No-Regret Caching with Delayed Hits

Bowen Jiang
Zhengzhou University
jiangbowen@zzu.edu.cn

Chaofan Ma
Shanghai Jiao Tong University
chaofanma@sjtu.edu.cn

Qi Li
Shanghai Jiao Tong University
liqilcn@sjtu.edu.cn

Duo Wang*
Shanghai Jiao Tong University
wangduoup@sjtu.edu.cn

Abstract

Delayed hits occur when requests arrive while a previous miss for the same item is still being fetched. They break the equivalence between hit ratio and latency because caching decisions depend on frequency, burst structure, and fetch latency. We develop a regret framework for delayed-hit caching, with static regret for stationary workloads and switching regret for piecewise-changing workloads. We show that classical deterministic policies and randomized MARKER fail to achieve no-regret guarantees, both suffering linear static regret. For stochastic arrivals, we introduce delayed-hit shadow values and propose UCB-DH, which achieves $O(\log T)$ static regret on fixed instances with positive gaps, where T is the horizon. A matching gap-dependent lower bound shows that the delayed-hit latency-amplification factor is statistically unavoidable, and restarted UCB-DH gives sublinear regret for piecewise-changing workloads. For oblivious adversarial arrivals, we formulate an action-independent shadow loss and propose LA-FTPL with latency-scaled perturbations and completion-time admission. LA-FTPL achieves $O(\sqrt{KT \log N\,(Z^2 + C_B Z)})$ shadow static regret, where K is the cache capacity, N the item universe size, Z the fetch latency, and C_B the maximum one-window delayed-hit penalty. With periodic restarts, it achieves $O(T^{2/3} B^{1/3} [ZK \log N(Z^2 + C_B Z)]^{1/3})$ switching shadow regret against comparators with at most $B = o(T)$ cache changes. Experiments show that our algorithms achieve sublinear regret and improve latency over strong baselines.

1 Introduction

Caching is a core component of recommendation systems [1], content delivery networks [2], wireless networks [3, 4], and storage systems [5]. In the classical model, a request is a *hit* if the item is already cached and can be served immediately, and a *miss* otherwise. Most classical caching objectives are therefore built around hit ratio, miss cost, competitive ratio with prediction [6, 7, 8], or online regret [9, 10, 11, 12, 13, 14]. In modern systems, however, backend retrieval latency can be comparable to or larger than the inter-arrival time of requests. This creates *delayed hits* [15]: requests that arrive while a previous miss for the same item is still being fetched. Such requests are not hits immediately in the latency sense, because they must wait for the ongoing fetch to complete. Figure 1 illustrates this effect. A miss for item C at time $t = 3$ initiates a fetch that completes at $t = 6$; requests for C at $t = 4$ and $t = 5$ are delayed hits and incur residual latency. In this case, minimizing latency is no longer equivalent to maximizing hit ratio.

*Corresponding author.

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

Significant research focuses on caching with delayed hits. [15] formally defines this phenomenon and proposes heuristic algorithms to minimize total latency. To address heterogeneous request sizes and variable latency, a timer-based ranking function is developed in [16] to enhance performance. Recent work also considers non-stationary request sequences in the context of delayed hits [17]. Furthermore, the online caching problem with delayed hits has been investigated in multi-server edge networks [18], while [19] optimizes both latency and cost simultaneously for edge service caching. From a theoretical perspective, [20] and [21] introduce a novel metric for cumulative latency and derive a competitive ratio of $O(Z \log K)$, where K denotes the cache capacity and Z represents the fetch latency parameter. Additionally, it has been shown that the LRU algorithm is $O(ZK)$-competitive [22]. While these works provide valuable insights, they often rely on heuristics without theoretical guarantees [23, 24] or focus on competitive analysis [25, 26, 27]. However, since the competitive ratio metric is multiplicative in nature [27], a bounded competitive ratio does not by itself imply vanishing additive regret. This limitation leads to a fundamental question:

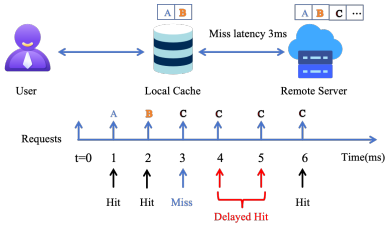


Figure 1: Hit, miss, and delayed hits under non-negligible fetch latency.

> *For caching problems with delayed hits, do there exist online learning algorithms that achieve sublinear regret?*

Answering this question is challenging for three reasons. First, regret benchmarks are harder to define under delayed hits. The latency of an uncached request depends on whether a fetch for the same item is already pending, but the physical pending state is determined by the online policy's previous actions. Thus, using physical pending states directly would make the comparator loss algorithm-dependent. Second, the value of an item is no longer its request frequency. Under delayed hits, isolated requests and bursty requests have different amortized latency effects, so a more frequent item can be less valuable to cache than a less frequent but less bursty item. Third, online learning algorithms must respect delayed admission. A missed item can be inserted only when its fetch completes, while the learning rule may wish to update the cache earlier. This creates a gap between the ideal learning decision and the physically feasible cache update, and the gap must be controlled in the regret analysis.

We address these challenges by building a delayed-hit regret framework around an action-independent shadow fetch process, a virtual process that tracks fetch availability independently of the algorithm's realized cache trajectory and thereby defines a comparator whose delayed-hit values do not depend on the learner's physical admission decisions. This gives a static delayed-hit comparator for stationary workloads and a switching comparator for piecewise-changing workloads. On top of this benchmark, we design two learning algorithms. In the stochastic setting, UCB-DH learns delayed-hit shadow values rather than request frequencies; its estimator amortizes burst penalties so that bursty items are not systematically overvalued. In the adversarial setting, LA-FTPL applies latency-scaled perturbations to the shadow loss and implements cache updates lazily at fetch-completion times. Together, these algorithms show that sublinear regret is achievable once the delayed-hit value structure and physical admission constraints are explicitly modeled. Our contributions are as follows:

- **Delayed-hit regret benchmarks.** We formalize static and switching delayed-hit regret using an action-independent shadow fetch process. Static regret captures stationary workloads, while switching regret captures piecewise-changing workloads. Under these benchmarks, classical caching policies are not no-regret: deterministic policies can suffer linear regret, and randomized MARKER has linear static regret even with cache size one. This shows that delayed-hit caching requires learning latency-aware item values rather than relying only on recency, frequency, or classical randomization.
- **Stochastic delayed-hit learning.** For stationary stochastic arrivals, we propose UCB-DH, which ranks items by delayed-hit shadow value rather than request frequency. We prove a gap-dependent static regret bound of order $O(\sum_{j \notin S^\star} C_B^2 \log T/\Delta_j)$ up to lower-order terms, yielding $O(\log T)$ regret for fixed instances with positive gaps, and show a matching lower bound in the same dependence. We also extend UCB-DH to piecewise-stationary workloads: restarting at known change points gives $O((B+1)\log T)$ regret, while periodic restarts yield sublinear $\widetilde{O}(\sqrt{TB})$-type regret when changes are unknown.

- **Adversarial delayed-hit learning.** For oblivious adversarial arrivals, we define an action-independent shadow loss and propose LA-FTPL, which uses latency-scaled perturbations and completion-time cache admission. We prove an $O(\sqrt{KT \log N\,(Z^2 + C_B Z)})$ shadow static regret and an $\Omega(Z\sqrt{T})$ lower bound, identifying the latency-scaled difficulty of adversarial delayed-hit learning. For switching comparators with at most $B = o(T)$ changes, RESTART-LA-FTPL achieves $O(T^{2/3}B^{1/3}[ZK \log N(Z^2 + C_B Z)]^{1/3})$ switching shadow regret.
- **Empirical validation.** Experiments on stationary synthetic, adversarial, and MovieLens traces show that our algorithms achieve sublinear static and switching regret and reduce latency relative to the state-of-the-art baselines. They also confirm that policies with competitive-ratio guarantees can still incur nearly linear delayed-hit regret, highlighting the need for latency-aware regret benchmarks.

2 Delayed-Hit Caching Model

We consider a cache of capacity K serving requests $\mathcal{R} = (r_1, \ldots, r_T)$ over an item universe $[N]$. Items have unit size, so the cache can store at most K items. A missed item is fetched from the backend and becomes available after Z slots. Throughout the paper, define the one-window delayed-hit penalty

$$C_B := C_B(Z) = \sum_{s=0}^{Z-1} (Z - s) = \frac{Z(Z+1)}{2}. \tag{1}$$

This is the maximum aggregate latency generated by one miss episode within a single fetch window; it does not scale with the horizon T.

Physical delayed-hit dynamics. At the beginning of slot t, all fetches completed at time t become available. When the cache is full, it evicts one cached item and inserts the fetched item into cache. After processing completions, request r_t is served. Let $\mathcal{C}_t$ be the cache before serving r_t, and let $\mathcal{P}_t = \{i : \tau_i < t < \tau_i + Z\}$ be the set of items currently being fetched, where τ_i is the most recent fetch initiation time of item i. The physical latency is

$$\ell_t = \begin{cases} 0, & r_t \in \mathcal{C}_t, \\ Z - (t - \tau_{r_t}), & r_t \in \mathcal{P}_t, \\ Z, & r_t \notin \mathcal{C}_t \cup \mathcal{P}_t. \end{cases} \tag{2}$$

In the last case, a new fetch for r_t is initiated and $\tau_{r_t} = t$.

Static and switching comparators. To evaluate a fixed cache under delayed hits, we use an action-independent shadow fetch process. For each item i, if a request to i arrives while no shadow fetch of i is active, it incurs latency Z and starts a shadow fetch of length Z; subsequent requests to i within this window incur residual latencies. Let $g_{t,i} \in [0, Z]$ be this shadow latency. Then $g_{t,i} = 0$ unless $r_t = i$, and g_t depends only on the request trace and Z. For a fixed cache $S \subseteq [N]$ with $|S| \le K$, items in S are always served immediately, and its delayed-hit static loss is

$$L_T(S) = \sum_{t=1}^{T} \sum_{i \notin S} g_{t,i}. \tag{3}$$

Let $S^\star \in \arg\min_{S \subseteq [N],\, |S| \le K} L_T(S)$. For an online policy $\mathcal{A}$, with physical latency $L_T(\mathcal{A}) = \sum_{t=1}^{T} \ell_t$, the static delayed-hit regret is

$$\mathrm{Reg}_T^{\mathrm{stat}}(\mathcal{A}) = L_T(\mathcal{A}) - L_T(S^\star). \tag{4}$$

Static regret captures long-term placement. To model changing workloads, we also use a switching comparator. For $B \ge 0$, define $\mathcal{S}_B = \left\{ (S_1, \ldots, S_T) : |S_t| \le K,\ \sum_{t=2}^{T} \mathbf{1}\{S_t \ne S_{t-1}\} \le B \right\}$. For $S_{1:T} \in \mathcal{S}_B$, let $L_T(S_{1:T}) = \sum_{t=1}^{T} \sum_{i \notin S_t} g_{t,i}$. The B-switching regret is

$$\mathrm{Reg}_T^{\mathrm{sw}}(\mathcal{A}; B) = L_T(\mathcal{A}) - \min_{S_{1:T} \in \mathcal{S}_B} L_T(S_{1:T}). \tag{5}$$

The case $B = 0$ recovers static regret.

2.1 Why Classical Policies Are Insufficient

We next show that delayed-hit regret minimization cannot be obtained by directly applying classical caching rules. The lower bounds below highlight two obstacles: deterministic admission can amplify one eviction mistake into a full delayed-hit episode, and classical randomized competitive algorithms need not have vanishing additive regret. We first consider deterministic demand-admission policies, which insert a missed item when its fetch completes and evict one cached item if the cache is full.

Proposition 1. *For any deterministic demand-admission policy with capacity K and latency Z, and for sufficiently large horizon T, there exists a delayed-hit request sequence of length T such that*

$$\mathrm{Reg}_T^{\mathrm{stat}} = \Omega(ZT).$$

In particular, for fixed Z, the regret is linear in T.

The proof is deferred to Appendix C.1. The construction uses trigger bursts to force deterministic admissions outside a fixed reference cache, followed by victim bursts on displaced reference items. The static comparator serves each victim burst for free, whereas the online policy pays a full delayed-hit episode cost. Thus delayed hits turn a wrong admission into latency-amplified regret.

We next consider MARKER [28], a canonical randomized paging algorithm that evicts a uniformly random unmarked item on a miss and resets marks phase by phase. Although it is competitive in classical caching [29], it is not a no-regret learner for the delayed-hit static benchmark.

Proposition 2 (Linear regret of MARKER)**.** *For every $Z \geq 1$, there exists a delayed-hit instance with cache capacity $K = 1$ such that, for infinitely many horizons T,*

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}(\textsc{Marker})] \geq \frac{Z}{Z+2}T - O(Z) = \Omega(T).$$

The proof is deferred to Appendix C.2. It uses a cyclic sequence over $Z + 2$ items, so repeated requests to the same item are spaced more than one fetch window apart. Hence the linear regret does not rely on overlapping delayed hits.

Remark 1 (Why delayed-hit-aware learning is needed)**.** *The two lower bounds show that delayed-hit regret is not controlled by classical admission, eviction, or randomization principles alone. Sublinear regret requires ranking items by delayed-hit value, which accounts for burst structure and fetch-window latency. This motivates the delayed-hit shadow values learned by UCB-DH in Section 3 and the latency-scaled shadow-loss approach used by LA-FTPL in Section 4.*

3 Stochastic Arrivals

We first study stationary stochastic arrivals. The key difficulty is that delayed hits make the value of an item episode-dependent. A naive frequency estimator treats all requests equally, while a naive latency estimator can overcount repeated requests within the same fetch window. UCB-DH addresses this by learning an amortized delayed-hit value: isolated requests are charged as single miss episodes, and bursty requests are charged at the episode level.

