# WORD-CTR: WRITE ONCE, READ DEEP FOR EFFICIENT LONG-SEQUENCE CLICK-THROUGH RATE PREDICTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Click-through rate (CTR) prediction often scores many candidate items for the same user, making repeated access to long behavior histories costly. Reusing a candidate-independent user state can reduce this repeated work, provided that different candidates can still obtain distinct information from it. We present **WordCTR** (**Write Once, Read Deep**), which represents the retained behavior history as a fixed-size, multi-head associative memory. Its writer fits a linear key-to-value map in each head using content- and recency-weighted ridge regression. Stacked reader blocks combine non-sequential user, candidate, and context fields and make successive queries to the same memory without revisiting the behavior sequence. For fixed model dimensions, candidate-side reading is independent of history length once the memory is available, while memory construction remains history-dependent. We evaluate WordCTR against 17 baselines on three public datasets and an InHouse dataset that includes a behavior stream of up to 1024 events. WordCTR obtains the best reported AUC, GAUC, and NE on the public datasets, and the best AUC and NE on InHouse. We also examine cached and uncached inference across history lengths and candidate counts. In the measured single-request GPU workload, candidate-side read-and-score time varies little across tested history lengths, while caching reduces total latency with a larger benefit at higher tested candidate counts.

## 1 INTRODUCTION

Click-through rate (CTR) prediction often scores multiple candidate items for the same user. Historical interactions provide signals about user interests, but directly matching every candidate against every retained behavior incurs candidate-dependent work that grows with history length. Let $N$ be the number of retained behaviors and $K$ the number of candidates, as in Section 3. Full candidate-to-history attention has $O(KN)$ interaction cost when feature dimensions are held fixed. This motivates a reusable user representation that can still produce candidate-dependent readouts.

CTR models address long histories in different ways. Target-aware attention, as in DIN (Zhou et al., 2018), accesses behaviors for each target; retrieval methods such as SIM and TWIN (Pi et al., 2020; Chang et al., 2023) first select relevant behaviors. Other systems already move part of the history processing away from each ranking request: MIMN uses a User Interest Center to maintain user interest states (Pi et al., 2019), SDIM separates behavior hashing into a behavior sequence encoder (Cao et al., 2022), and TWIN V2 compresses histories through offline clustering (Si et al., 2024). Figure 1 illustrates three computation paths: full-sequence modeling for each candidate, candidate-dependent retrieval followed by fine-grained modeling, and writing a shared memory before candidate-conditioned reading. Cross-candidate reuse is therefore an established design choice. The question we study is how a fixed-size, candidate-independent memory can be constructed from the retained history and queried repeatedly by a CTR predictor.

We propose **WordCTR** (**Write Once, Read Deep**). Its writer maps the retained behavior sequence into a fixed-size, multi-head associative memory. Each head fits a linear key-to-value map by ridge regression, with write weights determined by both behavior content and recency. Its reader takes non-sequential user, candidate, and context fields, performs field interaction, and repeatedly queries



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

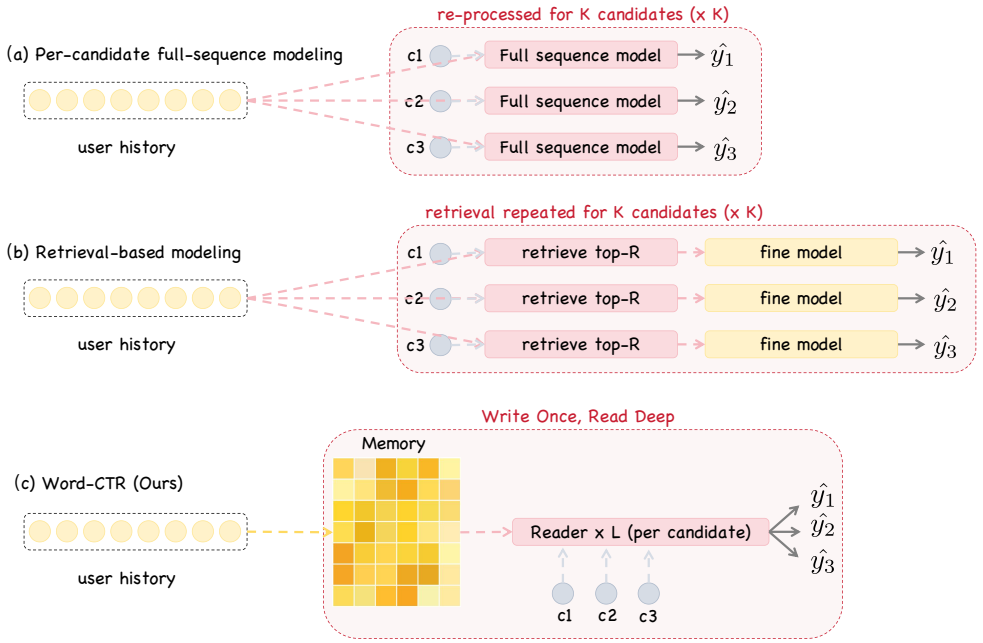


Figure 1: Comparison of scoring $K$ candidates against one retained user history. (a) A full-sequence model is applied to each candidate. (b) Candidate-dependent retrieval selects $R$ behaviors before fine-grained modeling. (c) WordCTR writes a fixed-size memory once for a given user state and applies $L$ reader blocks per candidate to query the shared memory.

