Two-Timescale Adaptive Memory for CTR Prediction under Temporal Distribution Shift

Abstract

Click-through rate (CTR) prediction is important in online advertising and recommendation systems. The CTR prediction can be challenging in practice, especially under temporal distribution shift. Existing methods often treat historical adaptation over model training and within-day information online updates separately, limiting their ability to exploit both heterogeneous historical data and newly available feedback. We propose Two-Timescale Adaptive Memory (TTAM), a causal framework that combines context-dependent historical weighting with adaptive-persistence online calibration. At the slower timescale, TTAM learns which historical horizons to use and how quickly their weights should change. At the faster timescale, it adapts calibration memory using delayed within-day feedback. We theoretically establish stability properties and regret guarantees for calibration aggregation. Experiments show that TTAM improves robustness across 14 controlled drift regimes and increases downstream bidding value. On the full Criteo and Avazu datasets, TTAM achieves strong performance against competing baselines under rolling-origin evaluation, supporting the value of adaptive memory at both timescales.

1 Introduction

Click-through rate (CTR) prediction is a central task in online advertising and recommendation systems. It estimates the probability that a user will click on a displayed advertisement or recommended item. These estimates help platforms select relevant content and help advertisers decide how much to *bid* for an advertising opportunity. Accurate predictions therefore support both user experience and the efficient allocation of advertising budgets. Maintaining reliable CTR predictions is consequently an important practical objective.

The CTR prediction can be challenging in practice. One major factor is *temporal distribution shift*—user interests evolve, advertising campaigns change, and the composition of traffic varies across days and hours—making the choice of training history involved. Using a long history provides more observations but may retain outdated patterns, while recent data allows faster adaptation but can discard useful information and produce less reliable estimates. Moreover, these changes need not affect all users or advertisements universally: a recent shift may make older observations less relevant for one group while leaving them useful for another. Moreover, changes can also occur between scheduled model updates (e.g., daily model retraining), creating a need to use newly observed responses without waiting for another full training cycle. These practical difficulties motivate methods that balance responsiveness with the retention of useful knowledge [15].

Existing approaches address several aspects of this problem. Adaptive rolling-window methods use past prediction errors to determine how much history is relevant, while expert-based methods combine multiple models that respond differently to changing data [8]. Another type of algorithms adjusts predictions as the latest outcomes become available by adapting online learning approaches [7]. These approaches motivate different forms of temporal adaptation, but each can fail under certain circumstances. A single historical weighting rule can overlook differences across users and advertisements and learn only an average pattern. Likewise, historical model selection alone does not use feedback arriving between retraining cycles, which could provide additional signals. On the other hand, continually adapting to new information could accelerate the forgetting of older patterns that might still be useful. These tradeoffs motivate a more integrated treatment of temporal adaptation.

In this paper, we propose Two-Timescale Adaptive Memory (TTAM), a causal framework that addresses these decisions together. At the slower timescale, TTAM performs a full training cycle and combines predictors trained on different lengths of history. Their relative weights depend on the features of each sample, allowing different users and advertisements to draw on different historical information. The method also controls how quickly these weights should change, balancing responsiveness to new conditions with stability over time. At the faster timescale, TTAM uses newly available click outcomes to adjust predictions between consecutive training cycles. It adapts how much fresh information is used, rather than committing to a single memory duration. Both components use feedback only

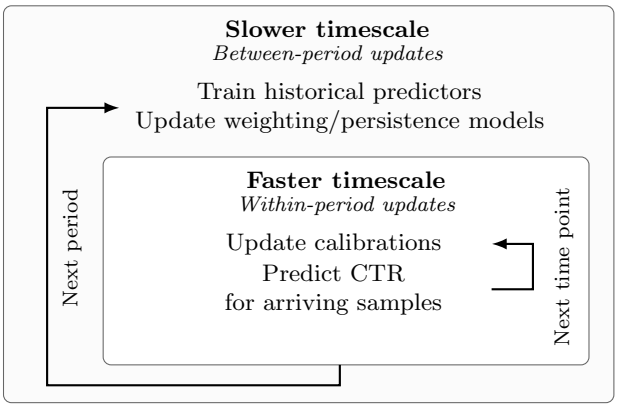

Figure 1: TTAM follows two nested update loops. Historical models and weighting mechanisms are updated between periods, while calibration and prediction proceed within each period. All updates use only available feedback.

after it becomes available. The central improvement is therefore to make temporal memory adaptive at both levels: the history supporting the prediction and the recent feedback used to refine it. Fig. 1 illustrates the pipeline of the TTAM algorithm.

Contributions. Our contributions are threefold.

1. We introduce a causal two-timescale adaptive-memory framework. Its slow layer, AMG-TP, uses context-dependent weights over multiple training horizons and a learned persistence parameter to maintain continuity in these weights over time. Its fast layer, AP-OPS, combines reset, short-memory, and long-memory Platt calibrators using online feedback. The two layers learn both *which* historical information remains useful and *how long* an adaptation state should persist.

2. We provide theoretical guarantees for the framework. We establish regret bounds for the calibration policy against a switching sequence of memory experts. These results connect TTAM to classical expert-tracking theory and extend the comparison beyond a single fixed calibration map to changing calibration memories under delayed feedback.