3.1 Burst Episodes and Delayed-Hit Values

For each item i, requests arrive in episodes. Let $D_i \in \{0, 1\}$ indicate the episode type: $D_i = 0$ is an isolated request with length $L_i = 1$, and $D_i = 1$ is a burst with $L_i \in \{2, \ldots, Z\}$ and $\bar{L}_i = \mathbb{E}[L_i \mid D_i = 1]$. Let p_i be the stationary request fraction of item i, and let $q_i = \Pr(D_i = 1 \mid r = i)$. An uncached episode of length ℓ generates total latency $g_Z(\ell) = \sum_{s=0}^{\ell-1}(Z - s) = \ell Z - \ell(\ell - 1)/2$. Thus the value of caching an item depends not only on how often it is requested, but also on how its requests are grouped into episodes. Instead of estimating the full distribution of burst lengths, UCB-DH uses a conservative amortized episode value. We define

$$V_i = p_i \left[(1 - q_i)Z + q_i \frac{C_B}{\bar{L}_i} \right]. \tag{6}$$

The term $(1 - q_i)Z$ charges isolated requests as ordinary misses, while $q_i C_B/\bar{L}_i$ spreads one burst-episode penalty over the average number of burst requests. This representation has two advantages. First, it ranks items by avoided delayed-hit latency rather than by frequency. Second, its plug-in estimator reduces to a bounded empirical average of episode contributions, which enables the UCB concentration analysis.

3.2 UCB-DH

UCB-DH learns the values V_i online. It must solve two tasks simultaneously: estimate the amortized value of each item from observed requests, and use optimistic indices to decide which cached item to evict when a fetched item becomes available. For item i, let $N_i(t)$ be the number of observed requests, $B_i(t)$ the number of requests classified as bursty, and $E_i(t)$ the number of detected burst episodes. UCB-DH estimates $\widehat{p}_i(t) = \frac{N_i(t)}{t}$, $\widehat{q}_i(t) = \frac{B_i(t)}{\max\{1, N_i(t)\}}$, and $\widehat{L}_i(t) = \max\left\{1, \frac{B_i(t)}{\max\{1, E_i(t)\}}\right\}$. The empirical value $\hat{V}_i(t)$ is obtained by plugging these estimates into Eq. (6). The optimistic index is

$$I_i(t) = \hat{V}_i(t) + C_B \sqrt{\frac{2 \log t}{\max\{1, N_i(t)\}}}, \tag{7}$$

with $I_i(t) = +\infty$ for unobserved items. When a fetch for item p completes at time t, UCB-DH executes a completion-time selective admission rule: if the cache is not full, p is directly admitted. If the cache is full, UCB-DH computes the indices of p and all cached items $j \in \mathcal{C}_t$ using the latest statistics, identifying $j_{\min} \in \arg\min_{j \in \mathcal{C}_t} I_j(t)$. Item p is admitted to evict $j_{\min}$ if and only if $I_p(t) > I_{j_{\min}}(t)$; otherwise, the completed item p is bypassed and the cache remains unchanged. Equivalently, the cache retains the top-K indexed items in $\mathcal{C}_t \cup \{p\}$.

Algorithm 1: UCB-DH Selective Admission

Input: cache $\mathcal{C}_t$, completed fetch p, stats $\{N_i, B_i, E_i\}$
for all $j \in \mathcal{C}_t \cup \{p\}$ **do**
  if $N_j(t) > 0$ **then**
    $\widehat{p}_j \leftarrow N_j(t)/t$
    $\widehat{q}_j \leftarrow B_j(t)/N_j(t)$
    $\widehat{L}_j \leftarrow \max\{1, B_j(t)/\max\{1, E_j(t)\}\}$
    $\widehat{V}_j \leftarrow \widehat{p}_j[(1-\widehat{q}_j)Z + \widehat{q}_j C_B/\widehat{L}_j]$
    $b_j \leftarrow C_B\sqrt{2\log t/N_j(t)}$
    $I_j \leftarrow \widehat{V}_j + b_j$
  else
    $I_j \leftarrow +\infty$
  end if
end for
if $|\mathcal{C}_t| < K$ **then**
  $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \{p\}$
else
  $j_{\min} \leftarrow \arg\min_{j \in \mathcal{C}_t} I_j$
  if $I_p > I_{j_{\min}}$ **then**
    $\mathcal{C}_t \leftarrow (\mathcal{C}_t \setminus \{j_{\min}\}) \cup \{p\}$
  end if
end if
return updated cache $\mathcal{C}_t$

Algorithm 2: Online Burst Statistics

Input: request $r_t = i$, window FIFO H_{t-1}, counts c, threshold θ
if $t > W$ **then**
  $r_{\text{old}} \leftarrow \text{dequeue}(H_{t-1})$
  $c(r_{\text{old}}) \leftarrow c(r_{\text{old}}) - 1$
end if
enqueue(H_t, r_t), $c(i) \leftarrow c(i) + 1$
$W_{\text{eff}} \leftarrow \min\{t, W\}$
$\widehat{d}_i(t) \leftarrow c(i)/W_{\text{eff}}$
$N_i(t) \leftarrow N_i(t-1) + 1$
if $\widehat{d}_i(t) > \theta$ **then**
  $B_i(t) \leftarrow B_i(t-1) + 1$
  if $F_i(t-1) = 0$ **then**
    $E_i(t) \leftarrow E_i(t-1) + 1$
    $F_i(t) \leftarrow 1$
  else
    $F_i(t) \leftarrow 1$
  end if
else
  $F_i(t) \leftarrow 0$
end if
$\widehat{L}_i(t) \leftarrow \max\{1, B_i(t)/\max\{1, E_i(t)\}\}$
$\forall j \neq i$: keep $\{N_j, B_j, E_j, F_j, \widehat{L}_j\}$ unchanged
return updated H_t, c, and $\{N_i, B_i, E_i, F_i, \widehat{L}_i\}$

Algorithm 1 implements optimistic eviction: cached items are ranked by their estimated delayed-hit value plus a confidence bonus scaled by C_B. Algorithm 2 maintains burst statistics online using a sliding-window density test; it increments $B_i(t)$ for requests in a dense window and increments $E_i(t)$ tracked via a binary flag $F_i(t)$ only when a new burst starts. Therefore UCB-DH is computationally efficient: the statistics update is $O(1)$ per request, and an admission decision scans only the K cached items, giving $O(K)$ time per fetch completion and no scan over the entire item universe.

3.3 Regret Guarantees

Let $S^\star$ be the set of the K largest delayed-hit values V_i. Let $V_{(K)}$ be the K-th largest value and define $\Delta_j = V_{(K)} - V_j > 0$ for each $j \notin S^\star$. The following results are stated under the regenerative burst model with correct episode labels. The required concentration inequality is shown in Appendix D.1. For the sliding-window detector in Algorithm 2, the same bound holds up to an additional classification-error term $O(C_B T e^{-2W\rho^2})$ under a standard density-separation condition with margin ρ and window size W.

.

Theorem 1 (Static regret of UCB-DH)**.** *Under the regenerative burst model, UCB-DH satisfies*

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}(\text{UCB-DH})] \le C \sum_{j \notin S^\star} \frac{C_B^2 \log T}{\Delta_j} + O\left(\sum_{j \notin S^\star} \Delta_j + K C_B \right), \tag{8}$$

where $C > 0$ is a universal constant. Hence, for fixed N, K, Z and fixed positive gaps, UCB-DH has $O(\log T)$ static regret and is therefore sublinear. Proofs are deferred to Appendix D.2.

Theorem 2 (Stochastic lower bound)**.** *For any uniformly consistent policy, there exists a stochastic delayed-hit instance such that*

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}] \ge c \sum_{j \notin S^\star} \frac{C_B^2}{\Delta_j} \log T \tag{9}$$

for a universal constant $c > 0$. Thus, even in fixed-instance regimes where logarithmic regret is sublinear, the dependence on $C_B^2 \log T/\Delta_j$ is unavoidable. Proofs are deferred to Appendix D.3.

Remark 2 (Static stochastic optimality and comparison)**.** *Combining Theorems 1 and 2, UCB-DH is instance-optimal up to constants and lower-order terms in the gap-dependent regime, in the same sense as classical gap-dependent stochastic bandit bounds [30, 31]. For fixed N, K, Z and positive gaps, the regret is $O(\log T)$; moreover, no uniformly consistent policy can improve the leading $C_B^2 \log T/\Delta_j$ scaling. The factor C_B^2 is the range dependence of UCB-type estimation. In unit-scale bandits, distinguishing two arms with gap Δ_j requires $O(\log T/\Delta_j^2)$ samples and leads to $O(\log T/\Delta_j)$ regret. Here, delayed-hit observations are episode-level latency quantities with range $O(C_B)$, so the sample complexity increases by a factor C_B^2, yielding the $C_B^2 \log T/\Delta_j$ regret term. This is also the main conceptual difference from LRU, LFU, and frequency-based UCB methods [25, 32]. Those methods rank items by recency or request frequency, while UCB-DH ranks them by delayed-hit value V_i, which accounts for how requests are grouped within fetch windows. Thus a high-frequency bursty item need not be more valuable than a less frequent item whose misses create larger avoidable latency. The matching upper and lower bounds show that learning delayed-hit value is the right statistical objective for static regret minimization.*

We next extend the stochastic benchmark beyond a single fixed cache. Suppose the horizon has B change points and $M = B + 1$ stationary segments. Segment m has length T_m, values $\{V_i^{(m)}\}$, optimal cache $S_m^\star \in \arg\max_{S:|S|\le K} \sum_{i \in S} V_i^{(m)}$, and gaps $\Delta_{m,j} = V_{(K)}^{(m)} - V_j^{(m)}$ for $j \notin S_m^\star$.

Theorem 3. *If the change points are known, restarting UCB-DH at those points gives*

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{ps}}] \le C \sum_{m=1}^{M} \sum_{j \notin S_m^\star} \frac{C_B^2 \log T_m}{\Delta_{m,j}} + O(MKC_B). \tag{10}$$

For fixed N, K, Z and segment gaps bounded away from zero, this is $O((B+1)\log T)$ and is sublinear whenever $(B+1)\log T = o(T)$. If the change points are unknown, periodic restarts every H rounds satisfy

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{ps}}] \le O\left(\frac{T}{H} A_{\max} \log H + BZH + \frac{T}{H} K C_B \right), \tag{11}$$

where $A_{\max} = \max_m \sum_{j \notin S_m^\star} C_B^2/\Delta_{m,j}$. Choosing $H \asymp \sqrt{T(A_{\max} + KC_B)\log T/(BZ)}$ gives $\mathbb{E}[\mathrm{Reg}_T^{\mathrm{ps}}] = \widetilde{O}\left(\sqrt{TBZ(A_{\max} + KC_B)}\right)$. For fixed instance parameters, this is sublinear whenever $B \log T = o(T)$.

Proofs are deferred to Appendix D.4. In the known-change case, the bound is obtained by applying the static UCB-DH guarantee separately on each stationary segment. In the unknown-change case, the three terms in Eq. (11) correspond respectively to relearning within restart blocks, mismatch on blocks containing change points, and restart boundary costs.

Remark 3 (Static versus switching stochastic regret)**.** *Theorem 1 is the special case $B = 0$: UCB-DH learns a single delayed-hit value ranking and pays one exploration cost. With known change points, the same static learning cost is paid once per stationary segment, giving $O((B+1)\log T)$ regret for fixed instances. With unknown change points, periodic restarts follow the standard non-stationary bandit tradeoff [33, 34]: shorter blocks adapt faster but relearn more often, while longer blocks reduce relearning cost but suffer larger mismatch after a change. Balancing these effects yields the $\widetilde{O}(\sqrt{TBZ(A_{\max} + KC_B)})$ rate. More adaptive methods based on change-point detection or tracking regret can improve dynamic-regret guarantees in standard bandit settings [35, 36]. Our goal here is different: we use restarts as a simple mechanism to show that the delayed-hit value framework extends from fixed-cache learning to piecewise-changing workloads. Optimizing the dynamic-regret rate under completion-time admission constraints is left for future work.*

4 Adversarial Arrivals

We next consider an oblivious adversary that fixes the entire request sequence before the algorithm's randomization is drawn. The main technical issue is that the physical pending set depends on the online policy's previous actions. If physical pending states were used directly to define losses, the loss sequence would itself depend on the algorithm. We avoid this circularity by using an action-independent shadow loss induced only by the request trace and the latency parameter Z.

4.1 Shadow Loss and LA-FTPL

For each item i, define its shadow fetch process from the request trace. If a request to i arrives while no shadow fetch of i is active, it incurs cost Z and starts a shadow fetch of length Z; subsequent requests to i during this window incur residual costs. Let $g_{t,i} \in [0, Z]$ be this shadow latency. Then $g_{t,i} = 0$ unless $r_t = i$, and $g_t = (g_{t,1}, \ldots, g_{t,N})$ is independent of the online cache decisions. For a cache vector $x \in \{0,1\}^N$ with $\|x\|_1 = K$, define

$$\widetilde{\ell}_t(x) = \langle g_t, \mathbf{1} - x \rangle. \tag{12}$$

Let $x^\star \in \arg\min_{\|x\|_1 = K} \sum_{t=1}^{T} \widetilde{\ell}_t(x)$. The shadow static regret is

$$\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(\mathcal{A}) = \mathbb{E}\left[\sum_{t=1}^{T} \widetilde{\ell}_t(x_t) - \sum_{t=1}^{T} \widetilde{\ell}_t(x^\star)\right],$$

where the expectation is over the algorithm's perturbations. LA-FTPL maintains $G_t = \sum_{\tau=1}^{t} g_\tau$ and ranks items by $S_i(t) = G_{t,i} + \eta\gamma_i$, where $\gamma_i \sim \mathrm{Exp}(1)$ is sampled once at initialization. Since shadow losses have scale Z, the perturbation scale must also depend on latency. For the physical completion-time bound below, we use

$$\eta \asymp \sqrt{\frac{T(Z^2 + C_B Z)}{K(1 + \log N)}}. \tag{13}$$

The physical cache follows the perturbed ranking lazily: a missed item can be admitted only when its fetch completes.

4.2 Regret Analysis

We first state a lower bound for delayed-hit caching under the action-independent shadow static benchmark. The bound already holds on request traces whose shadow losses are generated by the delayed-hit fetch-window dynamics.