the same memory through stacked blocks. The memory is computed independently of candidate items and can be reused while the user state is unchanged. This design differs from MIMN's recurrently updated memory slots and SDIM's hash-based interest aggregation, although these methods also support candidate-independent preprocessing.

For fixed model dimensions, constructing WordCTR's memory takes work that grows with $N$, while its cached size and candidate-side read cost do not depend on $N$. This is a property of the model's computation after the memory is available, rather than a claim that writing or maintaining the user state has no cost. The reader does not access the original behavior sequence or a separate recent-behavior stream. Our design combines a weighted regression memory with repeated, field-conditioned access; ridge-based associative memories and cross-candidate reuse each have prior art (Wang et al., 2026; Pi et al., 2019; Cao et al., 2022).

We evaluate predictive accuracy against 17 baselines on Amazon Electronics, KuaiVideo, TaobaoAds, and an InHouse dataset containing a behavior stream of up to 1024 events. WordCTR has the best reported AUC, GAUC, and NE among the evaluated methods on all three public datasets. On InHouse, it has the best AUC and NE. We separately measure inference latency for cached and uncached WordCTR and one MIMN implementation under a single-request GPU workload. The measurements show lower total latency for cached WordCTR than for the evaluated MIMN implementation across the tested history lengths and characterize how caching affects latency as the candidate count changes. These observations concern the tested implementations and workload, rather than every baseline or an online deployment.

Our main contributions are summarized as follows:

- We develop WordCTR's combination of candidate-independent, content- and recency-weighted multi-head ridge memory with stacked, field-conditioned reads over the same memory.
- We analyze the separation between history-dependent writing and history-independent candidate-side reading when the fixed-size memory is cached, and report measured user-side and candidate-side latency under the specified GPU workload.

- We compare predictive accuracy with 17 baselines on three public datasets and InHouse, and report latency against the measured MIMN implementation. WordCTR leads all three reported metrics on the public datasets and AUC and NE on InHouse.

## 2 Related Work

### 2.1 User Behavior Modeling for CTR Prediction

DIN (Zhou et al., 2018) uses target-aware attention to weight historical behaviors, and DIEN (Zhou et al., 2019) models the evolution of latent interests before target-aware prediction. Direct target-to-history interaction can require work that grows with the number of retained behaviors for each candidate. WordCTR retains candidate-conditioned access, but applies it to a shared fixed-size memory rather than the raw sequence.

### 2.2 Efficient Long-Sequence Modeling

Sequence compression (Dai et al., 2020; Jaegle et al., 2021; Bolya et al., 2023), sparse attention (Beltagy et al., 2020; Zaheer et al., 2020; Roy et al., 2021), and linear or approximate attention (Wang et al., 2020; Katharopoulos et al., 2020; Choromanski et al., 2021) reduce different costs of long-sequence processing. They do not, by themselves, specify how a CTR system should share user-state computation across candidates.

In long-sequence CTR prediction, UBR4CTR and SIM (Qin et al., 2020; Pi et al., 2020) retrieve candidate-relevant behaviors for subsequent modeling. ETA, ETA-Net, and TWIN (Chen et al., 2021; 2022; Chang et al., 2023) also address efficient target-to-history matching. TWIN V2 (Si et al., 2024) compresses histories with offline hierarchical clustering before online retrieval. SDIM (Cao et al., 2022) uses candidate and behavior hash signatures to aggregate matching behaviors, and separates behavior-sequence encoding from request-time scoring. These works show that offline preprocessing and reuse across candidates are established strategies.

Recent CTR models also integrate behavior-sequence modeling with non-sequential feature interaction. InterFormer (Zeng et al., 2025) learns interactions across heterogeneous feature modes, while OneTrans (Zhang et al., 2026) uses a shared Transformer and supports cross-request key–value (KV) caching. TokenFormer (Zhou et al., 2026) and EST (Liu et al., 2026) pursue related unified modeling with different attention designs. WordCTR instead fits a candidate-independent, multi-head associative memory over retained behaviors and applies stacked, field-conditioned reads to this memory without retrieving raw behaviors during scoring.

