EVALUATE BEFORE YOU COMMIT: LABEL-FREE CANDIDATE-UPDATE UTILITY FOR TEST-TIME ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time adaptation (TTA) can improve performance under distribution shift, yet deciding *where* to adapt does not determine whether the resulting parameter update is worth retaining. We formulate *candidate-update utility*: after a base adaptation operator produces a tentative state, should that realized update enter persistent model state? We introduce a label-free evaluate-before-commit controller that allocates candidate-generation compute using forward-only screening, evaluates each tentative update on independently sampled probe views using held-out predictive consistency and drift from a frozen pretrained reference, and transactionally rolls back low-utility candidates. Across CLIP and DINOv2 under synthetic corruption, natural domain shift, and a cross-hospital stress test, the proposed utility estimator improves harmful-update AUROC from 0.589 to 0.750 and AUPRC from 0.305 to 0.492 relative to predictive entropy. At the principal operating point, Utility-Aware control reduces the harmful committed-update rate (HCUR) from 27.0% to 11.0% while increasing the accuracy gain from $+2.20$ to $+2.40$ percentage points, halving candidate-update backward passes, and reducing adaptation-specific wall-clock time by $28\pm3\%$. Post-proposal verification also complements strong pre-update selection: adding Utility-Aware verification after Cross-Augmentation Similarity (CAS) reduces HCUR from 15.2% to 8.2%, while the proposed score achieves next-window harmful-update AUROC of 0.738 versus 0.641 for CAS. These results support pre-update selection and post-proposal candidate commitment as distinct and complementary control levels for TTA.

1 INTRODUCTION

Test-time adaptation (TTA) allows a pretrained model to update itself from unlabeled observations encountered after deployment, providing a practical mechanism for responding to distribution shift without labeled target-domain retraining (Wang et al., 2021; Niu et al., 2022; 2023). This flexibility, however, creates a stateful risk: once an adaptation update is retained, it changes the model from which subsequent predictions and updates are produced. Aggregate gains from TTA can therefore coexist with individual updates that degrade predictive performance.

Existing selective TTA methods primarily reason about the observation *before* adaptation. Predictive uncertainty, confidence, redundancy, augmentation agreement, or sample reliability can help decide whether an incoming observation should contribute to adaptation (Niu et al., 2022; 2023; Jiang et al., 2026b). These signals address an important question: *where should adaptation be attempted?* They do not directly determine whether the particular parameter update subsequently produced by the adaptation operator is beneficial. An uncertain prediction can produce a useful update, a negligible one, or a harmful one.

We study this complementary post-proposal decision. Let θ_t denote the persistent model state before processing target window B_t, and let a base adaptation operator construct a tentative state $\tilde{\theta}_t = \mathcal{U}(\theta_t; B_t)$. Rather than committing $\tilde{\theta}_t$ automatically, we ask whether the realized transition $\theta_t \to \tilde{\theta}_t$ should enter persistent state. For retrospective evaluation, we define its immediate oracle utility as $G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t)$, where R_t is labeled predictive risk available only offline. Thus, $G_t > 0$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

denotes a beneficial candidate and $G_t < 0$ a harmful one. The deployment problem is to obtain useful label-free evidence about this unavailable utility before persistence.

We address this problem with an evaluate-before-commit controller that separates adaptation into four stages: *screen–propose–evaluate–commit/rollback*. A causal forward-only signal first allocates a prescribed candidate-generation budget. Selected windows produce tentative updates using one set of semantic-preserving views, while independently sampled probe views withheld from optimization evaluate the realized candidate. The resulting candidate-update utility score rewards improved held-out predictive consistency while penalizing additional drift from a frozen pretrained reference. Candidates are committed only when their estimated utility exceeds a conservative margin; otherwise, both model and optimizer states are restored. Pre-update screening therefore determines *where adaptation compute is spent*, whereas post-proposal utility determines *whether the realized update persists*.

Experiments across CLIP and DINOv2 under synthetic corruption, natural domain shift, and a cross-hospital stress test support this separation. The proposed utility score improves harmful-update AUROC from 0.589 for predictive entropy to 0.750 and AUPRC from 0.305 to 0.492. At the principal operating point, Utility-Aware control reduces the harmful committed-update rate (HCUR) from 27.0% under Always Adapt to 11.0% while increasing the accuracy gain from $+2.20$ to $+2.40$ percentage points and reducing candidate-update backward passes by 50%. Post-proposal verification remains useful after strong pre-update selection: adding it after Cross-Augmentation Similarity (CAS) (Jiang et al., 2026b) reduces HCUR from 15.2% to 8.2%. The proposed utility score also reaches next-window harmful-update AUROC 0.738, compared with 0.641 for CAS, indicating that candidate-level information is not confined to the immediate evaluation window.

Our contributions are threefold:

- **Candidate-update utility as a distinct TTA control problem.** We separate deciding whether an observation should trigger adaptation from deciding whether the particular state transition produced by adaptation should persist.
- **A label-free evaluate-before-commit controller.** We combine causal compute allocation, independently sampled probe-view evaluation, frozen-pretrained anchoring, conservative commitment, and transactional restoration of model and optimizer state around a base adaptation operator.
- **Mechanism-matched evidence for post-proposal verification.** Candidate-update utility predicts harmful updates better than predictive uncertainty, remains informative for next-window effects, and further reduces harmful commitments after CAS preselection while preserving predictive performance.

2 Related Work

Test-time adaptation. TTA updates a pretrained model using unlabeled observations available during inference. TENT (Wang et al., 2021) performs entropy minimization over normalization parameters. EATA (Niu et al., 2022) filters unreliable and redundant samples while regularizing adaptation to reduce forgetting, whereas SAR (Niu et al., 2023) combines reliable-sample selection with sharpness-aware optimization and recovery. Recent vision-language TTA methods have explored test-time prompt optimization (Shu et al., 2022), dynamic adapters (Karmanov et al., 2024), retrieved visual knowledge (Lee et al., 2025), distributional estimation (Han et al., 2025), contrastive objectives (Lafon et al., 2025), and loss-landscape-guided adaptation (Li et al., 2026). These methods primarily differ in how adaptation is initiated or how an adapted prediction or state is constructed. Our work is complementary: we wrap a base adaptation operator and ask whether the *realized candidate update* should be allowed to modify persistent model state.

Selective adaptation and recovery. The closest conceptual connection is selective adaptation. CAS (Jiang et al., 2026b) decides whether an incoming test sample should undergo adaptation using agreement across augmented predictions. This is a pre-update, sample-level decision. We instead condition on an instantiated candidate $\tilde{\theta}_t = \mathcal{U}(\theta_t; B_t)$ and evaluate whether that particular post-proposal state transition should persist. Our matched-budget CAS experiments directly test whether candidate-level verification provides information beyond strong pre-update selection.

Related continual-TTA methods address accumulated adaptation error through recovery. Adaptive and Selective Reset (ASR) (Lim et al., 2026) determines when and where an adapted model should be reset, while DO-ALL (Jang et al., 2026) uses compact source anchors and harm-adaptive blending to move unstable parameter groups toward the source model. We therefore do not claim rollback itself as novel. Our distinction lies in the decision timing and object: each realized transition $\theta_t \to \tilde{\theta}_t$ is treated as a tentative candidate, evaluated before entering persistent state, and, if rejected, both model and optimizer states are transactionally restored. Thus, our contribution is candidate-level pre-persistence verification of a concrete adaptation outcome.

Reliability under realistic TTA. Recent studies have shown that reduced entropy or increased confidence after adaptation need not imply improved correctness and have broadened TTA evaluation toward calibration, robustness, stability, and realistic online behavior (Zanella et al., 2025; Sheng et al., 2025; Jiang et al., 2026a). These findings motivate our separation between predictive uncertainty, which characterizes the current prediction, and candidate-update utility, which characterizes the consequence of a specific parameter transition. Accordingly, the frozen pretrained model in our utility estimator is used only as a conservative reference for candidate-induced drift, not as a correctness oracle.

3 Utility-Aware Test-Time Adaptation

We consider an unlabeled test stream of windows $\{B_t\}_{t=1}^{T}$, where $B_t = \{x_{t,i}\}_{i=1}^{n_t}$, and let θ_t denote the persistent model state before processing window t. Our controller wraps a base adaptation operator $\mathcal{U}$ and separates adaptation into four stages: *screen–propose–evaluate–commit/rollback*, as illustrated in Fig. 1. Screening determines where candidate-generation compute is spent; evaluation determines whether the realized candidate should modify persistent state.

3.1 Candidate-Update Control

Let $z_t \in \{0, 1\}$ indicate whether window t is allocated candidate-generation compute. For a selected window, independently sampled update views V_t^{upd} are used by the base adaptation operator to construct the tentative state

$$\tilde{\theta}_t = \mathcal{U}\left(\theta_t; V_t^{\mathrm{upd}}\right). \tag{1}$$

The candidate is therefore the realized state transition $\theta_t \to \tilde{\theta}_t$, rather than the input window itself.

Let $a_t \in \{0, 1\}$ denote the subsequent commitment decision, with $a_t \le z_t$. Persistent state evolves according to

$$\theta_{t+1} = \begin{cases} \tilde{\theta}_t, & a_t = 1, \\ \theta_t, & a_t = 0. \end{cases} \tag{2}$$

For retrospective evaluation, the immediate oracle utility of a candidate is $G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t)$, where R_t is labeled predictive risk available only offline. Thus, $G_t > 0$ denotes a beneficial candidate and $G_t < 0$ a harmful one. Our deployment objective is to construct a label-free score $\widehat{G}_t$ whose sign and ordering are informative about G_t before the candidate is allowed to persist. This is a stage-wise decision problem: we estimate the immediate value of the current candidate rather than claiming globally optimal control over the future test trajectory.

3.2 Forward-Only Candidate Allocation

Candidate construction is more expensive than ordinary inference in our setting, so we first allocate a prescribed candidate-generation budget using a forward-only statistic. For a K-class predictor $p_{\theta_t}(x) = \mathrm{softmax}(f_{\theta_t}(x))$, we use normalized predictive entropy

$$s_t = \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{H(p_{\theta_t}(x_{t,i}))}{\log K}, \qquad H(p) = -\sum_{k=1}^{K} p_k \log p_k. \tag{3}$$

For target candidate fraction ρ, the forward-only screening proposal is

$$\tilde{z}_t = \mathbb{I}[s_t \ge \kappa_{\rho,t}], \tag{4}$$

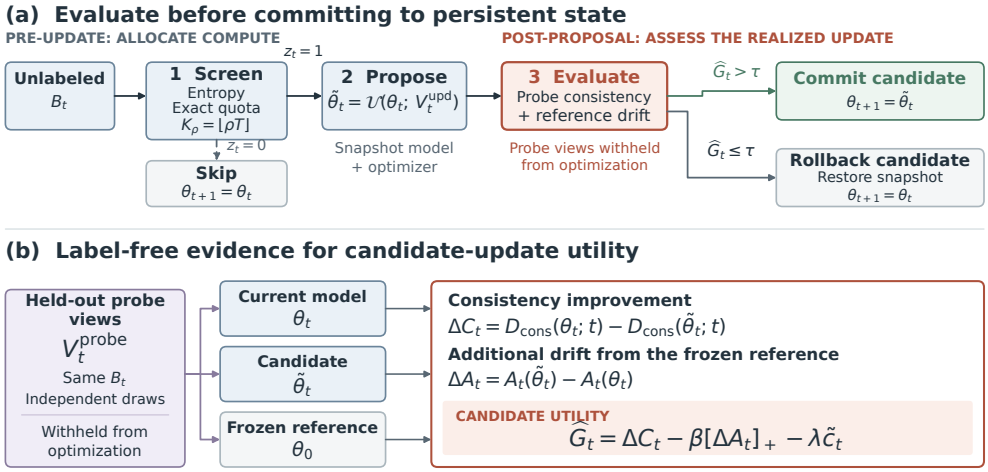

Figure 1: **Overview of Utility-Aware TTA.** (a) Forward-only screening allocates candidate-generation compute under an exact causal quota; tentative updates are evaluated before persistence, with rejected candidates restoring model and optimizer state. (b) Label-free candidate utility combines held-out probe-view consistency improvement with additional drift from a frozen pretrained reference.

where $\kappa_{\rho,t}$ is determined causally from screening information available before the current decision. In our finite-stream experiments, a remaining-quota correction maps this proposal to the final allocation $z_t \in \{0, 1\}$ and guarantees exactly $K_\rho = \lfloor \rho T \rfloor$ candidate windows. Neither target labels nor oracle utilities are used for allocation. Importantly, s_t determines only *where candidate-generation compute is spent*; it is not treated as evidence that the candidate subsequently produced by $\mathcal{U}$ will be beneficial. The exact causal quota rule is provided in the appendix.

3.3 Post-Proposal Candidate Evaluation

As illustrated in Fig. 1(b), candidate construction and candidate evaluation use separate augmentation draws. Update views V_t^{upd} are used exclusively by $\mathcal{U}$, whereas independently sampled probe views V_t^{probe} of the same underlying observations are withheld from optimization and used only to evaluate the realized candidate.

Held-out consistency. For observation $x_{t,i}$, let $\{v_{t,i,j}^{\text{probe}}\}_{j=1}^{M}$ denote its probe views and define their mean predictive distribution

$$\bar{p}_{\theta,t,i} = \frac{1}{M} \sum_{j=1}^{M} p_\theta\left(v_{t,i,j}^{\text{probe}}\right). \tag{5}$$

We measure disagreement across probe predictions using generalized Jensen–Shannon divergence,

$$D_{\text{cons}}(\theta; t) = \frac{1}{n_t} \sum_{i=1}^{n_t} \left[H(\bar{p}_{\theta,t,i}) - \frac{1}{M} \sum_{j=1}^{M} H\left(p_\theta(v_{t,i,j}^{\text{probe}})\right) \right]. \tag{6}$$

The candidate's held-out consistency gain is $\Delta C_t = D_{\text{cons}}(\theta_t; t) - D_{\text{cons}}(\tilde{\theta}_t; t)$. Thus, $\Delta C_t > 0$ means that the candidate increases predictive agreement on views that did not contribute to its optimization. Consistency alone is not a correctness guarantee, however, because an update may become more consistent around an incorrect prediction.

Frozen-pretrained anchor. To penalize unsupported candidate-induced drift, we retain the pretrained state θ_0 as a frozen reference and define

$$A_t(\theta) = \frac{1}{n_t M} \sum_{i=1}^{n_t} \sum_{j=1}^{M} D_{\text{JS}}\left(p_\theta(v_{t,i,j}^{\text{probe}}) \,\|\, p_{\theta_0}(v_{t,i,j}^{\text{probe}})\right). \tag{7}$$

The additional drift introduced by the candidate is

$$\Delta A_t = A_t(\tilde{\theta}_t) - A_t(\theta_t), \qquad [\Delta A_t]_+ = \max(0, \Delta A_t). \tag{8}$$

Only increases in disagreement with the pretrained predictor are penalized; a candidate that moves the current model toward the reference incurs no anchor penalty. The frozen pretrained model serves as a conservative reference for candidate-induced drift, not as a correctness oracle.