3. We validate the two adaptation layers through controlled experiments and evaluations on two public CTR datasets, Criteo and Avazu. Rolling-origin evaluation demonstrates consistent improvements in robustness and prediction accuracy from TTAM's adaptive-memory design. The prediction gains under recurring and local drift translate into greater bidding value under an unchanged autobidding policy.

2 Related Work

Learning under temporal distribution shift. Learning under temporal distribution shift concerns prediction when the data distribution changes over time. Such changes may affect the distribution of input features, the relationship between features and outcomes, or both. As a result, patterns learned from earlier observations may become less useful for future prediction. This problem is closely related to concept drift in data streams, for which Gama et al. [4], Lu et al. [16] review methods for detecting changes and adapting predictive models.

Existing methods differ in how they incorporate new observations while retaining useful historical information. Detection-based methods monitor prediction errors and trigger model updates or replacement when there is evidence of a change [3]. Window-based methods restrict learning to recent observations and adjust the window length to balance responsiveness with the amount of available data [2]. Instead of discarding older observations, forgetting methods gradually reduce their influence through temporal weights. For example, Bennett and Clarkson [1] learn a weighting mechanism from predictive performance, allowing the effective training history to adapt over time. Historical information can also guide the choice among candidate predictors. Han et al. [8] adapt the amount of recent history used to estimate current prediction risk and select a model. Ensemble methods instead combine several predictors and adjust their influence according to observed performance. They may also introduce new predictors or remove poorly performing ones as the stream evolves [14].

Mixture-of-experts and adaptive aggregation. Mixture-of-experts (MoE) models combine several predictors through a learned gating model. The gate assigns weights according to the input, allowing different samples to use different combinations of experts. Jacobs et al. [11] develop this approach by jointly learning the experts and their gating weights, encouraging specialization across different parts of the data. Hierarchical architectures extend this idea through multiple levels of gating, allowing the input space to be divided into progressively finer regions [12].

In a changing environment, the usefulness of an expert may also vary over time. For CTR prediction, Liu et al. [15] propose AdaMoE, which uses an adaptive weighting policy to balance responsiveness to new patterns with retention of previously learned information. A related literature on online expert aggregation studies how to update combination weights from sequential prediction errors. Exponential weighting gives greater influence to experts with smaller cumulative losses. To

accommodate changes in which expert performs best, Herbster and Warmuth [10] introduce Fixed Share, which redistributes part of the weight among experts and provides guarantees against switching sequences of experts. The sharing rate controls how readily the learner can shift between experts. Rather than choosing this rate in advance, Learn-α combines Fixed Share learners with different switching rates and updates their relative weights using observed losses [17].

Online calibration. Probability calibration concerns agreement between predicted probabilities and observed event frequencies among samples receiving similar predictions. Post-hoc calibration methods seek this agreement by adjusting the outputs of a trained model. Platt scaling fits a sigmoid transformation of model scores using labeled calibration data [18]. A simpler alternative is temperature scaling, which rescales logits using a single parameter. Guo et al. [5] demonstrate its effectiveness for calibrating neural network predictions.

In a changing environment, a correction fitted to historical data may become outdated, motivating calibration methods that update as new outcomes are observed. For adjusting the predictions of a trained model, Gupta and Ramdas [7] introduce OPS, which updates the slope and intercept through online logistic regression using log loss. They also combine OPS with an additional calibration procedure to obtain guarantees for adversarial outcome sequences. Online optimization provides tools for these sequential updates. In particular, Online Newton Step achieves logarithmic regret relative to the best fixed parameters for exp-concave losses under standard boundedness assumptions [9]. When outcomes are delayed, the learner must make further predictions before receiving feedback on earlier ones. Joulani et al. [13] provide general reductions for adapting online learning algorithms to this setting and characterize how delays affect regret.

3 Preliminaries

Chronological CTR stream. The time horizon is divided into deployment periods $d = 1, \dots, D$. In CTR prediction, a period is usually a day or a fixed temporal block. The prediction models are trained between consecutive periods using only the data available at the time of training. Within each period, observations occur at a finite set of equally spaced time points $t = 1, \dots, T$, which may represent hours within a day.

The samples in period d are denoted by

$$\left\{ \left\{ (\boldsymbol{x}^{d,t,i}, y^{d,t,i}) \right\}_{i=1}^{n_{d,t}} \right\}_{t=1}^{T}, \tag{1}$$

where $\boldsymbol{x}^{d,t,i}$ contains the context of a sample, such as the advertiser and advertisement category. The click label satisfies $y^{d,t,i} \in \{0, 1\}$, with $y^{d,t,i} = 1$ if the advertisement is clicked and 0 otherwise. At time point t of period d, the contexts of all $n_{d,t}$ arriving samples $\{\boldsymbol{x}^{d,t,i}\}_{i=1}^{n_{d,t}}$ are observed before predictions are made.

Each label becomes available after a uniform delay $\delta \geq 0$, measured in time steps. At time t of period d, the learner has access to labels from earlier periods whose feedback delays have elapsed, together with labels from time points

$$1 \leq t' \leq \min\{t - 1, t - \delta\} \tag{2}$$

in the current period. The restriction $t' < t$ excludes outcomes from the current time point even when $\delta = 0$.

