for the rest of the rollout.

3.5 Interpretable Policy and Distillation

Rolling out our policies results in dynamic subsets of acquisitions that vary from instance to instance. Here, we wish to gain further insight into *what subsets are acquired for what instances*, essentially distilling what (variable) information is important depending on instance characteristics. While scarcely explored in previous studies, such knowledge is often important in real-world problems; for example, in healthcare applications, an understanding of policy acquisitions distills variable informational importance according to different patient sub-populations (e.g., younger underweight

[3]We implement $\mathbf{o} \subseteq [D]$ as a binary mask and $x_{\mathbf{o}}$ as $x \odot \mathbf{o}$.

patients vs. older obese patients). Here, we propose a knowledge-distillation strategy to yield interpretable policies and a breakdown of possible acquisitions by rules.

Step-wise Student Decision Trees We collect state-action pairs from expert rollouts using an imitation framework (e.g., DAgger [24]). We propose to distill a TAFA policy (e.g., § 3.2 or § 3.4) into an intrinsically interpretable student policy by training a *step-wise* ensemble of decision trees. Specifically, we maintain one tree per acquisition step t, denoted as $\{\text{Tree}_t\}_{t=1}^{K}$, where $K \leq D$ is the maximum number of acquisition steps considered in the rollouts. Training a separate Tree_t per acquisition step decomposes the one difficult global decision problem into multiple simpler local sub-problems, allowing each tree to focus on the specific acquisition stage. The decision tree Tree_t trains on teacher rollout observations $(x_{\mathbf{o}}, \mathbf{o})$'s where $|\mathbf{o}| = t$ and outputs the discrete template index $v = \text{Tree}_t(x_{\mathbf{o}}, \mathbf{o}) \in \{1, ..., B\} \cup \{\varnothing\}$, indicating which template $\mathbf{b}^{(v)} \in \mathcal{B}$ to follow next.

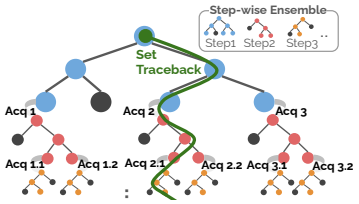


Figure 3: Step-wise Ensemble; at each step, non-terminating (colored) leaves select features and advance to the next tree level, reusing the same tree structure to build complete acquisition paths that form interpretable rules.

Tree Stitching for Subset/Rule(s) Extraction At the leaves of Tree_t, the policy shall either: 1) terminate, yielding the acquired subset with a traceback; or 2) select a template, with a corresponding feature to acquire (using the feature selection step; eq. 5). When not terminating, the policy continues by unrolling Tree_{t+1} with the updated current set $|o| = t + 1$. Thus, the ensemble's policy can be represented as a single large stitched tree (reusing the appropriate Tree_t at levels), where all leaves terminate acquisition with a specified subset.

Hence, one may collect all possible acquisitions by aggregating the unique subsets found at the leaves of the stitched tree (see Fig. 3). Moreover, for each unique subset, one can perform a traceback to collect one or more (if the acquired subset occurred at multiple leaves) rules that determine the respective acquisition set, i.e., *acquire set*$_1$: $\{x_6, x_7, x_9\}$ `if` $\text{rule}_{1.1}$: $0.0 < x_6 \leq 1.0$ `or` $\text{rule}_{1.2}$: ...; *acquire set*$_2$: $\{x_1, x_4, x_6\}$ `if` $\text{rule}_{2.1}$: $x_6 \leq 0.0 \wedge x_1 \leq 0.0$; etc. Note that sets and rules can be ordered by training set frequency for further distillation.

4 Experiments

We evaluate different variants of our proposed TAFA method against existing baselines across three key dimensions: prediction accuracy, feature acquisition efficiency, and computational runtime. Specifically, the TAFA variants we considered are as follows: 1) `tafa-knn` Gumbel templates (§ 3.4) with k-NN policy (eq. 4, § 3.2); 2) `tafa-actor` Gumbel templates with Gumbel policy (§ 3.4); 3) `tafa-interp` distilled step-wise decision tree policy (§ 3.5) with mutation-searched templates (§ 3.3). Code is available at `https://github.com/busFred/tafa-demo`.

Experiments span one synthetic dataset, `cube`-$\sigma = 0.3$ ($D = 20$) [25, 7]; six real-world benchmark datasets: the Big Five Personality questionnaire data[4] (`big5`, $D = 50$), Gas sensor home activity monitoring (`gas`, $D = 10$) [26], Electrical Grid stability data (`grid`, $D = 11$) [27], MNIST (`mnist`, $D = 784$, 28×28 pixels) [28], Fashion MNIST (`fashion`, $D = 784$, 28×28 pixels) [29], and grayscale CIFAR-10 (`cifar10`, $D = 1024$, 32×32 pixels) [30]; and the EngineDB automotive engine-fault dataset ($D = 11$) [31], used for an interpretability case study (§ H.3). We emphasize that, due to the difficulty of AFA in high-dimensional spaces, prior AFA studies typically operate in low-dimensional settings (e.g., SEFA [11] acquires among 20 prechosen MNIST features). In contrast, we retain the full-resolution MNIST setting ($D = 784$, 28×28 pixels) and evaluate acquisition over all 784 candidate features without such preselection. All datasets except EngineDB are standard AFA benchmarks used in prior work (e.g., ACO [9], SEFA [11], DIME [5], GSMRL [8]); among them, the image datasets act as high-dimensional stress tests. EngineDB [31] (§ H.3) is deliberately chosen to reflect a practical acquisition scenario, automotive troubleshooting, where diagnostic measurements are inherently costly.

Consistent with prior work, we use cross-entropy as the prediction loss and equal feature costs. The trade-off parameter λ in eq. 1 is application-specific; in practice it can be selected on a validation set

[4] `https://openpsychometrics.org/_rawdata/`

from the resulting accuracy–cost curve to meet a budget. We therefore sweep each method's trade-off parameter to trace the full trade-off (TAFA's λ grids in Table 1). Unless otherwise noted, we use a neural network as the arbitrary subset feature predictor $\widehat{y}(x_{\mathbf{o}})$. Appx. G details the architectures.

Baselines We compare against AACO [9] (non-greedy oracle approximation), SEFA [11] (stochastic latent, non-RL), DIME [5] (discriminative CMI), EDDI [4] (generative CMI), GSMRL [8] (model-based RL), and `static` (a fixed subset acquired for every instance). All baselines use their official implementations and sweep their respective cost/benefit trade-off parameters to trace accuracy vs. cost.

4.1 Benchmark Dataset Evaluations

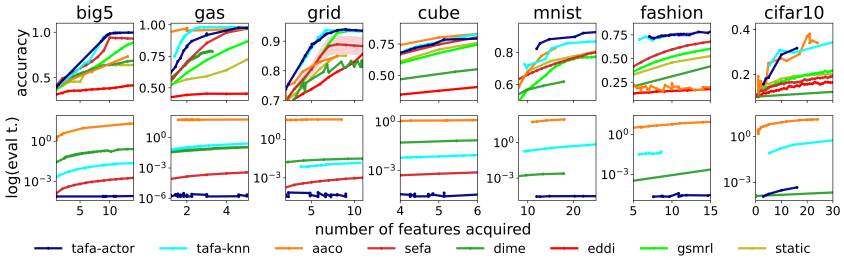


Figure 4: Acc. vs. *log-scaled* inference time vs. num. of features acquired compared to baselines. Tick marks for methods are computed by adjusting respective cost/benefit trade-off hparams.

The TAFA variants generally match or exceed the strongest baselines in accuracy while requiring fewer acquired features (Fig. 4, top; Table 2 summarizes a matched mid-range budget), with especially clear gains on `big5`, `grid`, `mnist`, and `fashion`; on `gas`, `cube` (at low budgets), and `cifar10` (where absolute accuracy is low for all methods), AACO is competitive in accuracy. At the same time, the log-scale inference-time plots (Fig. 4, bottom) show that TAFA is substantially faster than competing non-greedy methods, particularly AACO, whose online subset evaluation incurs inference costs that are orders of magnitude higher on every dataset. This speedup comes from moving computation offline: TAFA's one-time template search ranges from minutes on small tabular datasets to roughly 11–18 hours on `mnist` and `fashion`, comparable to training EDDI, GSMRL, or SEFA on high-dimensional data though higher than DIME, and requires no search at inference (§ G.3; Alg. 1 search only, excluding Alg. 2 and predictor training). Among our variants, `tafa-actor` provides the strongest computational profile, reflecting the benefit of amortized template selection, while `tafa-knn` remains highly competitive in accuracy and still provides substantial speedups over the KNN-based AACO baseline. Overall, TAFA provides the most consistent combination of acquisition performance and inference efficiency, suggesting that **the learned template abstraction provides an effective efficiency–expressivity tradeoff**: it captures useful joint feature structure without requiring exhaustive online planning over feature subsets.

4.2 TAFA Variants Ablation

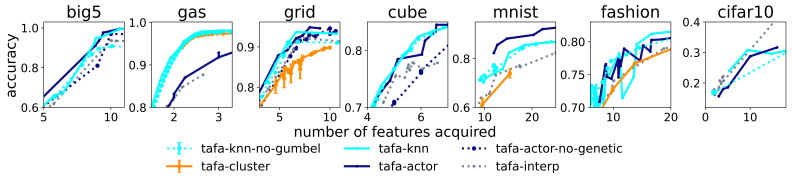


Figure 5: Accuracy vs. number of features acquired for ablation (see below for descriptions).

Here we compare different variants of TAFA and validate our design. From our results (Fig. 5), we see the following takeaways.

1. **Our continuous-relaxation optimization often helps find better templates.** We see that in several datasets (e.g., `big5`, `grid`) the performance of a policy using templates fine-tuned with the continuous relaxation (`tafa-knn`, § 3.4) improves over the same policy with the templates before the relaxation tuning (`tafa-knn-no-gumbel`).

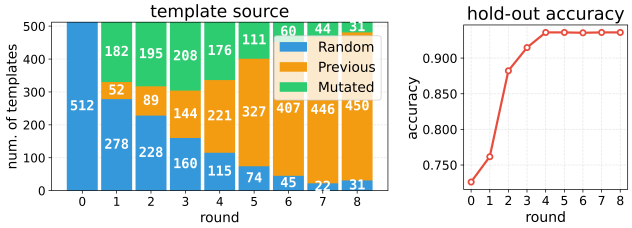


Figure 6: Template composition (left) and validation accuracy (right) over the course of mutation on `big5` dataset.