3.4 Candidate-Update Utility and Commitment

We combine the two post-proposal signals into the label-free risk-adjusted utility surrogate

$$\widehat{G}_t^{\mathrm{risk}} = \Delta C_t - \beta[\Delta A_t]_+, \tag{9}$$

where $\beta \geq 0$ controls the penalty on additional pretrained-model drift. Let $\tilde{c}_t \geq 0$ denote normalized incremental candidate cost. The full compute-aware score is

$$\widehat{G}_t = \Delta C_t - \beta[\Delta A_t]_+ - \lambda \tilde{c}_t, \tag{10}$$

and the deployed commitment rule is

$$a_t = z_t \, \mathbb{I}[\widehat{G}_t > \tau]. \tag{11}$$

Equivalently, a candidate is retained when $\widehat{G}_t^{\mathrm{risk}} > \tau + \lambda \tilde{c}_t$. In our primary experiments, candidate cost is constant across selected windows and normalized to $\tilde{c}_t = 1$, so $\lambda + \tau$ reduces to an effective commitment margin. We therefore do not attribute the empirical gains to heterogeneous cost-aware scheduling.

Candidate evaluation is transactional. Before constructing $\tilde{\theta}_t$, the adaptable model parameters and optimizer state are snapshotted. If the commitment rule in Eq. (11) accepts the candidate, the tentative model and optimizer states are committed; otherwise, both are restored. Computation spent on a rejected candidate is still counted, but the rejected update cannot affect subsequent adaptation through either model parameters or optimizer statistics. Full pseudocode, the exact causal quota controller, and implementation details are provided in the appendix.

4 Experimental Setup

Models, shifts, and adaptation protocol. We evaluate two pretrained vision foundation-model families. CLIP ViT-B/16 (Radford et al., 2021) is our primary backbone, while DINOv2 ViT-B/14 (Oquab et al., 2024) provides an architecturally distinct generalization test. For CLIP, prediction uses a frozen text classifier and adaptation is restricted to LayerNorm affine parameters of the vision encoder. For DINOv2, only backbone LayerNorm affine parameters are adapted and the classification head remains frozen. DINOv2 ImageNet-C experiments use the released ImageNet linear classifier, whereas Office-Home and Camelyon17-WILDS use linear heads trained from source-domain labels only, before the target stream is observed.

The base candidate generator shared by all mechanism-matched controls performs one entropy-minimization update of the LayerNorm affine parameters using AdamW with learning rate 10^{-4}. Target data are processed causally in windows of at most 64 observations, with two independently sampled update views per observation; Utility-Aware additionally uses $M = 4$ independently sampled probe views. Update and probe views are separate augmentation draws of the same underlying observations, and probe views are withheld from candidate optimization.

The primary configuration uses $\beta = 1$, effective commitment margin $m = \lambda + \tau = 0.05$, and candidate-generation fraction $\rho = 0.50$. We additionally evaluate $\rho \in \{0.10, 0.25, 0.50, 0.75, 1.00\}$. These choices are fixed before inspection of target labels from the principal evaluation benchmarks. Complete optimizer, augmentation, classifier, transactional-state, and causal quota-controller details are provided in the appendix.

We evaluate synthetic corruption on ImageNet-C (Hendrycks & Dietterich, 2019), natural domain shift on Office-Home (Venkateswara et al., 2017), and a cross-hospital distribution-shift stress test on Camelyon17-WILDS (Koh et al., 2021). ImageNet-C uses all 15 standard corruptions at severity

5, and Office-Home uses all 12 directed source-to-target transfers. Camelyon17-WILDS follows the official source/test-hospital split protocol with mild semantic-preserving transformations. CIFAR-10-C (Hendrycks & Dietterich, 2019) is used during development and controlled ablation analysis rather than as a principal benchmark supporting the main claims.

Baselines and mechanism-matched controls. We distinguish literature baselines from controls designed to isolate the candidate-commitment mechanism. TENT (Wang et al., 2021), EATA (Niu et al., 2022), SAR (Niu et al., 2023), and TPT (Shu et al., 2022), where compatible with the model family, retain their native parameterization, optimization, filtering, and reset semantics.

Mechanism-matched controls share the same base candidate generator, target stream, adaptable parameters, optimizer, learning rate, and update objective. *Always Adapt* generates and commits a candidate on every window. *Random Gate* allocates exactly $K_\rho = \lfloor \rho T \rfloor$ candidate windows without using predictive information and commits every generated candidate. *Entropy Gate* uses the same causal entropy allocation and candidate-generation budget as Utility-Aware but commits every generated candidate. *Utility-Aware* uses the same allocation rule and candidate generator but evaluates each realized candidate before persistence. An *Oracle Utility Gate*, which observes the offline oracle utility G_t, is used only as a retrospective one-step reference.

We additionally evaluate CAS (Jiang et al., 2026b) under a matched candidate-generation budget. *CAS Gate* uses CAS as its pre-update selection rule and commits every candidate generated by its trajectory, whereas *CAS+Utility* applies the proposed post-proposal commitment rule after CAS selection. The two policies use the same CAS pre-update selection rule, base candidate operator, prescribed candidate-generation budget, and evaluation seeds. This comparison tests whether post-proposal evaluation provides additional decision-relevant information after strong pre-update selection.

Evaluation metrics. We evaluate predictive performance, candidate-utility estimation, committed-update risk, and computational cost. Predictive performance is reported using classification accuracy, change in accuracy relative to the frozen model, and expected calibration error (ECE).

For each generated candidate, $h_t = \mathbb{I}[G_t < 0]$ denotes its retrospective immediate harm label. Harmful-update discrimination is measured by AUROC and AUPRC, with larger harm scores indicating greater predicted risk. Benefit-oriented utility ordering is measured by Spearman correlation between $\widehat{G}_t$ and the oracle utility G_t.

To test whether candidate-level information extends beyond the current evaluation window, we define the retrospective next-window utility

$$G_t^{\mathrm{next}} = R_{t+1}(\theta_t) - R_{t+1}(\tilde{\theta}_t), \tag{12}$$

where both states are evaluated on B_{t+1} before any adaptation to B_{t+1}. Target labels are used only offline to construct this diagnostic; G_t^{next} is never observed by the deployed controller. We report next-window harmful-update AUROC and AUPRC, Spearman correlation with G_t^{next}, and agreement between immediate and next-window harm labels.

Committed-update risk is quantified by the Harmful Committed-Update Rate (HCUR),

$$\mathrm{HCUR} = \frac{\sum_t \mathbb{I}[a_t = 1 \wedge G_t < 0]}{\sum_t \mathbb{I}[a_t = 1]}, \tag{13}$$

whenever at least one candidate is committed. Because HCUR can be reduced trivially through excessive abstention, we report it together with commitment coverage among generated candidates, $\mathrm{CommitRate} = \sum_t a_t / \sum_t z_t$. Reporting HCUR together with Commit Rate prevents a policy that commits almost no candidates from appearing artificially safe solely because of abstention.

Computational efficiency is reported using candidate-update backward passes and adaptation-specific wall-clock time. The latter includes all adaptation-specific computation spent on tentative candidates, including candidates that are subsequently rolled back.

Statistical protocol. Principal experiments use three independent random seeds, with identical target-stream realizations across mechanism-matched methods within each seed. The CAS complementarity and next-window analyses use five independent seeds. We report paired comparisons with

Table 1: Label-free identification of harmful candidate updates. AUROC and AUPRC are oriented so that larger values indicate greater predicted harm. For the proposed estimator, these metrics use $-\widehat{G}_t$ as the harm score, whereas Spearman r_s is computed between $\widehat{G}_t$ and the benefit-oriented oracle utility G_t.

Diagnostic	AUROC ↑	AUPRC ↑	Spearman r_s ↑
Predictive entropy	0.589	0.305	0.140
$1 - \max$ confidence	0.580	0.290	0.120
Reversed prediction margin	0.600	0.310	0.150
Augmentation disagreement	0.640	0.350	0.230
Frozen-anchor disagreement	0.620	0.330	0.200
Proposed $\widehat{G}_t$	**0.750**	**0.492**	**0.460**

95% confidence intervals; primary uncertainty estimates use 10,000 paired bootstrap replicates, with hierarchical resampling across shift units where applicable.

Target labels from the principal evaluation streams are isolated from screening, candidate generation, candidate-utility estimation, threshold application, and commitment decisions, and are used only retrospectively for evaluation and computation of oracle quantities.

5 Results

5.1 Candidate-Update Utility Predicts Harm Beyond Predictive Uncertainty

We test whether pre-update predictive uncertainty is sufficient to characterize the utility of the candidate update subsequently produced. Table 1 compares label-free diagnostics for identifying harmful candidate updates, defined retrospectively by $G_t < 0$. Predictive entropy is the normalized pre-update entropy score s_t in Eq. (3); $1 - \max$ confidence and reversed prediction margin are computed from the current model's pre-update predictive distribution; and augmentation disagreement and frozen-anchor disagreement are $D_{\mathrm{cons}}(\theta_t; t)$ and $A_t(\theta_t)$, respectively, as defined in Eqs. (6) and (7). These baselines characterize the pre-update predictive state, whereas $\widehat{G}_t$ evaluates the realized transition $\theta_t \to \tilde{\theta}_t$.

Predictive entropy achieves harmful-update AUROC/AUPRC of $0.589/0.305$, whereas the proposed estimator reaches $0.750/0.492$, corresponding to paired improvements of $+0.161$ AUROC (95% CI $[0.130, 0.191]$) and $+0.187$ AUPRC (95% CI $[0.145, 0.225]$). Spearman rank correlation with G_t likewise increases from $r_s = 0.140$ for predictive entropy to $r_s = 0.460$ for $\widehat{G}_t$. Thus, the realized candidate's behavior on independently sampled probe views provides information about update harm beyond that captured by pre-update predictive uncertainty.

5.2 Utility-Aware Commitment Reduces Harm Without Trading Away Predictive Performance

We next test whether candidate-level information improves online state transitions. Table 2 reports the principal CLIP ViT-B/16 ImageNet-C operating point. Always Adapt improves accuracy over the frozen model by 2.20 ± 0.12 percentage points, but $27.0 \pm 2.4\%$ of its committed updates are retrospectively harmful. Utility-Aware increases the accuracy gain to 2.40 ± 0.16 points while reducing HCUR to $11.0 \pm 1.4\%$. Relative to Always Adapt, this is a 16.0-percentage-point absolute HCUR reduction (95% CI $[13.4, 18.5]$), equivalent to a 59.3% relative reduction (95% CI $[51.3, 66.6]$). The paired accuracy difference is $+0.20$ percentage points (95% CI $[+0.09, +0.28]$).

The matched-budget controls show that reducing candidate-generation frequency alone does not account for this improvement. At $\rho = 0.50$, Random Gate and Entropy Gate obtain HCUR values of 24.1% and 18.2%, with accuracy gains of $+1.10$ and $+1.70$ points, respectively, whereas Utility-Aware reaches 11.0% HCUR and $+2.40$ points. Utility-Aware commits 62% of generated candidates, corresponding to persistent updates on approximately 31% of stream windows, so its HCUR reduction is not obtained through near-complete abstention. Calibration follows the same pattern:

Table 2: Principal operating-point results. ΔAcc and ΔECE are relative to the frozen model. Commit denotes commitment coverage among generated candidates. Random Gate, Entropy Gate, and Utility-Aware use the same $\rho = 0.50$ candidate-generation fraction. Backward-pass and adaptation-time reductions are relative to Always Adapt. Values are mean±SD over three independent seeds where applicable.

Method	ΔAcc. ↑ (pp)	ΔECE ↓ (pp)	HCUR ↓ (%)	Commit (%)	Back. red. ↑ (%)	Time red. ↑ (%)
Frozen	0.00	0.00	–	–	–	–
Always Adapt	$+2.20 \pm 0.12$	$+0.66 \pm 0.11$	27.0 ± 2.4	100	0	0
Random Gate	$+1.10 \pm 0.13$	$+0.44 \pm 0.09$	24.1 ± 2.2	100	50	42 ± 2
Entropy Gate	$+1.70 \pm 0.12$	$+0.28 \pm 0.06$	18.2 ± 1.7	100	50	40 ± 2
Utility-Aware	$\mathbf{+2.40 \pm 0.16}$	$+0.08 \pm 0.04$	$\mathbf{11.0 \pm 1.4}$	62	50	28 ± 3

Table 3: Performance across model families and distribution shifts. Accuracy entries are improvements (percentage points) over the corresponding frozen model and are three-seed means. Established methods use their native protocols. The final column reports HCUR for Always Adapt → Utility-Aware. TPT is evaluated only with CLIP.

Setting	Always	TENT	EATA	SAR	TPT	Utility	HCUR A→U
CLIP / ImageNet-C	+2.20	+1.68	+2.01	+2.22	+1.91	+2.40	27.0 → 11.0
CLIP / Office-Home	+1.43	+0.94	+1.23	+1.42	+1.18	+1.59	20.4 → 9.3
DINOv2 / ImageNet-C	+1.59	+1.18	+1.49	+1.63	–	+1.78	22.1 → 10.2
DINOv2 / Office-Home	+1.17	+0.82	+1.03	+1.19	–	+1.36	19.2 → 9.1
DINOv2 / Camelyon17-WILDS	+0.71	+0.31	+0.52	+0.72	–	+0.88	24.7 → 12.1

ΔECE decreases from $+0.66 \pm 0.11$ percentage points under Always Adapt to $+0.08 \pm 0.04$ under Utility-Aware.

5.3 Comparison with Established TTA Methods and Generalization

We next compare system-level predictive performance with established TTA methods under their native protocols and evaluate transfer across model families and distribution shifts. Table 3 summarizes the results.

Utility-Aware attains the largest accuracy improvement at the displayed precision in all five evaluated settings. Across these settings, HCUR under Always Adapt ranges from 19.2% to 27.0%, whereas Utility-Aware reduces it to 9.1–12.1%.

These comparisons establish competitiveness across the evaluated systems but do not isolate the effect of post-proposal verification because the literature methods differ in adaptation objectives, adaptable parameter subsets, optimizers, selection mechanisms, and state protocols. The mechanism-matched experiments in Section 5.2, which control the base candidate operator and candidate-generation budget, provide the more direct evidence for the contribution of post-proposal verification. Camelyon17-WILDS is interpreted only as a cross-hospital distribution-shift stress test, not as clinical validation.

5.4 Post-Proposal Verification Complements Strong Pre-Update Selection

We next ask whether post-proposal verification remains useful after a strong pre-update selector has already determined where adaptation should be attempted. Table 4 compares CAS Gate with CAS+Utility under the same five-seed evaluation. Utility-Aware at the principal operating point is included as a reference and is not a five-seed mean from the matched CAS comparison.

CAS Gate obtains a $+2.03$-point accuracy gain with 15.2% HCUR. Applying Utility-Aware verification after CAS selection reaches $+2.47$ points and reduces HCUR to 8.2%. Relative to CAS Gate, CAS+Utility reduces HCUR by 7.0 percentage points (95% CI $[4.0, 10.1]$) and increases accuracy by 0.44 percentage points (95% CI $[0.18, 0.69]$); HCUR is lower for CAS+Utility in all five independent seeds.

Table 4: Complementarity with strong pre-update selection. CAS Gate and CAS+Utility are evaluated over the same five independent seeds. Utility-Aware is shown as the principal $\rho = 0.50$ reference. Commit denotes commitment coverage among generated candidates.

Method	ΔAcc. (pp) $\uparrow$	HCUR (%) $\downarrow$	Commit (%)
CAS Gate	$+2.03$	15.2	100
Utility-Aware	$+2.40$	11.0	62
CAS+Utility	$+2.47$	8.2	59

The distinction persists beyond the immediate evaluation window. Using the retrospective next-window utility in Eq. (12), predictive entropy obtains harmful-update AUROC/AUPRC of $0.571/0.292$, CAS obtains $0.641/0.352$, and the proposed utility score reaches $0.738/0.457$. Relative to CAS, the paired improvements are $+0.097$ AUROC (95% CI $[0.056, 0.136]$) and $+0.105$ AUPRC (95% CI $[0.065, 0.145]$).