Mixture of experts. A mixture-of-experts (MoE) model combines the outputs of K predictive models, referred to as experts. For an input $\boldsymbol{x}$, let $\boldsymbol{p}(\boldsymbol{x}) = (p_1(\boldsymbol{x}), \dots, p_K(\boldsymbol{x}))^\top$ denote the vector of expert predictions. A gating model W_ϕ assigns a weight vector

$$\mathbf{w}(\boldsymbol{x}) = \mathrm{softmax}(W_\phi(\boldsymbol{x})) \in \Delta_K, \tag{3}$$

where Δ_K is the probability simplex in $\mathbb{R}^K$. The mixture prediction is

$$q(\boldsymbol{x}) = \sum_{k=1}^{K} w_k(\boldsymbol{x}) p_k(\boldsymbol{x}) = \mathbf{w}(\boldsymbol{x})^\top \boldsymbol{p}(\boldsymbol{x}). \tag{4}$$

The gating model is trained to assign larger weights to experts that perform well for the current input. Because the weights depend on the input, different samples may use different combinations of experts.

Online Platt scaling. Online Platt scaling (OPS) adjusts predicted logits using feedback received during deployment [6]. For a base prediction probability $q \in (0, 1)$, the calibrated prediction is

$$\widehat{p} = \mathrm{sigmoid}(a \, \mathrm{logit}(q) + b), \tag{5}$$

where $\mathrm{logit}(q) = \log(\frac{q}{1-q})$ and $\mathrm{sigmoid}(z) = (1 + \exp(-z))^{-1}$. The slope a adjusts the scale of the prediction logits, while the intercept b shifts them. The choice $(a, b) = (1, 0)$ leaves the base prediction unchanged.

OPS adjusts the slope and intercept through online learning updates based on newly available labels. For example, a projected gradient update takes the form

$$\boldsymbol{\theta}_{r+1} = \Pi_\Theta(\boldsymbol{\theta}_r - \eta_r \nabla L_r(\boldsymbol{\theta}_r)), \tag{6}$$

where $\boldsymbol{\theta}_r = (a_r, b_r)^\top$, η_r is the learning rate, and Π_Θ denotes projection onto the parameter domain Θ. Here, L_r is the average log loss on the data available for update r. Other online learning methods, such as Online Newton Step, can also be used.

4 Two-Timescale Adaptive Memory

Adaptive historical memory. Our first algorithmic block adapts the mixture-of-experts framework to temporal distribution shift by treating models trained on different historical horizons as experts. A gating model assigns sample-specific weights to these experts. Learned temporal persistence blends these weights with a memory of previous expert allocations, balancing responsiveness and stability.

TTAM maintains $K = 5$ experts with historical horizons $\mathcal{H} = \{1, 3, 7, 14, \text{expanding}\}$. Before each period d, expert k is trained using available data from its corresponding historical horizon. For sample (d, t, i), the expert predictions are collected in $\boldsymbol{p}^{d,t,i} = (p_1^{d,t,i}, \ldots, p_K^{d,t,i})^\top$, where $p_k^{d,t,i} = f_k^d(\boldsymbol{x}^{d,t,i})$.

Before period $d \geq 2$, we update the gating parameters ϕ and persistence parameters ψ using available feedback from the preceding period. These updates are based on the regularized log loss of the historical mixture. We also update the memory using the weights deployed during period $d-1$:

$$\overline{\boldsymbol{w}}^{d-1} = \frac{\sum_{t=1}^{T} \sum_{i=1}^{n_{d-1,t}} \boldsymbol{w}^{d-1,t,i}}{\sum_{t=1}^{T} n_{d-1,t}}, \tag{7}$$

$$\boldsymbol{m}^{d-1} = (1-\rho)\boldsymbol{m}^{d-2} + \rho\, \overline{\boldsymbol{w}}^{d-1}, \tag{8}$$

where $\rho \in [0, 1]$ controls the rate of memory updates. The first period uses initialized model parameters and $\boldsymbol{m}^0 = \mathbf{1}/K$.

Let $\boldsymbol{s}^{d-1}$ summarize the information available before period d, including recent expert losses, disagreement among experts, empirical CTR, changes in expert weights, and indicators of distribution shift. The persistence model determines

$$\beta_d = \text{sigmoid}\big(R_\psi(\boldsymbol{s}^{d-1})\big) \in [0, 1]. \tag{9}$$

The model parameters, memory, and persistence parameter remain fixed throughout period d.

As each sample arrives, the gating model produces

$$\widetilde{\boldsymbol{w}}^{d,t,i} = \text{softmax}\big(W_\phi\left(\boldsymbol{x}^{d,t,i}, \boldsymbol{p}^{d,t,i}, \boldsymbol{s}^{d-1}\right)\big) \in \Delta_K. \tag{10}$$

The deployed weights and combined prediction are

$$\boldsymbol{w}^{d,t,i} = (1-\beta_d)\widetilde{\boldsymbol{w}}^{d,t,i} + \beta_d \boldsymbol{m}^{d-1}, \tag{11}$$

$$q^{d,t,i} = \left(\boldsymbol{w}^{d,t,i}\right)^\top \boldsymbol{p}^{d,t,i}. \tag{12}$$

A larger β_d preserves the recent allocation of expert weights. A smaller β_d gives greater influence to the current sample-specific gate.

Adaptive online calibration. Our second algorithmic block adapts OPS into the long-term training process to further leverage the local information. It combines three calibrators: a reset calibrator that reproduces the classic OPS calibrator, and short-memory and long-memory calibrators that retain their states across periods.