2. **Genetic optimization often provides a helpful initialization to continuous relaxation optimization of templates.** We see that in several datasets (e.g., `big5`, `grid`, `cube`), continuous relaxation optimization over randomly initialized templates (`tafa-actor-no-genetic`) is not as effective as when fine-tuning the genetic templates (`tafa-actor`, § 3.3).
3. **Our interpretable distillation retains performance.** We see relatively small degradation in accuracy/cost trade-offs utilizing our distillation techniques (`tafa-interp`, § 3.5) despite the added interpretability of simple rules (see § 4.3).
4. **Our iterative genetic approach converges rapidly.** We visualize the genetic optimization procedure (Alg. 1) in Fig. 6 for the `big5` dataset as a function of rounds. The resulting policy quickly converges in terms of average accuracy (Fig. 6, right). Moreover, by visualizing the source of the template set at each round (Fig. 6; `random`, a randomly generated template; `mutated`, a mutated template from the last round's optimized template collection; `previous` a direct template from the last round's optimized template collection), we see that our mutations are providing useful alternatives in early rounds, and the optimized set converges relatively quickly after (choosing to preserve previous templates); analogous results for MNIST are shown in § C.4.
5. **Learned templates outperform generic feature grouping.** To test whether simpler grouping suffices, `tafa-cluster` builds templates from hierarchical feature agglomeration (number of groups chosen at the elbow of the merge-distance curve), forming each template from a sampled subset of groups with one representative feature per group. Averaged over datasets at matched budgets (§ G.2), `tafa-knn` and `tafa-actor` outperform `tafa-cluster` by 7.4 and 9.8 accuracy points at a lower budget and by 15.3 and 15.1 points at a higher budget, with the largest gaps on `big5`, `cube`, and `mnist`; clustering is competitive on `gas` and `fashion`.

Observations 1 and 2 show that our optimization techniques are complementary and pair well together; observation 3 shows the effectiveness of our interpretable policies; observations 4 and 5 further validate the design of our optimization and the benefit of learned templates. Additional ablations studying the distillation strategy can be found in Appx. H.

4.3 Interpretability Analysis

Qualitative Analysis on `cube-`σ dataset We use the `cube-`$\sigma = 0.3$ dataset [25, 7], a synthetic dataset with $D = 20$ features and 8 classes where each instance contains 17 uniform noise features and 3 informative Gaussian features. For class $y = c$, only the contiguous block $(c, c+1, c+2)$ is normally distributed (class distribution in Fig. 7, Gaussian means in blue cells), while all the remaining features are uniform noise.

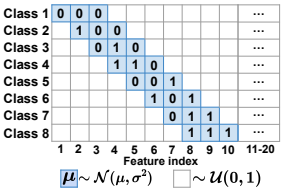


Figure 7: `cube-`σ dataset.

Fig. 8 (top left) shows an example of the extracted subsets of acquisitions and the respective rules (§ 3.5). First, we note the efficacy of our method to very succinctly distill down the behavior of the policy for nearly all instances (99.9%) into 5 distinct subsets. This is in stark contrast to previous works' black-box policies. Second, we can confirm that the learned policy is behaving reasonably based on a quick analysis of the subsets/rules. For example, `Set3` is selected if x_6 is negative ($x_6 \leq -0.019$), which rules out any class where x_6 is uniform ($y \notin \{1, 2, 3, 7, 8\}$), with the negative value indicating that a $\mathcal{N}(0, \sigma^2)$ class for x_6 is more likely ($y \in \{4, 5\}$), and correspondingly acquires x_5 to resolve the ambiguity. `Set2` is selected based on checking if x_6 is likely uniform ($-0.004 < x_6 \leq 1.043$) and x_7 is smaller ($x_7 \leq 0.492$, more likely $\mathcal{N}(0, \sigma^2)$ or potentially uniform), indicating that the class may be $y \in \{1, 2, 3, 7, 8\}$, and correspondingly acquires x_4 and

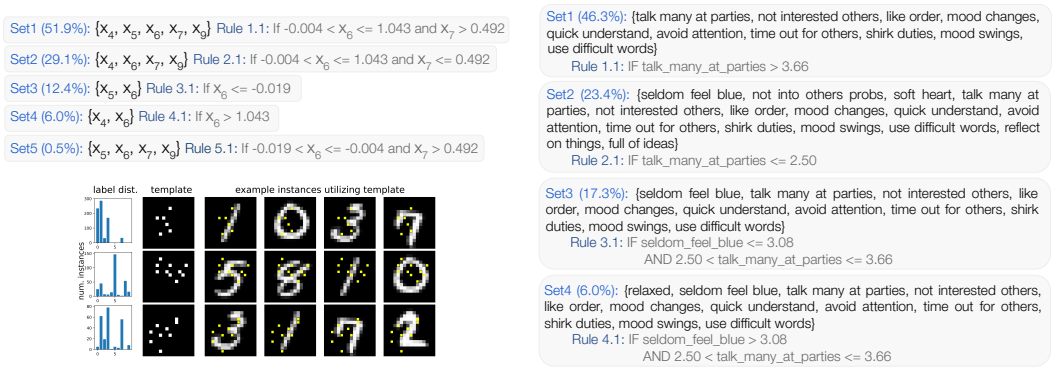


Figure 8: (Top left) Extracted rules on the `cube` dataset. (Right) Extracted rules on the `big5` dataset. (Bottom left) Visualization of feature templates found for the `mnist` dataset; yellow pixels in the "example instances utilizing templates" column represent the pixels that are acquired by the TAFA agent to classify the digits of interest.

x_9 (resolving possibilities of classes $1, 2, 3$ and $7, 8$, respectively). `Set1` corresponds to the previous case, but when x_7 is large (likely either $\mathcal{N}(1, \sigma^2)$ or potentially uniform), $y \in \{1, 2, 3, 5, 8\}$; thus, the policy additionally acquires x_5 to resolve class ambiguity.

Psychological Data Analysis In Big Five Personality questionnaire data (`big5`) instances contain 50 survey answers rated on a 5-point agreement scale. Following Valancius et al. [9], the label is a 4-class target (0-3) corresponding to Emotional Stability (ES) score quartiles. We see that our method is again able to succinctly distill a policy to a handful of rules, acquiring informative features for emotional stability (Fig. 8, right).

Visualized Templates TAFA variants provide direct interpretability through their learned templates. For MNIST, templates and their mapped instances (Fig. 8, bottom left) highlight distinctive digit curves and edges, exposing canonical acquisition subsets not provided by prior methods.

EngineDB Analysis In § H.3, we further analyze our policies on EngineDB to illustrate their use in automotive troubleshooting, a practical setting where diagnostic measurements are costly.

5 Conclusion

TAFA addresses a previously underexplored question in AFA: *what canonical feature subsets drive cost–benefit performance?* It learns a compact dictionary of informative feature templates, enabling non-greedy acquisition with a reduced policy search space and faster inference. We formulate template discovery as a submodular set optimization problem and use mutation-guided search to construct an initial dictionary. A continuous relaxation then jointly optimizes the template dictionary and acquisition policy, refining the templates to balance predictive performance and acquisition cost under the learned policy. We also develop a step-wise decision-tree distillation framework that produces interpretable template-based acquisition policies. Across synthetic and real-world benchmarks, TAFA achieves strong accuracy–cost–runtime tradeoffs.

Acknowledgments and Disclosure of Funding

This research was partly funded by National Science Foundation (NSF) grant DMS2324394 and by National Institutes of Health (NIH) grants 1R01AA02687901A1, 1OT2OD032581-02-321, and 1OT2OD038045-01. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the NIH and NSF.

References

[1] Maytal Saar-Tsechansky, Prem Melville, and Foster Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, 2009.

[2] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

[3] Arman Rahbar, Linus Aronsson, and Morteza Haghir Chehreghani. A survey on active feature acquisition strategies. *arXiv preprint arXiv:2502.11067*, 2025.

[4] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. Eddi: Efficient dynamic discovery of high-value information with partial vae. In *International Conference on Machine Learning*, pages 4234–4243. PMLR, 2019.

[5] Soham Gadgil, Ian Connick Covert, and Su-In Lee. Estimating conditional mutual information for dynamic feature selection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=0ju2Qu9jvn.

[6] Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari. 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*, pages 375–390. Springer, 2011.

[7] Hajin Shim, Sung Ju Hwang, and Eunho Yang. Joint active feature acquisition and classification with variable-size set encoding. *Advances in neural information processing systems*, 31, 2018.

[8] Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In *International conference on machine learning*, pages 6450–6459. PMLR, 2021.

[9] Michael Valancius, Maxwell Lennon, and Junier Oliva. Acquisition conditioned oracle for nongreedy active feature acquisition. In *International Conference on Machine Learning*, pages 48957–48975. PMLR, 2024.

[10] Zihan Ding and Hao Dong. Challenges of reinforcement learning. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pages 249–272, 2020.

[11] Alexander Norcliffe, Changhee Lee, Fergus Imrie, Mihaela Van Der Schaar, and Pietro Liò. Stochastic encodings for active feature acquisition. *arXiv preprint arXiv:2508.01957*, 2025.

[12] Osman Berke Guney, Ketan Suhaas Saichandran, Karim Elzokm, Ziming Zhang, and Vijaya B Kolachalama. Active feature acquisition via explainability-driven ranking. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 20748–20765. PMLR, 2025. URL https://proceedings.mlr.press/v267/guney25a.html.

[13] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[14] Andrew Silva, Taylor Killian, Ivan Jimenez, Sung-Hyun Son, and Matthew Gombolay. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, pages 1855–1865. PMLR, 2020.

[15] Sascha Marton, Tim Grams, Florian Vogt, Stefan Lüdtke, Christian Bartelt, and Heiner Stuckenschmidt. Mitigating information loss in tree-based reinforcement learning via direct optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978. doi: 10.1007/BF01588971.

[17] Matt Kusner, Wenlin Chen, Quan Zhou, Zhixiang (Eddie) Xu, Kilian Weinberger, and Yixin Chen. Feature-cost sensitive learning with submodular trees of classifiers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), June 2014. ISSN 2159-5399. doi: 10.1609/aaai.v28i1.8967.

[18] Alan Miller. *Subset selection in regression*. chapman and hall/CRC, 2002.

[19] Andreas Krause and Daniel Golovin. Submodular function maximization. In Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli, editors, *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014. doi: 10.1017/CBO9781139177801.004.

[20] Alan M Frieze. A cost function property for plant location problems. *Mathematical Programming*, 7(1):245–248, 1974.

[21] John H Holland. Genetic algorithms and the optimal allocation of trials. *SIAM journal on computing*, 2(2):88–105, 1973.

[22] Stephanie Forrest. Genetic algorithms. *ACM computing surveys (CSUR)*, 28(1):77–80, 1996.

[23] SN Sivanandam and SN Deepa. *Genetic algorithms*. Springer, 2008.

[24] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[25] Thomas Rückstieß, Christian Osendorfer, and Patrick van der Smagt. Minimizing data consumption with sequential online feature selection. *International Journal of Machine Learning and Cybernetics*, 4(3):235–243, April 2012. ISSN 1868-808X. doi: 10.1007/s13042-012-0092-x.

[26] Flavia Huerta and Ramon Huerta. Gas sensors for home activity monitoring. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C5BS4F.

[27] Vadim Arzamasov. Electrical Grid Stability Simulated Data . UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C5PG66.

[28] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[29] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

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

[31] Mary Vergara, Leo Ramos, Néstor Diego Rivera-Campoverde, and Francklin Rivas-Echeverría. Enginefaultdb: A novel dataset for automotive engine fault classification and baseline results. *IEEE Access*, 11:126155–126171, 2023. ISSN 2169-3536. doi: 10.1109/access.2023.3331316.