Moreover, $r_s(\widehat{G}_t, G_t^{\text{next}}) = 0.418$, while $r_s(G_t, G_t^{\text{next}}) = 0.503$, and immediate and next-window harm labels agree for 72.4% of candidates. Within the CAS-selected subset, $\widehat{G}_t$ remains informative, reaching next-window AUROC 0.752 and AUPRC 0.468 at a harm prevalence of 0.151. These results support pre-update selection and post-proposal candidate commitment as distinct and complementary control levels.

5.5 Robustness, Compute Trade-Offs, and Oracle Headroom

Additional analyses support the robustness of the principal result. Utility-Aware achieves a larger accuracy gain than Random Gate and Entropy Gate at every evaluated candidate-generation fraction $\rho \in \{0.10, 0.25, 0.50, 0.75, 1.00\}$; at $\rho = 1$, only Utility-Aware can still reject low-utility realized updates. Component ablations reduce HCUR from 27.0% under Always Adapt to 22.4% with screening, 15.3% with held-out post-proposal evaluation, 11.4% with frozen-pretrained anchoring, and 11.0% for the full controller. Sensitivity analyses over β, m, and M show smooth trade-offs around the selected $(\beta, m, M) = (1, 0.05, 4)$ configuration (Appendix B).

At $\rho = 0.50$, Utility-Aware reduces candidate-update backward passes by 50% and adaptation-specific wall-clock time by $28 \pm 3\%$. The fixed-cost setting does not establish heterogeneous cost-aware scheduling. The retrospective one-step Oracle Utility Gate reaches a $+2.68$-point accuracy gain with no harmful commitments, indicating remaining headroom in label-free utility estimation rather than a deployable or globally optimal policy.

6 Discussion and Conclusion

Our results support two distinct control decisions in TTA: where adaptation should be attempted and whether the realized update should persist. At the principal operating point, Utility-Aware reduces HCUR from 27.0% to 11.0% while increasing the accuracy gain from $+2.20$ to $+2.40$ percentage points; after CAS preselection, post-proposal verification further reduces HCUR from 15.2% to 8.2%.

The controller remains deliberately myopic and empirical and does not provide a safety guarantee: probe transformations may fail to preserve task semantics, the frozen pretrained reference may be imperfect under substantial shift, and the evaluation is limited to classification with fixed candidate cost and a LayerNorm entropy-minimization candidate family. Generalization to substantially different adaptation operators remains to be established. Camelyon17-WILDS is used only as a cross-hospital distribution-shift stress test, not as clinical validation.

Overall, we recast selective TTA as two-level control: adaptation opportunity is assessed before candidate construction, whereas candidate utility is evaluated after an update is realized but before persistence. Across the evaluated models and shifts, evaluate-before-commit reduces harmful persistence while preserving predictive gains; at the principal operating point, the controller also reduces adaptation computation. Future work can address longer-horizon utility, task-specific verification, heterogeneous compute allocation, and broader candidate-update operators.

AI USE STATEMENT

OpenAI ChatGPT was used to support refinement of the research question and conceptual framing, discussion of methodology and experimental design, and interpretation and presentation of results. It was also used for literature exploration and summarization, assistance with mathematical notation and pseudocode, design and refinement of scientific figures, identification of limitations and reviewer concerns, and manuscript organization, drafting, and language editing. All AI-assisted work was reviewed and verified by the authors, who take responsibility for the final content of the submission, including its text, claims, and artifacts.

ETHICS STATEMENT

This work studies label-free control of TTA with the goal of reducing harmful model updates and unnecessary adaptation computation under distribution shift. The proposed controller does not provide a formal safety guarantee: candidate utility is estimated using an empirical surrogate, and incorrect commit or rollback decisions remain possible. Reduced HCUR on the evaluated benchmarks therefore should not be interpreted as establishing reliability under arbitrary or previously unseen distribution shifts.

The experiments use established benchmark datasets and involve no new collection of personal data or human-subject recruitment. Camelyon17-WILDS is used only as a cross-hospital distribution-shift stress test; the reported results do not constitute clinical validation. The proposed method is not intended for autonomous diagnosis, treatment selection, or other safety-critical medical decision making without application-specific validation, appropriate human oversight, and compliance with relevant institutional and regulatory requirements.

More generally, TTA changes model behavior after deployment and may amplify errors when its assumptions fail. Deployment in consequential settings should therefore retain monitoring, rollback mechanisms, domain-specific evaluation, and appropriate human or system-level safeguards. The present experiments do not provide a comprehensive assessment of energy consumption or establish fairness across demographic or protected subgroups; such properties require explicit evaluation in the intended deployment context.

REPRODUCIBILITY STATEMENT

We provide methodological and experimental information to support reproducibility of the reported experiments. Section 3 specifies the complete Utility-Aware TTA formulation, including forward-only candidate allocation, tentative update construction, held-out probe-view evaluation, frozen-pretrained anchoring, candidate-utility estimation, and transactional commit/rollback. Section 4 specifies the datasets, model families, baselines, candidate-generation budgets, evaluation metrics, statistical protocol, and computational accounting.

The appendices provide seed-level results and bootstrap procedures, matched-budget controls and ablations, cross-model and cross-shift evaluations, hyperparameter sensitivities, and implementation details, including model and classifier construction, optimizer and transactional optimizer-state handling, augmentation parameters, the causal quota controller, software and hardware configuration, and implementation-level pseudocode. Target labels are isolated from all deployed adaptation decisions and are used only retrospectively for evaluation and computation of oracle utility.

REFERENCES

Zongbo Han, Jialong Yang, Guangyu Wang, Junfan Li, Qianli Xu, Mike Zheng Shou, and Changqing Zhang. DOTA: Distributional test-time adaptation of vision-language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 142772–142798, 2025.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Hyun-Kurl Jang, Jihun Kim, Hyeokjun Kweon, and Kuk-Jin Yoon. Distill once, adapt life-long: Exploring dataset distillation for continual test-time adaptation. *arXiv preprint arXiv:2606.20196*, 2026.

Jingyan Jiang, Yaru Sun, Xiao Chen, Jiazhen Huang, Caiting Li, Zhijian He, Yin Chen, and Pingting Hao. Respect your zero-shot uncertainty: Conservative calibration for test-time-adapted vision-language models. *arXiv preprint arXiv:2608.05945*, 2026a.

Siru Jiang, Yuwei Liang, Jian Liang, Ran He, and Tieniu Tan. To adapt or not to adapt? selective adaptation for vision-language models. *arXiv preprint arXiv:2609.08367*, 2026b.

Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. Efficient test-time adaptation of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14162–14171, 2024.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664. PMLR, 2021.

Marc Lafon, Gustavo Vargas Hakim, Clément Rambour, Christian Desrosiers, and Nicolas Thome. CLIPTTA: Robust contrastive vision-language test-time adaptation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 71849–71883, 2025.

Youngjun Lee, Doyoung Kim, Junhyeok Kang, Jihwan Bang, Hwanjun Song, and Jae-Gil Lee. RA-TTA: Retrieval-augmented test-time adaptation for vision-language models. In *International Conference on Learning Representations*, 2025.

Aodi Li, Liansheng Zhuang, Xiao Long, Houqiang Li, and Shafei Wang. Flatness guided test-time adaptation for vision-language models. In *International Conference on Learning Representations*, 2026.

Taejun Lim, Joongwon Hwang, and Kibok Lee. When and where to reset matters for long-term test-time adaptation. In *International Conference on Learning Representations*, 2026.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905. PMLR, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Lijun Sheng, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. The illusion of progress? a critical look at test-time adaptation for vision-language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14274–14289, 2022.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5018–5027, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Maxime Zanella, Clément Fuchs, Christophe De Vleeschouwer, and Ismail Ben Ayed. Realistic test-time adaptation of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25103–25112, 2025.

Table 5: Seed-level principal results. Random Gate, Entropy Gate, and Utility-Aware use candidate-generation fraction $\rho = 0.50$, while Always Adapt generates and commits a candidate on every window. Accuracy is reported as percentage-point improvement relative to the frozen pretrained model. HCUR denotes the harmful committed-update rate.

Metric	Method	Seed 0	Seed 1	Seed 2	Mean ± SD
ΔAcc. (pp)	Always Adapt	2.09	2.18	2.33	2.20 ± 0.12
	Random Gate	0.98	1.08	1.24	1.10 ± 0.13
	Entropy Gate	1.59	1.68	1.83	1.70 ± 0.12
	Utility-Aware	2.24	2.41	2.55	2.40 ± 0.16
HCUR (%)	Always Adapt	24.9	26.5	29.6	27.0 ± 2.4
	Random Gate	22.0	24.0	26.3	24.1 ± 2.2
	Entropy Gate	16.6	18.1	19.9	18.2 ± 1.7
	Utility-Aware	9.7	10.8	12.5	11.0 ± 1.4

A Statistical Details and Seed-Level Results

A.1 Seed-Level Results

Principal mechanism-matched experiments use three independent random seeds. Within each seed, all methods receive the same target-stream realization. Table 5 reports the seed-level accuracy and HCUR values underlying the principal results in the main paper.

The same qualitative pattern holds across all three seeds: Utility-Aware attains higher accuracy than Always Adapt while committing a substantially smaller fraction of harmful updates. Randomly reducing candidate generation does not reproduce this behavior, while entropy-based screening provides an intermediate reduction in HCUR.

A.2 Paired Bootstrap Inference

Primary inferential comparisons use 10,000 bootstrap replicates. Whenever two methods or diagnostics are evaluated on the same underlying observations, candidate windows, or stream realizations, resampling is paired so that the comparison preserves their matched structure. Two-sided 95% confidence intervals are obtained from the 2.5th and 97.5th percentiles of the bootstrap distribution.

Candidate-harm discrimination. For harmful-update detection, candidate windows are resampled with replacement while preserving the pairing between competing diagnostic scores. AUROC and AUPRC are computed with larger scores indicating greater predicted harm; hence the proposed utility uses $-\widehat{G}_t$ as its harm score. The proposed utility achieves AUROC 0.750 compared with 0.589 for predictive entropy, corresponding to $\Delta\mathrm{AUROC} = +0.161$ (95% CI $[0.130, 0.191]$). For AUPRC, the corresponding values are 0.492 and 0.305, giving $\Delta\mathrm{AUPRC} = +0.187$ (95% CI $[0.145, 0.225]$).

Harmful committed updates. For the principal Always-Adapt versus Utility-Aware comparison, stream-matched resampling gives an absolute HCUR reduction of $27.0 - 11.0 = 16.0$ percentage points (95% CI $[13.4, 18.5]$), corresponding to a relative reduction of 59.3% (95% CI $[51.3, 66.6]\%$).

Predictive accuracy. Accuracy comparisons use matched examples or evaluation units. Utility-Aware improves accuracy by $+2.40$ percentage points relative to the frozen model, compared with $+2.20$ for Always Adapt. The paired difference is $\Delta\mathrm{Acc} = +0.20$ percentage points (95% CI $[+0.09, +0.28]$).

Thus, under the stated paired-bootstrap analysis, the observed accuracy gain of Utility-Aware over Always Adapt is positive across the reported 95% confidence interval.

A.3 CAS and Next-Window Statistical Comparisons

The CAS complementarity and next-window analyses use five independent seeds.

For CAS selection, adding post-proposal Utility-Aware verification reduces HCUR from 15.2% to 8.2%, corresponding to an absolute reduction of 7.0 percentage points (95% CI $[4.0, 10.1]$). Accuracy increases by $+0.44$ percentage points (95% CI $[+0.18, +0.69]$), and CAS+Utility yields lower HCUR than CAS Gate in all five seeds.

For retrospective next-window harm prediction, the proposed candidate-utility score achieves AUROC 0.738, compared with 0.641 for CAS, giving $\Delta\mathrm{AUROC} = +0.097$ (95% CI $[0.056, 0.136]$). The corresponding AUPRC improvement is $+0.105$ (95% CI $[0.065, 0.145]$).

The benefit-oriented candidate score has Spearman correlation $r_s = 0.418$ with next-window oracle utility, while immediate and next-window oracle utilities have $r_s = 0.503$. Immediate and next-window harm labels agree on 72.4% of evaluated candidates.

A.4 Hierarchical Resampling Across Shift Units

For comparisons aggregated across multiple distribution shifts, bootstrap resampling preserves the hierarchical structure of the benchmark. Each replicate first resamples the higher-level shift unit and then resamples observations or candidate windows within each selected unit. Within each selected shift unit, the relevant second-stage units are resampled individually with replacement, while preserving pairing across compared methods or diagnostics, as applicable; no additional temporal block structure is imposed. ImageNet-C uses corruption trajectories as shift units, while Office-Home uses directed source-to-target domain transfers. This procedure avoids treating all observations from the same shift as independent replicates and prevents larger shift units from dominating uncertainty estimates solely because they contain more observations.

A.5 Wall-Clock Measurement Protocol

Wall-clock efficiency is measured using repeated complete adaptation runs after system warm-up. Within each comparison, all policies use identical hardware, model checkpoint, target-stream order, window size, numerical precision, software environment, and data-loading configuration.

Adaptation-specific timing includes forward computation required by the adaptation procedure, candidate-update backward computation, update- and probe-view processing, frozen-anchor evaluation, utility-score computation, and computation spent on candidates that are subsequently rolled back. Ordinary inference shared by all compared methods is excluded.

At the principal operating point, $\rho = 0.50$, Utility-Aware reduces candidate-update backward passes by exactly 50% relative to Always Adapt. Repeated end-to-end measurements yield an adaptation-specific wall-clock reduction of $28 \pm 3\%$.

B Additional Experimental Results, Ablations, and Sensitivity Analyses

Sensitivity-analysis protocol. Unless otherwise stated, the analyses in this appendix use the primary CLIP ViT-B/16 ImageNet-C setting and the same three random seeds as the principal experiments. The selected configuration is

$$(\beta, m, M) = (1.0, 0.05, 4), \qquad m = \lambda + \tau,$$

with candidate-generation fraction $\rho = 0.50$. These choices were fixed before inspection of target labels from the principal evaluation benchmarks. The sweeps below are therefore used to characterize robustness around the selected operating point rather than to retrospectively select hyperparameters that maximize the reported test results.

B.1 Matched-Budget Controls

Table 6 expands the mechanism-matched comparison at the principal operating point. Random Gate, Entropy Gate, and Utility-Aware all use the same candidate-generation fraction $\rho = 0.50$ and therefore construct tentative updates on the same number of target windows. Always Adapt instead generates and commits a candidate on every window.

Table 6: Matched-budget comparison at the principal operating point. Commit denotes commitment coverage among generated candidates. Backward-pass and adaptation-time reductions are measured relative to Always Adapt.

Method	ΔAcc. (pp)	HCUR (%)	Commit	Back. red. ↑	Time red. ↑
Always Adapt	$+2.20 \pm 0.12$	27.0 ± 2.4	100%	0%	0%
Random Gate	$+1.10 \pm 0.13$	24.1 ± 2.2	100%	50%	$42 \pm 2\%$
Entropy Gate	$+1.70 \pm 0.12$	18.2 ± 1.7	100%	50%	$40 \pm 2\%$
Utility-Aware	$\mathbf{+2.40 \pm 0.16}$	$\mathbf{11.0 \pm 1.4}$	62%	50%	$28 \pm 3\%$

Table 7: Accuracy improvement as a function of candidate-generation fraction ρ. Values are percentage-point improvements relative to the frozen pretrained model.