For the historical-memory prediction $q^{d,t,i}$, let $z^{d,t,i}$ be its logit after clipping to $[\epsilon, 1-\epsilon]$, where $0 < \epsilon < 1/2$. Calibrator $k \in \{1, 2, 3\}$ produces

$$c_k^{d,t,i} = \text{sigmoid}\Big(a_k^{d,t} z^{d,t,i} + b_k^{d,t}\Big). \tag{13}$$

The reset calibrator initializes $(a_1^{d,1}, b_1^{d,1}) = (1, 0)$ for any period d and uses the classic OPS updates. The persistent calibrators use discounted Online Newton Step updates with short and long memory scales. All updates use only feedback whose delay has elapsed.

The final prediction combines the calibrators:

$$\widehat{p}^{d,t,i} = \sum_{k=1}^{3} \pi_k^{d,t} c_k^{d,t,i}, \qquad \boldsymbol{\pi}^{d,t} \in \Delta_3. \tag{14}$$

The weights are initialized at $\boldsymbol{u} = (1-\lambda, \lambda/2, \lambda/2)^\top$, where $\lambda \in [0, 1]$. For feedback batch r, let L_k^r denote the average log loss of calibrator k on its previously issued predictions. The weights are updated by

$$\widetilde{\pi}_k^{r+1} \propto \pi_k^r \exp(-\eta_m L_k^r), \tag{15}$$

$$\boldsymbol{\pi}^{r+1} = (1-\alpha)\widetilde{\boldsymbol{\pi}}^{r+1} + \alpha \boldsymbol{u}, \tag{16}$$

where $\boldsymbol{\pi}^r$ denotes the weights before update r, $\eta_m > 0$ is the learning rate, and $\alpha \in [0, 1]$ is the average parameter. The averaging toward $\boldsymbol{u}$ maintains weight on alternative memory policies, allowing them to regain influence when conditions change.

Overall algorithm. The complete procedure is in Algorithm 1.

5 Theoretical Guarantees

We first establish properties of the slower timescale updates. We then bound the cumulative prediction loss at the faster timescale relative to a sequence that can switch among the three calibrators. The results do not require a stochastic model of temporal shift.

5.1 Properties of slower timescale updates

Lemma 1 (Historical weights and predictions)**.** *Suppose* $\widetilde{\boldsymbol{w}}^{d,t,i}, \boldsymbol{m}^{d-1} \in \Delta_K$, $\beta_d, \rho \in [0, 1]$, *and* $p_k^{d,t,i} \in [\epsilon, 1-\epsilon]$ *for every expert* k. *Then*

$$\boldsymbol{w}^{d,t,i}, \boldsymbol{m}^d \in \Delta_K, \qquad q^{d,t,i} \in [\epsilon, 1-\epsilon]. \tag{17}$$

Algorithm 1 Two-Timescale Adaptive Memory (TTAM)

Require: Historical horizons $\mathcal{H}$; initial training data; parameters $\rho, \lambda, \alpha, \eta_m, \epsilon, \delta$; OPS and discounted ONS configurations.

Initialize ϕ, ψ, $\boldsymbol{m}^0 = \mathbf{1}/K$, $\boldsymbol{\pi} = \boldsymbol{u} = (1-\lambda, \lambda/2, \lambda/2)^\top$, the three calibrators at $(a_k, b_k) = (1, 0)$, their optimizer states, and empty feedback queues.
for $d = 1, \ldots, D$ **do**
  if $d > 1$ **then**
    Update ϕ, ψ using available feedback and the regularized historical-mixture loss.
    Update $\boldsymbol{m}^{d-1}$ using Eq. (8).
  end if
  Train each expert f_k^d on its historical horizon using available data from periods before d.
  Construct $\boldsymbol{s}^{d-1}$ and compute β_d using Eq. (9). $\boldsymbol{\pi}$, and their pending feedback queues.
  for $t = 1, \ldots, T$ **do**
    Update the calibrator parameters $(a_k^{d,t}, b_k^{d,t})$ and weights $\boldsymbol{\pi}^{d,t}$
    Observe the arriving contexts $\{\boldsymbol{x}^{d,t,i}\}_{i=1}^{n_{d,t}}$.
    for $i = 1, \ldots, n_{d,t}$ **do**
      Compute expert predictions $p_k^{d,t,i} = f_k^d(\boldsymbol{x}^{d,t,i})$.
      Compute $\widetilde{\boldsymbol{w}}^{d,t,i}$ and $\boldsymbol{w}^{d,t,i}$ using Eqs. (10) and (11).
      Compute $q^{d,t,i} = (\boldsymbol{w}^{d,t,i})^\top \boldsymbol{p}^{d,t,i}$ and its clipped logit $z^{d,t,i}$.
      Compute $c_k^{d,t,i}$ using the current calibrator parameters.
      Output $\widehat{p}^{d,t,i} = \sum_{k=1}^3 \pi_k^{d,t} c_k^{d,t,i}$.
    end for
  end for
end for

Moreover, for either click outcome $y \in \{0, 1\}$,

$$\ell(y, q^{d,t,i}) \le \sum_{k=1}^{K} w_k^{d,t,i} \ell(y, p_k^{d,t,i}). \tag{18}$$

Proof. The deployed weights and the updated memory are convex combinations of probability vectors. The combined prediction is therefore a convex combination of the expert predictions. The loss bound follows from convexity of log loss in the predicted probability. □

Thus, the historical weights remain nonnegative and sum to one. The combined prediction has log loss no larger than the weighted average of the historical experts' losses.