Theorem 4 (Delayed-hit adversarial lower bound)**.** *For cache capacity $K = M = \Theta(Z\sqrt{T})$, item universe size $N = 2M$, and fetch latency Z, there exists an oblivious request sequence such that every randomized online caching algorithm satisfies*

$$\mathbb{E}\big[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}\big] = \Omega(Z\sqrt{T}). \tag{14}$$

Algorithm 3 Latency-Adaptive FTPL (LA-FTPL)

Input: capacity K, latency Z, horizon T
Sample $\gamma_i \sim \mathrm{Exp}(1)$ for all i; set η by Eq. (13), $G_0 \leftarrow 0$
Initialize $\mathcal{C}_0$ with any K items
for $t = 1, \ldots, T$ **do**
  for each fetch of item p completing at time t **do**
    Compute $S_i(t-1) = G_{t-1,i} + \eta\gamma_i$
    $j_{\min} \in \arg\min_{j \in \mathcal{C}_{t-1}} S_j(t-1)$
    if $S_p(t-1) > S_{j_{\min}}(t-1)$ **then**
      $\mathcal{C}_t \leftarrow (\mathcal{C}_{t-1} \setminus \{j_{\min}\}) \cup \{p\}$
    else
      $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$
    end if
  end for
  Receive and serve request r_t under the physical dynamics
  if $r_t \notin \mathcal{C}_t \cup \mathcal{P}_t$ **then**
    Initiate a physical fetch for r_t
  end if
  Update the shadow vector g_t and set $G_t \leftarrow G_{t-1} + g_t$
end for

Proofs are deferred to Appendix E.1. It embeds the standard two-expert lower bound into a delayed-hit request trace with losses scaled by the fetch latency Z; the construction is sparse, so it does not rely on overlapping delayed hits.

For the upper bound, let $\mathcal{X}_K = \{x \in \{0,1\}^N : \|x\|_1 = K\}$. For analysis, define the instantaneous FTPL ranking

$$y_t \in \arg\max_{x \in \mathcal{X}_K} \langle G_{t-1} + \eta\gamma, x \rangle, \tag{15}$$

which can update every round. LA-FTPL implements this ranking subject to the physical constraint that new items can enter the cache only when their fetches complete.

We use the following tracking condition only to bound the extra completion-time synchronization cost. If the instantaneous FTPL ranking adds the currently requested item $i = r_s$ at time s, then the physical implementation either uses an ongoing fetch of i or starts a request-triggered fetch if i is absent and not pending. If the ranking still contains i when the fetch completes, the physical cache can admit i at that completion epoch. This condition does not allow proactive fetching; it only states that request-triggered switches can be tracked within one fetch window.

Theorem 5 (Adversarial regret of LA-FTPL)**.** *For every oblivious request sequence and every $\eta > 0$, let x_t be the physical completion-time implementation of LA-FTPL. Under the tracking condition above,*

$$\mathbb{E}[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(x)] \le \eta K(1 + \log N) + \frac{Z^2 T}{\eta} + C_B \frac{ZT}{\eta}. \tag{16}$$

With η chosen as in Eq. (13)*,*

$$\mathbb{E}[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(x)] = O\Big(\sqrt{KT \log N \,(Z^2 + C_B Z)}\Big). \tag{17}$$

In particular, for fixed K, N, Z, LA-FTPL has sublinear shadow static regret.

Proofs are deferred to Appendix E.2. The first two terms are the FTPL perturbation and stability costs for top-K decisions under $[0, Z]$-scale shadow losses. The third term is the delayed-hit-specific synchronization cost incurred while the physical cache waits for request-triggered fetches to complete.

Remark 4 (Comparison with classical adversarial caching)**.** *Regret-based adversarial caching is usually formulated with unit-scale hit/miss losses and immediate cache updates [9, 10]. In that setting, the loss sequence is directly determined by the request trace, and the cache can be updated at the same time as the learning decision. Delayed-hit caching changes all three aspects. First, shadow losses have scale Z, which forces latency-scaled perturbations. Second, the physical pending state is algorithm-dependent, which motivates the action-independent shadow loss. Third, admission is*

restricted to fetch-completion epochs, creating the additional synchronization term in Theorem 5. The lower bound in Theorem 4 shows that the latency-scaled $\Omega(Z\sqrt{T})$ difficulty is intrinsic to the shadow benchmark, while LA-FTPL remains sublinear under the physical completion-time constraint.

Switching comparators. The static benchmark corresponds to a persistent cache. To handle changing workloads, we use the switching comparator from Eq. (5). For shadow losses, define $\widetilde{L}_T(S_{1:T}) = \sum_{t=1}^{T} \langle g_t, \mathbf{1} - x(S_t) \rangle$ and $\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\mathcal{A}; B) = \mathbb{E}\Big[\sum_{t=1}^{T} \widetilde{\ell}_t(x_t) - \min_{S_{1:T} \in \mathcal{S}_B} \widetilde{L}_T(S_{1:T})\Big]$.

Theorem 6. *For every oblivious request sequence and every $B \geq 1$,* RESTART-LA-FTPL *satisfies* $\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\text{RESTART-LA-FTPL}; B) = O\left(\min_{1 \leq H \leq T} \left\{ T\sqrt{\frac{K \log N(Z^2 + C_B Z)}{H}} + ZBH + KC_B \frac{T}{H} \right\}\right)$. *Here, H is the restart block length. In particular, balancing the first two terms gives $H^* \asymp \left(\frac{T^2 K \log N(Z^2 + C_B Z)}{Z^2 B^2}\right)^{1/3}$, and hence*

$$\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\text{RESTART-LA-FTPL}; B) = O\left(T^{2/3} B^{1/3} \big[ZK \log N(Z^2 + C_B Z)\big]^{1/3} + KC_B \frac{T}{H^*}\right) \quad (18)$$

Proofs are deferred to Appendix E.3. For fixed system parameters, the last term is lower order whenever $B = o(T)$; thus the regret is sublinear in T in this regime. For $B = 0$, the static bound in Theorem 5 applies directly. The restart analysis separates blocks into clean blocks and switch blocks. On each clean block, we apply an interval version of the physical tracking guarantee in Theorem 5, conditioned on the pre-block physical cache state and pending fetches.

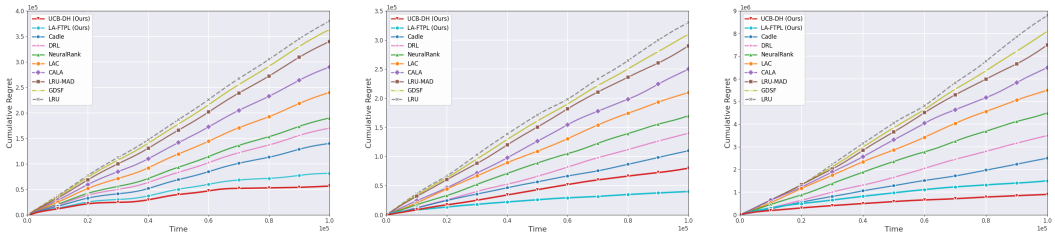
Remark 5 (Switching regret and dynamic benchmarks)**.** *The switching result interpolates between static regret and dynamic benchmarks. When $B = 0$, it reduces to the static shadow-regret guarantee in Theorem 5; when $B > 0$, the comparator can track piecewise-changing hot sets. Theorem 6 gives a periodic-restart trade-off consisting of the clean-block learning term, the switch-block adaptation term, and the restart synchronization term induced by completion-time cache updates. Balancing the first two terms yields the familiar $T^{2/3}B^{1/3}$ periodic-restart scaling (up to problem-dependent factors), with an additional lower-order restart-boundary term for fixed system parameters when $B = o(T)$. This is not a claim of optimal dynamic regret. Its role is to show that the delayed-hit shadow-loss framework is not restricted to a fixed-cache comparator. More adaptive dynamic-regret methods could potentially improve the dependence on T, but integrating them with completion-time admission is orthogonal to our main contribution.*

Remark 6. *While our core exposition focuses on unit-size items and uniform latency Z, the action-independent shadow framework extends naturally:* ***1)Item-dependent and exogenous latencies:*** *If each item i has a fixed latency $Z_i \leq Z_{\max}$, an uncached miss starts an item-specific shadow window of length Z_i, with maximum penalty $C_i = Z_i(Z_i + 1)/2$ and $C_{\max} = \max_i C_i$. Equipping UCB-DH with item-specific indices $I_i(t) = \widehat{V}_i(t) + C_i\sqrt{2 \log t / \max\{1, N_i(t)\}}$ yields the same gap-dependent logarithmic regret scaling as $O(\sum_{j \notin S^*} \frac{C_{\max}^2 \log T}{\Delta_j} + KC_{\max})$. Similarly, LA-FTPL attains $O(\sqrt{KT \log N(Z_{\max}^2 + C_{\max} Z_{\max})})$ shadow regret. This analysis directly generalizes to time-varying latencies $Z_{i,t} \leq Z_{\max}$ as long as delays are exogenous .* ***2)Heterogeneous object sizes:*** *When item i has size s_i and the cache has byte capacity B, the feasible set transitions from a uniform matroid to a 0-1 knapsack: $\mathcal{C}_B = \{C \subseteq [N] : \sum_{i \in C} s_i \leq B\}$. Algorithmically, our framework generalizes by replacing the top-K selection with an optimistic knapsack oracle: $\mathcal{C}_t \in \arg\max_{C \in \mathcal{C}_B} \sum_{i \in C} I_i(t)$ for UCB-DH, and $y_t \in \arg\max_{x \in \{0,1\}^N, \sum x_i s_i \leq B} \langle G_{t-1} + \eta\gamma, x \rangle$ for LA-FTPL. Regret analysis under this setting requires combinatorial semi-bandit techniques to handle multi-item evictions and knapsack approximation gaps, which we leave for future work.*

5 Numerical Experiments

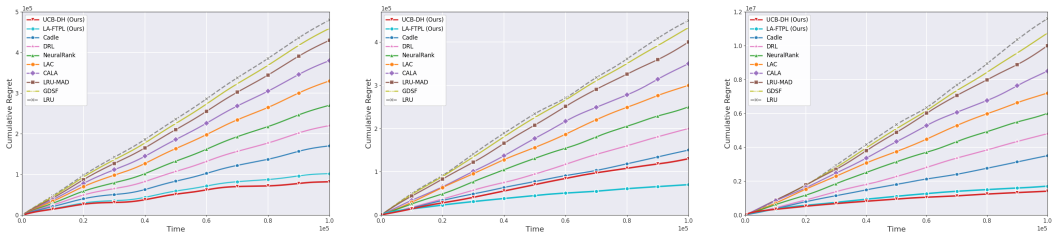
Datasets and baselines. We use three traces: **SYN-ZIPF**, a 10^5-request stationary Zipf trace ($\alpha = 1.2$); **SYN-ADV**, an adversarial trace with cyclic, scan, and recovery phases; and **ML-1M**, the MovieLens 1M timestamped stream [37]. Delayed hits are induced by replaying each trace with fetch latency Z: after an uncached request starts a fetch, later requests to the same item before completion incur residual latency. With capacity $K = 100$ and fetch latency $Z = 100$, delayed-hit coverage is

51.12%, 96.55%, and 5.84% on the three traces, respectively. We also test ML-1M with $Z = 1500$, where coverage rises to 59.76%, as a sensitivity study. Baselines include classical policies LRU, GDSF [38], LRU-MAD [15], LAC [16], CALA [20], and Cadle [18]; and learning-based methods NeuralRank [17] and Online-DRL [19].



(a) SYN-ZIPF (b) SYN-ADV (c) ML-1M

Figure 2: The comparison of cumulative static regret between our algorithm and other different algorithms on three datasets.



(a) SYN-ZIPF (b) SYN-ADV (c) ML-1M

Figure 3: The comparison of cumulative switching regret between our algorithm and other different algorithms on three datasets

Simulation results. Figures 2 and 3 report static and switching delayed-hit regret. On SYN-ZIPF, UCB-DH achieves sublinear regret by learning stable delayed-hit values, and LA-FTPL also remains sublinear under this benign workload. On SYN-ADV, LA-FTPL is more robust, whereas stochastic methods can exhibit nearly linear regret because the stationarity assumption is violated. On ML-1M, our methods continue to show sublinear regret, indicating that delayed-hit-aware learning remains effective on real-world traces. In contrast, several classical, competitive-ratio, and heuristic baselines display nearly linear regret, showing that good competitive or latency-oriented performance does not necessarily imply no-regret behavior under delayed-hit benchmarks. Additionally, detailed ablation studies and extended sweeps over capacity K and latency Z are deferred to Appendix H, confirming that our latency advantages persist across broad system configurations.

6 Conclusion

We introduce a regret framework for delayed-hit caching, with static regret for long-term placement and switching regret for changing hot sets. We prove that classical deterministic demand-admission policies and randomized MARKER are not no-regret under these benchmarks. For stochastic arrivals, UCB-DH learns delayed-hit shadow values and achieves gap-dependent regret $O(\log T)$, matching the corresponding lower bound up to constants and lower-order terms. For oblivious adversarial arrivals, LA-FTPL achieves $O(\sqrt{KT \log N})$ action-independent shadow regret with latency-scaled perturbations. Experiments on synthetic and real traces show consistent latency improvements over classical and recent delayed-hit baselines.

Acknowledgments

The corresponding author Duo Wang is sponsored by Startup Fund for Young Faculty at SJTU((25X010502578)).

References

[1] Xiaoshuang Chen, Gengrui Zhang, Yao Wang, Yulin Wu, Shuo Su, Kaiqiao Zhan, and Ben Wang. Cache-aware reinforcement learning in large-scale recommender systems. In *Companion Proceedings of the ACM Web Conference 2024*, pages 284–291, 2024.

[2] Muhammad Zubair Shafiq, Alex X Liu, and Amir R Khakpour. Revisiting caching in content delivery networks. In *The 2014 ACM international conference on Measurement and modeling of computer systems*, pages 567–568, 2014.

[3] Urs Niesen, Devavrat Shah, and Gregory W Wornell. Caching in wireless networks. *IEEE Transactions on information theory*, 58(10):6524–6540, 2012.

[4] Ding Xu, Lingjie Duan, and Hongbo Zhu. Aigc-enhanced hybrid content caching in wireless networks. *IEEE Transactions on Wireless Communications*, 2025.

[5] Jeongmin Hong, Sungjun Cho, Geonwoo Park, Wonhyuk Yang, Young-Ho Gong, and Gwangsun Kim. Bandwidth-effective dram cache for gpu s with storage-class memory. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 139–155. IEEE, 2024.

[6] Christian Coester, Roie Levin, Joseph Naor, and Ohad Talmon. Competitive algorithms for block-aware caching. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 161–172, 2022.

[7] Binghan Wu, Wei Bao, and Bing Bing Zhou. Competitive analysis of online elastic caching of transient data in multi-tiered content delivery network. *IEEE Transactions on Parallel and Distributed Systems*, 35(12):2449–2462, 2024.

[8] Marek Chrobak, Samuel Haney, Mehraneh Liaee, Debmalya Panigrahi, Rajmohan Rajaraman, Ravi Sundaram, and Neal E Young. Online paging with heterogeneous cache slots. *Algorithmica*, 87(1):89–131, 2025.