[32] Aaron M Roth, Nicholay Topin, Pooyan Jamshidi, and Manuela Veloso. Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy. *arXiv preprint arXiv:1907.01180*, 2019.

[33] Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Machine Learning*, 113(8):5847–5890, 2024.

[34] Saher Esmeir and Shaul Markovitch. Anytime induction of cost-sensitive trees. *Advances in Neural Information Processing Systems*, 20, 2007.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Appendices and Supplemental Materials

A Limitation Statement

While TAFA improves the interpretability of acquisition decisions, integrating these interpretable acquisitions with an interpretable downstream classifier remains an important direction for future work. This would further extend transparency across the full acquisition-to-prediction pipeline. We further note the following limitations.

- **Scope of the theory.** The $(1 - 1/e)$ guarantee (Appx. E) applies only to plain greedy selection under the 0-1 loss and uniform acquisition cost; it does not cover the genetic search (Alg. 1) with cross-entropy loss used in our experiments.
- **Dependence on the predictor.** Template quality depends on the calibration of the arbitrary-subset predictor on partial inputs and, for `tafa-knn`, on the quality of the nearest-neighbor loss estimate.
- **Offline cost and fixed templates.** Templates are learned offline with a non-trivial one-time cost and are fixed at inference time, which may make the method brittle under test-time distribution shift.
- **Data requirements.** Learning templates that cover diverse instance types requires sufficient labeled training data, which may be scarce in some medical or industrial settings.
- **Scope of evaluation.** Our evaluation follows established AFA benchmarks; validation on large-scale, naturally costed domains (e.g., clinical settings) remains future work.

B Initial Feature Identification

After the first round ($r = 0$) of Alg. 1 returns the template bank $\widehat{\mathcal{B}}^{(0)}$, an initial feature $o_{\text{init}} \in [D]$ is identified and pinned into every template, i.e. $\mathbf{b} \leftarrow \mathbf{b} \cup \{o_{\text{init}}\}$ for all $\mathbf{b} \in \widehat{\mathcal{B}}^{(0)}$; all subsequent rounds then keep o_{init} in every candidate. Below, $\widehat{\mathcal{B}}$ denotes the template bank at the time of identification. This post-hoc identification decouples the combinatorial template search from the choice of o_{init}, avoiding the need to pre-specify a starting feature before search. We describe three identification procedures below. Unless otherwise specified, TRYCOSTTHENIDENTIFY is used in all experiments.

B.1 Frequency-Based Identification (COUNTBASED)

Let $\widehat{\mathcal{B}} = \{\mathbf{b}^{(i)}\}_{i=1}^{B}$ denote the template bank and let $\widehat{\mathcal{B}}(\cdot)$ denote its associated precomputed oracle rewards over the training set $\mathcal{D}_{\text{trn}}$. For each instance $(x^{(n)}, y^{(n)}) \in \mathcal{D}_{\text{trn}}$, define the best-matching template index

$$i^{(n)} = \underset{i \in [B]}{\arg\max} \; -e\Big(x^{(n)}_{\mathbf{b}^{(i)}},\, y^{(n)}\Big),$$

where e is the negative cumulative reward objective (eq. 1). Each instance is thus associated with the template that yields the highest oracle reward. The feature frequencies are then accumulated over these best-matching templates:

$$f_d = \sum_{n=1}^{N} \mathbb{1}\Big[d \in \mathbf{b}^{(i^{(n)})}\Big], \quad d \in [D].$$

The initial feature is selected as the most frequently occurring feature across all best-matching templates:

$$o_{\text{init}} = \underset{d \in [D]}{\arg\max} \; f_d.$$

Intuitively, this selects the feature that is most commonly present in the template that best serves each training instance, making it a universally useful first observation.

B.2 Template-Marginal Frequency (FROMTEMPLATES)

This procedure simplifies the count-based approach by ignoring per-instance template assignments and instead counting feature occurrences across the entire template bank:

$$f_d = \sum_{i=1}^{B} \mathbb{1}\left[d \in \mathbf{b}^{(i)}\right], \quad d \in [D].$$

The initial feature is again the most frequent:

$$o_{\text{init}} = \arg\max_{d \in [D]} \; f_d.$$

Unlike COUNTBASED, this procedure does not require precomputed oracle rewards and operates directly on the binary template masks, making it the lightest of the three methods. It implicitly favors features that are broadly included across many templates, reflecting structural importance across the template library.

B.3 Cost-Aware Exhaustive Search (TRYCOSTTHENIDENTIFY)

This procedure directly optimizes the oracle reward criterion over all candidate initial features. For each $d \in [D]$, construct a modified template bank in which feature d is pinned into every template:

$$\mathbf{b}_d^{(i)} = \mathbf{b}^{(i)} \cup \{d\}, \quad i \in [B],$$

and evaluate the expected best-template oracle reward under this modification:

$$\bar{R}(d) = \frac{1}{N} \sum_{n=1}^{N} \max_{i \in [B]} \; -e\left(x^{(n)}_{\mathbf{b}_d^{(i)}},\, y^{(n)}\right).$$

The initial feature is the candidate that maximises this expected reward:

$$o_{\text{init}} = \arg\max_{d \in [D]} \; \bar{R}(d).$$

Because this requires D separate evaluations of the full oracle reward over $\mathcal{D}_{\text{trn}}$, it is the most expensive of the three procedures; however, it directly measures the marginal improvement each candidate feature provides when added to the template bank, making it the most principled choice. Our ablation (§ C.6) confirms that all three strategies yield comparable performance, so the choice of identification procedure is not a critical hyperparameter.

C Additional Ablations

C.1 Sensitivity to Initialization

Alg. 1 employs three complementary mechanisms to guard against poor initialization: stochastic mutation of high-performing templates, elitist retention of the previous round's best solutions, and random injection of fresh candidates at every round. To empirically confirm that these mechanisms confer robustness, we repeated the full template search with five independent random seeds and report the resulting accuracy–cost tradeoff curves. The variance across seeds is uniformly low across all reported datasets (Fig. 9), confirming that Alg. 1 does not require careful initialization to achieve strong performance.

C.2 Gumbel-Softmax Temperature Ablation

`tafa-actor` employs a temperature annealing schedule that decays τ from 2.0 to 0.1 over the course of training. The elevated initial temperature produces smoother relaxed template weights, while gradual decay makes the relaxation increasingly discrete. Fig. 10 compares this schedule against fixed-temperature baselines at $\tau \in \{0.1, 0.5, 1.0, 2.0\}$. The annealing schedule consistently outperforms all static alternatives, suggesting that the dynamic schedule resolves the fundamental tension between exploration quality and decision sharpness that any fixed temperature must compromise on.

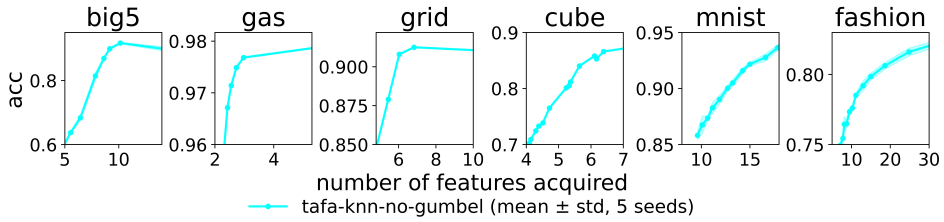


Figure 9: Sensitivity to initialization. Accuracy vs. number of features acquired by `tafa-knn-no-gumbel` (mutation-searched templates, no Gumbel refinement) for template search repeated with five independent random seeds; each point is the mean over seeds at a fixed cost trade-off λ and the shaded band shows $\pm$ one standard deviation across seeds. The bands are barely visible, indicating that the resulting accuracy–cost curves are stable across seeds. Axis ranges are zoomed to the informative region.

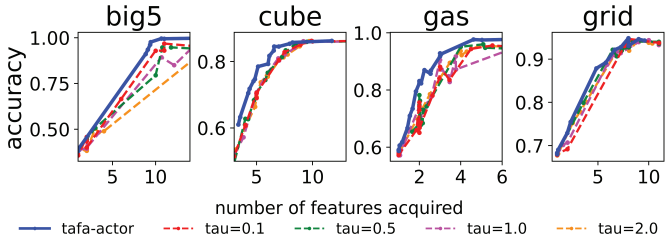


Figure 10: Gumbel-Softmax temperature ablation comparing the annealing schedule ($\tau : 2.0 \to 0.1$) against fixed-temperature baselines $\tau \in \{0.1, 0.5, 1.0, 2.0\}$. The annealing schedule consistently outperforms all static baselines.

C.3 Template Repository Size Ablation

The template repository size B is a tunable capacity parameter analogous to model width in neural networks: larger collections provide greater expressivity at the cost of increased storage and retrieval overhead. Fig. 11 shows accuracy vs. the number of features acquired for several values of B across datasets. Performance improves rapidly for small B and saturates beyond a modest threshold (size $B = 500$ already suffices for `mnist` and `fashion`), consistent with the diminishing-returns property implied by the submodular structure of the template selection objective (Thm. 3.1). This confirms that TAFA does not require a large template bank to perform well.

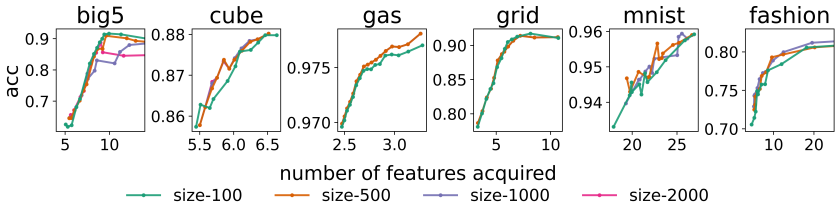


Figure 11: Accuracy vs. the number of features acquired with various template repository sizes B. Performance saturates beyond a modest threshold, demonstrating that TAFA is not sensitive to the choice of B once the repository is sufficiently expressive.

C.4 Mutation Rounds on MNIST

Fig. 6 in the main paper shows the mutation round analysis for the `big5` dataset. Here we report analogous results for `mnist` in Fig. 12 and Fig. 13, confirming that the same trends hold across dimensionally diverse datasets: convergence is rapid, with accuracy plateauing at about 0.90 after round 3, and the template composition shifts from random sampling to mutation and then retention

(random templates fall from 512/512 in round 0 to 34/512 by round 8, while retained templates grow to 439/512).

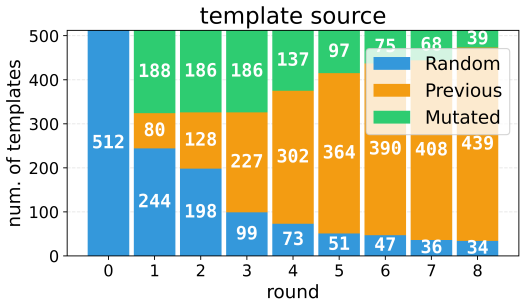


Figure 12: Template composition per mutation round on `mnist` (512 templates per round). Bars show the number of templates that are newly sampled at random (blue), retained from the previous round (orange), or produced by mutation (green). The share of random templates drops from 100% in round 0 to 34/512 in round 8, while retained templates grow to 439/512, mirroring the trend observed for `big5` in Fig. 6.