Lemma 2 (Effect of persistence)**.** *For every sample (d, t, i),*

$$\begin{aligned} &\left\|\boldsymbol{w}^{d,t,i} - \boldsymbol{m}^{d-1}\right\|_1 \\ &\quad = (1-\beta_d)\left\|\widetilde{\boldsymbol{w}}^{d,t,i} - \boldsymbol{m}^{d-1}\right\|_1. \end{aligned} \tag{19}$$

The memory update satisfies

$$\left\|\boldsymbol{m}^d - \boldsymbol{m}^{d-1}\right\|_1 \le 2\rho. \tag{20}$$

Proof. Subtracting $\boldsymbol{m}^{d-1}$ from the deployed weight vector gives the first identity. For the second bound, the memory update gives

$$\boldsymbol{m}^d - \boldsymbol{m}^{d-1} = \rho\left(\overline{\boldsymbol{w}}^d - \boldsymbol{m}^{d-1}\right). \tag{21}$$

Both vectors on the right belong to Δ_K, so their ℓ_1 distance is at most two. □

A larger β_d keeps the sample-specific weights closer to the stored memory. The parameter ρ separately limits how much that memory changes between consecutive periods.

5.2 Regret bound for faster timescale updates

For each time point (d, t), define the average log losses

$$L_k^{d,t} = \frac{1}{n_{d,t}} \sum_{i=1}^{n_{d,t}} \ell\Big(y^{d,t,i}, c_k^{d,t,i}\Big), \tag{22}$$

$$\widehat{L}^{d,t} = \frac{1}{n_{d,t}} \sum_{i=1}^{n_{d,t}} \ell\big(y^{d,t,i}, \widehat{p}^{d,t,i}\big). \tag{23}$$

These losses are evaluated on the predictions originally issued at that time.

Let $k^{d,t} \in \{1, 2, 3\}$ specify a calibrator for each time point. The comparison may switch between the reset, short-memory, and long-memory calibrators. Its regret is

$$\mathcal{R} = \sum_{d=1}^{D} \sum_{t=1}^{T} \left(\widehat{L}^{d,t} - L_{k^{d,t}}^{d,t}\right). \tag{24}$$

Theorem 1 (Regret of adaptive online calibration)**.** *Let $N = DT$, $\lambda \in (0, 1)$, $\alpha \in (0, 1)$, and $\eta_m > 0$. Initialize the calibration weights at $\boldsymbol{u} = (1-\lambda, \lambda/2, \lambda/2)^\top$, and define*

$$u_{\min} = \min\left\{1 - \lambda, \frac{\lambda}{2}\right\}. \tag{25}$$

Assume $0 \le L_k^{d,t} \le L_{\max}$. Each time point's loss vector is used once, in chronological order, after its feedback becomes available. At any prediction time, at most $\lceil \delta \rceil$ earlier loss vectors remain unavailable.

For any comparison sequence that switches calibrators at most S times, where $0 \le S \le N-1$, set

$$C_S = (S+1)\log\frac{1}{u_{\min}} + S\log\frac{1}{\alpha} + (N-1-S)\log\frac{1}{1-\alpha}. \tag{26}$$

Then

$$\mathcal{R} \le \frac{C_S}{\eta_m} + \frac{\eta_m L_{\max}^2 N(1+2\lceil\delta\rceil)}{8} + \alpha L_{\max} N\lceil\delta\rceil. \tag{27}$$

Proof. First consider the weights obtained by processing each loss vector before the next prediction. The fixed-share update assigns probability at least

$$u_{\min}^{S+1}\alpha^S(1-\alpha)^{N-1-S} \tag{28}$$

to every comparison sequence with at most S switches. The exponential-weights argument and Hoeffding's lemma therefore bound its regret by $C_S/\eta_m + \eta_m L_{\max}^2 N/8$.

For delayed feedback, compare the deployed weights with those obtained after processing all earlier loss vectors. One exponential update changes the weights by at most $\eta_m L_{\max}/2$ in ℓ_1 distance. This follows from Hoeffding's lemma and Pinsker's inequality. The subsequent sharing step adds at most 2α to this change.

The two weight vectors differ by at most $\lceil\delta\rceil$ updates. Since losses lie in $[0, L_{\max}]$, the resulting increase in weighted loss at each time point is at most

$$\lceil\delta\rceil\left(\frac{\eta_m L_{\max}^2}{4} + \alpha L_{\max}\right). \tag{29}$$

Summing over the N time points gives the stated bound. Finally, convexity of log loss gives $\widehat{L}^{d,t} \le \sum_{k=1}^{3}\pi_k^{d,t}L_k^{d,t}$, so the same bound applies to the issued predictions. □

The theorem compares TTAM with a calibrator sequence chosen in hindsight. It therefore allows different calibration memories to be useful at different times. The comparison uses the same historical predictions and the same three calibrators as TTAM.

For a fixed feedback delay and fixed $u_{\min} > 0$, choose $\alpha = (S+1)/(N+1)$ and

$$\eta_m = \sqrt{\frac{8C_S}{L_{\max}^2 N(1+2\lceil\delta\rceil)}}. \tag{30}$$

If $S = o(N)$, the regret bound is $o(N)$. Thus, the average excess loss vanishes when the number of calibrator changes grows more slowly than the number of prediction time points.