ρ	Random Gate	Entropy Gate	Utility-Aware
0.10	+0.28	+0.57	**+0.91**
0.25	+0.72	+1.16	**+1.76**
0.50	+1.10	+1.70	**+2.40**
0.75	+1.61	+2.02	**+2.49**
1.00	+2.20	+2.20	**+2.58**

Randomly withholding half of the candidate-generation opportunities leaves HCUR relatively close to unrestricted adaptation, indicating that reduced adaptation frequency alone does not explain the reliability improvement. Entropy-based screening provides a larger reduction in HCUR, while Utility-Aware further reduces harmful persistence despite using the same candidate-generation budget. The smaller wall-clock reduction of Utility-Aware relative to Random Gate and Entropy Gate reflects the additional forward computation required to evaluate tentative candidates before commitment.

B.2 Candidate-Generation Budget Sweep

We next vary the candidate-generation fraction ρ while leaving the corresponding policy otherwise unchanged. Table 7 reports accuracy improvement relative to the frozen model.

Utility-Aware obtains the largest predictive gain at every evaluated candidate-generation fraction. The $\rho = 1.00$ setting is particularly informative. Random Gate and Entropy Gate then reduce to Always Adapt because every window generates a candidate and every generated candidate is committed. Utility-Aware can still reject low-utility realized updates. Candidate generation and candidate persistence are therefore distinct control decisions even when candidate-generation coverage is complete.

B.3 Generalization Across Models and Distribution Shifts

Table 8 examines whether the reduction in harmful committed updates transfers beyond the primary CLIP/ImageNet-C setting.

Utility-Aware reduces HCUR in all five evaluated model–shift combinations. The effect is observed for both CLIP and DINOv2 and under synthetic corruption, natural domain shift, and the cross-hospital stress test. These experiments support transfer of the candidate-verification mechanism beyond the primary configuration, while not establishing reliability under arbitrary or previously unseen distribution shifts.

B.4 Component Ablation

We progressively introduce the components of the proposed controller to separate the effects of pre-update compute allocation and post-proposal candidate evaluation. Table 9 summarizes the resulting predictive performance, committed-update risk, and computational cost.

Table 8: Cross-model and cross-shift behavior of Utility-Aware commitment. ΔAcc. is measured relative to the corresponding frozen model. Camelyon17-WILDS is used only as a cross-hospital distribution-shift stress test.

Model	Shift	Utility ΔAcc. (pp)	HCUR Always (%)	HCUR Utility (%)
CLIP ViT-B/16	ImageNet-C	+2.40	27.0	11.0
CLIP ViT-B/16	Office-Home	+1.59	20.4	9.3
DINOv2 ViT-B/14	ImageNet-C	+1.78	22.1	10.2
DINOv2 ViT-B/14	Office-Home	+1.36	19.2	9.1
DINOv2 ViT-B/14	Camelyon17-WILDS	+0.88	24.7	12.1

Table 9: Component ablation in the primary setting. Backward-pass and adaptation-time reductions are relative to Always Adapt.

Configuration	ΔAcc. (pp)	HCUR (%)	Back. red. ↑	Time red. ↑
Always Adapt	+2.20	27.0	0%	0%
Screening only	+1.66	22.4	50%	42%
+ Held-Out Probe	+2.08	15.3	50%	33%
+ Frozen Anchor	+2.37	11.4	50%	29%
Full Utility-Aware	**+2.40**	**11.0**	50%	28%

Screening alone provides the principal reduction in candidate-update backward computation but yields only a modest reduction in HCUR. Adding held-out post-proposal evaluation reduces HCUR from 22.4% to 15.3%, indicating that the behavior of the realized candidate contains information unavailable to pre-update screening alone. Frozen-pretrained anchoring further reduces HCUR while improving predictive performance. The full controller reaches 11.0% HCUR with a $+2.40$-percentage-point accuracy gain.

These results isolate the two control levels underlying the proposed framework: screening determines where adaptation computation is allocated, whereas post-proposal evaluation determines whether the resulting parameter transition persists.

B.5 Sensitivity to the Frozen-Pretrained Anchor

We vary the anchor coefficient β while fixing $m = 0.05$, $M = 4$, and $\rho = 0.50$. Table 10 reports three-seed means.

Without the frozen-pretrained anchor, candidate verification remains useful but harmful-update discrimination is weaker. Moderate anchoring improves both utility ordering and the resulting risk–performance trade-off. Increasing β beyond the selected value continues to reduce HCUR but eventually suppresses useful adaptation and weakens utility discrimination. The selected $\beta = 1$ therefore lies within a stable moderate range rather than representing a narrowly isolated operating point.

B.6 Sensitivity to the Effective Commitment Margin

All standard candidates in the reported experiments have constant normalized cost $\tilde{c}_t = 1$. Hence $\widehat{G}_t = \widehat{G}_t^{\mathrm{risk}} - \lambda$, and the decision $\widehat{G}_t > \tau$ is equivalent to $\widehat{G}_t^{\mathrm{risk}} > \lambda + \tau \equiv m$. Thus, under the fixed-cost setting, λ and τ affect the commitment decision only through the effective margin m. Table 11 reports the resulting risk–coverage trade-off as m is varied.

Increasing m produces the expected risk–coverage trade-off. More conservative commitment reduces both commitment coverage and HCUR, but an excessively large margin also rejects beneficial candidates and reduces the accuracy gain. The selected $m = 0.05$ lies near the empirical knee of this trade-off. Because changing m changes only the operating threshold applied to the same candidate ranking, it does not alter the corresponding ranking-based AUROC.

Table 10: Sensitivity to the frozen-pretrained anchor coefficient β in the primary CLIP ViT-B/16 ImageNet-C setting.

β	ΔAcc. (pp)	HCUR (%)	AUROC	Spearman r_s
0.00	+2.11	15.3	0.690	0.310
0.25	+2.27	13.6	0.716	0.371
0.50	+2.36	12.1	0.736	0.421
1.00	**+2.40**	11.0	**0.750**	**0.460**
2.00	+2.33	9.6	0.743	0.432
4.00	+2.09	8.1	0.711	0.352

Table 11: Sensitivity to the effective commitment margin m. Commit denotes commitment coverage among generated candidates. Values are three-seed means.

m	Commit	ΔAcc. (pp)	HCUR (%)
−0.05	87%	+2.19	17.8
0.00	79%	+2.31	14.6
0.05	62%	**+2.40**	11.0
0.10	49%	+2.36	9.1
0.15	35%	+2.18	7.1
0.25	21%	+1.79	5.2

B.7 Sensitivity to the Number of Probe Views

We vary the number of independently sampled probe views M while fixing $\beta = 1$, $m = 0.05$, and $\rho = 0.50$. Table 12 reports the resulting reliability–compute trade-off.

Increasing the number of probe views improves candidate-utility estimation, but the gains exhibit diminishing returns. Moving from $M = 1$ to $M = 4$ substantially improves harmful-update AUROC and reduces HCUR. Increasing M further yields smaller reliability gains while progressively increasing verification cost. At $M = 16$, adaptation-specific computation is approximately 4% slower than Always Adapt. We therefore use $M = 4$ as a practical reliability–compute operating point.

B.8 Calibration

Table 13 reports change in expected calibration error relative to the frozen model.

Unrestricted adaptation improves accuracy but produces substantially larger calibration deterioration. Utility-Aware limits this degradation while retaining the strongest accuracy improvement among the mechanism-matched methods at the principal operating point. Because the frozen model has $\Delta\mathrm{ECE} = 0$ by definition, these results should be interpreted as relative calibration preservation rather than an improvement over the frozen reference.

B.9 Myopic Oracle Headroom

We compare Utility-Aware with the retrospective Oracle Utility Gate, which has access to the unavailable immediate oracle utility G_t. The oracle is used only to characterize one-step headroom in candidate-utility estimation. Table 14 summarizes this retrospective comparison.

The oracle eliminates harmful committed updates and increases the accuracy gain from $+2.40$ to $+2.68$ percentage points. This gap indicates remaining headroom in label-free candidate-utility estimation. It should not be interpreted as a deployable baseline or as a globally optimal sequential adaptation policy: G_t uses retrospective labels and measures only immediate candidate utility.

B.10 Interpretation and Scope

The additional analyses support four aspects of the proposed mechanism. First, reducing adaptation frequency alone is insufficient: matched-budget random allocation leaves HCUR close to unrestricted adaptation. Second, post-proposal evaluation contributes information beyond pre-update

Table 12: Sensitivity to the number of probe views M in the primary setting. Values are three-seed means. Negative time reduction denotes additional adaptation-specific overhead relative to Always Adapt.

M	ΔAcc. (pp)	HCUR (%)	AUROC	Spearman r_s	Time red. $\uparrow$
1	$+2.04$	15.6	0.691	0.341	37%
2	$+2.23$	13.1	0.726	0.401	33%
4	$+2.40$	11.0	0.750	0.460	28%
8	$+2.44$	10.1	0.764	0.489	18%
16	$+2.46$	9.7	0.771	0.501	-4%

Table 13: Change in expected calibration error relative to the frozen model. Lower deterioration is better.

Method	ΔECE (pp)
Frozen	0.00
Always Adapt	$+0.66 \pm 0.11$
Random Gate	$+0.44 \pm 0.09$
Entropy Gate	$+0.28 \pm 0.06$
Utility-Aware	$+0.08 \pm 0.04$

screening, as shown by the component ablation and by continued gains at $\rho = 1.00$. Third, the selected values of β, m, and M lie within smooth empirical trade-offs rather than depending on isolated parameter settings. Finally, the reduction in harmful committed updates transfers across the evaluated model families and distribution shifts.

The computational-cost interpretation is deliberately narrower. Because the standard candidate has constant normalized cost $\tilde{c}_t = 1$, the current experiments demonstrate compute-budgeted candidate allocation and a cost-adjusted commitment margin. They do not establish preferential selection among candidate updates with heterogeneous computational costs. Evaluating that more general setting is left to future work.

C Implementation and Reproducibility Details

This appendix specifies the implementation used for the reported experiments and complements the methodological description in Section 3 and the experimental protocol in Section 4.

C.1 Model and Classifier Configuration

CLIP. The primary experiments use CLIP ViT-B/16 (Radford et al., 2021). Only LayerNorm affine parameters (γ, β) in the vision encoder are eligible for test-time updates. The text encoder, class prototypes, pretrained logit scale, and all remaining vision-encoder parameters remain frozen throughout adaptation.

For class k, we construct a frozen text prototype using the fixed CLIP prompt ensemble. Let $\mathcal{T} = \{T_q\}_{q=1}^{Q}$ denote the prompt templates and let c_k denote the normalized human-readable class name. Each prompted text embedding is normalized as

$$e_{k,q} = \frac{g_{\text{text}}(T_q(c_k))}{\|g_{\text{text}}(T_q(c_k))\|_2}. \tag{14}$$

The class prototype is obtained by averaging the template embeddings and renormalizing:

$$w_k = \frac{\frac{1}{Q}\sum_{q=1}^{Q} e_{k,q}}{\left\|\frac{1}{Q}\sum_{q=1}^{Q} e_{k,q}\right\|_2}. \tag{15}$$

The same fixed prompt-template construction is used across the CLIP classification experiments. Target labels are not used to select templates or perform target-dependent prompt engineering.

Table 14: Retrospective one-step oracle comparison. The Oracle Utility Gate uses target labels through G_t and is therefore not deployable.

Method	ΔAcc. (pp)	HCUR (%)
Utility-Aware	+2.40	11.0
Oracle Utility Gate	+2.68	0.0

Given normalized image representation

$$u_\theta(x) = \frac{g_{\mathrm{img},\theta}(x)}{\|g_{\mathrm{img},\theta}(x)\|_2}, \tag{16}$$

the class logit is

$$f_\theta(x)_k = \exp(\alpha_0)\, u_\theta(x)^\top w_k, \tag{17}$$

where α_0 is the pretrained CLIP logit-scale parameter and remains frozen during TTA. Consequently, adaptation-induced changes in CLIP predictions arise only from the eligible LayerNorm affine parameters of the vision encoder.

DINOv2. Generalization experiments use DINOv2 ViT-B/14 (Oquab et al., 2024). ImageNet-C experiments use the released ImageNet linear classifier. For Office-Home and Camelyon17-WILDS, task-specific linear classifiers are trained from source-domain labels before the target stream is observed. During this source-head training stage, the DINOv2 backbone remains frozen and only the linear classifier is optimized. No target-domain labels are used.

For Office-Home, each source domain is divided into a stratified 90%/10% source-training/source-validation split. For Camelyon17-WILDS, the official source training and in-distribution validation splits are used. Classifier and hyperparameter selection therefore depend only on source-side information.

The classifier operates on frozen representations derived from the final four transformer blocks. The source head is trained using SGD with momentum 0.9, zero weight decay, batch size 128, cosine learning-rate scheduling, and 10 epochs. The learning rate is selected using source-validation accuracy from $\{10^{-5}, 2\times10^{-5}, 5\times10^{-5}, 10^{-4}, 2\times10^{-4}, 5\times10^{-4}, 10^{-3}, 2\times10^{-3}, 5\times10^{-3}, 10^{-2}, 2\times10^{-2}, 5\times10^{-2}, 10^{-1}\}$. Once selected, the task-specific classifier remains frozen for the complete target stream. During TTA, only LayerNorm affine parameters of the DINOv2 backbone are updated.

C.2 Candidate-Generation Configuration

The base candidate generator performs one entropy-minimization update on each window allocated candidate-generation computation. Table 15 summarizes the principal implementation configuration.

For every generated candidate, two independently sampled update views per observation are used exclusively for entropy minimization. Four independently sampled probe views per observation are generated separately and withheld from candidate optimization. Thus, update and probe views are independent augmentation draws of the same underlying target observations; probe-view draws used for candidate evaluation are not reused by the update step.

C.3 Candidate Construction and Transactional Optimizer State

Let ϕ_t denote the adaptable LayerNorm affine parameters of the persistent model at window t, and let ω_t denote the corresponding persistent AdamW optimizer state. Before candidate construction, both states are snapshotted:

$$(\phi_t^{\mathrm{snap}}, \omega_t^{\mathrm{snap}}) \leftarrow (\phi_t, \omega_t). \tag{18}$$

The adaptation loss is computed exclusively from the update views:

$$\mathcal{L}_{\mathrm{ent}} = \frac{1}{|\mathcal{V}_t^{\mathrm{upd}}|} \sum_{v \in \mathcal{V}_t^{\mathrm{upd}}} H(p_{\theta_t}(v)). \tag{19}$$

Table 15: Implementation configuration used in the reported experiments.

Component	Configuration
Primary backbone	CLIP ViT-B/16
Generalization backbone	DINOv2 ViT-B/14
CLIP classifier	Frozen prompt-ensemble text classifier
DINOv2 ImageNet-C head	Released ImageNet linear head
DINOv2 Office-Home head	Source-trained frozen linear head
DINOv2 Camelyon17 head	Source-trained frozen linear head
Adapted parameters	LayerNorm affine parameters only
Base TTA objective	Entropy minimization
Candidate-update steps	1
Optimizer	AdamW
Learning rate	1×10^{-4}
Adam coefficients	$(0.9, 0.999)$
Weight decay	0
Optimizer state	Persistent and transactional
Maximum window size	64
Update views / observation	2
Probe views / observation	4
Random-resized-crop scale	$[0.8, 1.0]$
Horizontal-flip probability	0.5
Strong color jitter	Disabled
Anchor coefficient β	1.0
Normalized candidate cost $\tilde{c}_t$	1
Implemented (λ, τ)	$(0.05, 0)$
Effective margin $m = \lambda + \tau$	0.05
Primary candidate-generation fraction ρ	0.50
Candidate-generation sweep	$\{0.10, 0.25, 0.50, 0.75, 1.00\}$
Screening statistic	Normalized predictive entropy
Rolling screening history W	100 windows
Screening warm-up W_0	20 windows
Candidate-budget enforcement	Exact causal quota
Principal random seeds	$\{0, 1, 2\}$
Numerical precision	BF16 mixed precision
Bootstrap replicates	10,000

One AdamW step produces tentative parameter and optimizer states,

$$(\tilde{\phi}_t, \tilde{\omega}_t) = \mathrm{AdamWStep}\left(\phi_t, \omega_t, \nabla_\phi \mathcal{L}_{\mathrm{ent}}; \eta = 10^{-4}\right). \tag{20}$$

All non-adapted model parameters remain unchanged. Both $\tilde{\phi}_t$ and $\tilde{\omega}_t$ remain tentative until candidate evaluation has completed.