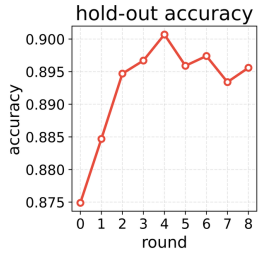


Figure 13: Validation accuracy vs. mutation rounds on `mnist`. Performance converges within a small number of rounds, mirroring the trend observed for `big5` in Fig. 6.

C.5 Template Switching Behavior at Inference

Although each instance is assigned to a single template at each acquisition step, the policy is free to *switch* its target template at subsequent steps as new feature values are observed. Fig. 14 tracks the average number of template transitions per instance as a function of the number of features acquired, across datasets. The policy does switch templates in practice, with switching frequency increasing at higher acquisition budgets, confirming that the `tafa-actor` policy utilizes the full dynamic capacity of the template bank rather than committing to a single template per instance.

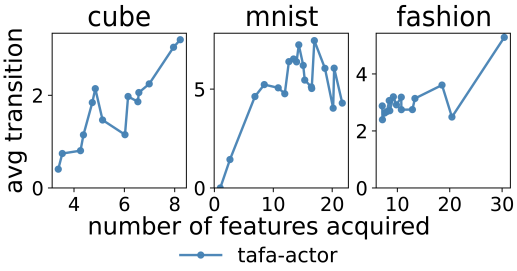


Figure 14: Average number of template transitions per instance vs. number of features acquired at inference time for `tafa-actor`. The policy actively switches templates as acquisition progresses, demonstrating non-trivially dynamic behavior.

C.6 Cold-Start Strategy Ablation

As described in Appx. B, three post-hoc strategies are available for identifying o_{init} from the round-0 template bank: CountBased, FromTemplates, and TryCostThenIdentify. Init is a fourth, non-post-hoc baseline in which o_{init} is fixed before the search by a simple feature-selection procedure, as in ACO [9]. Fig. 15 compares the accuracy–cost tradeoffs obtained under each strategy across datasets. Performance is largely insensitive to the choice of strategy, supporting the conclusion that o_{init} identification is not a critical hyperparameter for TAFA.

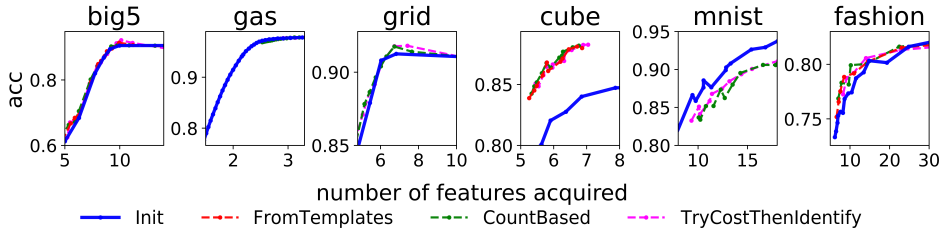


Figure 15: Accuracy vs. number of features acquired under the three post-hoc initial feature identification strategies (CountBased, FromTemplates, TryCostThenIdentify) and a fixed-in-advance baseline (Init, ACO-style feature selection). Performance is robust to the choice of strategy across datasets.

D Submodularity of Template Set Optimization

Before showing the submodular property for our proposed template search problem, we first restate the "facility location assignment" problem described in Krause and Golovin [19]. Further details regarding submodularity and its applications can be found in Krause and Golovin [19].

Facility Location Suppose we wish to select some locations out of a set $\mathcal{V} \equiv [V]$ to open up facilities in order to serve a collection of M customers. If we open up a facility at location v, then it provides a service of value $R_{m,v}$ to customer $m \in \mathcal{M} \equiv [M]$, where $R \in \mathbb{R}^{M \times V}$. If each customer chooses the facility with the highest value, the total value provided to all customers can be modeled with the following set function:

$$f(\widetilde{\mathcal{V}}) \coloneqq \sum_{m \in \mathcal{M}} \max_{\widetilde{v} \in \widetilde{\mathcal{V}}} R_{m,\widetilde{v}}, \qquad f(\varnothing) \coloneqq 0 \tag{D.1}$$

where $\widetilde{\mathcal{V}} \subseteq \mathcal{V}$ is a potential set of locations to open up facilities. If $R_{i,j} \geq 0,\ \forall i, j$, then $f(\cdot)$ is a monotone submodular function [20].

Theorem D.1. *The negated template collection objective $-g(\mathcal{B})$, with g defined in eq. 2, is a submodular set function of $\mathcal{B} \subseteq \mathcal{V} \coloneqq \mathfrak{P}([D])$, with the convention $g(\varnothing) \coloneqq \mathbb{E}_{x,y\sim\mathcal{P}}[\max_{\boldsymbol{b}\in\mathcal{V}} e(x_{\boldsymbol{b}}, y)]$; equivalently, g is supermodular.*

Proof. To prove that our template set optimization has a submodular property, we first rewrite our minimization problem as a maximization one, which is trivial, as maximization and minimization problems differ by a negative sign. That is, we define a maximization objective function $h : 2^{\mathcal{V}} \to \mathbb{R}$, where $\mathcal{V} = \mathfrak{P}([D])$ is the finite universe of candidate templates (so $|\mathcal{V}| = 2^D$), such that $\min_{\mathcal{B}\in\mathfrak{S}_B} g(\mathcal{B}) \Leftrightarrow \max_{\mathcal{B}\in\mathfrak{S}_B} h(\mathcal{B})$. As we do not have knowledge about the underlying data generation process, we replace the expectation with its empirical counterpart and ignore the constant normalization factor term in computing the average.

$$h(\mathcal{B}) = \sum_{x^{(n)},y^{(n)}\in\mathcal{D}} \left[\max_{\mathbf{b}\in\mathcal{B}} -l\left(\widehat{y}\left(x_{\mathbf{b}}^{(n)}\right), y^{(n)}\right) - \lambda \sum_{b\in\mathbf{b}} c(b) \right], \qquad \mathcal{B} \neq \varnothing,$$

where symbols g and $\mathfrak{S}_B$ are defined in eq. 2 and eq. 3, respectively, so that $h(\mathcal{B}) = -|\mathcal{D}|\, g(\mathcal{B})$ on nonempty collections (with the expectation replaced by its empirical counterpart); the value $h(\varnothing)$ is

fixed below. The strategy for proving that h has the submodular property is to show, through algebraic manipulation, that the rewritten version of h has the same form as eq. D.1.

$$\begin{aligned} h(\mathcal{B}) &= \sum_{x^{(n)},y^{(n)}\in\mathcal{D}} \max_{\mathbf{b}^{(b)}\in\mathcal{B}} -l\left(\widehat{y}\left(x^{(n)}_{\mathbf{b}^{(b)}}\right), y^{(n)}\right) - \lambda \sum_{j\in\mathbf{b}^{(b)}} c(j) \\ &\overset{\text{eq. 1}}{=} \sum_{x^{(n)},y^{(n)}\in\mathcal{D}} \max_{\mathbf{b}^{(b)}\in\mathcal{B}} -e\left(x^{(n)}_{\mathbf{b}^{(b)}}, y^{(n)}\right) \\ &= \sum_{n\in\mathcal{D}} \max_{b\in\mathcal{B}} -H_{n,b} \end{aligned} \tag{D.2}$$

where $H \in \mathbb{R}^{|\mathcal{D}|\times|\mathcal{V}|}$ is a matrix over the universe $\mathcal{V} = \mathfrak{P}([D])$ of candidate templates and $H_{n,b} = e\left(x^{(n)}_{\mathbf{b}^{(b)}}, y^{(n)}\right)$ denotes the n^{th} row and b^{th} column of H.[5] Thus eq. D.2 has the facility-location form of eq. D.1 with $R_{n,b} = -H_{n,b}$.

Two points remain. First, submodularity on nonempty collections holds for any real-valued R: for nonempty $\mathcal{S} \subseteq \mathcal{T} \subseteq \mathcal{V}$ and $v \in \mathcal{V} \setminus \mathcal{T}$, each summand satisfies

$$\begin{aligned} \max_{b\in\mathcal{S}\cup\{v\}} R_{n,b} - \max_{b\in\mathcal{S}} R_{n,b} &= \left[R_{n,v} - \max_{b\in\mathcal{S}} R_{n,b}\right]_+ \\ &\geq \left[R_{n,v} - \max_{b\in\mathcal{T}} R_{n,b}\right]_+ \\ &= \max_{b\in\mathcal{T}\cup\{v\}} R_{n,b} - \max_{b\in\mathcal{T}} R_{n,b}, \end{aligned}$$

because $\max_{b\in\mathcal{S}} R_{n,b} \leq \max_{b\in\mathcal{T}} R_{n,b}$; summing over n gives diminishing returns for h. Second, the convention $h(\varnothing) = 0$ in eq. D.1 is consistent with this inequality at $\mathcal{S} = \varnothing$ when $R \geq 0$, but fails in general for losses such as cross-entropy, for which $R_{n,b} \leq 0$ (then $h(\{v\}) - h(\varnothing) < 0 \leq h(\mathcal{T}\cup\{v\}) - h(\mathcal{T})$). We therefore set $\rho_n \coloneqq \min_{b\in\mathcal{V}} R_{n,b}$ (finite since $\mathcal{V}$ is finite) and define $h(\varnothing) \coloneqq \sum_n \rho_n$; equivalently, we shift each row to $R'_{n,b} = R_{n,b} - \rho_n \geq 0$ and take $h(\varnothing) = 0$, which is eq. D.1 with nonnegative entries and is therefore monotone submodular on all of $2^{\mathcal{V}}$ [20]. The shift adds the constant $-\sum_n \rho_n$ to h on every nonempty collection, so it leaves eq. 3, the greedy criterion eq. 6, and Alg. 1 unchanged. Finally, the per-instance inequality above holds pointwise in (x, y), so taking expectations under $\mathcal{P}$ (with $\rho(x, y) = \min_{\mathbf{b}\in\mathcal{V}} -e(x_{\mathbf{b}}, y)$, which gives the convention for $g(\varnothing)$ in the theorem statement) yields the same conclusion for the population objective g. This completes the proof. □

E Theoretical Guarantee of Template Greedy Search

In this section, we show that a special choice of task objective and cost function makes eq. D.2 a nonnegative monotone submodular set function without any row shift, so that the $(1 - 1/e)$ guarantee holds for the objective itself. Nemhauser et al. [16] prove a celebrated result that the greedy algorithm provides a good approximation to the optimal solution to an NP-hard optimization problem, which we restate below

Theorem E.1. *(Nemhauser et al. [16]; general m as stated in Krause and Golovin [19]) Fix a nonnegative monotone submodular function $f : 2^{\mathcal{V}} \to \mathbb{R}_+$ and let $\{\mathcal{S}_i\}_{i\geq 0}$ be the greedily selected sets, initialized at $\mathcal{S}_0 = \varnothing$, where $\mathcal{S}_i = \mathcal{S}_{i-1} \cup \{\arg\max_{u\in\mathcal{V}} \Delta(u|\mathcal{S}_{i-1})\}$ and Δ is the discrete derivative. Then for all positive integers k and m,*

$$f(\mathcal{S}_m) \geq \left(1 - e^{-m/k}\right) \max_{\mathcal{S}:|\mathcal{S}|\leq k} f(\mathcal{S})$$

In particular, for $m = k$, $f(\mathcal{S}_k) \geq (1 - 1/e) \max_{|\mathcal{S}|\leq k} f(\mathcal{S})$.