6 Numerical experiments

We evaluate TTAM's two adaptation layers through complementary studies. Rolling-origin experiments on Criteo and Avazu examine the faster timescale updates. Controlled temporal-shift experiments examine the slower timescale updates and the role of learned persistence. We also evaluate whether the resulting prediction improvements increase clicks under an unchanged bidding policy.

6.1 Rolling-origin evaluation

Datasets and prediction models. We use the full Criteo Attribution and Avazu datasets. Criteo contains 16,468,027 observations across 31 days. Avazu contains 40,428,967 observations across 10 days. Observations are ordered chronologically, without random splitting across days. For Avazu, timestamps determine the chronological order, while identifiers and timestamps are excluded from the prediction features.

The historical predictors use logistic regression with ℓ_2 regularization and hashed categorical features. They share the same features and model class and differ in their training histories. For each prediction day, the predictors are trained using earlier days.

Evaluation protocol. We evaluate Criteo on days 16–30 and Avazu on days 5–9, giving 15 and 5 rolling origins, respectively. These dataset indices are zero-based. At each origin, the three most recent eligible days form the inner validation window. We select one common configuration by averaging validation log loss across seeds $0, 1, 2$. The selected configuration is replayed over the preceding prediction stream to initialize its online state. We then evaluate the next day and advance the origin.

The historical-mixture settings and calibration-memory settings are selected separately at each origin. The calibrator optimizer settings are held fixed at values selected in the earlier development experiments. The AP-OPS selection grid includes $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$. Thus, the selection procedure can recover OPS through $\lambda = 0$.

Click labels become available after a 30-minute delay. Calibration updates occur every 15 minutes on Criteo and every hour on Avazu. During evaluation, newly available labels may update subsequent predictions. Persistent calibration states continue across day boundaries.

Comparison methods. To isolate the faster timescale updates, every calibration method receives the same causal mixture of predictors trained on the previous 3 days, the previous 7 days, and expanding

history. This common input is held fixed across calibration comparisons.

We compare five methods: OPS, which resets its calibration parameters each day; Reset-ONS, which uses discounted Online Newton Step updates and resets each day; Persistent-ONS, which carries the same type of calibrator across days; AP-OPS, which combines reset, short-memory, and long-memory calibrators; and a no-slope version of AP-OPS with $a = 1$.

The persistent experts in AP-OPS use memory half-lives of 4 and 16 hours. Reset-ONS and Persistent-ONS share a half-life selected through inner validation. The no-slope variant uses the configuration selected for full AP-OPS.

Evaluation measures. The primary measure is log loss. We first average results across seeds within each evaluation day and then average equally across days. Differences between methods are paired by day. We report 95% paired bootstrap confidence intervals over evaluation days and the number of origins at which each method improves on OPS.

6.2 Rolling-origin prediction performance

Table 1 reports the results. AP-OPS achieves the lowest mean log loss among the compared calibration methods on both datasets. Relative to OPS, it reduces mean log loss by 0.000172 on Criteo and 0.000136 on Avazu. It improves on OPS at all 15 Criteo origins and all 5 Avazu origins. Both paired confidence intervals lie below zero.

The component comparisons show that the source of improvement differs between datasets. On Criteo, Reset-ONS accounts for most of the gain. Persistence and aggregation provide smaller additional reductions in mean loss. On Avazu, Reset-ONS does not improve the mean loss of OPS. Persistent-ONS has a lower mean loss, although its confidence interval includes zero. Full AP-OPS produces the largest reduction and improves performance at every origin.

Fixing the slope at $a = 1$ removes most of the improvement on Criteo and increases loss on Avazu. These results support updating both the slope and intercept.

The selected λ is positive at every origin. Criteo selects 0.75 for the first twelve origins and 0.25 for the remaining three. Avazu selects values from $\{0.25, 0.5\}$. Thus, the selected configurations use the additional calibration memories. Separate implementation checks verify that $\lambda = 0$ reproduces OPS prediction by prediction.

6.3 Historical adaptation under controlled shift

Experimental setup. We generate logistic CTR streams with 120 temporal blocks and 3,000 observations per block. The study covers fourteen regimes. These include stationary data, abrupt and gradual changes, recurring patterns, local changes, opposing local changes, mixed changes, and opposing recurring patterns. We also include recurrence periods of 9, 11, 17, and 21 blocks, irregular recurrence, and a stream containing several consecutive types of shift. The additional recurrence periods differ from the historical horizons used by the experts.

We use five development seeds to select the AMG-TP configuration and twelve disjoint seeds for confirmation. The final 30% of each stream is used for evaluation. The comprehensive fixed-persistence comparison summarizes all seventeen seeds.

AMG-TP uses historical horizons $\{1, 3, 7, 14, \text{expanding}\}$. The historical predictors share the same features and backbone. Comparison methods include expanding-history training, adaptive rolling-window selection (ARW), AdaMoE, Fixed Share, Learn-α, context gates, and a three-gate ensemble. We also compare fixed persistence levels and multiscale gates with fixed temporal-smoothing strengths.

In the experimental implementation, the preceding period's weights are recomputed after updating the gating and persistence models. The recomputation keeps that period's original state and memory inputs fixed. The average of these weights is then used to update the memory.

Robustness across shift regimes. We first examine whether learned persistence remains effective when different regimes favor different memory durations. For each regime, we evaluate fixed values $\beta \in \{0, 0.05, \ldots, 1\}$. Excess log loss is measured relative to the best fixed value for that regime, selected in hindsight. We report the average excess across regimes and the largest regime-level excess.