If the candidate is committed,

$$(\phi_{t+1}, \omega_{t+1}) = (\tilde{\phi}_t, \tilde{\omega}_t). \tag{21}$$

If the candidate is rejected,

$$(\phi_{t+1}, \omega_{t+1}) = (\phi_t^{\mathrm{snap}}, \omega_t^{\mathrm{snap}}). \tag{22}$$

Transactional restoration therefore prevents a rejected candidate from affecting subsequent adaptation through either model parameters or the AdamW first- and second-moment states.

C.4 Probe-View Consistency

For observation $x_{t,i}$, let $\{v_{t,i,j}^{\mathrm{probe}}\}_{j=1}^{M}$ denote its independently sampled probe views, with $M = 4$ at the principal operating point. Their mean predictive distribution is

$$\bar{p}_{\theta,t,i} = \frac{1}{M} \sum_{j=1}^{M} p_\theta(v_{t,i,j}^{\mathrm{probe}}). \tag{23}$$

We measure generalized Jensen–Shannon disagreement as

$$D_{\mathrm{cons}}(\theta;t)=\frac{1}{n_t}\sum_{i=1}^{n_t}\left[H(\bar{p}_{\theta,t,i})-\frac{1}{M}\sum_{j=1}^{M}H\Big(p_\theta(v_{t,i,j}^{\mathrm{probe}})\Big)\right]. \tag{24}$$

The held-out consistency improvement produced by the candidate is

$$\Delta C_t=D_{\mathrm{cons}}(\theta_t;t)-D_{\mathrm{cons}}(\tilde{\theta}_t;t). \tag{25}$$

Thus, $\Delta C_t>0$ indicates increased agreement across views withheld from candidate optimization. It remains a label-free surrogate rather than a guarantee of predictive correctness.

C.5 Frozen-Pretrained Anchor

The pretrained model θ_0 is retained as a frozen reference throughout the target stream. Anchor disagreement is

$$A_t(\theta)=\frac{1}{n_tM}\sum_{i=1}^{n_t}\sum_{j=1}^{M}D_{\mathrm{JS}}\left(p_\theta(v_{t,i,j}^{\mathrm{probe}})\,\|\,p_{\theta_0}(v_{t,i,j}^{\mathrm{probe}})\right). \tag{26}$$

Candidate-induced change in anchor disagreement is

$$\Delta A_t=A_t(\tilde{\theta}_t)-A_t(\theta_t), \tag{27}$$

with only increased drift penalized:

$$[\Delta A_t]_+=\max(0,\Delta A_t). \tag{28}$$

The pretrained predictor therefore serves as a conservative reference for candidate-induced drift rather than as a correctness oracle.

C.6 Implemented Candidate-Utility Score

The candidate-utility score is

$$\widehat{G}_t=\Delta C_t-\beta[\Delta A_t]_+-\lambda\tilde{c}_t. \tag{29}$$

Within the fixed-cost setting used by the reported commitment policy, candidate cost is normalized as

$$\tilde{c}_t=1. \tag{30}$$

The decision $\widehat{G}_t>\tau$ is therefore equivalent to

$$\Delta C_t-\beta[\Delta A_t]_+>\lambda+\tau\equiv m. \tag{31}$$

At the principal operating point,

$$\beta=1,\qquad m=0.05, \tag{32}$$

so a generated candidate is committed iff

$$\Delta C_t-[\Delta A_t]_+>0.05. \tag{33}$$

The implementation uses $(\lambda,\tau)=(0.05,0)$. Because candidate cost is constant in this setting, only the combined margin $m=\lambda+\tau$ affects the binary commitment decision. The reported experiments therefore do not independently evaluate preferential selection among candidate updates with heterogeneous computational costs.

C.7 Exact Causal Candidate-Quota Controller

For a stream containing T windows and target candidate-generation fraction ρ, define

$$K_\rho = \lfloor \rho T \rfloor. \tag{34}$$

Before processing window t, let

$$r_t = K_\rho - \sum_{j=1}^{t-1} z_j, \qquad \ell_t = T - t + 1, \tag{35}$$

where r_t is the remaining candidate quota and ℓ_t is the number of remaining windows.

For $t > W_0$, the causal entropy threshold is

$$\kappa_{\rho,t} = Q_{1-\rho}\left(\{s_j\}_{j=\max(1,t-W)}^{t-1}\right), \qquad W = 100, \tag{36}$$

and the provisional forward-only allocation is

$$\tilde{z}_t = \mathbb{I}\left[s_t \geq \kappa_{\rho,t}\right]. \tag{37}$$

During the first $W_0 = 20$ windows,

$$\tilde{z}_t \sim \text{Bernoulli}(\rho). \tag{38}$$

The final allocation applies the remaining-quota correction:

$$z_t = \begin{cases} 0, & r_t = 0, \\ 1, & r_t = \ell_t, \\ \tilde{z}_t, & \text{otherwise.} \end{cases} \tag{39}$$

This construction guarantees

$$\sum_{t=1}^{T} z_t = K_\rho \tag{40}$$

without examining future target observations. For the evaluated even-length streams at the principal operating point $\rho = 0.50$, $K_\rho = T/2$, so candidate-generation computation is allocated to exactly half of the windows, yielding an exact 50% reduction in candidate-update backward passes relative to Always Adapt.

C.8 Reference Pseudocode

Algorithm 1 gives implementation-level pseudocode for the complete transactional controller.

Transactional restoration removes the effects of every rejected candidate from both the persistent model state and the optimizer trajectory.

C.9 Augmentation Protocol

Update and probe views are sampled independently from the same mild, semantic-preserving augmentation family:

- random resized cropping with scale in $[0.8, 1.0]$;
- horizontal flipping with probability 0.5; and
- no strong color jitter.

For Camelyon17-WILDS, cropping retains at least 80% of the original patch area, and stain- or color-altering transformations are disabled. These transformations are used as mild invariance probes rather than as guarantees of semantic preservation under arbitrary transformations.

Algorithm 1 Transactional Utility-Aware Test-Time Adaptation

Require: pretrained state θ_0; target stream $\{B_t\}_{t=1}^{T}$; candidate-generation fraction ρ; $\beta = 1$; $m = 0.05$; $W = 100$; $W_0 = 20$

$\theta \leftarrow \theta_0$
initialize persistent AdamW state ω
$K \leftarrow \lfloor \rho T \rfloor$
$q \leftarrow 0$
$\mathcal{S} \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  compute ordinary prediction and normalized entropy score s_t
  $r \leftarrow K - q$, $\ell \leftarrow T - t + 1$
  if $r = 0$ **then**
    $z_t \leftarrow 0$
  else if $r = \ell$ **then**
    $z_t \leftarrow 1$
  else if $t \leq W_0$ **then**
    $\tilde{z}_t \sim \mathrm{Bernoulli}(\rho)$
    $z_t \leftarrow \tilde{z}_t$
  else
    $\kappa \leftarrow Q_{1-\rho}(\text{last } W \text{ scores in } \mathcal{S})$
    $\tilde{z}_t \leftarrow \mathbb{I}[s_t \geq \kappa]$
    $z_t \leftarrow \tilde{z}_t$
  end if
  append s_t to $\mathcal{S}$
  if $z_t = 0$ **then**
    continue
  end if
  $q \leftarrow q + 1$
  snapshot $(\theta^{\mathrm{snap}}, \omega^{\mathrm{snap}}) \leftarrow (\theta, \omega)$
  independently sample $\mathcal{V}_t^{\mathrm{upd}}$ and $\mathcal{V}_t^{\mathrm{probe}}$
  $(\tilde{\theta}, \tilde{\omega}) \leftarrow \mathcal{U}(\theta, \omega; \mathcal{V}_t^{\mathrm{upd}})$
  compute ΔC_t using $\mathcal{V}_t^{\mathrm{probe}}$
  compute $[\Delta A_t]_+$ relative to frozen θ_0
  $\widehat{G}_t^{\mathrm{risk}} \leftarrow \Delta C_t - \beta[\Delta A_t]_+$
  if $\widehat{G}_t^{\mathrm{risk}} > m$ **then**
    $(\theta, \omega) \leftarrow (\tilde{\theta}, \tilde{\omega})$ ▷ commit
  else
    $(\theta, \omega) \leftarrow (\theta^{\mathrm{snap}}, \omega^{\mathrm{snap}})$ ▷ rollback
  end if
end for

C.10 Randomness and Target-Label Isolation

The principal experiments use seeds

$$\{0, 1, 2\}. \tag{41}$$

Each seed controls augmentation sampling, warm-up candidate-allocation proposals, stream-order randomness where applicable, and framework random-number generators. Mechanism-matched methods receive identical target-stream realizations within each seed.

The CAS complementarity and next-window analyses use five independent seeds, as specified in the corresponding experimental protocol.

Target labels from the principal evaluation streams are inaccessible to every deployed decision. They are not used to compute screening scores, allocate candidate-generation computation, construct tentative updates, generate probe views, compute ΔC_t or ΔA_t, select the commitment margin, or make commit/rollback decisions.

Table 16: Software and hardware environment used for the reported experiments.

Item	Configuration
GPU	NVIDIA A100 80GB
GPU count	1
Operating system	Ubuntu 22.04 LTS
Python	3.11
PyTorch	2.14.0
TorchVision	0.29.0
PyTorch CUDA runtime	13.2
System CUDA toolkit	13.3.1
Mixed precision	BF16
OpenCLIP	`open_clip_torch` 3.3.0
DINOv2 implementation	Official Meta repository
DINOv2 repository revision	`7764ea0`

Target labels are revealed only retrospectively after the corresponding online trajectory has completed and are used to compute predictive metrics, oracle utility G_t, harmful-update labels, HCUR, and statistical comparisons.

C.11 Compute Accounting and Timing

Adaptation-specific wall-clock time includes:

- adaptation-specific forward computation;
- candidate-update backward computation;
- update- and probe-view processing;
- frozen-anchor evaluation;
- candidate-utility computation; and
- computation spent on candidates that are subsequently rolled back.

Ordinary inference shared by all compared methods is excluded from the reported adaptation-time reduction. Candidate-update backward passes and wall-clock time are reported separately.

Timing measurements are obtained from repeated complete end-to-end adaptation runs after system warm-up. Within each comparison, methods use identical hardware, model checkpoint, target-stream order, window size, numerical precision, software environment, and data-loading configuration. Individual target windows within a single run are not treated as independent timing replicates.

At the principal operating point, Utility-Aware reduces candidate-update backward passes by exactly 50% relative to Always Adapt and reduces measured adaptation-specific wall-clock time by $28\pm3\%$.

C.12 Software and Hardware Environment

Table 16 reports the software and hardware environment used for the reported experiments. The system CUDA toolkit and the CUDA runtime against which the PyTorch binary was built are reported separately because they need not be identical.

C.13 DINOv2 Source-Head Reproducibility

Table 17 summarizes the source-only classifier-training protocol used for Office-Home and Camelyon17-WILDS. Learning-rate selection uses source-side validation only, and the selected checkpoint is fixed before the target stream is processed.

Together with Sections 3 and 4, this appendix provides implementation details to support reproduction of the screen–propose–evaluate–commit/rollback procedure used in the reported experiments.

Table 17: DINOv2 source-head training protocol.

Component	Configuration
Backbone during head training	Frozen DINOv2 ViT-B/14
Feature representation	Final four transformer blocks
Classifier	Linear
Optimizer	SGD
Momentum	0.9
Weight decay	0
Batch size	128
Epochs	10
Scheduler	Cosine annealing
LR selection	Source-validation accuracy
Office-Home validation	Stratified 10% source split
Camelyon17 validation	Official source/ID-validation split
Target labels for head training	None
Head during TTA	Frozen

D Theoretical Interpretation of Conservative Commitment

This appendix provides a formal interpretation of the candidate-commitment rule. The analysis is deliberately local: it concerns whether a realized candidate improves the current window-level predictive risk. It does not establish global optimality of the resulting sequential adaptation trajectory, nor does it provide a formal safety guarantee.

D.1 Oracle Candidate Utility

Let θ_t denote the persistent model state before adaptation at window t, and let $\tilde{\theta}_t$ denote the tentative state produced by the base adaptation operator. As defined in Section 3, the retrospective one-step oracle utility is

$$G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t), \tag{42}$$

where $R_t(\cdot)$ is labeled predictive risk on the current evaluation window. Hence

$$G_t > 0 \tag{43}$$

denotes an immediately beneficial candidate, whereas

$$G_t < 0 \tag{44}$$

denotes an immediately harmful candidate.

Because target labels are unavailable during deployment, G_t is used only for retrospective evaluation. The proposed label-free risk-adjusted pre-cost score is

$$\widehat{G}_t^{\mathrm{risk}} = \Delta C_t - \beta[\Delta A_t]_+, \tag{45}$$

and the full candidate-utility score is

$$\widehat{G}_t = \widehat{G}_t^{\mathrm{risk}} - \lambda \tilde{c}_t. \tag{46}$$

The candidate is committed iff

$$\widehat{G}_t > \tau, \tag{47}$$

or equivalently,

$$\widehat{G}_t^{\mathrm{risk}} > \tau + \lambda \tilde{c}_t. \tag{48}$$

D.2 Conditional Bounded-Error Interpretation

The commitment margin admits a simple conditional conservative interpretation.

Proposition 1 (bounded-error interpretation). Suppose that, for candidate t, the pre-cost label-free score satisfies

$$\left|\widehat{G}_t^{\mathrm{risk}} - G_t\right| \le \epsilon_t. \tag{49}$$

If the controller commits only when

$$\widehat{G}_t^{\mathrm{risk}} > \tau + \lambda\tilde{c}_t \tag{50}$$

and

$$\tau + \lambda\tilde{c}_t \ge \epsilon_t, \tag{51}$$

then every committed candidate satisfying Eq. (49) has positive one-step oracle utility:

$$G_t > 0. \tag{52}$$

Proof. From Eq. (49),

$$G_t \ge \widehat{G}_t^{\mathrm{risk}} - \epsilon_t. \tag{53}$$

For a committed candidate,

$$\widehat{G}_t^{\mathrm{risk}} > \tau + \lambda\tilde{c}_t. \tag{54}$$

Therefore,

$$G_t > \tau + \lambda\tilde{c}_t - \epsilon_t \ge 0, \tag{55}$$

where the final inequality follows from Eq. (51). □

Interpretation. Proposition 1 is conditional and does not assert that the proposed estimator satisfies a known or uniform error bound in practice. Rather, it clarifies the role of the commitment margin: if candidate-utility estimation error were bounded by a known amount, a margin at least as large as that bound would convert the point estimate into a conservative one-step commit rule.

Equivalently, commitment requires the margin-adjusted surrogate

$$\widehat{G}_t^{\mathrm{risk}} - (\tau + \lambda\tilde{c}_t) > 0. \tag{56}$$

The experiments evaluate the empirical usefulness of this principle when no deterministic estimation-error bound is assumed.