As previously noted in Appx. D, for the facility location outline in eq. D.1, showing $R_{i,j} \geq 0, \forall i, j$ implies that $f(\cdot)$ is a monotone submodular function [20]. Here, we note that in a special case the rewards $-H_{n,b}$ are nonnegative without any shift, so Thm. E.1 applies to h directly. Suppose we are

[5] We abuse the notation in the following proof and assume that a set is ordered and the n^{th} element of a set $\mathcal{D}$ is denoted as $n \in \mathcal{D}$.

Table 1: TAFA Method Experimental Configurations

Dataset	Architecture	Hidden Units	Activation	λ range (low, high, step)	k-neighs.
big5	2-layer FCN	[256, 256]	ReLU	0.0, 0.31, 0.02	10
cube	Linear Layer	—	—	0.0, 0.31, 0.02	10
gas	2-layer FCN	[256, 256]	ReLU	0.0, 0.31, 0.02	10
grid	2-layer FCN	[128, 128]	ReLU	0.0, 1.161, 0.02	10
mnist	3-layer FCN	[512, 512, 512]	ReLU	0.0, 0.10, 0.01	100
fashion	2-layer FCN	[512, 512]	ReLU	0.05, 0.0751, 0.0005	100
cifar10	VGG-11 CNN	[4096, 4096]	ReLU	0.0, 0.15, 0.05	100

designing a policy to solve a classification task; if we choose a λ-shifted negative zero-one loss as our task loss function, i.e. $l\left(\widehat{y}\left(x_{\mathbf{b}}\right), y\right) = -\mathbb{1}\left(\widehat{y}\left(x_{\mathbf{b}}\right) = y\right) - \lambda$ and a constant function of $\frac{1}{D}$ for all features as the acquisition cost $c(\cdot)$, then $-H_{n,b} = \mathbb{1}\left(\widehat{y}\left(x^{(n)}_{\mathbf{b}^{(b)}}\right) = y^{(n)}\right) + \lambda - \lambda \sum_{j \in \mathbf{b}^{(b)}} \frac{1}{D} \geq 0$ for any $\lambda \in \mathbb{R}_+$ (since $0 \leq |\mathbf{b}^{(b)}| \leq D$), which makes this special case a nonnegative monotone submodular function that enjoys the theoretical approximation guarantee specified in Thm. E.1. Here no row shift ρ_n (Appx. D) is needed, so the $(1 - 1/e)$ bound is relative to the objective with the λ-shifted loss. We acknowledge that for many practical choices of task loss and acquisition cost, $-H$ has negative entries, so the guarantee holds only relative to the shifted objective $h - \sum_n \rho_n$ and can be loose; yet, our experiments show that the greedy method could still provide decent performance.

F Lower Bound on the AFA MDP Value Function

Let $V(x_{\mathbf{o}})$ denote the optimal AFA MDP value using predictor $\widehat{y}$, and there is no cardinality restriction on further acquisitions. Define the collection of all additional acquisition sets and the collection of template completions, respectively, as

$$\begin{aligned} \mathcal{A}_{\mathbf{o}} &\coloneqq \mathfrak{P}([D] \setminus \mathbf{o}), \\ \mathcal{B}_{\mathbf{o}} &\coloneqq \{\mathbf{b} \setminus \mathbf{o} : \mathbf{b} \in \mathcal{B}\} \subseteq \mathcal{A}_{\mathbf{o}}. \end{aligned}$$

By the argument of Thm. C.1 of Valancius et al. [9], followed by restricting the maximization to template completions, we obtain

$$\begin{aligned} V(x_{\mathbf{o}}) &\geq \mathrm{ACO}(x_{\mathbf{o}}) \\ &= \max_{\mathbf{u} \in \mathcal{A}_{\mathbf{o}}} \left\{ -\mathbb{E}_{y, x_{\mathbf{u}} | x_{\mathbf{o}}} \left[l\left(\widehat{y}(x_{\mathbf{o} \cup \mathbf{u}}), y\right) \right] - \lambda \sum_{j \in \mathbf{u}} c(j) \right\} \\ &\geq \max_{\mathbf{u} \in \mathcal{B}_{\mathbf{o}}} \left\{ -\mathbb{E}_{y, x_{\mathbf{u}} | x_{\mathbf{o}}} \left[l\left(\widehat{y}(x_{\mathbf{o} \cup \mathbf{u}}), y\right) \right] - \lambda \sum_{j \in \mathbf{u}} c(j) \right\}. \end{aligned}$$

The final inequality follows from $\mathcal{B}_{\mathbf{o}} \subseteq \mathcal{A}_{\mathbf{o}}$. Since $\mathbf{o} \cup (\mathbf{b} \setminus \mathbf{o}) = \mathbf{o} \cup \mathbf{b}$, the last expression is the negative of the minimized TAFA cost criterion in eq. 4, evaluated using exact conditional expected losses.

G Experimental Setup

All experiments were conducted on a Lenovo ThinkStation P520 workstation equipped with an Intel Xeon W-2295 18-core processor running at 3.0 GHz (4.8 GHz boost) and 512 GB of DDR4 ECC memory. The system runs Ubuntu 24.04 LTS with Python 3.13.2.

All algorithms were implemented using single-threaded execution to ensure fair comparison across methods. The datasets used in our experiments are publicly available and were preprocessed according to standard practices in the literature. Both the neural network architecture and the trade-off parameters used in our experiments are shown in Table 1; we use the AdamW optimizer with learning rate 10^{-3}. For all baselines, we use the official implementation in their respective GitHub repositories.

Table 2: Accuracy at a matched mid-range budget and inference-time speedup over AACO.

Dataset	#feat.	`tafa-actor`	`tafa-knn`	Best baseline	Speedup vs. AACO
`big5`	7.5	$\mathbf{0.830 \pm 0.004}$	0.786	0.705 ± 0.002 (sefa)	1.5×10^5
`gas`	3.1	0.927	**0.981**	0.968 (aaco)	2.8×10^7
`grid`	6.5	0.919 ± 0.001	**0.932**	0.869 ± 0.002 (gsmrl)	7.1×10^5
`cube`	5.0	0.767	0.767	**0.802** (aaco)	3.5×10^4
`mnist`	16.5	**0.887**	0.837	0.755 (static)	2.8×10^6
`fashion`	10.0	$\mathbf{0.751 \pm 0.012}$	—	0.596 ± 0.010 (sefa)	2.3×10^5
`cifar10`	15.0	**0.311**	0.297	0.278 (aaco)	2.5×10^4

Mean $\pm$ standard error over independent runs; no $\pm$: single run. Speedup: ratio of mean seconds/sample of AACO to tafa-actor; "—": no operating point at that budget. Bold: best accuracy per row. Full tables in § G.1.

All timing measurements exclude data loading and preprocessing time, focusing solely on the algorithm execution time. Memory usage was monitored throughout the experiments to ensure no memory constraints affected the results.

For the results of `tafa-interp` in Fig. 5, we train an ensemble of trees with each tree's leaf limit of 4 (big5), 4 (cube), 16 (gas), 8 (grid), 8 (mnist), 8 (fashion), and 8 (cifar10). For the DDT and SYMPOL baselines, we use the official implementations provided in the respective papers' GitHub repositories. To adapt DDT and SYMPOL for AFA tasks, we define the goal of the RL as minimizing the predictive loss while minimizing the acquisition cost: $\text{reward} = -l\left(\widehat{y}\left(x_{\mathbf{o}}\right), y\right) - \lambda \sum_{u \in \mathbf{o}} c(u)$, following Li and Oliva [8]. For consistent results, DDT, SYMPOL, and `tafa-interp` are trained with the same classifier $\widehat{y}$ and fixed λ. To quantify the tree's policy interpretability, we follow the common practice in RL interpretable literature by using reward as the metric and leaf count as a proxy for rule complexity, as fewer leaves would imply shorter paths and simpler rules [32, 33].

G.1 Tabulated Results

Tables 3–9 give the numbers underlying Fig. 4. Each run contributes one value per budget (its own accuracy/time curve linearly interpolated at the budget); we report the mean and standard error over runs.

Table 3: Tabulated results on `big5`.

Accuracy at #features =					
Method (n)	1.0	4.2	7.5	10.8	14.0
tafa-actor (n=1–4)	0.394 ± 0.002	$\mathbf{0.611 \pm 0.004}$	$\mathbf{0.830 \pm 0.004}$	0.992 ± 0.002	**0.997**
tafa-knn (n=1)	0.389	0.559	0.786	**0.995**	0.993
aaco (n=1)	0.382	0.499	0.587	0.675	—
sefa (n=4)	0.410 ± 0.008	0.570 ± 0.001	0.705 ± 0.002	0.938 ± 0.007	0.931 ± 0.008
gsmrl (n=4)	0.369 ± 0.003	0.507 ± 0.007	0.619 ± 0.004	0.767 ± 0.012	0.888 ± 0.010
dime (n=1)	0.379	0.519	0.598	0.650	0.688
eddi (n=4)	0.317 ± 0.006	0.358 ± 0.008	0.373 ± 0.010	0.390 ± 0.009	0.415 ± 0.012
static (n=1)	**0.424**	0.457	0.595	0.639	0.639
Inference time (s/sample) at #features =					
Method (n)	1.0	4.2	7.5	10.8	14.0
tafa-actor (n=2)	$(7.81 \pm 0.07) \times 10^{-5}$	$(7.92 \pm 0.12) \times 10^{-5}$	$(8.04 \pm 0.18) \times 10^{-5}$	$(8.07 \pm 0.11) \times 10^{-5}$	$(8.27 \pm 0.22) \times 10^{-5}$
tafa-knn (n=1)	2.17×10^{-3}	7.61×10^{-3}	1.40×10^{-2}	2.05×10^{-2}	2.76×10^{-2}
aaco (n=1)	1.40×10^{0}	7.24×10^{0}	1.22×10^{1}	1.68×10^{1}	2.09×10^{1}
sefa (n=1)	1.33×10^{-4}	5.77×10^{-4}	1.03×10^{-3}	1.47×10^{-3}	1.92×10^{-3}
dime (n=1)	3.26×10^{-2}	1.02×10^{-1}	1.64×10^{-1}	2.04×10^{-1}	2.61×10^{-1}

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

G.2 Clustering-Based Templates

`tafa-cluster` replaces the learned templates with templates built from classical feature clustering, and keeps everything else (kNN-based cost estimation over templates, acquisition procedure, classifier) unchanged. Templates are built from the training features only, as follows.

Table 4: Tabulated results on `gas`.