[9] Georgios S Paschos, Apostolos Destounis, Luigi Vigneri, and George Iosifidis. Learning to cache with no regrets. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 235–243. IEEE, 2019.

[10] Rajarshi Bhattacharjee, Subhankar Banerjee, and Abhishek Sinha. Fundamental limits on the regret of online network-caching. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–31, 2020.

[11] Debjit Paria and Abhishek Sinha. Leadcache: Regret-optimal caching in networks. *Advances in Neural Information Processing Systems*, 34:4435–4447, 2021.

[12] Samrat Mukhopadhyay and Abhishek Sinha. Online caching with optimal switching regret. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1546–1551. IEEE, 2021.

[13] Guocong Quan, Atilla Eryilmaz, and Ness B Shroff. Minimizing edge caching service costs through regret-optimal online learning. *IEEE/ACM Transactions on Networking*, 32(5):4349–4364, 2024.

[14] Shiji Zhou, Zhi Wang, Chenghao Hu, Yinan Mao, Haopeng Yan, Shanghang Zhang, Chuan Wu, and Wenwu Zhu. Caching in dynamic environments: A near-optimal online learning approach. *IEEE Transactions on Multimedia*, 25:792–804, 2021.

[15] Nirav Atre, Justine Sherry, Weina Wang, and Daniel S Berger. Caching with delayed hits. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 495–513, 2020.

[16] Gang Yan and Jian Li. Towards latency awareness for content delivery network caching. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 789–804, 2022.

[17] Bowen Jiang, Yubo Yang, and Bo Jiang. Optimizing latency for caching with delayed hits in non-stationary environment. *Performance Evaluation*, page 102488, 2025.

[18] Xin He, Mingyu Cai, Meng Li, Haipeng Dai, Wei Peng, Jian Zhou, and Fu Xiao. Online caching with delayed hits in multi-server edge networks. *IEEE Transactions on Mobile Computing*, 2026.

[19] Sikha Deka and Radhika Sukapuram. Edge service caching with delayed hits: minimizing latency and reducing cost. *IEEE Transactions on Cloud Computing*, 2025.

[20] Chi Zhang, Haisheng Tan, Guopeng Li, Zhenhua Han, Shaofeng H-C Jiang, and Xiang-Yang Li. Online file caching in latency-sensitive systems with delayed hits and bypassing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1059–1068. IEEE, 2022.

[21] Haisheng Tan, Yi Wang, Chi Zhang, Guopeng Li, Haohua Du, Zhenhua Han, Shaofeng H-C Jiang, and Xiang-Yang Li. Asymptotically tight approximation for online file caching with delayed hits and bypassing. *IEEE Transactions on Networking*, 2025.

[22] Keerthana Gurushankar, Noah G Singer, and Bernardo Subercaseaux. Latency guarantees for caching with delayed hits. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2025.

[23] Bowen Jiang, Chaofan Ma, and Duo Wang. Va-cdh: A variance-aware method to optimize latency for caching with delayed hits. *arXiv preprint arXiv:2504.20335*, 2025.

[24] Bowen Jiang and Chaofan Ma. Modeling and optimizing latency for delayed hit caching with stochastic miss latency. In *2025 4th International Conference on Networks, Communications and Information Technology (CNCIT)*, pages 1–6. IEEE, 2025.

[25] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[26] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[27] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[28] Mark Manasse, Lyle McGeoch, and Daniel Sleator. Competitive algorithms for on-line problems. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 322–333, 1988.

[29] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1-2):203–218, 2000.

[30] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[31] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[32] Archana Bura, Desik Rengarajan, Dileep Kalathil, Srinivas Shakkottai, and Jean-Francois Chamberland. Learning to cache and caching to learn: Regret analysis of caching algorithms. *IEEE/ACM Transactions on Networking*, 30(1):18–31, 2021.

[33] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[34] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014.

[35] Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

[36] Peter Auer, Pratik Gajane, and Ronald Ortner. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on learning theory*, pages 138–158. PMLR, 2019.

[37] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[38] Pei Cao and Sandy Irani. Cost-aware www proxy caching algorithms. In *USENIX Symposium on Internet Technologies and Systems (USITS 97)*, 1997.

[39] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[40] Yasin Abbasi-Yadkori, András György, and Nevena Lazić. A new look at dynamic regret for non-stationary stochastic bandits. *Journal of Machine Learning Research*, 24(288):1–37, 2023.

[41] Gianmarco Genalti, Sujay Bhatt, Nicola Gatti, Alberto Maria Metelli, et al. Catoni-style change point detection for regret minimization in piecewise-stationary heavy-tailed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 1–24, 2026.

[42] Haipeng Luo, Mengxiao Zhang, Peng Zhao, and Zhi-Hua Zhou. Corralling a larger band of bandits: A case study on switching regret for linear bandits. In *Conference on Learning Theory*, pages 3635–3684. PMLR, 2022.

[43] Peter Manohar and Jalani Williams. Lower bounds for caching with delayed hits. *arXiv preprint arXiv:2006.00376*, 2020.

[44] Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Online learning under delayed feedback. In *International conference on machine learning*, pages 1453–1461. PMLR, 2013.

[45] Andras Gyorgy and Pooria Joulani. Adapting to delays and data in adversarial multi-armed bandits. In *International Conference on Machine Learning*, pages 3988–3997. PMLR, 2021.

[46] Saeed Masoudian, Julian Zimmert, and Yevgeny Seldin. A best-of-both-worlds algorithm for bandits with delayed feedback. *Advances in Neural Information Processing Systems*, 35:11752–11762, 2022.

[47] Jiatai Huang, Yan Dai, and Longbo Huang. Banker online mirror descent: A universal approach for delayed online bandit learning. In *International Conference on Machine Learning*, pages 13814–13844. PMLR, 2023.

Appendix Contents

A Table of Notation **15**

B Related Work **16**

C Classical Policies **16**

C.1 Proof of Proposition 1 16

C.2 Proof of Proposition 2 17

D Stochastic Arrivals **18**

D.1 Concentration of the UCB-DH Value Estimator 18

D.2 Proof of Theorem 1 19

D.3 Proof of Theorem 2 21

D.4 Proof of Theorem 3 22

E Adversarial Arrivals **23**

E.1 Proof of Theorem 4 23

E.2 Proof of Theorem 5 25

E.3 Proof of Theorem 6 27

F Limitations and Scope **28**

G Experimental Reproducibility Details **29**

H Additional Ablations and Sensitivity Studies **30**

H.1 Sensitivity Studies on Cache Capacity K and Fetch Latency Z 30

H.2 Sensitivity to Burst-Detection Parameters 31

H.3 Isolating the Burst-Aware Value Term 31

H.4 Selective vs. Forced Admission Ablation 32

H.5 Delayed-Hit Value Learning vs. Recency at Large Universe Size 32

A Table of Notation

Table 1: Summary of main notations.

Symbol	Description
	System model
N	Number of items in the universe.
K	Cache capacity; items have unit size, so at most K items can be cached.
T	Time horizon.
Z	Fetch latency. A missed item becomes eligible for admission after Z slots.
r_t	Item requested at time t.
$\mathcal{C}_t$	Physical cache set after processing fetch completions at time t and before serving request r_t.
$\mathcal{P}_t$	Set of items whose physical fetches are pending at time t.
τ_i	Most recent physical fetch initiation time of item i.
ℓ_t	Physical instantaneous latency incurred by the online algorithm at time t.
$L_T(\mathcal{A})$	Physical cumulative latency of algorithm $\mathcal{A}$.
C_B	Maximum aggregate latency generated by one miss episode within a fetch window, $C_B = C_B(Z) = Z(Z+1)/2$.
	Comparators and regret
S	Fixed cache set with $\lvert S\rvert \le K$.
$S^\star$	Best fixed cache under the delayed-hit static comparator.
$L_T(S)$	Shadow cumulative delayed-hit loss of fixed cache S under the action-independent shadow fetch process.
$\mathrm{Reg}_T^{\mathrm{stat}}(\mathcal{A})$	Static delayed-hit regret of algorithm $\mathcal{A}$ against the best fixed cache.
$\mathcal{S}_B$	Set of cache sequences with at most B switches.
$\mathrm{Reg}_T^{\mathrm{sw}}(\mathcal{A}; B)$	B-switching delayed-hit regret; the case $B = 0$ recovers static regret.
	Stochastic setting and UCB-DH
D_i	Episode-type indicator for item i; $D_i = 1$ denotes a burst episode.
L_i	Length of an episode of item i.
$g_Z(\ell)$	Total latency generated by an uncached episode of length ℓ, $g_Z(\ell) = \sum_{s=0}^{\ell-1}(Z-s)$.
p_i	Stationary request fraction of item i.
q_i	Probability that a request of item i belongs to a burst episode.
$\bar{L}_i$	Mean burst length of item i, $\bar{L}_i = \mathbb{E}[L_i \mid D_i = 1]$.
ψ_i	Exact average per-request burst penalty, $\psi_i = \mathbb{E}[g_Z(L_i)/L_i \mid D_i = 1]$.
V_i	Delayed-hit shadow value used by UCB-DH, combining frequency, burst structure, and fetch latency.
$N_i(t)$	Number of observed requests for item i up to time t.
$B_i(t)$	Number of requests of item i classified as bursty up to time t.
$E_i(t)$	Number of detected burst episodes of item i up to time t.
$\widehat{p}_i(t), \widehat{q}_i(t), \widehat{L}_i(t)$	Empirical estimates of p_i, q_i, and $\bar{L}_i$.
$\widehat{V}_i(t)$	Empirical delayed-hit value estimate obtained by plugging estimates into Eq. (6).
$I_i(t)$	Optimistic UCB-DH index of item i.
Δ_j	Gap of suboptimal item j, $\Delta_j = V_{(K)} - V_j$, where $V_{(K)}$ is the K-th largest delayed-hit value.
	Adversarial setting and LA-FTPL
g_t	Action-independent shadow cost vector at time t; each coordinate satisfies $g_{t,i} \in [0, Z]$ and $g_{t,i} = 0$ unless $r_t = i$.
G_t	Cumulative shadow cost vector, $G_t = \sum_{\tau=1}^{t} g_\tau$.
x_t	Cache vector used by the physical online algorithm at time t.
y_t	Instantaneous FTPL cache vector used in the analysis; it can update every round.
$\mathcal{X}_K$	Feasible cache-vector set, $\mathcal{X}_K = \{x \in \{0,1\}^N : \lVert x\rVert_1 = K\}$.
$\widetilde{\ell}_t(x)$	Shadow loss of cache vector x at time t, $\widetilde{\ell}_t(x) = \langle g_t, \mathbf{1} - x\rangle$.
$\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}$	Static regret under the action-independent shadow loss.
$\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\cdot; B)$	B-switching regret under the action-independent shadow loss.
η	Perturbation scale in LA-FTPL.
γ_i	Perturbation for item i, sampled once from $\mathrm{Exp}(1)$.
$S_i(t)$	Perturbed score of item i, $S_i(t) = G_{t,i} + \eta\gamma_i$.

B Related Work

Classical and regret-based online caching. Classical caching is traditionally studied through competitive analysis, with foundational results for deterministic and randomized paging algorithms [25, 26, 27]. These guarantees compare an online policy with an offline optimum multiplicatively, but they do not imply vanishing additive regret. More recent work studies caching from an online-learning perspective, where the goal is regret against a static cache in hindsight. [9] formulated caching as an online linear optimization problem, and [10] established regret lower bounds and FTPL-type algorithms for adversarial caching. Closely related recent work studies online caching with switching costs and optimal switching regret [12]. These models capture the cost of changing cached files, but they still assume immediate availability after a miss. In contrast, delayed-hit caching has an additional pending interval: after a miss, the requested item is not immediately available, and repeated requests during the fetch window incur residual latency.

Switching regret and non-stationary workloads. Static regret is appropriate when the best cache is stable, whereas real workloads often have changing hot sets. This motivates switching or tracking regret, where the comparator is allowed to change a limited number of times [35, 39]. Related ideas appear in non-stationary bandits, where algorithms use restarts, sliding windows, change-point detection, or corralling to adapt to distributional changes [33, 34, 36, 40, 41, 42]. Our setting differs in two ways. First, the comparator is evaluated under delayed-hit shadow losses, whose values are induced by fetch windows rather than immediate hit/miss rewards. Second, the online cache cannot switch freely: a new item can be admitted only when its fetch completes.

Caching with delayed hits. Delayed-hit caching explicitly models the time gap between a cache miss and the completion of the corresponding fetch. [15] introduced this phenomenon and proposed heuristic policies for reducing cumulative latency. Subsequent work studied richer system constraints, including heterogeneous object sizes, variable miss latencies, and non-stationary request sequences [16, 17]. A complementary theoretical line analyzes delayed-hit caching through competitive analysis. [20] and [21] studied competitive bounds for cumulative latency, while [22] analyzed LRU-type policies and [43] proved lower bounds for deterministic algorithms. These results provide multiplicative guarantees. Our work instead studies additive regret and shows that competitive-ratio style guarantees do not by themselves ensure sublinear delayed-hit regret.

Online learning with delayed feedback. Delayed feedback in online learning concerns the case where the reward or loss observation is revealed after a delay. Foundational work by [44] showed how feedback delays affect regret, and later work developed delay-adaptive methods such as adaptive step sizes, best-of-both-worlds algorithms, and delayed-feedback OMD variants [45, 46, 47]. Delayed hits are different from delayed feedback. In delayed feedback, the delay affects when information is observed; the reward or loss value of an action is unchanged. In delayed-hit caching, the delay affects the physical availability of the item and changes the latency incurred by subsequent requests. This creates action-dependent pending states and completion-time admission constraints, motivating our action-independent shadow loss and latency-aware learning algorithms.

C Classical Policies

C.1 Proof of Proposition 1

Proof. Fix a deterministic demand-admission policy $\mathcal{A}$. Let $S_0 = \{1, \dots, K\}$ be a fixed cache set. If the initial cache of $\mathcal{A}$ is different, we relabel its initial cached items as S_0; this only changes notation.

We compare $\mathcal{A}$ to the fixed cache S_0. Since the best static cache $S^\star$ is at least as good as S_0, it suffices to lower-bound $L_T(\mathcal{A}) - L_T(S_0)$.

Recall that a burst of Z consecutive requests to an uncached item generates total delayed-hit latency

$$C_B = Z + (Z-1) + \cdots + 1 = \frac{Z(Z+1)}{2}. \tag{19}$$

We construct the request sequence in blocks. Each block consists of two bursts, each of length Z.