### 2.3 Memory and Associative Memory

MIMN (Pi et al., 2019) updates bounded external memory slots as behaviors arrive and uses a User Interest Center to maintain user states outside the prediction request. WordCTR does not use MIMN's recurrent slot updates: it fits multi-head linear maps to a retained history and makes candidate-conditioned queries to those maps from successive reader blocks. Associative retrieval through fast weights and linear attention predates WordCTR (Katharopoulos et al., 2020; Schlag et al., 2021). Test-time regression further formalizes weighted regression as an associative-memory construction (Wang et al., 2026). Our contribution is the specified CTR architecture and its evaluated behavior, rather than the ridge objective or candidate-independent caching in isolation.

## 3 Problem Formulation

For a user $u$, let $\mathcal{S}_u = (e_1, \ldots, e_N)$ denote a chronological behavior history, ordered from oldest to newest. A ranking example also contains user and context fields $g_u$, a candidate item $c$, and a binary click label $y \in \{0, 1\}$. A CTR model predicts

$$\hat{y}_{u,c} = \sigma(f_\theta(g_u, c, \mathcal{S}_u)), \tag{1}$$

where $f_\theta$ outputs a logit and $\sigma$ is the logistic sigmoid.

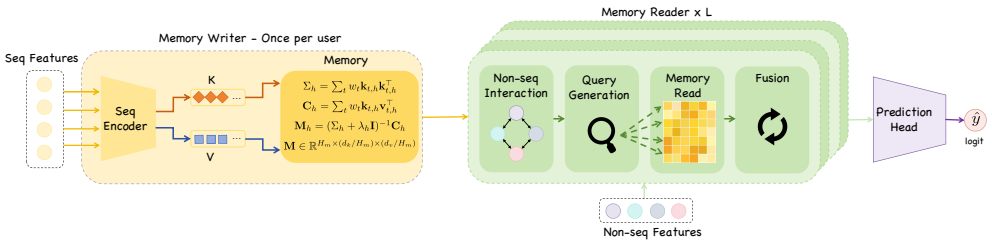


Figure 2: Overview of **WordCTR**. The Seq Encoder prepares retained behavior streams for candidate-independent memory writing; stacked reader blocks query the resulting fixed-size, multi-head memory using non-sequential features.

The serving system commonly scores many candidates $\mathcal{C}_u = \{c_1, \ldots, c_K\}$ for the same user state. Full target attention incurs candidate-dependent work of order $O(KN)$, ignoring feature dimensions, because every candidate attends to all $N$ behaviors. Our objective is to decompose the predictor as

$$\mathcal{M}_u = \mathrm{Write}_\theta(\mathcal{S}_u), \qquad \hat{y}_{u,c} = \sigma(\mathrm{Read}_\theta(g_u, c; \mathcal{M}_u)), \tag{2}$$

such that (i) $\mathcal{M}_u$ is independent of the candidate and can be reused, (ii) its size is independent of $N$ for fixed model hyperparameters, and (iii) $\mathrm{Read}$ remains target-aware. The complete predictor is trained end to end with binary cross entropy.

# 4 Method

## 4.1 Overview

Figure 2 shows the stages of WordCTR. The *Seq Encoder* transforms the retained behavior streams into key- and value-path representations. The candidate-independent *Memory Writer* uses these representations to construct a fixed-size, multi-head associative memory once for a given user state. In parallel, a feature tokenizer maps non-sequential fields to $F$ global field tokens. A stack of $L$ *Memory Reader* blocks updates these tokens while querying the same read-only memory; a prediction head maps the final field tokens to a click logit.

## 4.2 Sequence Encoder

The Seq Encoder in Figure 2 prepares the retained behavior history for candidate-independent memory writing. Sequence features belonging to the same semantic behavior group are summed position-wise, while distinct groups remain separate. The resulting streams are concatenated for memory construction; a learned stream embedding identifies the source stream. Let $z_t$ denote the resulting token at position $t$, and let $a_t \geq 0$ denote its age within its semantic behavior stream, with the newest valid behavior assigned age zero.

A causal depthwise mixer adds local order information to the key path. Let $m_t \in \{0, 1\}$ indicate whether position $t$ is valid, let $\kappa$ be the convolution kernel size, and let $\omega_j$ be its channel-wise weights. With zero padding on the left, the mixer and age-augmented value path are

$$\bar{z}_t = m_t \,\mathrm{RMSNorm}(z_t), \tag{3}$$

$$h_t = \sum_{j=0}^{\kappa-1} \omega_j \odot \bar{z}_{t-j}, \tag{4}$$

$$z_t^{\mathrm{key}} = m_t \left[ z_t + W_{\mathrm{mix}} \,\mathrm{SiLU}(h_t) \right], \tag{5}$$

$$b_t = \min(\lfloor \log_2(a_t + 1) \rfloor, B_{\mathrm{age}} - 1), \tag{6}$$

$$z_t^{\mathrm{val}} = z_t^{\mathrm{key}} + E_{\mathrm{age}}[b_t]. \tag{7}$$

Here $\bar{z}_s = 0$ for positions before the start of the sequence. The depthwise convolution processes each channel separately, and its output at $t$ depends only on the current token and the preceding $\kappa - 1$

tokens, never on later tokens. A projection and residual connection combine this local context with $z_t$; the mask zeros out invalid output positions. We use $\kappa = 4$.