Accuracy at #features =					
Method (n)	1.2	2.2	3.1	4.1	5.1
tafa-actor (n=1)	0.623	0.863	0.927	0.960	0.975
tafa-knn (n=1)	—	0.952	**0.981**	**0.981**	**0.981**
aaco (n=1)	**0.921**	**0.967**	0.968	0.969	0.970
sefa (n=1)	0.555	0.735	0.863	0.941	0.968
gsmrl (n=1)	0.520	0.640	0.728	0.807	0.870
dime (n=1)	0.577	0.725	0.789	—	—
eddi (n=1)	0.426	0.446	0.454	0.453	0.454
static (n=1)	0.528	0.572	0.596	0.649	0.728
Inference time (s/sample) at #features =					
Method (n)	1.2	2.2	3.1	4.1	5.1
tafa-actor (n=3)	$(2.04 \pm 0.21) \times 10^{-6}$	$(1.97 \pm 0.31) \times 10^{-6}$	$(2.05 \pm 0.15) \times 10^{-6}$	$(1.93 \pm 0.20) \times 10^{-6}$	$(2.06 \pm 0.17) \times 10^{-6}$
tafa-knn (n=1)	—	1.16×10^{-1}	1.68×10^{-1}	2.23×10^{-1}	2.79×10^{-1}
aaco (n=1)	—	5.92×10^{1}	5.79×10^{1}	5.89×10^{1}	5.99×10^{1}
sefa (n=1)	8.91×10^{-5}	1.62×10^{-4}	2.30×10^{-4}	3.04×10^{-4}	3.77×10^{-4}
dime (n=1)	3.89×10^{-2}	5.84×10^{-2}	7.53×10^{-2}	9.81×10^{-2}	1.14×10^{-1}

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

Table 5: Tabulated results on `grid`.

Accuracy at #features =					
Method (n)	2.0	4.2	6.5	8.8	11.0
tafa-actor (n=1–4)	$\mathbf{0.737 \pm 0.004}$	$\mathbf{0.847 \pm 0.005}$	0.919 ± 0.001	$\mathbf{0.939 \pm 0.003}$	0.935
tafa-knn (n=1)	—	0.838	**0.932**	0.935	—
aaco (n=1)	0.720	0.801	0.835	—	—
sefa (n=4)	0.695 ± 0.001	0.777 ± 0.011	0.865 ± 0.022	0.888 ± 0.023	0.883 ± 0.023
gsmrl (n=4)	0.701 ± 0.002	0.773 ± 0.001	0.869 ± 0.002	0.932 ± 0.001	$\mathbf{0.936 \pm 0.000}$
dime (n=4)	0.681 ± 0.006	0.749 ± 0.007	0.797 ± 0.006	0.816 ± 0.005	0.821 ± 0.004
eddi (n=1)	0.677	0.726	0.767	0.809	0.857
static (n=1)	0.695	0.751	0.819	0.851	—
Inference time (s/sample) at #features =					
Method (n)	2.0	4.2	6.5	8.8	11.0
tafa-actor (n=3)	$(4.61 \pm 0.54) \times 10^{-5}$	$(5.06 \pm 0.89) \times 10^{-5}$	$(4.88 \pm 0.72) \times 10^{-5}$	$(4.38 \pm 0.58) \times 10^{-5}$	—
tafa-knn (n=1)	—	7.48×10^{-3}	1.04×10^{-2}	1.31×10^{-2}	—
aaco (n=1)	3.15×10^{1}	3.42×10^{1}	3.49×10^{1}	—	—
sefa (n=1)	1.82×10^{-4}	3.88×10^{-4}	5.98×10^{-4}	8.27×10^{-4}	1.03×10^{-3}
dime (n=1)	1.65×10^{-2}	2.24×10^{-2}	2.67×10^{-2}	2.99×10^{-2}	3.19×10^{-2}

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

1. **Grouping.** Features (not samples) are clustered with agglomerative hierarchical clustering (`sklearn.cluster.FeatureAgglomeration`; default Ward linkage with Euclidean distance on the training matrix, so features with similar value profiles across samples are merged).
2. **Number of groups.** The number of groups k is chosen automatically at the elbow of the dendrogram: we fit the full tree, take the merge distance as a function of the number of clusters, and use the knee of this convex, decreasing curve (`kneed.KneeLocator`).
3. **Templates.** Each of the M templates ($M = 128$ for the tabular datasets, $M = 2048$ for MNIST) draws a number of groups uniformly from $\{1, \ldots, k\}$, selects that many distinct groups uniformly at random, and activates one uniformly random feature from each selected group (one representative per group).
4. **Initial feature.** The initial feature is fixed per dataset and forced to be active in every template, as in the learned-template variants.

Hence, `tafa-cluster` templates reflect feature redundancy (at most one feature per correlated group) but ignore the classifier and the downstream task, whereas TAFA learns templates optimized for accuracy and cost.

Table 10 compares `tafa-knn`, `tafa-actor` and `tafa-cluster` at two acquisition budgets per dataset. For each dataset, we take the range of average numbers of acquired features covered by all

Table 6: Tabulated results on `cube`.

Accuracy at #features =					
Method (n)	4.0	4.5	5.0	5.5	6.0
tafa-actor (n=1)	0.679	0.726	0.767	0.788	0.793
tafa-knn (n=1)	0.677	0.737	0.767	0.799	0.826
aaco (n=1)	**0.742**	**0.772**	**0.802**	**0.816**	**0.831**
sefa (n=1)	0.668	0.707	0.746	0.775	0.804
gsmrl (n=1)	0.596	0.637	0.678	0.711	0.745
dime (n=1)	0.468	0.490	0.511	0.531	0.549
eddi (n=1)	0.346	0.362	0.377	0.392	0.407
static (n=1)	0.611	0.659	0.707	0.733	0.758
Inference time (s/sample) at #features =					
Method (n)	4.0	4.5	5.0	5.5	6.0
tafa-actor (n=3)	$(3.11 \pm 0.35) \times 10^{-5}$	$(2.95 \pm 0.35) \times 10^{-5}$	$(3.38 \pm 0.35) \times 10^{-5}$	$(3.25 \pm 0.45) \times 10^{-5}$	$(3.38 \pm 0.35) \times 10^{-5}$
tafa-knn (n=1)	6.11×10^{-3}	6.76×10^{-3}	7.34×10^{-3}	8.17×10^{-3}	8.92×10^{-3}
aaco (n=1)	1.10×10^{0}	1.17×10^{0}	1.18×10^{0}	1.22×10^{0}	1.24×10^{0}
sefa (n=1)	5.15×10^{-4}	5.78×10^{-4}	6.41×10^{-4}	7.06×10^{-4}	7.71×10^{-4}
dime (n=1)	5.23×10^{-2}	5.74×10^{-2}	6.26×10^{-2}	6.77×10^{-2}	7.21×10^{-2}

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

Table 7: Tabulated results on `mnist`.

Accuracy at #features =					
Method (n)	8.0	12.2	16.5	20.8	25.0
tafa-actor (n=1)	—	**0.835**	**0.887**	**0.903**	**0.926**
tafa-knn (n=1)	—	0.741	0.837	0.861	0.868
aaco (n=1)	0.581	0.751	—	—	—
sefa (n=1)	**0.634**	0.689	0.734	0.770	0.799
gsmrl (n=1)	0.494	0.617	0.717	0.765	0.773
dime (n=1)	0.539	0.583	0.609	—	—
eddi (n=1)	0.184	0.207	0.230	0.248	0.272
static (n=1)	0.587	0.710	0.755	0.780	0.806
Inference time (s/sample) at #features =					
Method (n)	8.0	12.2	16.5	20.8	25.0
tafa-actor (n=1)	—	2.79×10^{-5}	2.98×10^{-5}	3.00×10^{-5}	3.00×10^{-5}
tafa-knn (n=1)	—	2.62×10^{-1}	3.90×10^{-1}	5.35×10^{-1}	7.09×10^{-1}
aaco (n=1)	—	6.08×10^{1}	8.35×10^{1}	—	—
dime (n=1)	1.48×10^{-3}	2.05×10^{-3}	2.36×10^{-3}	—	—

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

three methods, and define Cost 1 and Cost 2 as the 25th and 75th percentile points of this shared range. Each method's accuracy at a budget is read off its accuracy–budget curve by linear interpolation, giving one value per independent curve. For `tafa-cluster`, a curve is one repeat of the sweep over λ (three seeds per λ; the number of curves is shown in parentheses; mean $\pm$ standard error); `tafa-knn` and `tafa-actor` have a single run per λ and hence one curve (no standard error). The last rows average over the six datasets for which `tafa-cluster` was run; `cifar10` was not run. Averaged over datasets, `tafa-knn` (`tafa-actor`) outperforms `tafa-cluster` by 7.4 (9.8) accuracy points at Cost 1 and by 15.3 (15.1) points at Cost 2.

G.3 Training and Template-Search Cost

Table 11 reports the offline cost of each method in wall-clock hours. TAFA's cost is the template search of Alg. 1 (mutation search) only; the Gumbel refinement of Alg. 2 and predictor training are not included, so the TAFA column is a template-search time rather than a total offline cost. AACO is non-parametric and has no training stage, so its cost is paid entirely at inference. TAFA's offline cost

Table 8: Tabulated results on `fashion`.

Accuracy at #features =					
Method (n)	5.0	7.5	10.0	12.5	15.0
tafa-actor (n=1–4)	—	**0.719 ± 0.018**	**0.751 ± 0.012**	**0.780 ± 0.005**	**0.786**
tafa-knn (n=1)	—	0.695	—	—	—
aaco (n=1–4)	0.190	0.189 ± 0.009	0.206 ± 0.004	0.187 ± 0.010	0.190
sefa (n=4)	**0.416 ± 0.006**	0.521 ± 0.011	0.596 ± 0.010	0.646 ± 0.009	0.678 ± 0.007
gsmrl (n=4)	0.363 ± 0.007	0.452 ± 0.014	0.518 ± 0.016	0.565 ± 0.015	0.602 ± 0.014
dime (n=1)	0.206	0.259	0.312	0.365	0.418
eddi (n=1)	0.131	0.147	0.158	0.166	0.178
static (n=1)	0.330	0.394	0.458	0.492	0.526
Inference time (s/sample) at #features =					
Method (n)	5.0	7.5	10.0	12.5	15.0
tafa-actor (n=1)	—	—	2.62×10^{-5}	2.74×10^{-5}	2.89×10^{-5}
tafa-knn (n=1)	—	3.63×10^{-2}	—	—	—
aaco (n=1)	3.16×10^{0}	4.56×10^{0}	5.98×10^{0}	7.86×10^{0}	9.19×10^{0}
dime (n=1)	1.17×10^{-3}	1.68×10^{-3}	2.19×10^{-3}	2.70×10^{-3}	3.21×10^{-3}

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

Table 9: Tabulated results on `cifar10`.