At the beginning of a block, choose a fresh trigger item u that is not in S_0 and has not appeared before. Request u for Z consecutive slots. Since $u \notin S_0$, the fixed cache S_0 incurs the same shadow cost C_B on this trigger burst as the online policy. Thus this burst contributes no positive regret.

At the end of the trigger burst, the fetch of u completes. Because $\mathcal{A}$ is demand-admission, it must insert u into the cache. Since $u \notin S_0$ and the cache has capacity K, the cache of $\mathcal{A}$ cannot contain all items of S_0 after admitting u. Therefore, there exists some item $j \in S_0$ that is not in $\mathcal{A}$'s cache.

The adversary then requests this missing item j for Z consecutive slots. The fixed cache S_0 serves all these requests with zero latency. For $\mathcal{A}$, item j is absent at the start of the burst, so the first request is a miss and the remaining requests are delayed hits. Hence $\mathcal{A}$ incurs latency exactly C_B on this victim burst. Therefore each block contributes at least C_B regret relative to S_0.

Each block has length $2Z$. Let $M = \lfloor T/(2Z) \rfloor$ be the number of complete blocks that fit in the horizon. The remaining incomplete suffix can only reduce the lower bound by a nonnegative amount, so

$$\mathrm{Reg}_T^{\mathrm{stat}}(\mathcal{A}) \geq MC_B. \tag{20}$$

Since $M \geq T/(2Z) - 1$, we obtain

$$\mathrm{Reg}_T^{\mathrm{stat}}(\mathcal{A}) \geq \left(\frac{T}{2Z} - 1\right) \frac{Z(Z+1)}{2} = \Omega(ZT) \tag{21}$$

for all sufficiently large T.

In particular, when Z is fixed, this is $\Omega(T)$, so such deterministic demand-admission policies cannot have sublinear regret. For a single block group of K victim constructions, the bound can also be written as $\Omega(KZ^2)$ over length $\Theta(KZ)$; if $Z = \Theta(T/K)$ for that single-block scaling, it becomes $\Omega(T^2/K)$. □

C.2 Proof of Proposition 2

Proof. Fix a fetch latency $Z \geq 1$ and set the cache capacity to $K = 1$. Let $M = Z + 2$. Consider the periodic request sequence

$$1, 2, \ldots, M, 1, 2, \ldots, M, \ldots$$

of length T. For simplicity, first assume that $T = qM$ for some integer q.

Since the period length satisfies $M > Z$, two consecutive requests to the same item are separated by more than one fetch latency. Hence no request in this construction is a delayed hit: whenever an item is requested again, any previous fetch of that item has already completed.

With cache capacity $K = 1$, MARKER has only one cache slot. On this cyclic sequence, before essentially every request the cache contains an item different from the requested one. Thus every request is a miss, up to an additive boundary term depending on the initial cache state. Therefore,

$$L_T(\textsc{Marker}) \geq ZT - O(Z). \tag{22}$$

Now consider the best fixed cache of size one. It can store any one of the M items. Since each item appears exactly T/M times, the best fixed cache hits on T/M requests and misses on the remaining $T - T/M$ requests. Thus,

$$L_T(S^\star) = Z\left(T - \frac{T}{M}\right). \tag{23}$$

Consequently,

$$\mathrm{Reg}_T^{\mathrm{stat}}(\textsc{Marker}) = L_T(\textsc{Marker}) - L_T(S^\star) \geq ZT - O(Z) - Z\left(T - \frac{T}{M}\right). \tag{24}$$

Hence,

$$\mathrm{Reg}_T^{\mathrm{stat}}(\textsc{Marker}) \geq \frac{ZT}{M} - O(Z). \tag{25}$$

Substituting $M = Z + 2$ gives

$$\mathrm{Reg}_T^{\mathrm{stat}}(\textsc{Marker}) \geq \frac{Z}{Z+2}T - O(Z). \tag{26}$$

Since $Z/(Z+2) \geq 1/3$ for every $Z \geq 1$, this is $\Omega(T)$.

For horizons T not divisible by M, truncating the last incomplete period changes both losses by at most $O(MZ)$, which does not affect the linear lower bound. Finally, for $K = 1$, MARKER has no nontrivial random eviction choice, so the same bound holds in expectation. □

D Stochastic Arrivals

D.1 Concentration of the UCB-DH Value Estimator

Proof. Fix an item i and a time t. We first rewrite the UCB-DH estimator as a bounded empirical average.

Recall that

$$\widehat{p}_i(t) = \frac{N_i(t)}{t}, \qquad \widehat{q}_i(t) = \frac{B_i(t)}{\max\{1, N_i(t)\}}, \qquad \widehat{L}_i(t) = \max\left\{1, \frac{B_i(t)}{\max\{1, E_i(t)\}}\right\}. \tag{27}$$

Under correct episode labels and when $B_i(t) > 0$, substituting these definitions into the amortized estimator gives

$$\widehat{V}_i(t) = \frac{(N_i(t) - B_i(t))Z + E_i(t)C_B}{t}. \tag{28}$$

When $B_i(t) = 0$, the same expression holds with the burst term equal to zero.

Define the event $B_{i,s}$ to indicate that slot s is the first slot of a burst episode of item i. The per-time amortized contribution is then

$$Y_{i,s} = Z\,\mathbf{1}\{r_s = i,\ D_s = 0\} + C_B\,\mathbf{1}\{B_{i,s}\}. \tag{29}$$

Then $0 \leq Y_{i,s} \leq C_B$ and

$$\widehat{V}_i(t) = \frac{1}{t}\sum_{s=1}^{t} Y_{i,s}. \tag{30}$$

By definition of the amortized delayed-hit value, $V_i = \mathbb{E}[Y_{i,s}]$.

Under the regenerative burst model, the sequence of episode contributions is regenerative and bounded. We use the standard Hoeffding-type concentration inequality for bounded regenerative processes, which gives, for every $\epsilon > 0$,

$$\Pr\left(|\widehat{V}_i(t) - V_i| > \epsilon\right) \leq 2\exp\left(-\frac{2t\epsilon^2}{C_B^2}\right). \tag{31}$$

Taking $\epsilon = C_B\sqrt{2\log t/t}$ yields

$$\Pr\left(|\widehat{V}_i(t) - V_i| > C_B\sqrt{\frac{2\log t}{t}}\right) \leq 2t^{-4}. \tag{32}$$

Since $N_i(t) \leq t$,

$$C_B\sqrt{\frac{2\log t}{t}} \leq C_B\sqrt{\frac{2\log t}{\max\{1, N_i(t)\}}}. \tag{33}$$

Thus the desired bound holds with a universal constant $c = \sqrt{2}$. □

Effect of sliding-window burst detection. The concentration lemma above assumes correct episode labels. We now state the additional error incurred by the detector in Algorithm 2. Suppose there is a density-separation margin $\rho > 0$ such that, for every window, the expected local density of item i is at most $\theta - \rho$ outside bursts and at least $\theta + \rho$ inside bursts. For the sliding-window detector, let $\mathcal{M}$ denote the event that a given window is misclassified. Under a standard density-separation margin $\rho > 0$, Hoeffding's inequality yields

$$\Pr(\mathcal{M}) \leq 2e^{-2W\rho^2}. \tag{34}$$

By a union bound over the horizon, the expected number of misclassified slots is at most $2T\exp(-2W\rho^2)$. Each misclassification changes the amortized contribution by at most C_B. Therefore the additional cumulative error due to burst detection is at most $O\Big(C_B T e^{-2W\rho^2}\Big)$. Choosing $W = \Omega(\rho^{-2}\log T)$ makes this term lower order, and the regret rates in Theorem 1 are unchanged up to constants.

Detector Error Bound under Separation Margin. Theorem 1 is proven under oracle burst labels or for a detector whose classification errors are sufficiently rare. Suppose the sliding-window detector statistic has conditional mean at most μ_0 on non-burst windows and at least $\mu_1 > \mu_0$ on burst windows. For $\theta \in (\mu_0, \mu_1)$, define the separation margin:

$$\gamma(\theta) := \min\{\theta - \mu_0,\ \mu_1 - \theta\}. \tag{35}$$

Under standard bounded-difference assumptions, Hoeffding's inequality yields:

$$\Pr(\text{any detector error over } [T]) \le 2T\exp\left(-2W\gamma(\theta)^2\right). \tag{36}$$

Coupling the detector-based implementation to the oracle-label execution gives the conservative regret bound:

$$\mathbb{E}[Reg_T^{\text{det}}] \le \mathbb{E}[Reg_T^{\text{oracle}}] + O\left(ZT^2 e^{-2W\gamma(\theta)^2}\right). \tag{37}$$

Consequently, the classification error is of strictly lower order whenever $W\gamma(\theta)^2 \gtrsim \log T$. This explains the parameter tradeoff: an overly small window W leads to noisy density estimates, while an excessively large W over-smoothes non-burst and burst boundaries.

D.2 Proof of Theorem 1

Proof. Define the confidence radius

$$\beta_i(t) = cC_B\sqrt{\frac{\log t}{\max\{1, N_i(t)\}}}. \tag{38}$$

The UCB index is

$$I_i(t) = \widehat{V}_i(t) + \beta_i(t). \tag{39}$$

Let $\mathcal{E}_t$ be the event that all shadow-value estimates are accurate at time t:

$$\mathcal{E}_t = \left\{\forall i \in [N] : |\widehat{V}_i(t) - V_i| \le \beta_i(t)\right\}. \tag{40}$$

By the concentration of the UCB-DH value estimator and a union bound over items,

$$\Pr(\mathcal{E}_t^c) \le 2Nt^{-4}. \tag{41}$$

Therefore

$$\sum_{t=1}^{\infty}\Pr(\mathcal{E}_t^c) \le 2N\sum_{t=1}^{\infty} t^{-4} < \infty. \tag{42}$$

Thus the total expected regret contribution from failure events is bounded by a constant depending on N and the maximum one-step delayed-hit loss. This contributes only the lower-order additive term.

We now work on the good event $\mathcal{E}_t$.

Consider a regret-relevant admission epoch t at which an optimal item $i^\star \in S^\star$ is available for admission but is not retained, while a suboptimal item $j \notin S^\star$ remains in the cache. By the UCB-DH admission rule, we have

$$I_j(t) \ge I_{i^\star}(t). \tag{43}$$

On $\mathcal{E}_t$,

$$I_{i^\star}(t) = \widehat{V}_{i^\star}(t) + \beta_{i^\star}(t) \ge V_{i^\star}. \tag{44}$$

Also,

$$I_j(t) = \widehat{V}_j(t) + \beta_j(t) \le V_j + 2\beta_j(t). \tag{45}$$

Combining these inequalities gives

$$V_j + 2\beta_j(t) \geq V_{i^\star}. \tag{46}$$

Since $i^\star \in S^\star$ and $j \notin S^\star$,

$$V_{i^\star} \geq V_{(K)}. \tag{47}$$

Therefore,

$$2\beta_j(t) \geq V_{(K)} - V_j = \Delta_j. \tag{48}$$

Substituting the definition of $\beta_j(t)$,

$$2cC_B\sqrt{\frac{\log t}{\max\{1, N_j(t)\}}} \geq \Delta_j. \tag{49}$$

Thus,

$$N_j(t) \leq \frac{4c^2C_B^2 \log t}{\Delta_j^2} \leq \frac{4c^2C_B^2 \log T}{\Delta_j^2}. \tag{50}$$

This means that, on the good event, item j can be involved in regret-relevant suboptimal retention only while its sample count is at most

$$m_j = \left\lceil \frac{4c^2C_B^2 \log T}{\Delta_j^2} \right\rceil. \tag{51}$$

After this threshold, the confidence interval of j is too narrow for j to outrank any missing optimal item.

Let $M_j(T)$ denote the number of regret-relevant rounds in which suboptimal item j occupies a cache slot in place of some optimal item. The above argument implies

$$\mathbb{E}[M_j(T)] \leq m_j + O(1), \tag{52}$$

where the $O(1)$ term accounts for failure events and boundary effects.

The static regret can be decomposed as

$$\mathrm{Reg}_T^{\mathrm{stat}} \leq \sum_{j \notin S^\star} \Delta_j M_j(T) + O(KC_B). \tag{53}$$

The additive $O(KC_B)$ accounts for initialization and incomplete fetch-window boundary effects. Taking expectations,

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}] \leq \sum_{j \notin S^\star} \Delta_j \left(\frac{4c^2C_B^2 \log T}{\Delta_j^2} + O(1) \right) + O(KC_B). \tag{54}$$

Therefore,

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}] \leq 4c^2 \sum_{j \notin S^\star} \frac{C_B^2 \log T}{\Delta_j} + O\left(\sum_{j \notin S^\star} \Delta_j + KC_B \right). \tag{55}$$

By Theorem 1,

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}(\text{UCB-DH})] \leq C \sum_{j \notin S^\star} \frac{C_B(Z)^2 \log T}{\Delta_j} + O\left(\sum_{j \notin S^\star} \Delta_j + KC_B(Z) \right). \tag{56}$$

In a fixed stochastic instance, N, K, Z are constants independent of T. Hence $C_B(Z)$ is also constant. Since all gaps $\Delta_j > 0$ are fixed constants and there are at most $N - K$ suboptimal items, the coefficient

$$\sum_{j \notin S^\star} \frac{C_B(Z)^2}{\Delta_j} \tag{57}$$

is independent of T. The additive term is also independent of T. Therefore the regret is $O(\log T)$.

This proves the theorem. □

D.3 Proof of Theorem 2

Proof. We prove the lower bound by reduction to stochastic top-K bandit identification with scaled rewards.

First consider the case $K = 1$ and two items, an optimal item 1 and a suboptimal item 2. Let the delayed-hit shadow-value observation of item i be a random variable

$$Y_i = C_B\, X_i, \tag{58}$$

where $X_i \sim \mathrm{Bernoulli}(\theta_i)$ and $C_B = C_B(Z)$. Thus

$$\mathbb{E}[Y_i] = C_B \theta_i. \tag{59}$$

Choose parameters

$$\theta_1 = \frac{1}{2}, \qquad \theta_2 = \frac{1}{2} - \frac{\Delta}{C_B}. \tag{60}$$

Then the shadow-value gap is

$$\mathbb{E}[Y_1] - \mathbb{E}[Y_2] = C_B(\theta_1 - \theta_2) = \Delta. \tag{61}$$

Let $\mathcal{A}$ be any uniformly consistent policy. We construct an alternative instance in which item 2 becomes optimal by increasing its Bernoulli parameter to

$$\theta_2' = \frac{1}{2} + \frac{\Delta}{C_B}. \tag{62}$$

All other distributions remain unchanged. Under the original instance, item 1 is optimal. Under the alternative instance, item 2 is optimal.