D.3 Fixed-Cost Equivalence

The reported commitment experiments use candidates with identical normalized cost,

$$\tilde{c}_t = 1. \tag{57}$$

Lemma 1 (effective-margin equivalence). When $\tilde{c}_t = 1$ for all generated candidates, the binary commitment policy depends on λ and τ only through

$$m = \lambda + \tau. \tag{58}$$

Proof. Substituting $\tilde{c}_t = 1$ into Eq. (48) gives

$$\widehat{G}_t^{\mathrm{risk}} > \lambda + \tau = m. \tag{59}$$

Thus, any pair (λ, τ) with the same sum m induces the same binary commitment decisions. □

This equivalence motivates reporting the effective margin m in the sensitivity analysis of Appendix B.6. It also makes clear that λ and τ are not separately identifiable as decision parameters from the fixed-cost experiments.

At the principal operating point,

$$\lambda = 0.05, \qquad \tau = 0, \qquad m = 0.05. \tag{60}$$

With $\beta = 1$, the implemented decision is therefore

$$\Delta C_t - [\Delta A_t]_+ > 0.05. \tag{61}$$

D.4 The Commitment Margin Changes Decisions, Not Ranking

For a fixed set of generated candidates and fixed β, the effective margin changes the binary operating point but not the ordering induced by the candidate score.

Lemma 2 (ranking invariance). For fixed β, changing m leaves the ranking of generated candidates induced by

$$\widehat{G}_t^{\mathrm{risk}} = \Delta C_t - \beta[\Delta A_t]_+ \tag{62}$$

unchanged.

Proof. The effective margin appears only in the binary comparison

$$\widehat{G}_t^{\mathrm{risk}} > m \tag{63}$$

and is not part of the score itself. Therefore, for any two generated candidates i and j, whether

$$\widehat{G}_i^{\mathrm{risk}} > \widehat{G}_j^{\mathrm{risk}} \tag{64}$$

holds is independent of m. □

Consequently, threshold-free diagnostics computed from the same candidate scores and candidate set, such as AUROC, AUPRC, and Spearman correlation, are unchanged when only m is varied. In contrast, commitment coverage, HCUR, and predictive performance can change because they depend on the resulting binary commitment decisions.

D.5 Exact Causal Candidate-Budget Guarantee

The candidate-generation controller also provides an exact deterministic budget property. For a target candidate-generation fraction ρ, define

$$K_\rho = \lfloor \rho T \rfloor \tag{65}$$

as the required number of windows on which a tentative candidate is constructed.

Immediately before window t, let

$$r_t = K_\rho - \sum_{j=1}^{t-1} z_j \tag{66}$$

denote the remaining candidate quota and let

$$\ell_t = T - t + 1 \tag{67}$$

denote the number of remaining windows. Recall the causal quota correction

$$z_t = \begin{cases} 0, & r_t = 0, \\ 1, & r_t = \ell_t, \\ \tilde{z}_t, & \text{otherwise}, \end{cases} \tag{68}$$

where $\tilde{z}_t$ is the provisional forward-only screening decision.

Proposition 2 (exact causal candidate quota). For every realization of the provisional screening decisions $\{\tilde{z}_t\}_{t=1}^T$, the controller satisfies

$$\sum_{t=1}^{T} z_t = K_\rho. \tag{69}$$

Proof. At the beginning of every window, feasibility requires

$$0 \le r_t \le \ell_t. \tag{70}$$

If $r_t = 0$, setting $z_t = 0$ preserves feasibility and prevents the quota from being exceeded. If $r_t = \ell_t$, setting $z_t = 1$ is necessary to meet the remaining quota. Otherwise,

$$0 < r_t < \ell_t. \tag{71}$$

Because r_t and ℓ_t are integers, either provisional decision remains feasible: if $z_t = 0$, then $r_{t+1} = r_t \le \ell_t - 1 = \ell_{t+1}$; if $z_t = 1$, then $0 \le r_t - 1 = r_{t+1} \le \ell_t - 1 = \ell_{t+1}$. Thus, the invariant

$$0 \le r_t \le \ell_t \tag{72}$$

is preserved until the stream is exhausted. At $t = T + 1$, $\ell_{T+1} = 0$, so feasibility implies $r_{T+1} = 0$, and hence Eq. (69) follows. □

The controller uses only the current quota state, the current screening score, and screening history from already observed windows. It therefore enforces the prescribed candidate-generation budget without accessing future target observations.

D.6 Commitment Risk–Coverage Interpretation

For a fixed candidate-generation allocation and fixed candidate scores, increasing the effective commitment margin produces a nested sequence of committed sets. Define

$$\mathcal{C}(m) = \left\{ t : z_t = 1,\ \widehat{G}_t^{\mathrm{risk}} > m \right\}. \tag{73}$$

For any $m_2 > m_1$,

$$\mathcal{C}(m_2) \subseteq \mathcal{C}(m_1). \tag{74}$$

Therefore, increasing the margin can only decrease or preserve commitment coverage. It does *not* mathematically guarantee a monotonic reduction in empirical HCUR, because harmful and beneficial candidates need not be perfectly ordered by the estimated utility score.

The decreasing HCUR observed across the evaluated margin sweep in Appendix B.6 is therefore an empirical property of the estimated candidate-score ordering rather than a consequence imposed by the threshold definition. The controller provides a mechanism for trading commitment coverage against estimated candidate risk; the quality of that trade-off depends on how informative $\widehat{G}_t^{\mathrm{risk}}$ is about realized candidate utility.

D.7 Scope of the Formal Results

The preceding results are intentionally limited in scope.

First, G_t is a myopic one-step utility. A candidate that improves predictive risk on the current evaluation window may change persistent model state in ways that help or harm later windows. Consequently, $G_t > 0$ does not imply that committing the candidate is globally optimal over the future adaptation trajectory. The next-window analysis in Section 5 provides an additional retrospective diagnostic but does not establish long-horizon optimality.

Second, Proposition 1 is conditional on an estimation-error bound. The proposed method does not establish a distribution-free guarantee of

$$\left| \widehat{G}_t^{\mathrm{risk}} - G_t \right| \le \epsilon \tag{75}$$

under arbitrary distribution shift.

Third, the reported experiments use fixed normalized candidate cost and therefore validate budgeted candidate generation and conservative candidate commitment, but not optimal scheduling among heterogeneous-cost adaptation operators.

Finally, neither the formal interpretation nor the empirical reduction in HCUR constitutes a formal safety certification. The role of the analysis is to clarify the controller's decision structure and the assumptions under which a positive commitment margin admits a conservative interpretation.

D.8 Summary

The analysis clarifies four properties of the proposed controller. The commitment margin admits a conditional bounded-error interpretation; under constant candidate cost, λ and τ collapse to a single effective margin; changing that margin changes the binary operating point without changing

the underlying score ranking; and the causal candidate-generation controller exactly satisfies its prescribed quota without access to future observations.

Together, these properties formalize the separation between the two control levels used throughout the paper: screening controls where adaptation compute is spent, utility controls whether the realized update persists.

E Baseline Protocols and Fair-Comparison Controls

This appendix specifies the comparison protocols used in the experiments. We distinguish between two classes of comparisons. Mechanism-matched controls share the same candidate-generation operator as Utility-Aware TTA and are designed to isolate the effects of candidate allocation and post-proposal commitment. Literature baselines retain their defining adaptation mechanisms and are used to assess system-level competitiveness against established TTA approaches.

The literature baselines evaluated in this work are TENT (Wang et al., 2021), EATA (Niu et al., 2022), SAR (Niu et al., 2023), and Test-Time Prompt Tuning (TPT) (Shu et al., 2022). Cross-Augmentation Similarity (CAS) (Jiang et al., 2026b) is additionally used as a strong pre-update selection baseline for the complementarity experiment.

E.1 Common Evaluation Protocol

Within each model–dataset–seed setting, compared methods receive the same ordered target stream, preprocessing pipeline, evaluation examples, and corresponding initial model checkpoint. Unless a method intrinsically requires a different adaptation unit, the same stream partitioning is used. The principal window-level experiments use windows of at most 64 observations.

The principal experiments use three independent seeds, $\{0, 1, 2\}$, with identical target-stream realizations across mechanism-matched methods within each seed. The CAS complementarity and next-window analyses use five independent seeds, as specified in the corresponding experimental protocol.

No deployed method receives target labels during adaptation. Target labels from the evaluation streams are revealed only retrospectively for metric computation and oracle analysis. A fresh model state is restored at the beginning of every independent method–dataset–seed trajectory. Within a continual TTA trajectory, adaptation state persists across successive windows unless resetting is an intrinsic component of the baseline.

E.2 Mechanism-Matched and Literature-Level Comparisons

The mechanism-matched controls are Always Adapt, Random Gate, Entropy Gate, and Utility-Aware. They use the same base candidate generator, adaptable parameter set, update objective, optimizer configuration, and target stream. Their differences are restricted to where candidates are generated and whether a realized candidate is evaluated before persistence.

In contrast, TENT, EATA, SAR, and TPT retain their defining optimization, parameter-selection, filtering, regularization, recovery, and state semantics. They are therefore system-level competitors rather than candidate-generator-matched controls. We do not replace their native adaptation mechanisms with the proposed candidate generator merely to make them superficially identical.

This distinction is important: the literature baselines assess external competitiveness, whereas the mechanism-matched controls provide the controlled test of the paper's central post-proposal commitment hypothesis.

E.3 Frozen Model

The Frozen baseline performs ordinary inference without modifying model state, so $\theta_{t+1} = \theta_t = \theta_0$ for all t. It defines the zero-adaptation reference from which changes in predictive performance and calibration are reported unless otherwise stated.

E.4 Always Adapt

Always Adapt uses the same base candidate-generation operator as Utility-Aware: LayerNorm affine parameters are adaptable, the classifier remains frozen, the adaptation objective is entropy minimization, and one AdamW candidate-update step is performed with learning rate 10^{-4}, coefficients $(0.9, 0.999)$, and zero weight decay. Two independently sampled update views per observation are used for candidate construction.

The optimizer state persists across the stream. A candidate is generated on every window and committed unconditionally, so $(\theta_{t+1}, \omega_{t+1}) = (\tilde{\theta}_t, \tilde{\omega}_t)$. No post-proposal candidate-utility evaluation is performed.

Because Always Adapt and Utility-Aware use the same underlying candidate operator, their comparison measures the joint effect of restricting candidate generation and evaluating realized candidates before persistence. The matched-budget controls below separate these two factors.

E.5 Random Matched-Budget Gate

Random Gate uses the same candidate generator as Always Adapt and Utility-Aware but constructs candidates on exactly $K_\rho = \lfloor \rho T \rfloor$ of the T target windows. Immediately before window t, define

$$r_t = K_\rho - \sum_{j=1}^{t-1} z_j, \qquad \ell_t = T - t + 1, \tag{76}$$

where r_t is the remaining candidate quota and ℓ_t is the number of remaining windows.

To obtain a causal uniform sample without replacement while satisfying the exact quota, Random Gate uses

$$z_t \sim \text{Bernoulli}\left(\frac{r_t}{\ell_t}\right), \qquad 0 < r_t < \ell_t, \tag{77}$$

with $r_t = 0$ and $r_t = \ell_t$ deterministically forcing $z_t = 0$ and $z_t = 1$, respectively. Consequently,

$$\sum_{t=1}^{T} z_t = K_\rho \tag{78}$$

without access to future target observations.

Every generated candidate is committed. Random Gate therefore controls for the possibility that performance changes arise merely because adaptation is attempted less frequently.

E.6 Entropy Matched-Budget Gate

Entropy Gate uses the same normalized predictive-entropy screening statistic, causal screening history, exact quota controller, and candidate generator as Utility-Aware. Thus, Entropy Gate and Utility-Aware construct candidates on the same number of windows under the same candidate-generation fraction ρ.

Whenever $z_t = 1$, Entropy Gate commits the resulting candidate unconditionally, so $(\theta_{t+1}, \omega_{t+1}) = (\tilde{\theta}_t, \tilde{\omega}_t)$.

Accordingly, the Entropy Gate versus Utility-Aware comparison holds fixed the target stream, candidate generator, adaptable parameters, optimizer, screening statistic, causal screening protocol, and candidate-generation budget. The additional operation performed by Utility-Aware is evaluation of the realized candidate followed by commit or rollback.

This comparison provides the most direct mechanism-matched test of whether post-proposal candidate information adds value beyond pre-update predictive uncertainty.

E.7 Mechanism-Matched Factorization

Table 18 summarizes the mechanism-matched comparison.

Table 18: Mechanism-matched controls. "Same" denotes the common LayerNorm-only entropy-minimization candidate operator described in Appendix C.

Method	Candidate	Exact budget	Screening	Post-proposal evaluation
Always Adapt	Same	No	None	None
Random Gate	Same	Yes	Random	None
Entropy Gate	Same	Yes	Entropy	None
Utility-Aware	Same	Yes	Entropy	Utility

The progression Always → Random → Entropy → Utility separates three questions: whether reducing candidate-generation frequency alone changes behavior, whether predictive uncertainty improves allocation of candidate-generation compute, and whether evaluating the realized candidate improves the persistence decision.

At the principal operating point, $\rho = 0.50$, Random Gate, Entropy Gate, and Utility-Aware construct candidates on exactly the same number of target windows.

E.8 TENT

TENT (Wang et al., 2021) is evaluated using its standard online entropy-minimization mechanism with a ViT-compatible normalization parameterization. For CLIP and DINOv2, LayerNorm affine scales and biases are adaptable while the task classifier remains frozen.

TENT performs one online update per target window and maintains adaptation state continually across the stream. Optimization uses SGD with learning rate 10^{-3}, momentum 0.9, and zero weight decay. No episodic reset is applied between consecutive target windows.

E.9 EATA

EATA (Niu et al., 2022) retains its defining selective-adaptation and anti-forgetting mechanisms. LayerNorm affine parameters are adaptable and the classifier remains frozen. Optimization uses one update opportunity per target window with SGD, learning rate 10^{-3}, momentum 0.9, and zero weight decay.

Let C denote the number of task classes. Reliable-sample filtering uses $E_0 = 0.4 \log C$, and the redundancy threshold is $\epsilon_{\text{red}} = 0.05$. The anti-forgetting term uses $\alpha_{\text{Fisher}} = 2000$, with Fisher information estimated from $N_{\text{Fisher}} = 2000$ source-side examples.

For ImageNet-C, the Fisher reference set is drawn from clean ImageNet source data. For Office-Home and Camelyon17-WILDS, only the corresponding source-domain or source-training data are used. Target labels are not used in Fisher estimation; pseudo-labels from the fixed source model are used according to the baseline protocol.

EATA's moving prediction statistics persist across the target stream, and its native reliability and redundancy filters determine which observations contribute gradients.

E.10 SAR

SAR (Niu et al., 2023) is evaluated using its native sharpness-aware reliable entropy-minimization mechanism. It adapts the normalization affine parameters selected by the ViT implementation, including its native parameter-selection rule, rather than replacing its update operator with the candidate generator used by Utility-Aware.

Optimization uses SAM with SGD as the base optimizer, with learning rate 10^{-3}, momentum 0.9, zero weight decay, and SAM neighborhood radius $\rho_{\text{SAM}} = 0.05$. Reliable-entropy filtering uses $E_0 = 0.4 \log C$, and the exponential-moving-average recovery threshold is $e_m = 0.2$.

SAR otherwise operates continually across the target stream. Any model recovery triggered by its native criterion is retained because recovery is an intrinsic component of SAR rather than an externally imposed reset.

Table 19: Literature-baseline implementation settings. These methods retain their native adaptation mechanisms and are not candidate-generator-matched controls.