Table 2 shows that AMG-TP has the lowest mean and worst-regime excess among the tested methods. Its mean excess is 0.0019, compared with 0.0047 for the best single fixed persistence across all regimes. Its worst-regime excess is 0.0166, compared with 0.0254 for that fixed choice.

Figure 2 explains the benefit of adaptation. Stationary, abrupt, gradual, and local regimes favor little persistence. Recurring and heterogeneous streams generally favor greater persistence. The best fixed value ranges from 0 to 0.9, so one constant does not perform equally well across these conditions.

Dataset	Method	Log loss	Difference	95% confidence interval	Origins won
Criteo	OPS	0.608288	—	—	—
	Reset-ONS	0.608120	-0.000167	$[-0.000218, -0.000125]$	15/15
	Persistent-ONS	0.608118	-0.000170	$[-0.000220, -0.000128]$	15/15
	AP-OPS	**0.608116**	**-0.000172**	$[-0.000220, -0.000131]$	**15/15**
	No-slope	0.608285	-0.000002	$[-0.000053, +0.000047]$	6/15
Avazu	OPS	0.401636	—	—	—
	Reset-ONS	0.401655	+0.000019	$[-0.000090, +0.000133]$	3/5
	Persistent-ONS	0.401587	-0.000049	$[-0.000143, +0.000057]$	3/5
	AP-OPS	**0.401500**	**-0.000136**	$[-0.000181, -0.000098]$	**5/5**
	No-slope	0.401865	+0.000229	$[+0.000081, +0.000397]$	0/5

Table 1: Rolling-origin evaluation on Criteo and Avazu. Differences are measured relative to OPS; negative values indicate improvement. Results average seeds within each day and then weight evaluation days equally. Confidence intervals use paired resampling of days.

Method	Mean excess	Worst excess
AMG-TP	**0.0019**	**0.0166**
Smoothed gate (0.1)	0.0022	0.0195
Three-gate ensemble	0.0022	0.0246
Learn-α	0.0029	0.0273
Fixed Share	0.0040	0.0275
Fixed $\beta = 0.8$	0.0047	0.0254
ARW	0.0059	0.0305
Smoothed gate (0.001)	0.0080	0.0468
AdaMoE	0.0175	0.0428
Two-expert context gate	0.0191	0.0608
Expanding history	0.0519	0.2037

Table 2: Excess log loss relative to the best fixed persistence within each regime. The summary covers fourteen regimes and seventeen seeds. The smoothed-gate rows use the indicated regularization strengths. Lower values are better.

Performance within individual regimes. Table 3 presents representative results on the twelve confirmation seeds. AMG-TP performs well under recurring and local changes. Its recurring-pattern improvement also holds at a period of 17 blocks, which does not match an expert horizon. Learn-α performs better on irregular recurrence and the heterogeneous stream. These results show where context-dependent historical weighting is most useful.

The learned persistence also changes with the observed conditions. In the heterogeneous stream, its mean value is approximately 0.36 during abrupt change, 0.27 during local change, and 0.82 during recurring patterns. Figure 3 shows these changes over time. The method learns this behavior from past observations without receiving the regime labels.

6.4 Downstream bidding performance

We pass the frozen CTR predictions to the same second-price auction and pacing policy. Only the prediction method changes. The evaluation uses eight seeds and compares realized clicks at the same 25% spend target.

Table 4 reports the change relative to ARW. AMG-TP increases clicks by 1.53% under recurring drift and 2.09% under local drift, with improvements on all eight seeds. The gains are smaller under abrupt and opposing-local changes. The mixed regime produces a 1.02% reduction. Thus, the prediction improvements under recurring and local drift translate into greater bidding value without changing the bidding policy.

References

[1] S. Bennett and J. Clarkson. Time series prediction under distribution shift using differentiable forgetting. *arXiv preprint arXiv:2207.11486*, 2022.

[2] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[3] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46 (4):1–37, 2014.