Accuracy at #features =					
Method (n)	0.0	7.5	15.0	22.5	30.0
tafa-actor (n=1)	—	0.237	**0.311**	—	—
tafa-knn (n=1)	—	**0.276**	0.297	0.320	**0.343**
aaco (n=1)	—	0.260	0.278	**0.347**	—
sefa (n=1)	**0.101**	0.149	0.163	0.180	0.192
gsmrl (n=1)	0.100	0.156	0.182	0.203	0.217
dime (n=1)	0.100	0.105	0.111	0.116	0.122
eddi (n=1)	0.100	0.126	0.144	0.161	0.164
static (n=1)	—	0.168	0.192	0.197	0.200
Inference time (s/sample) at #features =					
Method (n)	0.0	7.5	15.0	22.5	30.0
tafa-actor (n=1)	—	2.36×10^{-4}	4.39×10^{-4}	—	—
tafa-knn (n=1)	—	1.27×10^{-1}	3.03×10^{-1}	5.56×10^{-1}	8.10×10^{-1}
aaco (n=1)	—	7.52×10^{0}	1.11×10^{1}	1.34×10^{1}	—
dime (n=1)	1.28×10^{-4}	2.35×10^{-3}	4.57×10^{-3}	6.79×10^{-3}	9.01×10^{-3}

Mean $\pm$ standard error over n independent runs; no $\pm$: single run. "—": no operating point near that budget. Bold: best mean per column. Inference time is unavailable for gsmrl, eddi, and static, and for sefa where not recorded.

is of the same order as that of the RL and generative baselines on high-dimensional data, and higher than DIME on most datasets. This cost is paid once per dataset and trade-off value, and no search is performed at inference time.

On `fashion` and `cifar10`, TAFA's search takes less time than EDDI and GSMRL (and than SEFA on `cifar10`), but more than SEFA on `fashion` and far more than DIME. On small tabular datasets, TAFA's search is longer than DIME on `gas`, `grid`, and `cube`, and shorter on `big5`.

Table 10: TAFA vs. clustering-based templates at matched budgets.

Dataset	Cost	Budget	`tafa-knn`	`tafa-actor`	`tafa-cluster`
`big5`	1	3.65	0.532 (1)	0.565 (1)	0.508 ± 0.011 (3)
	2	8.94	0.950 (1)	0.920 (1)	0.562 ± 0.006 (3)
`gas`	1	2.83	0.980 (1)	0.911 (1)	0.972 ± 0.000 (3)
	2	4.15	0.981 (1)	0.961 (1)	0.975 ± 0.000 (3)
`grid`	1	4.93	0.872 (1)	0.880 (1)	0.816 ± 0.005 (6)
	2	7.64	0.936 (1)	0.938 (1)	0.877 ± 0.002 (6)
`cube`	1	4.18	0.703 (1)	0.697 (1)	0.381 ± 0.002 (3)
	2	4.77	0.753 (1)	0.743 (1)	0.396 ± 0.006 (3)
`mnist`	1	13.03	0.750 (1)	0.861 (1)	0.688 ± 0.011 (3)
	2	15.21	0.825 (1)	0.883 (1)	0.743 (1)
`fashion`	1	11.98	0.720 (1)	0.789 (1)	0.749 ± 0.001 (3)
	2	20.73	0.816 (1)	0.807 (1)	0.792 ± 0.002 (3)
`cifar10`	not run (no `tafa-cluster` experiment)				
Average (6 datasets)	1	–	0.760	0.784	0.686
Average (6 datasets)	2	–	0.877	0.875	0.724

Cost 1/Cost 2: 25th/75th percentile of the feature-count range shared by the three methods (Budget column). Accuracy is linearly interpolated from each curve at that budget; mean $\pm$ SE over repeat curves (number of curves in parentheses). Average rows: unweighted mean over datasets.

Table 11: Offline training and template-search cost (hours).

Dataset	TAFA search	DIME	EDDI	GSMRL	SEFA
`big5`	0.34 ± 0.01 (n=15)	1.35	1.13	1.14	1.55
`gas`	4.09 ± 0.22 (n=60)	2.12	–	–	–
`grid`	0.148 ± 0.003 (n=21)	0.056 ± 0.044 (n=4)	0.015	0.44	0.20
`cube`	2.564 ± 0.003 (n=13)	0.25	0.18	0.77	0.33
`mnist`	11.44 ± 0.19 (n=7)	7.66	–	–	–
`fashion`	17.91 ± 0.96 (n=16)	0.16	24.94	30.38	12.01
`cifar10`	5.26 ± 0.63 (n=11)	0.84	20.00	32.06	140.92
AACO	0 (no training; cost is incurred at inference)				

Mean $\pm$ standard error over n runs; no spread when $n = 1$ or only one recorded figure exists. All runs are single-threaded on the workstation in Appx. G. TAFA and DIME: span between the first and last TensorBoard timestamps of a run. TAFA run: one template search (Alg. 1) at one λ within the plotted range; DIME run: one training run shared across all λ. EDDI, GSMRL, SEFA: single recorded training times. Excludes the Gumbel refinement stage (Alg. 2). "–": no timing record.

H Additional Interpretability Result

H.1 Interpretable Baseline Comparisons

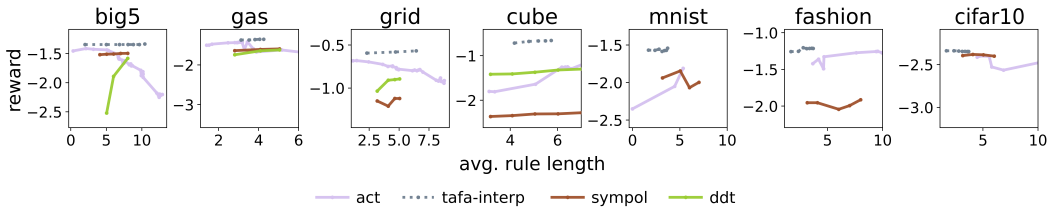


Figure 16: Reward vs. average rule length compared to baselines. Note that for visual clarity, we omit `ddt` on image datasets as its reward is below -15. A higher reward is better for comparable average rule length.

We compare our interpretable policy (`tafa-interp`) to three interpretable baselines: 1) differentiable decision tree [`ddt` 14], 2) symbolic tree-based on-policy RL [`sympol` 15], and 3) cost-sensitive tree [`act` 34]. DDT trains differentiable decision trees with PPO [35], then post-hoc discretizes them,

while SYMPOL directly optimizes the hard decision tree with PPO, avoiding discretization loss. Following prior work, we treat feature indices as actions and train with PPO to support non-greedy acquisition. For ACT, each internal tree test corresponds to acquiring that feature, and the total acquisition cost is the cost of the unique features queried along the root-to-leaf path. For consistent results, `tafa-interp`, `ddt`, and `sympol` are trained with the same classifier and fixed λ; see Appx. G for details. Across datasets, Fig. 16 shows `tafa-interp` consistently achieves the highest reward at comparable average rule length [32, 33] to `ddt` and `sympol`, indicating that `tafa-interp` can make better performance-cost-aware decisions under similar interpretability constraints compared to the baselines.

H.2 Ablative Studies for `tafa-interp`

Two variants of `tafa-interp` are trained; `tafa-interp-action-feature`: the decision tree predicts feature indices (eq. 5) instead of template indices; `tafa-interp-non-ensemble`: a single tree is shared across cardinalities. Fig. 17 shows that `tafa-interp` achieves higher reward at comparable leaf budgets than its variants, indicating that (1) acting on the template action space and (2) using step-wise trees are both beneficial for optimizing the tree policy under the reward objective.

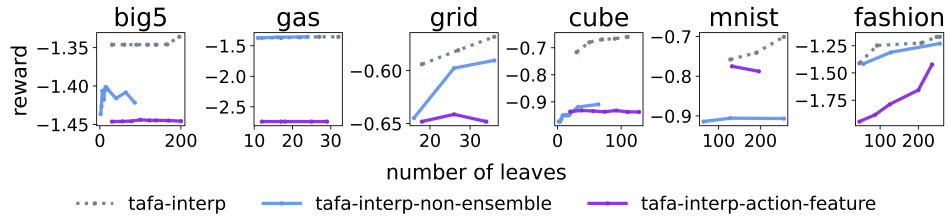


Figure 17: Reward vs. number of leaves comparing to `tafa-interp` variants. A higher reward is better for a comparable number of leaves.

H.3 Additional Analysis for `tafa-interp`

In this section, to showcase TAFA's applicability in interpretability-critical domains, we demonstrate how one can apply the TAFA framework to make intelligent choices of feature subsets to diagnose car engines recorded in the EngineDB dataset [31]. We use decision trees as predictors and employ only iterative mutate template search Alg. 1, forgoing continuous refinement Alg. 2. This configuration is appropriate for three reasons: First, automotive diagnostics often requires fully transparent decision logic for regulatory compliance and technician understanding. Decision trees provide this interpretability but are non-differentiable, precluding gradient-based template optimization. Second, the EngineDB dataset [31] (D=11 features, structured sensor measurements) has lower dimensionality and stronger domain structure than the high-dimensional problems (e.g., `fashion`, $D = 784$) where continuous refinement provides substantial gains (Fig. 5). Our ablation studies suggest genetic search alone suffices for such structured, moderate-dimensional spaces. Third, the quality of the discovered templates validates this choice: as analyzed below, the templates acquire diagnostically meaningful feature combinations (e.g., RPM, Speed, CO, O2 for low-speed conditions) that align with automotive engineering principles, without requiring gradient optimization. The genetic search remains effective when the problem structure is well-defined, as we show below.

Here, the task for our TAFA agent is to sequentially gather information to diagnose a car engine that could be in one of four states: `normal` engine, running `lean`/`rich`, or having an ignition `voltage` issue. See Vergara et al. [31] for further details. We choose average fuel `consumption` (L/100km) as the initial feature o_{init} since car owners often track their gas mileage and would notice it when one starts spending more on gasoline. We provide a brief background on failure modes recorded in the EngineDB dataset [31]; further details can be found in Vergara et al. [31].

- An engine running `rich` means it receives too much fuel relative to the amount of air in the combustion chamber, resulting in incomplete combustion that produces high carbon monoxide (CO) emissions, reduced fuel efficiency, and potential engine damage from fuel dilution of engine oil.

- An engine running `lean` occurs when there's too little fuel relative to air, causing higher combustion temperatures that can lead to engine knocking, overheating, and potential damage to pistons and valves, while also producing distinctive oxygen signatures in exhaust gases
- A `voltage` issue in the ignition circuit means the spark plugs receive insufficient electrical power to properly ignite the fuel-air mixture, resulting in misfires, rough idling, reduced power output, and poor fuel economy due to incomplete or delayed combustion

Below, we analyze two acquisition sets—set 4 and set 5—from Fig. 18 that handle mild to slightly elevated fuel consumption (6.69-7.13 L/100km), which can be challenging to diagnose. Additionally, Fig. 18 shows the interpretable rules extracted from our TAFA policy, whereas Fig. 19 and Fig. 20 show the downstream interpretable predictor that acts on the features selected with our TAFA policy.