Method	Adapted state	Optimizer	LR	Selection	State protocol
TENT	LN affine	SGD	10^{-3}	None	Continual
EATA	LN affine	SGD	10^{-3}	Reliability + redundancy	Continual
SAR	Selected norm affine	SAM(SGD)	10^{-3}	Reliable entropy	Continual + recovery
TPT	4 prompt tokens	AdamW	5×10^{-3}	Top-10% views	Episodic

E.11 Test-Time Prompt Tuning

TPT (Shu et al., 2022) is evaluated only with CLIP, for which its language-prompt parameterization is applicable. TPT retains its episodic test-time prompt-tuning protocol: the visual and text encoders remain frozen, while four prompt context tokens are optimized for each test example and reset before processing the next example.

The prompt is initialized with `a photo of a`, with $n_{\text{ctx}} = 4$. For each test example, TPT constructs a 64-view augmentation batch and uses the lowest-entropy 10% of views for its confidence-selection objective. One test-time optimization step is performed using AdamW with learning rate 5×10^{-3}, coefficients $(0.9, 0.999)$, and weight decay 0.01.

Because TPT adapts prompt parameters episodically and operates on an augmentation batch for each individual example, its computational unit is not equivalent to the persistent window-level candidate used by Utility-Aware. TPT is therefore included as a literature-level predictive-performance comparison rather than a mechanism-matched control. It is not evaluated with DINOv2 because DINOv2 has no language-prompt interface.

E.12 Literature-Baseline Configuration Summary

Table 19 summarizes the literature-baseline configurations.

E.13 CAS Matched-Budget Complementarity Control

CAS (Jiang et al., 2026b) is used to test whether post-proposal verification remains useful after a strong pre-update selector has already determined where adaptation should be attempted.

CAS Gate and CAS+Utility use the same CAS pre-update selection rule, the same base candidate operator, the same prescribed candidate-generation budget, and the same evaluation seeds. CAS Gate commits every candidate generated by its CAS selection trajectory, whereas CAS+Utility applies the proposed post-proposal utility evaluation before persistence.

Because the two policies can enter different persistent model states after their commitment decisions, subsequent CAS scores—and consequently the specific candidate windows selected later in the trajectory—need not be identical unless those opportunities are explicitly precomputed and frozen. We therefore interpret the matched comparison as holding fixed the pre-update selection mechanism, candidate operator, candidate-generation budget, and evaluation seeds, while varying whether realized candidates undergo post-proposal verification before entering persistent state.

This comparison tests whether adding candidate-level evaluation after strong CAS pre-update selection provides additional decision-relevant information. We do not otherwise equate the CAS selection mechanism with the entropy-based screening rule used by the principal Utility-Aware controller.

CAS Gate and CAS+Utility are evaluated on the same five independent seeds. The principal Utility-Aware operating point is shown separately as a reference and should not be interpreted as a five-seed mean from the matched CAS comparison.

E.14 Oracle Utility Gate

The Oracle Utility Gate is an offline reference rather than a deployable baseline. For a tentative candidate, it observes the retrospective one-step oracle utility $G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t)$ and commits iff $G_t > 0$.

Because computing G_t requires target labels, this controller is simulated only for retrospective oracle analysis and is never available to the deployed adaptation procedure. It measures one-step headroom in candidate-utility estimation; it is not a globally optimal sequential controller.

E.15 Hyperparameter Selection and Target-Label Isolation

Hyperparameters are fixed before evaluation on the principal labeled target streams. Target labels from those streams are not used for optimizer or learning-rate selection, adaptation-parameter selection, reliability-threshold selection, source-head checkpoint selection, prompt initialization, commitment-threshold selection, or per-dataset adaptation-policy tuning.

Literature baselines use the configurations specified above. For Utility-Aware, the principal configuration $(\beta, m, M) = (1.0, 0.05, 4)$ is fixed before inspection of the principal evaluation labels. The sensitivity experiments characterize behavior around this operating point rather than retrospectively redefining it from the final evaluation streams.

CIFAR-10-C is used during development and controlled ablation analysis; labels from the principal ImageNet-C, Office-Home, and Camelyon17-WILDS evaluation streams remain isolated from deployed adaptation decisions.

E.16 Stream Continuity and Reset Semantics

State handling follows each method's intended adaptation semantics.

Continual methods. Always Adapt, Random Gate, Entropy Gate, Utility-Aware, TENT, EATA, and SAR carry their relevant adaptation state across successive target windows. Each independent method–dataset–seed trajectory begins from the corresponding pretrained or source state.

For Utility-Aware, model and AdamW states are transactional: an accepted candidate commits both states, whereas a rejected candidate restores both states, as detailed in Appendix C. SAR's native recovery mechanism is preserved and is not treated as an externally imposed reset.

Episodic method. TPT restores its tunable prompt and optimizer state to their pre-adaptation initialization before processing each new test example, consistent with its episodic protocol.

These conventions are kept explicit because mixing episodic and continual state semantics can materially change TTA behavior.

E.17 Compute Accounting

For the reported adaptation-specific timing comparisons, computational cost is not inferred from the number of committed candidates alone. All computation required by the adaptation procedure is counted, including, where applicable, screening, augmentation generation, auxiliary forward passes, backward passes, candidate construction, candidate verification, frozen-anchor evaluation, state copying, rollback, and computation spent on candidates that are subsequently rejected.

Utility-Aware is therefore charged for the complete screen $\rightarrow$ propose $\rightarrow$ evaluate $\rightarrow$ commit/rollback pipeline. Random Gate and Entropy Gate are not charged for post-proposal verification operations that they do not perform.

Candidate-update backward-pass reduction and wall-clock reduction are reported separately. Methods with equal candidate-generation counts may have different runtime because their forward-pass, augmentation, filtering, verification, and state-management requirements differ.

In the principal ImageNet-C $\rho = 0.50$ comparison, the candidate-update backward-pass reduction is exactly 50% for Random Gate, Entropy Gate, and Utility-Aware relative to Always Adapt because

Table 20: Principal failure modes of label-free candidate-update control. The listed responses are diagnostic or conservative mitigations rather than guarantees.

Failure mode	Diagnostic pattern	Underlying issue	Potential response
Consistently wrong candidate	$\Delta C_t > 0$ despite $G_t < 0$	Probe views agree more strongly after adaptation, but around an incorrect prediction	Frozen-anchor penalty, richer probe views, or a more conservative commitment margin
Over-conservative anchoring	Large $[\Delta A_t]_+$ for a beneficial candidate	The pretrained predictor is itself mismatched to the shifted domain	Reduce β or use a more context-sensitive anchor
Insufficient probe diversity	Noisy or unstable ΔC_t	Too few independent probe views yield a high-variance consistency estimate	Increase M subject to the compute budget
Broken augmentation invariance	Probe disagreement reflects transformation artifacts	The assumed semantic-preserving transformations are inappropriate for the current domain	Use domain-appropriate probe transformations
Over-conservative commitment	Low commitment coverage and reduced adaptation gain	The effective margin rejects both harmful and beneficial candidates	Move to a less conservative risk–coverage operating point
Myopic utility mismatch	A locally beneficial candidate harms later windows, or vice versa	G_t measures one-step rather than long-horizon trajectory utility	Develop sequential or horizon-aware candidate valuation
Candidate-budget misallocation	Useful candidate opportunities are never generated	Screening determines where candidate-generation compute is spent before candidate utility can be evaluated	Improve screening or increase the candidate-generation fraction
Candidate-family mismatch	Available candidates cannot correct the dominant target-domain error	The base adaptation operator is too restricted for the shift	Consider richer candidate operators with appropriate safeguards

the prescribed quota generates candidates on exactly half of the target windows and the underlying candidate backward operation is identical.

E.18 Why Matched-Budget Controls Are Necessary

A selective adaptation policy can appear safer simply because it attempts adaptation less frequently. Candidate-generation opportunity must therefore be controlled when attributing differences in harmful persistent updates to the selection or commitment mechanism.

At the principal $\rho = 0.50$ operating point, Random Gate, Entropy Gate, and Utility-Aware construct candidates on exactly the same number of target windows, yet their mean HCUR values are 24.1%, 18.2%, and 11.0%, respectively. Always Adapt, which constructs and commits a candidate on every window, has mean HCUR 27.0%.

Thus, merely reducing candidate-generation frequency from every window to half of the stream does not account for the reduction achieved by Utility-Aware. Entropy-based allocation improves over random allocation, but post-proposal verification further reduces HCUR from 18.2% to 11.0% under the same candidate-generation budget and screening mechanism.

The corresponding accuracy gains are $+1.10$, $+1.70$, and $+2.40$ percentage points for Random Gate, Entropy Gate, and Utility-Aware, respectively. The mechanism-matched comparison therefore separates candidate frequency, pre-update compute allocation, and post-proposal persistence control.

F Failure Modes and Diagnostic Interpretation

Utility-Aware TTA reduces the frequency with which harmful candidate updates enter persistent model state, but it does not eliminate adaptation failure. This appendix clarifies the main failure modes implied by the controller and the diagnostic interpretation of its principal signals.

The estimated candidate utility $\widehat{G}_t = \Delta C_t - \beta[\Delta A_t]_+ - \lambda \tilde{c}_t$ is a label-free surrogate for the unavailable one-step oracle utility $G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t)$. A candidate can therefore be incorrectly accepted or rejected whenever the surrogate and oracle utility disagree.

F.1 Failure-Mode Summary

Table 20 summarizes the principal failure modes. The listed responses are diagnostic or conservative mitigations rather than guarantees.

F.2 Consistency Can Improve Around an Incorrect Prediction

Held-out consistency is informative because a useful candidate should often make predictions more stable across independently sampled semantic-preserving probe views. However, consistency is not equivalent to correctness.

A harmful candidate may have $\Delta C_t > 0$ if adaptation makes the probe views agree more strongly on the same incorrect class. This motivates the frozen-pretrained anchor rather than relying on probe consistency alone.

The component ablation in Appendix B.4 supports this interpretation: adding held-out post-proposal evaluation reduces HCUR relative to screening alone, and adding the frozen-pretrained anchor reduces it further. Nevertheless, the anchor does not turn consistency into a correctness guarantee. A sufficiently coherent but incorrect candidate can still obtain a positive estimated utility.

F.3 The Frozen Anchor Can Reject Useful Adaptation

The frozen pretrained model is used as a conservative reference, not as a ground-truth predictor. Under a sufficiently large but legitimate domain shift, a beneficial candidate may need to move substantially away from the pretrained predictions. In that case, $[\Delta A_t]_+$ can be large even when $G_t > 0$, causing the candidate utility estimator to reject a genuinely useful update.

The anchor sensitivity results illustrate this trade-off. Increasing β from 0 toward the selected value $\beta = 1$ improves candidate-harm discrimination and the resulting risk–performance trade-off. Stronger anchoring continues to reduce HCUR but eventually suppresses useful adaptation: at $\beta = 4$, HCUR is lower than at $\beta = 1$, while both predictive gain and harmful-update discrimination are reduced.

The anchor should therefore be interpreted as a conservative risk-control prior rather than an assumption that remaining close to the pretrained model is always desirable.

F.4 Probe-View Estimation Has a Reliability–Compute Trade-Off

Candidate evaluation relies on a finite number M of independently sampled probe views. Small M reduces verification cost but increases estimation variance.

The sensitivity analysis shows the trade-off directly. Increasing M from 1 to 4 improves harmful-update AUROC from 0.691 to 0.750 and reduces HCUR from 15.6% to 11.0%. The corresponding accuracy gain increases from $+2.04$ to $+2.40$ percentage points. Increasing M further produces smaller reliability gains: at $M = 8$, AUROC is 0.764 and HCUR is 10.1%, while at $M = 16$, AUROC reaches 0.771 and HCUR 9.7%.

These additional gains incur progressively greater verification cost. Adaptation-specific time is reduced by 28% at the selected $M = 4$ operating point, by 18% at $M = 8$, and is approximately 4% slower than Always Adapt at $M = 16$. Thus, $M = 4$ is an empirical reliability–compute compromise rather than a universally optimal number of probe views.

F.5 Semantic-Preserving Augmentations Are Domain Dependent

The probe mechanism assumes that independently sampled transformations preserve task-relevant semantics. If this assumption fails, disagreement among probe views can reflect transformation artifacts rather than candidate quality. Conversely, an augmentation family that is too weak may provide insufficient variation to expose unstable candidate behavior.

This issue is especially relevant in specialized domains. For the Camelyon17-WILDS cross-hospital stress test, strong color- or stain-altering transformations are disabled and cropping is restricted to the mild range specified in Appendix C.

The probe transformation family should therefore be regarded as part of the task- and domain-specific deployment specification rather than as a universally valid choice.

F.6 Conservative Commitment Trades Risk Against Coverage

Under the fixed-cost setting, increasing the effective commitment margin m produces a more selective set of committed candidates. The removed candidates, however, may include both harmful and beneficial updates.

The finalized margin sweep makes this trade-off explicit. As m increases from -0.05 to 0.25, commitment coverage decreases from 87% to 21% and HCUR decreases from 17.8% to 5.2%. Predictive gain, however, rises from $+2.19$ points to a maximum of $+2.40$ at the selected $m = 0.05$ and then falls to $+1.79$ at $m = 0.25$.

A very low HCUR obtained through aggressive abstention is therefore not sufficient evidence of a desirable operating point. HCUR should be interpreted jointly with commitment coverage, predictive performance, and computational cost.

F.7 Screening Can Miss Useful Candidate Opportunities

Post-proposal utility can be evaluated only after a window has been selected for candidate generation. When $z_t = 0$, no tentative candidate is constructed and no realized candidate utility can be estimated.

This creates an important asymmetry in the two-stage controller: candidate utility can reject a poor proposed update, but it cannot recover a beneficial candidate from a window for which screening never allocates candidate-generation compute.

The candidate-generation-fraction sweep partly characterizes this limitation. Predictive gain generally increases as more candidate opportunities are made available, indicating that aggressive compute restriction can discard useful adaptation opportunities. This is why screening is interpreted primarily as a compute-allocation mechanism, whereas post-proposal utility governs whether a realized update should persist.

F.8 One-Step Utility Is Not Long-Horizon Utility

The oracle quantity $G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t)$ measures immediate candidate benefit. In a persistent adaptation process, however, committing a candidate also changes the initialization from which later predictions and updates are produced.

A candidate with small immediate benefit may improve the state from which the model responds to later shifts. Conversely, an immediately beneficial candidate may move the model into a state from which later adaptation becomes less stable. Neither the one-step oracle nor the proposed label-free estimator captures these counterfactual long-horizon effects.

The next-window analysis provides evidence that the candidate score retains predictive value beyond the current evaluation window, but it remains a one-step future diagnostic rather than a solution to the full sequential control problem. The present work therefore asks whether the current realized candidate should persist, not which sequence of updates maximizes cumulative future utility.

F.9 The Candidate Generator Limits What the Controller Can Repair

The reported candidate generator adapts LayerNorm affine parameters in the visual backbone while keeping the task classifier frozen. This design isolates candidate-level commitment and limits the complexity of the adaptation operator, but it also constrains the class of errors that can be corrected.

If the dominant target-domain error arises from classifier mismatch or from a representation change outside the capacity of LayerNorm-only adaptation, then even an ideal candidate-utility estimator cannot produce a useful state transition if the base adaptation operator does not generate one.

Accordingly, the oracle headroom reported in the experiments should be interpreted as headroom within the evaluated candidate family rather than as the maximum performance attainable by arbitrary TTA. A natural extension is to evaluate multiple candidate operators—for example backbone, prompt, or classifier updates—under a common post-proposal utility controller.

Table 21: Qualitative interpretation of the two principal candidate signals.