[5] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

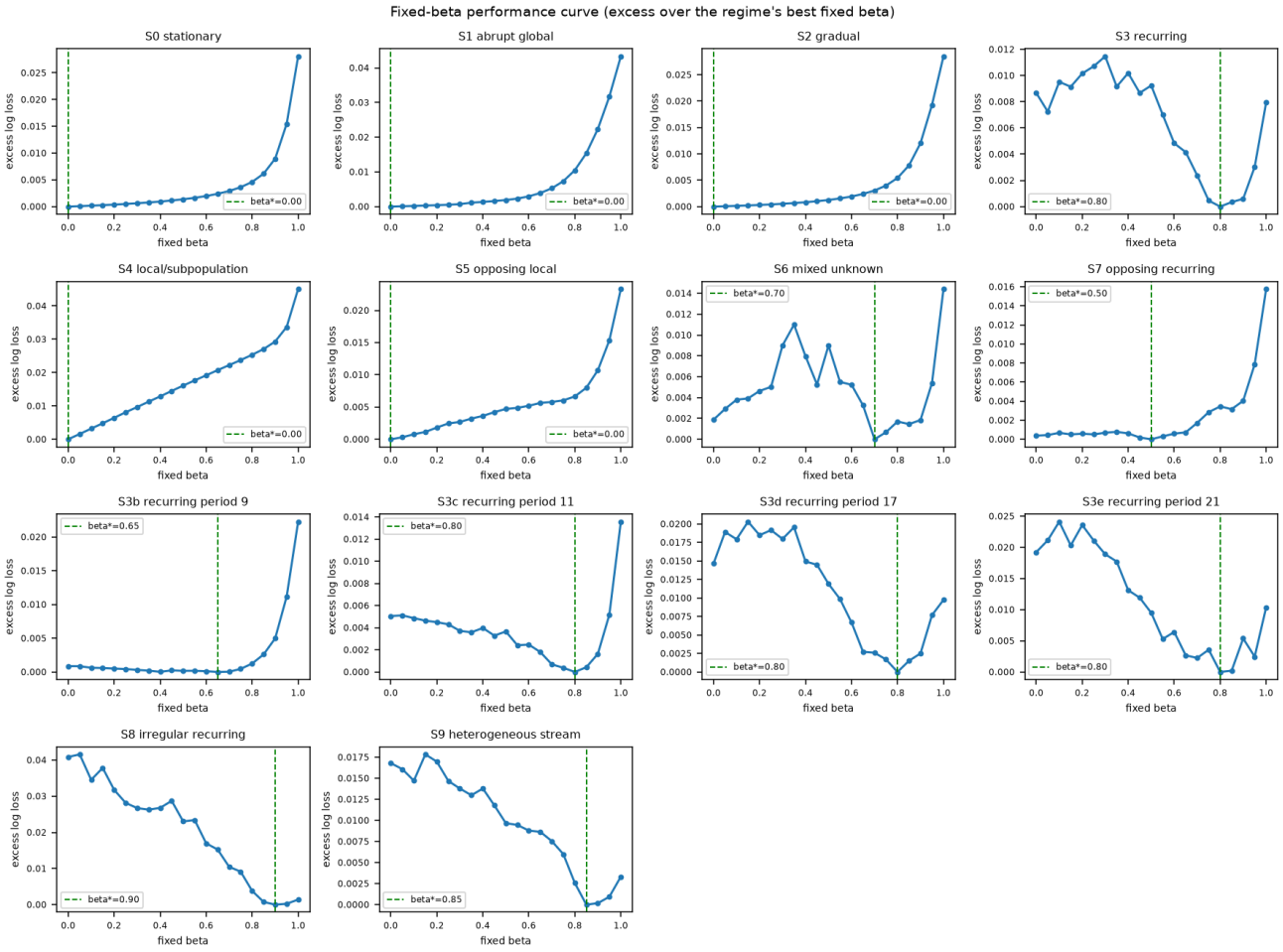

Figure 2: Effect of fixed persistence across temporal-shift regimes. Each panel reports excess log loss relative to the best fixed β for that regime. Different regimes favor different persistence levels.

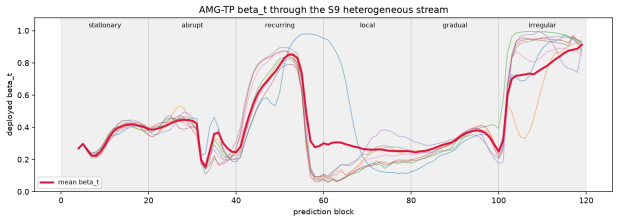

Figure 3: Learned persistence in a stream containing several types of temporal shift. Thin curves show individual seeds, and the thick curve shows their mean.

[6] C. Gupta and A. Ramdas. Online platt scaling with calibeating. In *International Conference on Machine Learning*, pages 12182–12204. PMLR, 2023.

[7] C. Gupta and A. Ramdas. Online platt scaling with calibeating. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12182–12204. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/gupta23c.html`.

[8] E. Han, C. Huang, and K. Wang. Model assessment and selection under temporal distribution shift. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17374–17392. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/han24b.html`.

[9] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

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

[11] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Regime	Expanding	ARW	Fixed Share	Learn-α	AMG-TP
Recurring, period 14	0.4357	0.4368	0.4328	0.4318	**0.4298**
Recurring, period 17	0.4346	0.4398	0.4289	0.4283	**0.4266**
Local shift	0.5190	0.4885	0.4864	0.4862	**0.4612**
Irregular recurrence	0.4914	0.4523	0.4500	**0.4459**	0.4608
Heterogeneous stream	0.4527	0.4470	0.4400	**0.4368**	0.4430

Table 3: Mean evaluation log loss on representative controlled regimes, using twelve confirmation seeds. Lower values are better.

Regime	Click change	Seeds won	Paired p
Recurring	+1.53%	8/8	0.008
Local	+2.09%	8/8	0.008
Abrupt	+0.17%	7/8	0.016
Opposing-local	+0.12%	5/8	0.383
Mixed	−1.02%	0/8	0.008

Table 4: Change in realized clicks relative to ARW at matched spend. All methods use the same auction and pacing policy.

[12] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[13] P. Joulani, A. Gyorgy, and C. Szepesvári. Online learning under delayed feedback. In *International conference on machine learning*, pages 1453–1461. PMLR, 2013.

[14] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[15] C. Liu, Y. Li, F. Teng, X. Zhao, C. Peng, Z. Lin, J. Hu, and J. Shao. On the adaptation to concept drift for ctr prediction. *arXiv preprint arXiv:2204.05101*, 2022.

[16] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[17] C. Monteleoni and T. Jaakkola. Online learning of non-stationary sequences. *Advances in Neural Information Processing Systems*, 16, 2003.

[18] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.