Here $B_{\text{age}}$ is the number of age buckets and $E_{\text{age}}$ is their learned embedding table. Logarithmic bucketing gives recent positions finer resolution: with the default $B_{\text{age}} = 9$, ages 0 and 1 receive different embeddings, whereas ages 200 and 201 share one. This follows the design intuition that adjacent recent behaviors may differ more in relevance than adjacent distant behaviors. Clipping bounds the table at $B_{\text{age}}$ embeddings regardless of sequence length. Adding $E_{\text{age}}[b_t]$ to the value path lets the memory encode coarse recency together with behavior content; the separate decay factor in Equation 11 controls how strongly each behavior is written. The resulting $z_t^{\text{key}}$ and $z_t^{\text{val}}$ are passed to the memory writer.

### 4.3 Memory writer: closed-form associative user memory

**Multi-head key and value spaces.** We project each behavior into a key and value,

$$k_t = W_k \,\mathrm{RMSNorm}(z_t^{\text{key}}), \qquad v_t = W_v \,\mathrm{RMSNorm}(z_t^{\text{val}}). \tag{8}$$

The vectors are divided into $H$ memory heads. We write $k_{t,h} \in \mathbb{R}^{d_k/H}$ and $v_{t,h} \in \mathbb{R}^{d_v/H}$ for head $h$, and normalize each $k_{t,h}$ to unit norm.

**Content- and recency-weighted writes.** Each behavior receives a scalar write weight determined jointly by its content and recency. The content gate is

$$\beta_t = \sigma\Big(w_\beta^\top z_t^{\text{key}} + b_\beta\Big), \tag{9}$$

where $\beta_t \in (0, 1)$ controls how strongly behavior $t$ is written according to its representation.

We parameterize the temporal decay rate as

$$r = \mathrm{softplus}(\theta_r), \tag{10}$$

where $\theta_r \in \mathbb{R}$ is an unconstrained trainable scalar. The softplus parameterization guarantees $r > 0$; $\theta_r$ is initialized such that $r = 10^{-3}$ at the beginning of training. A larger $r$ emphasizes recent behaviors more strongly, whereas a smaller $r$ preserves information over a longer time range.

The final write weight is

$$w_t = \beta_t \exp(-r a_t)\, \mathbb{I}[t \text{ is valid}]. \tag{11}$$

Thus, informative and recent behaviors receive larger weights, while padded positions contribute exactly zero. The decay rate and write weight are shared across memory heads; each head nevertheless learns its own key and value subspaces and ridge coefficient.

**Weighted ridge memory.** For each memory head, we fit one linear Key-to-Value map by solving

$$A_h^\star = \arg\min_{A \in \mathbb{R}^{d_{k,h} \times d_{v,h}}} \sum_{t=1}^{N} w_t \left\| v_{t,h} - A^\top k_{t,h} \right\|_2^2 + \lambda_h \|A\|_F^2, \tag{12}$$

where $d_{k,h} = d_k/H$, $d_{v,h} = d_v/H$, and $\lambda_h > 0$ is learned with a positive floor. Define

$$\Sigma_h = \sum_t w_t k_{t,h} k_{t,h}^\top, \qquad C_h = \sum_t w_t k_{t,h} v_{t,h}^\top. \tag{13}$$

The unique minimizer is

$$\mathcal{M}_h \equiv A_h^\star = (\Sigma_h + \lambda_h I)^{-1} C_h. \tag{14}$$

We form $\Sigma_h$ and $C_h$ directly and compute Equation 14 in FP32 with a batched Cholesky solve; no explicit matrix inverse is materialized. The positive ridge term guarantees a well-defined system even for short or padded histories. The solve is performed once during writing, producing the fixed-size memory $\mathcal{M}_u = \{\mathcal{M}_h\}_{h=1}^H$; subsequent reads require only matrix–vector products.

### 4.4 Memory reader: iterative field-conditioned access

The tokenizer produces global tokens $X^{(0)} \in \mathbb{R}^{F \times d}$ for user, context, candidate, dense, and multi-hot fields. The same memory $\mathcal{M}_u = \{\mathcal{M}_h\}_{h=1}^H$ is shared by all $L$ reader blocks and is never rewritten by the reader. Let $X^{(\ell-1)} \in \mathbb{R}^{F \times d}$ denote global field tokens entering layer $\ell$.