ΔC_t	ΔA_t	Interpretation
Large positive	Small/non-positive	Stronger evidence in favor of commitment
Large positive	Large positive	Improved consistency with substantial pretrained drift
Near zero	Small	Weak evidence of candidate benefit
Negative	Any	Candidate destabilizes held-out predictions

F.10 Extreme or Unseen Shifts Can Break Both Utility Components

The experiments cover synthetic corruption, natural domain shift, and a cross-hospital distribution-shift stress test, but they do not span all possible deployment conditions.

Under sufficiently severe or qualitatively different distribution shifts, both terms in the utility estimator can become unreliable. The consistency term ΔC_t may reward stable but semantically incorrect predictions, while the anchor term ΔA_t may penalize movement away from a pretrained model that is itself poorly aligned with the target domain.

The observed reduction in HCUR should therefore not be extrapolated into a distribution-free reliability or safety guarantee. Consequential deployment would require additional domain-specific evaluation, monitoring, operational rollback, and other appropriate safeguards.

F.11 Diagnostic Interpretation of Candidate Signals

The two principal candidate signals can remain diagnostically useful even when the final binary commitment decision is imperfect. Table 21 summarizes their qualitative interpretation.

The second case is particularly important. Increased probe consistency alone is insufficient evidence when the candidate simultaneously increases disagreement with the frozen pretrained reference. This interaction motivates combining the two quantities rather than thresholding either independently.

F.12 What the Controller Does and Does Not Guarantee

Two deterministic properties should be distinguished from the empirical reduction in harmful commitments.

First, the causal candidate-generation controller guarantees exactly $\sum_{t=1}^{T} z_t = \lfloor \rho T \rfloor$ generated candidates. Second, transactional commit/rollback guarantees that a rejected candidate does not alter the persistent model or optimizer state.

These properties do *not* imply that every committed candidate is beneficial, every rejected candidate is harmful, HCUR decreases monotonically under arbitrary distribution shifts, the selected margin is optimal for every deployment, the frozen anchor is always reliable, the probe transformation family always preserves semantics, or the one-step commitment policy is globally optimal. The method should not be interpreted as a formal safety mechanism.

The empirical contribution is therefore a reduction in observed harmful candidate persistence under the evaluated conditions, not elimination of adaptation failure.

F.13 Summary

The primary failure modes arise from imperfect correspondence between label-free candidate diagnostics and true predictive utility. Consistency can increase around an incorrect prediction; the pretrained anchor can become over-conservative; finite or domain-inappropriate probe views can distort candidate evaluation; conservative thresholds can suppress useful adaptation; and screening can prevent valuable candidate opportunities from being generated.

These limitations reinforce the distinction underlying the paper: pre-update screening determines where adaptation compute is allocated, whereas post-proposal candidate utility estimates whether the realized update should enter persistent model state. Neither signal is infallible.

G Dataset, Stream, and Evaluation-Unit Construction

This appendix specifies how each benchmark is converted into the causal target streams used for test-time adaptation. We make explicit the shift units, stream ordering, reset semantics, window construction, information available online, and labels used only for retrospective evaluation.

G.1 General Stream Construction

Each model–dataset experiment consists of one or more predefined shift units. A shift unit is processed as an independent continual adaptation trajectory.

For shift unit u, let $\mathcal{D}^{(u)} = \{(x_i, y_i)\}_{i=1}^{N_u}$. For experiment seed s, a deterministic random permutation $\pi^{(u,s)}$ is generated once before adaptation. Mechanism-matched methods within the same shift-unit–seed condition receive exactly the same ordered target observations. Literature baselines use the same underlying ordered examples wherever their native evaluation unit permits a direct stream match.

The permutation is independent of target labels and is introduced to prevent results from depending on arbitrary storage or class-directory ordering.

The ordered target observations are partitioned sequentially into windows B_t containing at most 64 observations. The final incomplete window is retained rather than discarded, and no observation is duplicated solely to construct a full final window. Thus, a shift unit containing N_u examples produces $T_u = \lceil N_u/64 \rceil$ target windows.

G.2 Reset Semantics Across Shift Units

Adaptation state persists across successive windows within the same shift unit. Independent shift units, however, do not share adapted state. Before a new shift unit begins, the relevant model, classifier, and optimizer states are restored to their corresponding pre-adaptation initialization.

Consequently, performance on one corruption type, domain-transfer direction, or benchmark split cannot alter the initialization of another independently reported shift unit. This reset rule is applied consistently to methods for which the corresponding benchmark unit defines an independent adaptation trajectory.

G.3 Causal Information Constraint

At target window t, the deployed controller can use only the current and previously observed target windows together with persistent state produced by previously committed updates.

Specifically, the controller may access the current unlabeled observations, current-model predictions, frozen-pretrained-reference predictions, independently sampled update and probe views, screening statistics from the current and previous windows, the remaining candidate-generation quota, and persistent model and optimizer state produced by previously committed candidates.

Target labels are unavailable to all deployed screening, candidate-generation, evaluation, and commitment decisions. Future target observations are likewise not inspected when making the current decision. Target labels are revealed only retrospectively for evaluation and computation of oracle quantities.

G.4 ImageNet-C

ImageNet-C (Hendrycks & Dietterich, 2019) provides corrupted versions of the ImageNet-1K validation set. We use the official pre-generated benchmark images rather than synthesizing corruptions online.

The primary evaluation uses all 15 standard corruption types at severity 5: Gaussian noise, shot noise, impulse noise, defocus blur, glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic transform, pixelate, and JPEG compression.

Each corruption type at severity 5 constitutes an independent shift unit. Within each corruption, the 50,000 corrupted ImageNet validation examples are permuted using the experiment seed and processed in consecutive windows of at most 64 observations. Each trajectory therefore contains 782 windows, with 16 observations in the final window.

At the principal candidate-generation fraction $\rho = 0.50$, the exact candidate-generation quota is $K_{0.5} = \lfloor 0.5 \times 782 \rfloor = 391$ windows.

The ImageNet classifier remains fixed throughout each trajectory. CLIP uses the frozen text classifier described in Appendix C, whereas DINOv2 uses its frozen ImageNet linear classifier. Aggregate ImageNet-C results are computed across the 15 independent corruption trajectories; adapted state is not carried from one corruption type into the next.

G.5 OFFICE-HOME

Office-Home (Venkateswara et al., 2017) contains 65 object classes across four domains: Art (Ar), Clipart (Cl), Product (Pr), and Real-World (Rw). The experiments evaluate all 12 directed source-to-target transfer settings: Ar→Cl, Ar→Pr, Ar→Rw, Cl→Ar, Cl→Pr, Cl→Rw, Pr→Ar, Pr→Cl, Pr→Rw, Rw→Ar, Rw→Cl, and Rw→Pr.

Each directed transfer is treated as an independent adaptation trajectory. For DINOv2, the task-specific linear classifier is trained using only labeled source-domain data as described in Appendix C, and the selected classifier is frozen before the target trajectory begins. For CLIP, the frozen text classifier is constructed independently of target labels and remains fixed.

For each directed transfer and seed, target-domain examples are permuted once using the seed-specific permutation and then processed sequentially in windows of at most 64 observations. The final incomplete window is retained.

After a directed-transfer trajectory is completed, its adapted model and optimizer state are discarded. The next transfer starts from its corresponding unadapted initialization. Target-domain labels remain hidden throughout the online trajectory and are revealed only retrospectively for evaluation.

The reported Office-Home results aggregate the 12 directed transfer conditions, with each transfer treated as a separate shift unit.

G.6 CAMELYON17-WILDS

Camelyon17-WILDS (Koh et al., 2021) is included as a cross-hospital distribution-shift stress test. We use the official WILDS split scheme. Under the dataset's hospital indexing, center 1 is the OOD validation hospital and center 2 is the OOD test hospital; the remaining source hospitals contribute to the official training distribution.

The DINOv2 classifier is trained from the official labeled source training split, with the official in-distribution validation split used for source-side model selection. The selected classifier is frozen before test-time adaptation. The principal adaptation trajectory uses the official OOD test split as the unlabeled target stream; its target labels are not exposed to the adaptation algorithm.

Target observations are permuted once for each experiment seed and processed sequentially in windows of at most 64 observations. Any final incomplete window is retained. Each method–seed run begins from the same DINOv2 source model and frozen classifier state. Hospital metadata are retained for dataset bookkeeping and retrospective evaluation but are not supplied as privileged input to either screening or candidate-utility estimation.

Relationship to the WILDS leaderboard protocol. The experiment evaluates an externally pretrained vision foundation model under the Camelyon17-WILDS distribution shift and is not presented as a standard WILDS leaderboard submission. Its purpose is to provide a controlled cross-hospital stress test of test-time adaptation rather than direct comparability with leaderboard settings that impose different restrictions on external pretraining. Likewise, the three-seed TTA protocol is the common random-seed protocol used in our adaptation experiments and should not be interpreted as reproducing every replication requirement of the separate WILDS leaderboard.

Domain-appropriate transformations. The Camelyon17 update and probe transformations use the same mild semantic-preserving family but are sampled independently. Random resized cropping uses scale $[0.8, 1.0]$ and horizontal flipping is applied with probability 0.5. Strong color, stain, and other appearance-altering transformations are disabled.

Camelyon17-WILDS is used solely as a distribution-shift stress test. These experiments do not constitute clinical validation or evidence for autonomous medical deployment.

G.7 CIFAR-10-C Development Protocol

CIFAR-10-C (Hendrycks & Dietterich, 2019) is used for development, debugging, controlled ablations, and implementation checks rather than as a principal benchmark supporting the paper's main claims.

It is used to check causal stream processing, candidate construction, transactional commit/rollback, exact candidate-generation quota enforcement, metric computation, seed determinism, and controlled ablations.

CIFAR-10-C is used during development and controlled ablation analysis. Target labels from the principal ImageNet-C, Office-Home, and Camelyon17-WILDS evaluation streams are not used to select the reported operating point. The principal configuration $(\beta, m, M) = (1.0, 0.05, 4)$ is fixed before inspection of the final target labels used for the principal reported evaluations.

G.8 Window-Level Candidate Definition

The fundamental adaptation decision is made at the target-window level. When candidate-generation compute is allocated to window B_t ($z_t = 1$), the base adaptation operator constructs one tentative state

$$\tilde{\theta}_t = \mathcal{U}\left(\theta_t; V_t^{\mathrm{upd}}\right), \tag{79}$$

where V_t^{upd} denotes the independently sampled update views.

A candidate is therefore the realized proposed state transition $\theta_t \to \tilde{\theta}_t$. Utility-Aware evaluates that candidate using independently sampled probe views of the same underlying observations, with probe draws sampled separately from the update views, before deciding whether the candidate should enter persistent state.

For retrospective evaluation, its one-step oracle utility is $G_t = R_t(\theta_t) - R_t(\tilde{\theta}_t)$. The unit used for candidate-harm analysis is therefore the generated candidate at window t, not an individual augmented view. A candidate is retrospectively harmful iff $G_t < 0$.

G.9 HCUR and Commitment Coverage

Let $a_t \in \{0, 1\}$ indicate whether the candidate generated at window t is committed, with $a_t \le z_t$. The Harmful Committed-Update Rate is

$$\mathrm{HCUR} = \frac{\sum_t \mathbb{I}[a_t = 1 \wedge G_t < 0]}{\sum_t \mathbb{I}[a_t = 1]}. \tag{80}$$

HCUR measures the fraction of updates entering persistent model state that are retrospectively harmful. It is defined only when at least one candidate is committed; if $\sum_t a_t = 0$, HCUR is reported as undefined rather than being assigned an artificial value of zero.

Commitment coverage among generated candidates is

$$\mathrm{CommitRate} = \frac{\sum_t a_t}{\sum_t z_t}. \tag{81}$$

This commitment-coverage metric is reported alongside HCUR in the experimental results. Reporting both quantities prevents an extremely conservative policy that commits almost no candidates from appearing artificially safe solely because of abstention.

G.10 Candidate-Generation Budget Versus Commitment

Candidate generation and candidate commitment occur at distinct control stages. For a trajectory of length T, the candidate-generation controller allocates exactly

$$\sum_{t=1}^{T} z_t = K_\rho = \lfloor \rho T \rfloor \tag{82}$$

windows to candidate construction, whereas the number of candidates entering persistent state satisfies $\sum_{t=1}^{T} a_t \leq K_\rho$.

Thus, the candidate-generation budget is not the commitment budget. At $\rho = 1$, every target window generates a candidate, but Utility-Aware can still reject candidates after evaluating their realized behavior. At smaller ρ, potentially useful candidates may never be generated because screening first determines where candidate-generation compute is allocated.

G.11 Exact Causal Candidate-Generation Budget

For each independent trajectory of known length T, the total candidate-generation quota $K_\rho = \lfloor \rho T \rfloor$ is fixed before adaptation begins. Knowledge of T determines only the total computational budget and does not expose future target observations.

Immediately before window t, the controller uses the remaining quota $r_t = K_\rho - \sum_{j<t} z_j$ and the number of remaining windows $\ell_t = T - t + 1$, together with the provisional screening decision computed from current and previously observed screening information.

The causal quota correction described and proved in Appendix D.5 guarantees $\sum_{t=1}^{T} z_t = K_\rho$ for every trajectory without inspecting future target observations.

G.12 Aggregation Across Shift Units

Evaluation preserves the natural shift-unit structure. For ImageNet-C, the 15 corruption trajectories constitute the shift units. For Office-Home, the 12 directed source-to-target transfers constitute the shift units. Camelyon17-WILDS contributes its independent cross-hospital test trajectory.

Metrics are computed from completed trajectories and subsequently aggregated according to the corresponding results table. For statistical comparisons spanning multiple shift units, individual observations or windows are not treated as exchangeable independent replicates. The hierarchical bootstrap described in Appendix A first resamples shift units and then resamples observations or windows within the selected units, preserving between-shift variation.

G.13 Matched Stream Realizations

Within a seed, Always Adapt, Random Gate, Entropy Gate, and Utility-Aware receive exactly the same ordered target observations. The same principle is used for literature baselines wherever their native evaluation unit permits a direct stream match.

Random Gate randomness affects only candidate-generation allocation; it does not alter the underlying target permutation. Differences among the mechanism-matched trajectories therefore cannot be attributed to different target-example orderings.

G.14 Dataset-Protocol Summary

Across the principal benchmarks, target observations are permuted deterministically by seed, processed causally in windows of at most 64 observations, and any final incomplete window is retained. Independent shift units begin from their corresponding pre-adaptation initialization.

G.15 Summary

The evaluation protocol maintains a strict separation between information available during online adaptation and labels used for retrospective evaluation. Each distribution-shift condition defines

Table 22: Dataset and stream protocol used in the experiments.

Dataset	Shift unit	Conditions	Target-label use
ImageNet-C	Corruption	15 corruptions, severity 5	Retrospective only
Office-Home	Directed transfer	12 source→target tasks	Retrospective only
Camelyon17-WILDS	OOD test hospital	Official WILDS test split	Retrospective only
CIFAR-10-C	Development condition	Development / ablation	Not used for final target tuning

an explicit adaptation trajectory; state is reset between independent shift units; target ordering is matched across mechanism-matched methods and does not depend on labels; incomplete final windows are retained; candidate-generation allocation is causal and satisfies its prescribed quota without future-observation inspection; and target labels are revealed only retrospectively.

The fundamental evaluation object is therefore a realized proposed model-state transition, $\theta_t \to \tilde{\theta}_t$, rather than merely an uncertain target observation. This construction directly supports the central question of the paper: whether a realized candidate update can be evaluated without target labels before it is allowed to enter persistent model state.