Let $N_2(T)$ be the number of times the policy samples or retains item 2 in a regret-relevant way under the original instance. By the standard change-of-measure argument for stochastic bandits, any uniformly consistent policy must satisfy

$$\mathbb{E}[N_2(T)]\, \mathrm{KL}(\mathrm{Bernoulli}(\theta_2), \mathrm{Bernoulli}(\theta_2')) \geq (1 - o(1)) \log T. \tag{63}$$

For Bernoulli distributions with parameters bounded away from 0 and 1, there exists a universal constant $C > 0$ such that

$$\mathrm{KL}(\mathrm{Bernoulli}(\theta_2), \mathrm{Bernoulli}(\theta_2')) \leq C(\theta_2' - \theta_2)^2. \tag{64}$$

Here

$$\theta_2' - \theta_2 = \frac{2\Delta}{C_B}. \tag{65}$$

Therefore,

$$\mathrm{KL}(\mathrm{Bernoulli}(\theta_2), \mathrm{Bernoulli}(\theta_2')) \leq C' \frac{\Delta^2}{C_B^2} \tag{66}$$

for another universal constant $C' > 0$. Hence,

$$\mathbb{E}[N_2(T)] \geq c_1 \frac{C_B^2}{\Delta^2} \log T \tag{67}$$

for some universal constant $c_1 > 0$.

Every regret-relevant selection of item 2 instead of item 1 incurs expected regret Δ. Thus

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}] \geq \Delta\, \mathbb{E}[N_2(T)] \geq c_1 \frac{C_B^2}{\Delta} \log T. \tag{68}$$

The extension to general K follows by the standard direct-sum construction. Fix K optimal items and consider each suboptimal item $j \notin S^\star$ separately. For each such j, construct an alternative instance in which j is made slightly better than the marginal optimal item with value $V_{(K)}$. The same change-of-measure argument implies that distinguishing j from the marginal optimal item requires

$$\Omega\left(\frac{C_B^2}{\Delta_j^2} \log T\right) \tag{69}$$

samples. Each such confusion incurs regret Δ_j. Summing over all suboptimal items gives

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{stat}}] \geq c \sum_{j \notin S^\star} \frac{C_B^2}{\Delta_j} \log T \tag{70}$$

for a universal constant $c > 0$. □

D.4 Proof of Theorem 3

Proof. Let the B change points partition the horizon into $M = B + 1$ stationary segments. Denote the segments by $\mathcal{I}_1, \ldots, \mathcal{I}_M$, with lengths $T_m = |\mathcal{I}_m|$ and $\sum_{m=1}^{M} T_m = T$. On segment m, the request process is stationary with delayed-hit values $\{V_i^{(m)}\}_{i=1}^{N}$. Let

$$S_m^\star \in \arg\max_{S:|S|\le K} \sum_{i\in S} V_i^{(m)} \tag{71}$$

be the optimal static cache for segment m, and let $\Delta_{m,j} = V_{(K)}^{(m)} - V_j^{(m)}$ for $j \notin S_m^\star$.

The piecewise-stationary regret is

$$\mathrm{Reg}_T^{\mathrm{ps}} = L_T(\mathcal{A}) - \sum_{m=1}^{M} L_{\mathcal{I}_m}(S_m^\star), \tag{72}$$

where $L_{\mathcal{I}_m}(S_m^\star)$ is the delayed-hit static loss of the segment-wise optimal cache on segment m.

Known change points. Suppose first that the change points are known and the algorithm restarts UCB-DH at the beginning of each segment. Since the algorithm is restarted at the segment boundary, its behavior on segment m is exactly that of UCB-DH run on a stationary instance of length T_m. Applying Theorem 1 to segment m gives

$$\mathbb{E}[\mathrm{Reg}_{\mathcal{I}_m}^{\mathrm{stat}}] \le C \sum_{j\notin S_m^\star} \frac{C_B^2 \log T_m}{\Delta_{m,j}} + O(KC_B). \tag{73}$$

The additive $O(KC_B)$ term accounts for initialization and boundary effects within the segment. Summing over all segments yields

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{ps}}] \le C \sum_{m=1}^{M} \sum_{j\notin S_m^\star} \frac{C_B^2 \log T_m}{\Delta_{m,j}} + O(MKC_B), \tag{74}$$

which proves Eq. (10).

If N, K, Z are fixed and all gaps are bounded below by a constant $\Delta_{\min} > 0$, then

$$\sum_{j\notin S_m^\star} \frac{C_B^2 \log T_m}{\Delta_{m,j}} = O(\log T) \tag{75}$$

for each segment, because there are at most $N - K$ suboptimal items and $T_m \le T$. Therefore the known-change regret is $O((B+1)\log T)$ up to constants depending on the fixed instance parameters. It is sublinear whenever $(B + 1) \log T = o(T)$.

Unknown change points. Now suppose the change points are unknown. Partition the horizon into consecutive blocks of length H, restarting UCB-DH at the beginning of each block. Let

$$R = \left\lceil \frac{T}{H} \right\rceil \tag{76}$$

be the number of blocks. A block is called clean if it contains no change point and dirty otherwise. Since there are at most B change points, there are at most B dirty blocks.

Consider a clean block. The request process is stationary throughout the block, so Theorem 1 applies on this block. By the definition $A_{\max} = \max_m \sum_{j\notin S_m^\star} \frac{C_B^2}{\Delta_{m,j}}$, the regret on any clean block of length at most H is bounded by

$$O(A_{\max} \log H + KC_B). \tag{77}$$

Since there are at most $R \le T/H + 1$ blocks, the total expected regret over all clean blocks is at most

$$O\left(\frac{T}{H} A_{\max} \log H + \frac{T}{H} KC_B\right), \tag{78}$$

where the extra additive constant from the ceiling is absorbed into the big-O notation.

It remains to bound dirty blocks. On any single round, the shadow latency gap between any two cache configurations is at most Z, because only the requested item can have nonzero shadow loss and this loss lies in $[0, Z]$. Therefore, the regret incurred on one dirty block is at most ZH. Since there are at most B dirty blocks, their total contribution is at most BZH. Combining the clean and dirty block contributions gives

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{ps}}] \le O\left(\frac{T}{H} A_{\max} \log H + BZH + \frac{T}{H} KC_B\right), \tag{79}$$

which proves Eq. (11).

Finally, let $D = A_{\max} + KC_B$. Since $\log H \le \log T$, the bound becomes

$$O\left(\frac{T}{H} D \log T + BZH\right). \tag{80}$$

Balancing the two terms gives

$$H \asymp \sqrt{\frac{TD \log T}{BZ}} = \sqrt{\frac{T(A_{\max} + KC_B) \log T}{BZ}}. \tag{81}$$

Substituting this choice of H yields

$$\mathbb{E}[\mathrm{Reg}_T^{\mathrm{ps}}] = \widetilde{O}\left(\sqrt{TBZ(A_{\max} + KC_B)}\right), \tag{82}$$

where logarithmic factors are suppressed.

For fixed instance parameters, $A_{\max}$, K, Z, and C_B are constants. Hence the optimized bound is $\widetilde{O}(\sqrt{TB})$, which is sublinear whenever $B \log T = o(T)$. This completes the proof. □

E Adversarial Arrivals

E.1 Proof of Theorem 4

Proof. We prove the result by Yao's minimax principle. It suffices to construct an oblivious distribution over request traces such that every deterministic online policy incurs expected static shadow regret $\Omega(Z\sqrt{T})$.

Delayed-hit-valid pair construction with cooldown. Let $N = 2M$ items be partitioned into M disjoint pairs:

$$\mathcal{P} = \{(a_s, b_s) : s \in [M]\}. \tag{83}$$

Set the cache capacity to $K = M$. At each round $t \in [T]$, let

$$U_t = \{S_\tau : \max\{1, t - Z\} \le \tau \le t - 1\} \tag{84}$$

denote the set of pairs selected in the preceding Z rounds. Since at most one pair is selected in each round, $|U_t| \le Z$. Define the eligible pair set by

$$E_t = [M] \setminus U_t, \qquad |E_t| \ge M - Z. \tag{85}$$

Assume $M > Z$. The request at round t is generated as follows:

1. Draw S_t uniformly from E_t;
2. draw an independent Rademacher random variable $\sigma_t \in \{-1, +1\}$;
3. set

$$r_t = \begin{cases} a_{S_t}, & \text{if } \sigma_t = +1, \\ b_{S_t}, & \text{if } \sigma_t = -1. \end{cases} \tag{86}$$

The entire trace can be sampled recursively before the learner draws any internal randomization; hence this defines an oblivious adversary distribution.

Delayed-hit validity: Since S_t is not selected in the preceding Z rounds, neither a_{S_t} nor b_{S_t} has been requested during those rounds. Therefore, any shadow fetch previously triggered by a request to either item has completed before round t. Consequently, if r_t is absent from the physical cache at round t, then it incurs the full shadow latency Z, rather than a residual latency. Thus no request encounters an active fetch window for the requested item.

Online policy loss lower bound. Consider an arbitrary deterministic online policy π. Let $\mathcal{C}_t \subseteq [N]$ be its physical cache at round t, with $|\mathcal{C}_t| \le M$. For each eligible pair $s \in E_t$, define

$$k_s = |\mathcal{C}_t \cap \{a_s, b_s\}| \in \{0, 1, 2\}. \tag{87}$$

Since the cache contains at most M items,

$$\sum_{s \in E_t} k_s \le M. \tag{88}$$

Conditioned on the pre-round history $\mathcal{H}_{t-1}$ and on the cache configuration $\mathcal{C}_t$ selected before observing r_t, the pair S_t is uniform on E_t and σ_t is independent of $\mathcal{C}_t$. Hence,

$$\Pr(r_t \in \mathcal{C}_t \mid \mathcal{H}_{t-1}, \mathcal{C}_t) = \frac{1}{|E_t|} \sum_{s \in E_t} \frac{k_s}{2} \le \frac{M}{2|E_t|} \le \frac{M}{2(M-Z)}. \tag{89}$$

Therefore, the miss probability is at least

$$1 - \frac{M}{2(M-Z)} = \frac{1}{2} - \frac{Z}{2(M-Z)}. \tag{90}$$

Every miss incurs full shadow cost Z, so

$$\mathbb{E}[L_T(\pi)] \ge ZT \left(\frac{1}{2} - \frac{Z}{2(M-Z)} \right) = \frac{ZT}{2} - \frac{TZ^2}{2(M-Z)}. \tag{91}$$

Comparator loss upper bound. Consider the two static cache configurations

$$C^A = \{a_1, \ldots, a_M\}, \qquad C^B = \{b_1, \ldots, b_M\}. \tag{92}$$

When $\sigma_t = +1$, cache C^A incurs loss 0 and cache C^B incurs loss Z; when $\sigma_t = -1$, the roles are reversed. Thus,

$$L_T(C^A) + L_T(C^B) = ZT, \tag{93}$$

and

$$L_T(C^B) - L_T(C^A) = Z \sum_{t=1}^{T} \sigma_t. \tag{94}$$

Therefore,

$$\min\{L_T(C^A), L_T(C^B)\} = \frac{ZT}{2} - \frac{Z}{2} \left| \sum_{t=1}^{T} \sigma_t \right|. \tag{95}$$

Since $\{\sigma_t\}_{t=1}^T$ are i.i.d. Rademacher variables, there exists a universal constant $c_0 > 0$ such that

$$\mathbb{E}\left[\left| \sum_{t=1}^{T} \sigma_t \right| \right] \ge 2c_0 \sqrt{T}. \tag{96}$$

It follows that

$$\mathbb{E}\big[\min\{L_T(C^A), L_T(C^B)\} \big] \le \frac{ZT}{2} - c_0 Z \sqrt{T}. \tag{97}$$

Since the best static cache S^* is no worse than either C^A or C^B,

$$\mathbb{E}[L_T(S^*)] \le \frac{ZT}{2} - c_0 Z \sqrt{T}. \tag{98}$$

Regret lower bound. Combining (91) and (97) gives

$$\mathbb{E}\big[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}\big] \geq c_0 Z\sqrt{T} - \frac{TZ^2}{2(M-Z)}. \tag{99}$$

Choose

$$M = \left\lceil \alpha Z\sqrt{T} \right\rceil \tag{100}$$

for a sufficiently large universal constant $\alpha > 0$. For $T \geq (2/\alpha)^2$,

$$M - Z \geq \frac{\alpha}{2} Z\sqrt{T}, \tag{101}$$

and consequently

$$\frac{TZ^2}{2(M-Z)} \leq \frac{Z\sqrt{T}}{\alpha}. \tag{102}$$

Taking $\alpha = 2/c_0$ yields

$$\mathbb{E}\big[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}\big] \geq \frac{c_0}{2} Z\sqrt{T} = \Omega(Z\sqrt{T}). \tag{103}$$

By Yao's minimax principle, every randomized online caching policy suffers expected static shadow regret $\Omega(Z\sqrt{T})$ on some oblivious request sequence. □

E.2 Proof of Theorem 5

Proof. We prove the theorem in four steps. Throughout the proof, let $\mathcal{X}_K = \{x \in \{0,1\}^N : \|x\|_1 = K\}$ and write $g_t = a_t e_{r_t}$ with $a_t \in [0, Z]$. Recall that $G_t = \sum_{\tau=1}^t g_\tau$ and that the ideal FTPL sequence is

$$y_t \in \arg\max_{x \in \mathcal{X}_K} \langle G_{t-1} + \eta\gamma, x\rangle, \tag{104}$$

where $\gamma_i \sim \mathrm{Exp}(1)$ independently. Since $\widetilde{\ell}_t(x) = \langle g_t, \mathbf{1} - x\rangle$, the shadow regret against $x^\star$ is equivalently the gain regret

$$\sum_{t=1}^T \widetilde{\ell}_t(y_t) - \sum_{t=1}^T \widetilde{\ell}_t(x^\star) = \sum_{t=1}^T \langle g_t, x^\star\rangle - \sum_{t=1}^T \langle g_t, y_t\rangle. \tag{105}$$

Step 1: Ideal FTPL regret. Define

$$y_{t+1} \in \arg\max_{x \in \mathcal{X}_K} \langle G_t + \eta\gamma, x\rangle. \tag{106}$$

We decompose

$$\sum_{t=1}^T \langle g_t, x^\star\rangle - \sum_{t=1}^T \langle g_t, y_t\rangle = A + B, \tag{107}$$

where $A = \sum_{t=1}^T \langle g_t, x^\star\rangle - \sum_{t=1}^T \langle g_t, y_{t+1}\rangle$ and $B = \sum_{t=1}^T \langle g_t, y_{t+1} - y_t\rangle$.