**Field interaction and shared-space queries.** The reader first applies self-attention over the small set of global fields:

$$\widehat{X}^{(\ell)} = X^{(\ell-1)} + \mathrm{MHA}_\ell(\mathrm{RMSNorm}(X^{(\ell-1)})). \tag{15}$$

For each field $f$ and memory head $h$, the layer constructs a normalized query

$$q_{f,h}^{(\ell)} = \mathrm{Norm}\left(\left[W_k \bar{x}_f^{(\ell)} + W_{q,\ell} \bar{x}_f^{(\ell)}\right]_h\right), \quad \bar{x}_f^{(\ell)} = \mathrm{RMSNorm}(\widehat{x}_f^{(\ell)}). \tag{16}$$

The base projection $W_k$ is shared with the writer, while $W_{q,\ell}$ is layer-specific and zero-initialized. Thus queries initially inhabit the write-key geometry and learn task-dependent deviations.

**Multi-head memory read.** Each normalized field query retrieves one value from every memory head:

$$o_{f,h}^{(\ell)} = \mathcal{M}_h^\top q_{f,h}^{(\ell)}, \qquad o_f^{(\ell)} = \mathrm{Concat}_{h=1}^{H} o_{f,h}^{(\ell)}. \tag{17}$$

Because $\mathcal{M}_u$ is reused unchanged, additional reader depth refines the query and the interpretation of retrieved evidence without revisiting the behavior sequence.

**Gated fusion and iterative refinement.** For reader output $o_f^{(\ell)}$, let $p_f^{(\ell)} = W_{o,\ell} o_f^{(\ell)}$. The projected readout is paired with an elementwise field–read match and fused through a learned gate:

$$u_f^{(\ell)} = W_{\mathrm{fuse},\ell}[p_f^{(\ell)}; \bar{x}_f^{(\ell)} \odot p_f^{(\ell)}], \tag{18}$$

$$g_f^{(\ell)} = \sigma\left(W_{g,\ell}[\bar{x}_f^{(\ell)}; p_f^{(\ell)}; \bar{x}_f^{(\ell)} \odot p_f^{(\ell)}]\right), \tag{19}$$

$$H^{(\ell)} = \widehat{X}^{(\ell)} + g^{(\ell)} \odot u^{(\ell)}, \tag{20}$$

$$X^{(\ell)} = H^{(\ell)} + \mathrm{FFN}_\ell(\mathrm{RMSNorm}(H^{(\ell)})). \tag{21}$$

Here $u^{(\ell)}$ and $g^{(\ell)}$ collect the field-wise terms. The updated fields form the next block's queries; after $L$ blocks, they are flattened and passed to an MLP prediction head.

### 4.5 Prediction and learning objective

After the final reader block, all global field tokens are flattened and passed through an MLP to produce the click logit $s_i$. The implementation uses one forward path for both training and inference and optimizes only the standard binary cross-entropy objective

$$\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{B}\sum_{i=1}^{B}\left[y_i \log \sigma(s_i) + (1-y_i)\log(1-\sigma(s_i))\right]. \tag{22}$$

### 4.6 Complexity and cached serving

The write stage accumulates the weighted ridge statistics in one pass over the behavior sequence and therefore scales linearly with $N$. It then solves $H$ fixed-size systems, one per memory head, independently of the candidate. Once the multi-head memory has been constructed, its size and the candidate-side read cost are independent of $N$; they depend only on the model dimensions, number of memory heads, number of fields, and reader depth. We therefore report user-side writing, cache storage, and candidate-side reading separately. Exact costs and comparisons are provided in Appendix B.

The comparison with full attention, retrieve-then-attend systems, and fixed user vectors is summarized in Appendix B. In particular, the claim is candidate-side independence from $N$ after a cache is available, not zero-cost history processing.

## 5 Experiments

We evaluate predictive performance, inference efficiency, and sensitivity to reader depth and temporal encoding.

Table 1: Statistics of the three public CTR benchmarks and the InHouse dataset. Seq./Non-Seq. denotes the numbers of sequential and non-sequential feature fields. The two InHouse sequence lengths correspond to two types of behavior streams.

| Dataset | #Interactions | #Users | #Items | #Fields (Seq./Non-Seq.) | Max Seq. Len. |
|---|---|---|---|---|---|
| Amazon Electronics | 2,993,570 | 63,001 | 192,403 | 2/4 | 100 |
| TaobaoAds | 25,029,426 | 1,061,768 | 827,006 | 3/19 | 50 |
| KuaiVideo | 13,661,383 | 10,000 | 3,240,282 | 4/5 | 100 |
| InHouse | 6,115,949 | 600,000 | 2,081,279 | 8/33 | 50/1024 |

### 5.1 Experimental setup

**Datasets.** We evaluate on three widely used public CTR datasets: Amazon Electronics, TaobaoAds, and KuaiVideo. Amazon Electronics is an e-commerce product-review benchmark containing timestamped user–product interactions (He & McAuley, 2016); its configuration contains two sequential and four non-sequential feature fields. TaobaoAds is an advertising CTR benchmark with heterogeneous user, ad, context, and behavior features, comprising three sequential and 19 non-sequential fields. KuaiVideo is a micro-video recommendation benchmark (Li et al., 2019) with four sequential and five non-sequential fields. The maximum retained behavior-sequence lengths are 100, 50, and 100 for Amazon Electronics, TaobaoAds, and KuaiVideo, respectively. We additionally construct an InHouse dataset sampled from nine days of production logs (2026/01/21–2026/01/29) at a leading e-commerce platform. It includes two types of user behavior streams with maximum sequence lengths of 50 and 1024, respectively. Table 1 summarizes the dataset statistics.