Acquire Set 4 (Low Speed, < 41.61 kph) Cases assigned to use this set of rules are being operated in a low-speed regime and are experiencing average or slightly low fuel efficiency; for these cases, our policy decides to acquire `RPM`, `Speed`, `CO`, and `O2` in addition to the initial feature (fuel `consumption`). From Fig. 18, we see that the majority of cases that get assigned to use this set of features are either `normal` or `rich`, and there are very few cases being `lean` or `voltage`. As previously observed in Vergara et al. [31], rich-running engines often have a very distinctive pattern in terms of chemicals released from the emission, which naturally explains our policy's decision to acquire both `CO` and `O2` at the same time. Besides the obvious choice, we think our policy's decision to acquire `RPM` is especially interesting. Recall that cases assigned to this rule are all in the low-speed regime, i.e., below 41.61 kph, suggesting the vehicle is highly likely to be stuck in city stop-and-go traffic where big throttle input is often applied to get the vehicle up to speed as the traffic light turns green. In this scenario, the engine revs up as the vehicle accelerates, meaning more gasoline is being burned and the tachometer reading (measured in `RPM`) is going to increase. In this case, a car having above-average fuel consumption could be due to the condition it was operated in, not because of any malfunction. From the visualization (Fig. 19) of our predictor using the chosen set of features, one can see that, among the three leaves that classify an engine as `normal`, the ancestors of two of the three leaves use `RPM` as one of the features, suggesting the importance of `RPM` in distinguishing `normal` running engine, and further verifying the intelligence of our TAFA policy.

Acquire Set 5 (High Speed, > 41.61 kph) In contrast to set 4, all cases assigned to this set are in the high-speed regime with average or slightly low fuel efficiency; regarding the features acquired, in addition to all the features in Set 4, this set additionally acquires rotational `force` (measuring the fuel-to-power conversion efficiency) and short-term fuel `rate` (in L/hr). Our policy's decision to acquire these two additional features, again, makes sense: at highway speeds, healthy engines generate strong force while faulty engines produce weak power despite having similar or slightly above average fuel consumption. Here, inspecting the predictor (Fig. 20), we again see both `force` and `rate` play an essential role in correctly diagnosing car engines, especially `force`; in the middle branch of Fig. 20, `force` is the second feature that the decision tree classifier splits on, showing the importance of `force` in diagnosing vehicles in the high-speed regime experiencing slightly above average fuel consumption.

Both acquisition strategies demonstrate TAFA's ability to learn interpretable diagnostic strategies aligned with automotive engineering principles. We again stress that all the patterns observed above were learned from data, and none of the prior knowledge described above (with the exception of choosing `consumption` as initial feature) was known to the TAFA policy at any point during training. Despite using a non-differentiable decision tree classifier as a predictor and therefore not using gradient-based template fine-tuning, our study shows the TAFA policy is still able to identify high-quality feature templates and allow users of our framework to gain useful insights into the reasoning made by our TAFA policy. This demonstrates TAFA's modularity: practitioners can select a predictor and optimization strategy based on domain constraints (interpretability, compute resources, dimensionality) without abandoning the template-based framework. The genetic search outlined in Alg. 1 remains effective, as we just showed above with the automotive diagnosis example.

Acquire Set 1 (Freq: 53.5%)

Features: {Consumption, CO}
Rule 1.1: If 7.1335 < Consumption ≤ 13.1900 (L/100km)

Normal	Rich	Lean	Voltage
38.3%	20.4%	20.1%	21.2%

Acquire Set 2 (Freq: 28.0%)

Features: {Consumption, Speed, O2}
Rule 2.1: If Consumption ≤ 6.6905 (L/100km)

Normal	Rich	Lean	Voltage
19.1%	12.9%	36.6%	31.4%

Acquire Set 3 (Freq: 10.4%)

Features: {Consumption, HC}
Rule 3.1: If Consumption > 13.1900 (L/100km)

Normal	Rich	Lean	Voltage
0.0%	34.4%	32.3%	33.3%

Acquire Set 4 (Freq: 3.3%)

Features: {RPM, Consumption, Speed, CO, O2}
Rule 4.1: If 6.6905 < Consumption ≤ 7.1335 (L/100km) and Speed ≤ 41.6110 (kph)

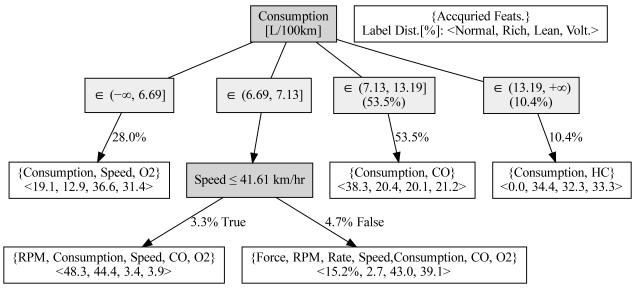
Normal	Rich	Lean	Voltage
48.3%	44.4%	3.4%	3.9%

Acquire Set 5 (Freq: 4.7%)

Features: {Force, RPM, Rate, Consumption, Speed, CO, O2}
Rule 5.1: If 6.6905 < Consumption ≤ 7.1335 (L/100km) and Speed > 41.6110 (kph)

Normal	Rich	Lean	Voltage
15.2%	2.7%	43.0%	39.1%

(a) Text-based representation of the extracted rules.



(b) Tree-based representation of the extracted rules, meaning this figure represents the same information as that of Fig. 18a.

Figure 18: Engine fault diagnosis rules extracted by TAFA in both text format and in tree format.

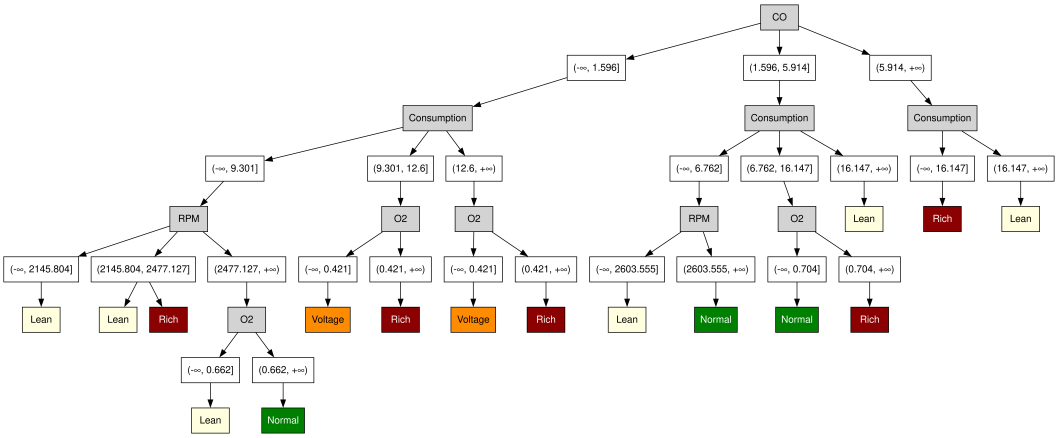


Figure 19: The decision tree classifier using set 4.

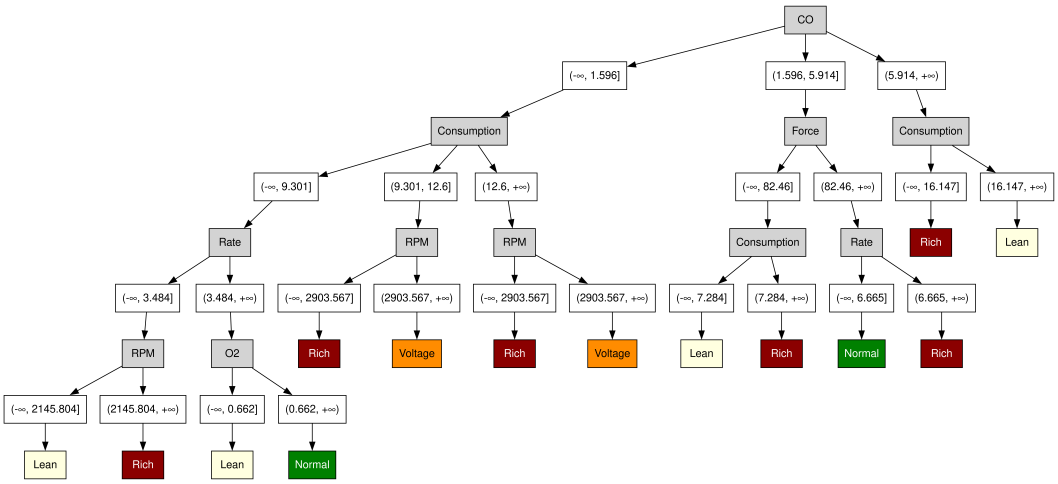


Figure 20: The decision tree classifier using set 5.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state five contributions—template-based non-greedy AFA, mutation-greedy search with Gumbel refinement, theoretical guarantees (submodularity and value lower bound), interpretable decision-tree distillation, and extensive experiments—and each is substantiated in the corresponding sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations can be found in Appx. A. The section discusses the scope of the theory (the $(1 - 1/e)$ guarantee covers only plain greedy under 0-1 loss and uniform cost, not the genetic search used in experiments), dependence on the arbitrary-subset predictor's calibration on partial inputs, the one-time offline cost and fixed templates at inference (potentially brittle under test-time distribution shift), data requirements for learning diverse templates, and that evaluation is limited to established AFA benchmarks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theorems are numbered and cross-referenced. Thm. 3.1 (submodularity) and Thm. 3.2 (value lower bound) have proof sketches in the main paper with full proofs in Appx. D–Appx. F.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Both algorithms are described with full pseudocode (Alg. 1, Alg. 2); all datasets are publicly available; neural network architectures, hyperparameters, and hardware details are reported in the Experimental Setup appendix (Appx. G). Code is available at `https://github.com/busFred/tafa-demo`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is available at `https://github.com/busFred/tafa-demo`. All datasets are publicly available and cited in § 4.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation details, including neural network architectures, trade-off hyperparameters λ, optimizer, and hardware, are fully specified in Appx. G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The tabulated results (§ G.1) report mean ± standard error over independent runs for every method and budget where more than one run is available; entries from a single run are marked as such, since running multiple seeds for all baselines across all datasets is computationally costly. Robustness to initialization is additionally validated via a 5-seed ablation in § C.1, with ± one standard deviation across seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appx. G reports the full hardware specification (Lenovo ThinkStation P520, Intel Xeon W-2295 18-core at 3.0 GHz, 512 GB DDR4 ECC RAM, Ubuntu 24.04 LTS) and notes that all algorithms were run with single-threaded execution.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses only publicly available datasets, involves no human subjects, and presents foundational ML research with no foreseeable direct negative applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational ML research on efficient feature acquisition; there is no direct path to negative societal applications and no specific societal impact that needs to be highlighted.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new datasets or high-risk models; it uses only publicly available benchmark datasets and releases an algorithm implementation.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and baseline implementations are cited with their original papers. Baseline code is used via official GitHub repositories as acknowledged in Appx. G.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Code for TAFA is publicly available at `https://github.com/busFred/tafa-demo` (with a README covering setup, dependencies, and usage). No new datasets are introduced.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as part of the core methodology; any LLM usage was limited to writing and editing assistance only.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.