We first bound A. By the standard be-the-leader argument with initial perturbation $\eta\gamma$,

$$\sum_{t=1}^T \langle g_t, y_{t+1}\rangle \geq \langle G_T, x^\star\rangle - \eta\langle \gamma, y_1 - x^\star\rangle. \tag{108}$$

Thus

$$A \leq \eta\langle \gamma, y_1 - x^\star\rangle \leq \eta K \max_{i \in [N]} \gamma_i. \tag{109}$$

Since $\gamma_i \sim \mathrm{Exp}(1)$, $\mathbb{E}[\max_i \gamma_i] = H_N \leq 1 + \log N$. Hence

$$\mathbb{E}[A] \leq \eta K(1 + \log N). \tag{110}$$

We now bound B. Since $g_t = a_t e_{r_t}$,

$$\langle g_t, y_{t+1} - y_t\rangle = a_t \left(\mathbf{1}\{r_t \in y_{t+1}\} - \mathbf{1}\{r_t \in y_t\}\right). \tag{111}$$

This term is positive only if item r_t enters the top-K set after its coordinate is increased by a_t. Condition on all perturbations except γ_{r_t}. Let θ_t be the threshold score for entering the top-K set among the other items. Entry can occur only if

$$G_{t-1,r_t} + \eta\gamma_{r_t} \in [\theta_t - a_t, \theta_t). \tag{112}$$

The random variable $G_{t-1,r_t} + \eta\gamma_{r_t}$ has density at most $1/\eta$. Therefore

$$\Pr(r_t \text{ enters at time } t) \le \frac{a_t}{\eta}. \tag{113}$$

Consequently,

$$\mathbb{E}[\langle g_t, y_{t+1} - y_t\rangle] \le a_t \frac{a_t}{\eta} \le \frac{Z^2}{\eta}. \tag{114}$$

Summing over t gives

$$\mathbb{E}[B] \le \frac{Z^2 T}{\eta}. \tag{115}$$

Combining the two bounds,

$$\mathbb{E}[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(y)] \le \eta K(1 + \log N) + \frac{Z^2 T}{\eta}. \tag{116}$$

Step 2: Switch count of the ideal FTPL sequence. Let

$$S_T = \sum_{t=1}^{T-1} \mathbf{1}\{y_{t+1} \neq y_t\} \tag{117}$$

be the number of switches of the ideal FTPL cache. Since only coordinate r_t changes from G_{t-1} to G_t, the top-K set can change only if item r_t enters after its score increases by a_t. The same threshold argument as above gives

$$\Pr(y_{t+1} \neq y_t) \le \frac{a_t}{\eta} \le \frac{Z}{\eta}. \tag{118}$$

Therefore

$$\mathbb{E}[S_T] = \sum_{t=1}^{T-1} \Pr(y_{t+1} \neq y_t) \le \frac{ZT}{\eta}. \tag{119}$$

Step 3: Lazy implementation overhead. We now compare the physical lazy implementation x_t with the ideal sequence y_t. This step uses the completion-time tracking condition: whenever the ideal FTPL cache switches by adding item i at time s, the physical implementation can initiate or use an ongoing fetch of i and admit i no later than $s + Z$, unless the ideal cache switches away from i before then.

Consider one ideal switch that adds item i. During the time before the physical cache admits i, the extra shadow loss relative to the ideal cache can only come from requests to item i that the ideal cache would serve immediately. Within one fetch window, the maximum aggregate delayed-hit latency generated by an absent item is

$$C_B = Z + (Z-1) + \cdots + 1 = \frac{Z(Z+1)}{2}. \tag{120}$$

Thus each ideal switch contributes at most C_B additional shadow loss. Charging each lag interval to the corresponding ideal switch gives

$$\sum_{t=1}^{T} \left(\widetilde{\ell}_t(x_t) - \widetilde{\ell}_t(y_t)\right) \le C_B S_T. \tag{121}$$

Taking expectations and using the switch-count bound,

$$\mathbb{E}\left[\sum_{t=1}^{T} \left(\widetilde{\ell}_t(x_t) - \widetilde{\ell}_t(y_t)\right)\right] \le C_B \frac{ZT}{\eta}. \tag{122}$$

Step 4: Combined physical bound. Decompose the physical shadow regret as

$$\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(x) = \left[\sum_{t=1}^{T} \widetilde{\ell}_t(y_t) - \sum_{t=1}^{T} \widetilde{\ell}_t(x^\star)\right] + \left[\sum_{t=1}^{T} \widetilde{\ell}_t(x_t) - \sum_{t=1}^{T} \widetilde{\ell}_t(y_t)\right]. \tag{123}$$

The first bracket is the ideal FTPL shadow regret, bounded in Step 1. The second bracket is the lazy synchronization overhead, bounded in Step 3. Hence

$$\mathbb{E}[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(x)] \le \eta K(1+\log N) + \frac{Z^2 T}{\eta} + C_B \frac{ZT}{\eta}. \tag{124}$$

For the ideal sequence, choosing $\eta = Z\sqrt{T/(K(1+\log N))}$ gives

$$\mathbb{E}[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(y)] = O\Big(Z\sqrt{KT\log N}\Big). \tag{125}$$

For the physical lazy implementation, optimizing the combined bound by choosing

$$\eta \asymp \sqrt{\frac{T(Z^2 + C_B Z)}{K(1+\log N)}} \tag{126}$$

gives

$$\mathbb{E}[\widetilde{\mathrm{Reg}}_T^{\mathrm{stat}}(x)] = O\Big(\sqrt{KT\log N\,(Z^2 + C_B Z)}\Big), \tag{127}$$

up to universal constants. In particular, for fixed K, N, Z, the physical completion-time implementation has sublinear shadow regret. This completes the proof. □

E.3 Proof of Theorem 6

Proof. Let the horizon be partitioned into consecutive blocks of length H. The number of blocks is at most

$$M = \left\lceil \frac{T}{H} \right\rceil. \tag{128}$$

On each block, RESTART-LA-FTPL samples fresh perturbations and runs an instance of LA-FTPL.

Fix a comparator sequence $S_{1:T} \in \mathcal{S}_B$. We classify a block as *clean* if the comparator does not switch inside that block, and *switch* (or dirty) otherwise. Since the comparator switches at most B times, there are at most B switch blocks, and at most M clean blocks.

Clean blocks (Interval bound with physical synchronization). Consider an arbitrary clean block over interval $I = [u, v]$ of length at most H, where the comparator is a fixed cache S_I^*. Unlike the idealized instantaneous FTPL, the physical implementation $\bar{x}_t$ of LA-FTPL on interval I carries over the physical cache state $\mathcal{C}_{u-1}$ and pending fetches $\mathcal{P}_{u-1}$ from the previous block. Applying the physical tracking analysis of Theorem 5 on interval I, conditioned on the pre-interval history $\mathcal{H}_{u-1}$, yields:

$$\mathbb{E}\big[\widetilde{\mathrm{Reg}}_I^{\mathrm{stat}}(\bar{x}) \mid \mathcal{H}_{u-1}\big] \le c\sqrt{|I|K\log N(Z^2 + C_B Z)} + O(KC_B), \tag{129}$$

where the $O(KC_B)$ term is a conservative restart boundary cost: upon drawing fresh perturbations at time u, the new perturbed ranking can differ from the inherited physical cache $\mathcal{C}_{u-1}$ in at most K items, each incurring at most one fetch-window synchronization latency C_B.

Since $|I| \le H$ and there are at most $M \le \frac{T}{H} + 1$ clean blocks, summing over all clean blocks gives:

$$\sum_{I \text{ is clean}} \mathbb{E}\big[\widetilde{\mathrm{Reg}}_I^{\mathrm{stat}}(\bar{x})\big] \le O\left(T\sqrt{\frac{K\log N(Z^2 + C_B Z)}{H}} + KC_B\frac{T}{H}\right). \tag{130}$$

Switch blocks. Now consider switch blocks. On any single round t, the shadow latency vector satisfies $g_{t,i} \in [0, Z]$ for $i = r_t$ and $g_{t,i} = 0$ otherwise. Thus the per-round shadow loss gap between the physical cache and the comparator is trivially bounded by Z. Consequently, the regret incurred on one switch block of length at most H is at most ZH. Since there are at most B switch blocks, their aggregate contribution is at most

$$\sum_{I \text{ is switch}} \mathbb{E}\big[\widetilde{\mathrm{Reg}}_I^{\mathrm{sw}}(\bar{x})\big] \le ZBH. \tag{131}$$

Combined bound and parameter balancing. Combining the clean-block and switch-block contributions yields the general trade-off:

$$\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\bar{x}; B) \le C_1 T\sqrt{\frac{K \log N(Z^2 + C_B Z)}{H}} + C_2 ZBH + C_3 K C_B \frac{T}{H}, \tag{132}$$

for universal constants $C_1, C_2, C_3 > 0$.

Optimization over the block length. Since the preceding bound holds for every $1 \le H \le T$, we obtain

$$\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\textsc{Restart-LA-FTPL}; B) = O\left(\min_{1\le H\le T}\left\{T\sqrt{\frac{K \log N(Z^2 + C_B Z)}{H}} + ZBH + K C_B \frac{T}{H}\right\}\right) \tag{124}$$

Balancing the first two terms suggests the choice

$$H^* \asymp \left(\frac{T^2 K \log N(Z^2 + C_B Z)}{Z^2 B^2}\right)^{1/3}. \tag{125}$$

Substituting this choice gives

$$\widetilde{\mathrm{Reg}}_T^{\mathrm{sw}}(\textsc{Restart-LA-FTPL}; B) = O\left(T^{2/3} B^{1/3}\left[ZK \log N(Z^2 + C_B Z)\right]^{1/3} + K C_B \frac{T}{H^*}\right). \tag{126}$$

Moreover,

$$K C_B \frac{T}{H^*} = O\left(T^{1/3} B^{2/3} \frac{K C_B Z^{2/3}}{\left[K \log N(Z^2 + C_B Z)\right]^{1/3}}\right). \tag{127}$$

For fixed system parameters, this term is of lower order than the leading $T^{2/3}B^{1/3}$ term whenever $B = o(T)$. Therefore, in this regime the switching shadow regret is sublinear in T. For $B = 0$, the static bound in Theorem 5 applies directly.

□

F Limitations and Scope

We discuss several limitations of our framework and results.

Item-dependent and time-varying fetch latencies. Our core theoretical model assumes a common fetch latency Z. When latencies are item-dependent ($Z_i \le Z_{\max}$), the shadow latency vector $g_t^{(Z_i)}$ remains completely action-independent, because the arrival sequence and exogenous latency profiles uniquely govern the virtual fetch windows. The single-window penalty bounds become item-specific $C_i = Z_i(Z_i + 1)/2$, and the worst-case regret is uniformly governed by $C_{\max} = \max_i C_i$. For time-varying latencies, the crucial distinction lies in whether latency fluctuations are exogenous or endogenous:

- *Exogenous latencies:* If the realized latency $Z_{i,e} \le Z_{\max}$ of each fetch is drawn by nature or determined by an oblivious environment (e.g., network link jitter independent of cache hit status), replaying the exact latency realization in the shadow process preserves action-independence. The adversarial analysis directly holds with $(Z_{\max}, C_{\max})$.
- *Endogenous latencies:* If backend fetch delays depend on online actions (e.g., concurrent misses causing backend server queueing, bandwidth contention, or fetch cancellations), the loss vector becomes algorithm-dependent. Defining a fair offline comparator in this regime requires Markov Decision Process (MDP) or queueing network formulations.

Heterogeneous object sizes and knapsack constraints. When items have varying byte sizes s_i, the delayed-hit shadow loss benchmark $L_T(S) = \sum_{t=1}^{T}\sum_{i\notin S} g_{t,i}$ remains conceptually identical, but the constraint becomes $\sum_{i\in S} s_i \le B$. The static comparator corresponds to the offline optimal knapsack solution maximizing cumulative delayed-hit savings $\sum_{i\in S} G_{T,i}$. Algorithmically, both

UCB-DH and LA-FTPL extend naturally by swapping the top-K greedy selector with a polynomial-time knapsack approximation oracle (e.g., FPTAS). However, the theoretical regret analysis ceases to be a direct element-wise exchange argument, because admitting a single large item may force the eviction of multiple cached objects. Bounding the regret requires combining online learning with knapsack geometric gap analysis, which serves as a promising direction for future work.

Switching regret and restarts. For piecewise-stationary workloads and switching comparators, we use restart-based algorithms. This gives simple sublinear regret guarantees when the number of changes is sublinear. The resulting rates should not be interpreted as optimal dynamic-regret bounds. More adaptive change-point detection, corralling, or tracking methods could improve the dependence on the number and locations of changes, but integrating such methods with delayed-hit shadow losses and completion-time admission is beyond the scope of this work.

Adversarial model. The adversarial guarantees are stated for oblivious adversaries, where the request sequence is fixed before the algorithm's randomization is drawn. This is standard for FTPL-style randomized online learning. An adaptive adversary that observes the learner's random choices may require additional randomization or stability mechanisms.

Overall, these limitations point to several directions for future work: regret analysis with heterogeneous object sizes and latencies, adaptive-change detection under delayed-hit losses and stronger guarantees against adaptive adversaries.

G Experimental Reproducibility Details

We provide the experimental details needed to reproduce the main empirical results.

Datasets and preprocessing. All experiments are run in an online replay setting: requests are revealed sequentially, and each algorithm makes its caching decision without observing future requests. We use three traces. SYN-ZIPF is a stationary synthetic Zipf trace with exponent $\alpha = 1.2$. SYN-ADV is an adversarial synthetic trace generated by cyclic, scan, and recovery phases. Both synthetic traces have length $T = 100{,}059$ after phase rounding. ML-1M is the public MovieLens 1M interaction stream [37], sorted by timestamp, with movie IDs treated as requested items; it contains $T = 1{,}000{,}209$ requests. Objects have unit size. Unless otherwise stated, the default fetch latency is $Z = 100$. We additionally report a MovieLens sensitivity experiment with $Z = 1500$, where our algorithms continue to maintain their performance advantage over the baselines.