**Baselines.** We compare WordCTR with 17 representative baselines from five groups: (1) **DNN-based CTR models**: DLRM (Naumov et al., 2019), DIN (Zhou et al., 2018), DCNv2 (Wang et al., 2021), and Wukong (Zhang et al., 2024); (2) **feature-interaction models**: DHEN (Zhang et al., 2022), AutoInt (Song et al., 2019), and HiFormer (Gui et al., 2023); (3) **efficient sequential models**: HSTU (Zhai et al., 2024) and SparseCTR (Lai et al., 2026); (4) **unified sequence and feature models**: InterFormer (Zeng et al., 2025), OneTrans (Zhang et al., 2026), MTGR (Han et al., 2025), EST (Liu et al., 2026), and TokenFormer (Zhou et al., 2026); and (5) **long-sequence interest models**: MIMN (Pi et al., 2019), SDIM (Cao et al., 2022), and LREA (Song et al., 2025). All 17 baselines are evaluated on each dataset.

### 5.2 Overall predictive performance

Table 2 reports AUC, GAUC, and NE on the three public datasets and the InHouse dataset. On the public datasets, WordCTR obtains the best reported values for all three metrics, demonstrating that compressing the behavior history into a unified associative memory can preserve predictive accuracy. On InHouse, WordCTR achieves the highest AUC (0.7005) and lowest NE (0.9196) among the reported methods. Its GAUC (0.5628) is below the best baseline, DHEN (0.5678), indicating that the overall gains do not uniformly translate to user-group ranking.

### 5.3 Inference efficiency

Figure 3 compares total inference latency for cached and uncached WordCTR and the measured MIMN implementation, varying history length $N$ at $K = 64$ or candidate count $K$ at $N = 100$. Each measurement processes one user request, scoring candidates in chunks of at most 64, on an NVIDIA A100-PCIE-40GB GPU with PyTorch 2.5.1+cu118 in FP32. After five warm-up iterations, we report the median of 20 timed repetitions, synchronizing CUDA before and after each call. Inputs are already on the GPU; timings exclude input generation and CPU-to-GPU transfer but include Python dispatch. Requests are measured serially, without an explicitly configured CPU thread count or custom CUDA stream.

**Scaling with history length.** At $K = 64$, increasing $N$ from 25 to 800 changes the median total latency of cached WordCTR from 5.25 to 5.69 ms and that of uncached WordCTR from 5.69 to 6.35 ms (Figure 3, left). In contrast, the measured MIMN implementation increases from 25.06 to

Table 2: Overall performance comparison against 17 baselines on each dataset. Bold: best; underline: second-best (including ties). ↑ indicates higher is better and ↓ indicates lower is better.