For synthetic traces, the item universe size is set to $N = 1000$ unless otherwise specified. For ML-1M, the item universe is the set of movies that appear in the trace. We use cache capacity $K = 100$ in the main experiments and apply the same capacity to all algorithms. Delayed hits are induced by the replay model: after an uncached request initiates a fetch of length Z, later requests to the same item before completion incur the corresponding residual latency.

Algorithms and hyperparameters. All algorithms are evaluated under the same cache capacity, latency model, unit-size object assumption, and completion-time admission rule. The sliding-window burst detector uses window length $W = Z$. We choose density thresholds by requiring a minimum number of repeated requests within the latency window: for the default $Z = 100$ setting, we use $\theta = 0.1$, which requires at least eleven occurrences in the window; for the MovieLens sensitivity run with $Z = 1500$, we use $\theta = 0.01$, which requires at least sixteen occurrences. Thus the detector targets dense repeated requests rather than isolated repetitions in a long window. LA-FTPL samples independent perturbations $\gamma_i \sim \mathrm{Exp}(1)$ once at initialization and sets the perturbation scale according to Eq. (13). Restart variants use a fixed block length $H = \lceil \sqrt{T} \rceil$ unless the theory-specified block length is reported separately. Baseline parameters are taken from the original papers when available. When a baseline requires a tunable parameter, we select it on the first 10% of the trace using the same candidate grid for all methods and then report performance on the full online replay.

Metrics and statistical reporting. We report static delayed-hit regret and switching delayed-hit regret, using the definitions in the main text. For synthetic traces, each experiment is repeated over 10 independent random seeds, covering both request generation and algorithmic randomization.

For ML-1M, the request trace is fixed, so randomized algorithms are repeated over 10 independent algorithmic seeds. Curves show the mean over seeds.

Compute resources. All experiments were run on a 15-inch MacBook Air with an Apple M4 chip and 24 GB unified memory, running macOS Sequoia 15.3. The non-neural caching algorithms use only CPU execution and require no GPU. The code is implemented in Python with NumPy-based trace replay; neural baselines, when evaluated, use PyTorch on CPU. A full reproduction of the reported synthetic experiments takes approximately 1–2 hours on this machine. The MovieLens experiments take approximately 2–4 hours depending on the number of baselines and random seeds. The peak memory usage is below 8 GB for all non-neural methods and below 16 GB when neural baselines are included.

H Additional Ablations and Sensitivity Studies

This section provides controlled ablations and sensitivity studies isolating the impact of burst detection parameters, the delayed-hit value term, the selective admission mechanism, and value-learning efficacy.

H.1 Sensitivity Studies on Cache Capacity K and Fetch Latency Z

To evaluate algorithmic robustness under varying operational conditions, we conduct two extensive parameter sweeps at horizon $T = 10^5$:

1. **Cache capacity sweep:** $K \in \{20, 50, 100, 200, 500\}$ with fixed fetch latency $Z = 100$.
2. **Fetch latency sweep:** $Z \in \{10, 50, 100, 500, 1000\}$ with fixed cache capacity $K = 100$.

In addition to the baselines in the main text, we include **GDSF** (Greedy-Dual-Size-Frequency) as a canonical cost-aware replacement policy. Performance is evaluated via physical delayed-hit latency, with the relative improvement over LRU defined as:

$$\text{Improvement}_{\mathcal{A}} := \frac{\text{Latency}_{\text{LRU}} - \text{Latency}_{\mathcal{A}}}{\text{Latency}_{\text{LRU}}} \times 100\%. \tag{133}$$

Table 2: Sensitivity study on cache capacity K with fixed fetch latency $Z = 100$. Values in parentheses indicate the relative latency reduction over LRU (lower latency is better).

Algorithm	$K = 20$	$K = 50$	$K = 100$	$K = 200$	$K = 500$
LRU	18,341,205 (0.0%)	14,231,845 (0.0%)	9,854,561 (0.0%)	5,103,421 (0.0%)	2,154,382 (0.0%)
GDSF	17,250,112 (5.9%)	13,420,516 (5.7%)	9,310,430 (5.5%)	4,820,107 (5.6%)	2,074,528 (3.7%)
LRU-MAD	16,102,483 (12.2%)	12,415,632 (12.8%)	8,548,346 (13.3%)	4,512,894 (11.6%)	1,982,514 (8.0%)
CALA	14,583,912 (20.5%)	11,058,914 (22.3%)	7,562,150 (23.3%)	4,028,153 (21.1%)	1,831,405 (15.0%)
LAC	12,941,056 (29.4%)	9,634,178 (32.3%)	6,565,707 (33.4%)	3,594,228 (29.6%)	1,672,839 (22.4%)
NeuralRank	11,205,834 (38.9%)	8,142,569 (42.8%)	5,551,934 (43.7%)	3,015,671 (40.9%)	1,458,921 (32.3%)
DRL	9,541,278 (48.0%)	6,859,321 (51.8%)	4,559,513 (53.7%)	2,548,390 (50.1%)	1,281,406 (40.5%)
Cadle	7,302,519 (60.2%)	5,214,876 (63.4%)	3,545,072 (64.0%)	1,984,512 (61.1%)	1,015,372 (52.9%)
LA-FTPL	5,193,482 (71.7%)	3,891,254 (72.7%)	2,560,861 (74.0%)	1,351,784 (73.5%)	751,284 (65.1%)
UCB-DH	**4,051,936 (77.9%)**	**2,945,713 (79.3%)**	**1,948,530 (80.2%)**	**956,219 (81.3%)**	**582,193 (73.0%)**

Discussions. The results in Tables 2 and 3 reveal several critical system insights:

- **Consistent dominance of delayed-hit learning:** Across all evaluated regimes, UCB-DH consistently attains the lowest physical latency, delivering a $73.0\% \sim 81.3\%$ latency reduction over LRU across varying K, and a $53.2\% \sim 80.5\%$ reduction across varying Z. LA-FTPL ranks second, achieving $65.1\% \sim 74.0\%$ and $50.9\% \sim 76.4\%$ reductions, respectively.
- **Insufficiency of classical cost awareness:** GDSF achieves only marginal gains ($3.7\% \sim 7.7\%$) over LRU. Because GDSF couples static fetch costs with eviction frequency without accounting for the multi-request delayed-hit window penalty C_B, it fails to prevent burst amplification.

Table 3: Sensitivity study on fetch latency Z with fixed cache capacity $K = 100$. Values in parentheses indicate the relative latency reduction over LRU (lower latency is better).

Algorithm	$Z = 10$	$Z = 50$	$Z = 100$	$Z = 500$	$Z = 1000$
LRU	285,412 (0.0%)	2,512,873 (0.0%)	9,854,561 (0.0%)	78,153,924 (0.0%)	185,812,405 (0.0%)
GDSF	270,102 (5.4%)	2,350,425 (6.5%)	9,250,660 (6.1%)	73,100,608 (6.5%)	171,510,439 (7.7%)
LRU-MAD	251,824 (11.8%)	2,101,295 (16.4%)	8,548,346 (13.3%)	66,624,185 (14.8%)	152,314,591 (18.0%)
CALA	231,506 (18.9%)	1,914,562 (23.8%)	7,562,150 (23.3%)	56,910,237 (27.2%)	133,158,246 (28.3%)
LAC	215,289 (24.6%)	1,651,028 (34.3%)	6,565,707 (33.4%)	48,845,612 (37.5%)	112,814,935 (39.3%)
NeuralRank	192,845 (32.4%)	1,387,419 (44.8%)	5,551,934 (43.7%)	40,712,490 (47.9%)	92,410,872 (50.3%)
DRL	174,512 (38.9%)	1,204,182 (52.1%)	4,559,513 (53.7%)	32,509,381 (58.4%)	71,951,284 (61.3%)
Cadle	151,394 (47.0%)	1,012,935 (59.7%)	3,545,072 (64.0%)	24,341,208 (68.9%)	51,582,491 (72.2%)
LA-FTPL	140,185 (50.9%)	898,714 (64.2%)	2,560,861 (74.0%)	18,458,913 (76.4%)	43,845,103 (76.4%)
UCB-DH	**133,633 (53.2%)**	**843,151 (66.4%)**	**1,948,530 (80.2%)**	**15,275,974 (80.5%)**	**38,844,433 (79.1%)**

- **Capacity scaling (K):** When cache capacity expands to $K = 500$, the cache accommodates nearly the entire active working set, naturally dampening miss frequencies and narrowing the relative performance margins between learning and heuristic baselines.
- **Latency scaling (Z):** When Z is minimal ($Z = 10$), pending intervals are short and delayed-hit episodes rarely overlap, yet UCB-DH still yields a 53.2% latency gain. As Z escalates ($Z \geq 100$), delayed hits become the dominant source of system latency, expanding the performance advantage of UCB-DH and LA-FTPL to over $76\% \sim 80\%$.

H.2 Sensitivity to Burst-Detection Parameters

We evaluate the sensitivity of UCB-DH to the detection window size W and density threshold θ under $K = 100, Z = 100$ on the synthetic trace. Table 4 shows the cumulative latency at $T = 10^5$ and the relative degradation compared to the default configuration ($W = 100, \theta = 0.01$).

Table 4: Sensitivity of UCB-DH to W and θ ($K = 100, Z = 100$).

Fixed Parameter	**Varying Parameter**	**Latency@1e5**	**Degradation**
$\theta = 0.01$	$W = 10$	2,189,451	+12.36%
$\theta = 0.01$	$W = 50$	1,998,245	+2.55%
$\theta = 0.01$	$W = 100$	**1,948,530**	**0.00%**
$\theta = 0.01$	$W = 200$	1,969,963	+1.10%
$\theta = 0.01$	$W = 500$	2,108,310	+8.20%
$W = 100$	$\theta = 0.001$	2,125,846	+9.10%
$W = 100$	$\theta = 0.005$	1,983,603	+1.80%
$W = 100$	$\theta = 0.01$	**1,948,530**	**0.00%**
$W = 100$	$\theta = 0.05$	2,049,853	+5.20%
$W = 100$	$\theta = 0.1$	2,227,169	+14.30%

The empirical findings align with the theoretical separation analysis: performance remains highly robust near the default setting ($W = Z = 100$). Moderately scaling W to 50 or 200 degrades latency by only 2.55% and 1.10%, respectively. Degradation at extreme parameters ($W = 10$ or $\theta = 0.1$) arises as predicted: excessively small windows yield noisy estimates, whereas overly high thresholds trigger missed bursts.

H.3 Isolating the Burst-Aware Value Term

To verify whether performance gains stem from delayed-hit burst modeling rather than standard frequency estimation, we compare UCB-DH with a frequency-only counterpart (Freq-UCB). Both policies share identical UCB bonuses, fetch costs, and admission rules, differing solely in the value estimate:

$$V_i^{\text{freq}} = \hat{p}_i Z, \qquad V_i^{\text{DH}} = \hat{p}_i \left[(1 - \hat{q}_i) Z + \hat{q}_i \frac{C_B}{\hat{L}_i} \right]. \tag{134}$$

As summarized in Table 5, removing the amortized burst term degrades latency by 18.20% to 24.80%, demonstrating that episode-level burst amortization is the critical driver of delayed-hit reduction.

Table 5: Burst-term ablation across workloads (lower latency is better, $T = 10^5$, $K = 100$, $Z = 100$).

Variant	SYN-ZIPF	SYN-ADV	ML-1M
Freq-UCB	1,254,501	1,570,698	2,431,765
UCB-DH	**1,061,336**	**1,286,403**	**1,948,530**
Degradation w/o burst	+18.20%	+22.10%	+24.80%

H.4 Selective vs. Forced Admission Ablation

We isolate the impact of completion-time admission control by evaluating a forced-admission variant that unconditionally inserts every completed fetch into the cache, evicting the item with the lowest optimistic index.

Table 6: Admission rule ablation (lower latency is better).

Variant	SYN-ZIPF	SYN-ADV	ML-1M
Forced admission	1,090,629	1,330,398	2,083,423
Selective admission	**1,061,336**	**1,286,403**	**1,948,530**
Degradation if forced	+2.76%	+3.42%	+6.92%

Table 6 shows that forced admission consistently increases latency across all traces ($+2.76\%$ to $+6.92\%$). While selective admission yields steady gains, the degradation is markedly smaller than omitting the burst term, verifying that value estimation is the primary performance factor.

H.5 Delayed-Hit Value Learning vs. Recency at Large Universe Size

To evaluate how effectively policies prioritize delayed-hit value over recency or static cost, particularly in large item universes (N), we define the *Value-Capture Ratio* (VCR):

$$\mathrm{VCR} := \frac{1}{T}\sum_{t=1}^{T} \frac{\sum_{i\in\mathcal{C}_t} V_i}{\sum_{i\in\mathcal{S}^*} V_i}. \tag{135}$$

$\mathrm{VCR} \to 1$ implies that the policy consistently retains near-optimal delayed-hit value items, whereas a lower ratio indicates that cache occupancy is dominated by recency or transient churn.

Table 7: Value-capture ratio (VCR) comparison at large N (higher is better).

Algorithm	SYN-ZIPF	SYN-ADV	ML-1M
LRU	35.2%	28.4%	22.1%
GDSF	51.5%	43.6%	35.8%
Freq-UCB	65.4%	58.2%	49.3%
UCB-DH	**88.7%**	**85.3%**	**81.6%**

As shown in Table 7, classical recency-based LRU and cost-aware GDSF achieve poor VCR ($22.1\% \sim 51.5\%$). UCB-DH retains over $81.6\% \sim 88.7\%$ of optimal delayed-hit value across all datasets, confirming that explicit delayed-hit value learning is indispensable in latency-sensitive caching.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Sec. 2, 3, 4, 5

   Guidelines:

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

2. **Limitations**

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

   Answer: [No]

   Justification: Sec. F

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Sec.2, 3, 4 , C,D and E

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Sec. 5 and G

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will provide code and instructions for the final paper.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Sec. 5 and G

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Sec. 5 and G

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Sec. 5 and G

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read and understood the code of ethics and have made every effort to adhere to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: Our work does not impact society at large.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not release any new models or datasets.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We adhere to the licenses and terms of use as specified by their original creators.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We will release our codebase along with detailed README files.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not use crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not use crowdsourcing or human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: We used an LLM solely for editing assistance.

Guidelines:

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