| | Amazon Electronics | | | TaobaoAds | | | KuaiVideo | | | InHouse | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | AUC↑ | GAUC↑ | NE↓ | AUC↑ | GAUC↑ | NE↓ | AUC↑ | GAUC↑ | NE↓ | AUC↑ | GAUC↑ | NE↓ |
| DLRM | .8567 | .8551 | .6878 | .6392 | .5626 | .9752 | .7348 | .6450 | .8878 | .6971 | .5673 | .9226 |
| DIN | .8562 | .8550 | .6833 | .6390 | .5626 | .9755 | .7445 | .6621 | .8801 | .7000 | .5653 | .9208 |
| DCNv2 | .8604 | .8585 | .6875 | .6409 | .5658 | .9720 | .7384 | .6519 | .8906 | .6989 | .5674 | .9209 |
| Wukong | .8658 | .8638 | .6596 | .6389 | .5640 | .9770 | .7415 | .6613 | .8922 | .6945 | .5639 | .9236 |
| DHEN | .8539 | .8518 | .6906 | .6438 | .5675 | .9697 | .7369 | .6495 | .8924 | .6977 | **.5678** | .9225 |
| AutoInt | .8514 | .8502 | .6908 | .6418 | .5648 | .9891 | .7345 | .6423 | .8858 | .6965 | .5661 | .9248 |
| HiFormer | .8620 | .8600 | .6886 | .6285 | .5616 | .9914 | .7334 | .6441 | .8998 | .6925 | .5587 | .9275 |
| OneTrans | .8667 | .8649 | .6718 | .6411 | .5656 | .9727 | .7402 | .6558 | .8822 | .6985 | .5649 | .9230 |
| HSTU | .8616 | .8597 | .6782 | .6329 | .5600 | .9752 | .7379 | .6484 | .8828 | .6960 | .5445 | .9271 |
| InterFormer | .8410 | .8396 | .7096 | .6350 | .5638 | .9765 | .7241 | .6208 | .9004 | .6874 | .5516 | .9274 |
| MTGR | .8629 | .8609 | .6731 | .6377 | .5643 | .9719 | .7384 | .6521 | .8927 | .6984 | .5662 | .9210 |
| SparseCTR | .8506 | .8494 | .6909 | .6387 | .5604 | .9730 | .7266 | .6310 | .9103 | .6938 | .5556 | .9245 |
| EST | .8483 | .8483 | .7404 | .6319 | .5595 | .9749 | .7376 | .6485 | .8926 | .6968 | .5637 | .9227 |
| TokenFormer | .8249 | .8273 | .7681 | .6314 | .5538 | .9776 | .7141 | .6081 | .9090 | .6932 | .5566 | .9261 |
| MIMN | .8662 | .8655 | .6777 | .6412 | .5625 | .9725 | .7381 | .6526 | .8851 | .6990 | .5628 | .9205 |
| SDIM | .8280 | .8278 | .7953 | .6382 | .5638 | .9750 | .7376 | .6513 | .8866 | .6967 | .5624 | .9240 |
| LREA | .8591 | .8570 | .6795 | .6413 | .5650 | .9724 | .7361 | .6484 | .8872 | .6989 | .5619 | .9205 |
| **WordCTR** | **.8790** | **.8779** | **.6292** | **.6520** | **.5687** | **.9672** | **.7464** | **.6658** | **.8752** | **.7005** | .5628 | **.9196** |

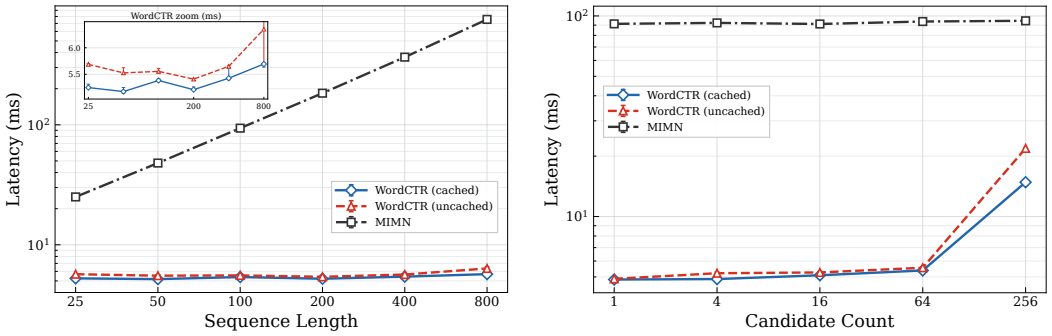


Figure 3: Measured total inference latency versus history length $N$ (left; $K = 64$ candidates) and candidate count $K$ (right; $N = 100$ behaviors). Points show medians over 20 measurements, error bars show the 10th–90th percentiles, and both vertical axes are logarithmic. The left inset enlarges the two WordCTR curves.

756.02 ms. Cached WordCTR has the lowest total latency at every tested length, and the inset makes the smaller but persistent difference between the two WordCTR variants visible. These measurements support the expected weak dependence of candidate-side reading on $N$ in this benchmark; they do not imply that memory writing itself is free.

**Scaling with candidate count.** At $N = 100$ and $K = 256$, cached WordCTR takes 14.83 ms, compared with 21.87 ms without caching and 94.56 ms for the measured MIMN implementation (Figure 3, right). Caching reduces WordCTR's measured total latency by 32.2% at this candidate count. The increase from 64 to 256 candidates includes the cost of additional 64-item scoring chunks, so the curve should not be interpreted as an isolated cost for one memory read. We do not extrapolate candidate-count scaling to $N = 800$, for which only the $K = 64$ setting was measured.

**User- and candidate-side costs.** Table 3 separates the two stages at $K = 64$. As $N$ grows from 25 to 800, WordCTR's user-side time increases from 2.02 to 2.48 ms, whereas the measured MIMN implementation increases from 24.24 to 755.86 ms. WordCTR's candidate-side read-and-score time

Table 3: Latency breakdown by history length $N$ at a fixed candidate count of $K = 64$ (ms). Stage and total values are separately measured medians and therefore need not sum exactly.

| $N$ | Method | User-side precompute | Candidate-side read and score | Total |
|---|---|---|---|---|
| 25 | WordCTR (cached) | 2.02 | 3.29 | 5.25 |
| 25 | MIMN | 24.24 | 0.91 | 25.06 |
| 100 | WordCTR (cached) | 2.07 | 3.36 | 5.38 |
| 100 | MIMN | 92.97 | 0.89 | 93.79 |
| 400 | WordCTR (cached) | 2.21 | 3.28 | 5.43 |
| 400 | MIMN | 364.11 | 0.87 | 366.44 |
| 800 | WordCTR (cached) | 2.48 | 3.28 | 5.69 |
| 800 | MIMN | 755.86 | 1.02 | 756.02 |

Table 4: Effects of the Memory Reader depth and temporal distance encoding. The best result for each metric and dataset within each comparison is shown in bold.

| | Amazon Electronics | | KuaiVideo | | TaobaoAds | |
|---|---|---|---|---|---|---|
| Setting | AUC ↑ | NE ↓ | AUC ↑ | NE ↓ | AUC ↑ | NE ↓ |
| *(a) Number of Memory Reader layers* | | | | | | |
| $L_{\mathrm{R}} = 1$ | 0.8733 | 0.6421 | **0.7471** | 0.8759 | 0.6510 | **0.9655** |
| $L_{\mathrm{R}} = 2$ | 0.8755 | 0.6358 | **0.7471** | **0.8744** | 0.6516 | 0.9686 |
| $L_{\mathrm{R}} = 3$ (default) | **0.8790** | **0.6292** | 0.7464 | 0.8752 | 0.6520 | 0.9672 |
| $L_{\mathrm{R}} = 4$ | 0.8744 | 0.6396 | 0.7449 | 0.8788 | **0.6528** | 0.9681 |
| *(b) Number of temporal distance buckets* | | | | | | |
| $B = 1$ | 0.8722 | 0.6442 | 0.7468 | 0.8762 | 0.6508 | 0.9671 |
| $B = 5$ | 0.8769 | 0.6329 | 0.7479 | **0.8745** | 0.6504 | **0.9668** |
| $B = 9$ (default) | **0.8790** | **0.6292** | 0.7464 | 0.8752 | **0.6520** | 0.9672 |
| $B = 13$ | 0.8771 | 0.6339 | **0.7481** | 0.8747 | 0.6512 | 0.9694 |

stays within 3.28–3.36 ms, which is higher than MIMN's 0.87–1.02 ms. Under this workload, the lower measured user-side cost more than offsets the higher candidate-side cost, keeping WordCTR's total latency within 5.25–5.69 ms.

### 5.4 Ablation Study

Table 4 varies Memory Reader depth $L_{\mathrm{R}}$ and age-bucket count $B = B_{\mathrm{age}}$. With $B = 1$, the age embedding cannot distinguish positions, although recency-dependent write weights remain active. We select $L_{\mathrm{R}} = 3$ and $B = 9$ as a shared compromise: both give the best Amazon Electronics AUC and NE in their respective sweeps, and every AUC/NE gap to the best setting on KuaiVideo and TaobaoAds is at most 0.0017. Appendix C reports a separate write-weight ablation.

## 6 Conclusion

In this work, we study cross-candidate history reuse for long-sequence CTR prediction through WordCTR's **Write Once, Read Deep** design. WordCTR encodes the retained user history into a fixed-size, multi-head associative memory using content- and recency-weighted ridge regression. Candidate-independent memory construction allows the same memory to be reused across candidate predictions; stacked reader blocks then make candidate-conditioned queries without revisiting the behavior sequence. Against 17 baselines, WordCTR obtains the best reported AUC, GAUC, and NE on Amazon Electronics, KuaiVideo, and TaobaoAds. On InHouse, which includes a behavior stream of up to 1024 events, it obtains the best AUC and NE, while its GAUC remains below the strongest baseline. In the measured single-request GPU workload, cached WordCTR has low total latency for tested histories up to 800 events and 32.2% lower total latency than its uncached variant at 256 candidates. The efficiency measurements cover the tested implementations and workload; writing and maintaining the memory still incur costs.

### ETHICS STATEMENT

This work studies predictive modeling of user interaction logs. Such logs may encode sensitive attributes, exposure bias, popularity bias, and historical inequities. The public experiments use anonymized benchmark data under their respective licenses. We do not attempt to infer protected attributes or identify individuals. A production deployment should apply data minimization, access control, retention limits, privacy review, bias monitoring, and user-consent requirements appropriate to the jurisdiction and application. Improvements in offline CTR metrics do not by themselves establish improvements in user welfare.

### REPRODUCIBILITY STATEMENT

The submission will include exact configuration files, preprocessing scripts, environment versions, and commands for every main-table and ablation run. Section 4 specifies the model and objective; Appendix A derives the memory solution; Appendix D records dataset-specific hyperparameters.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting and editing the paper text. We have not used generative AI tools for generating synthetic or experimental data, formulating mathematical claims, assisting with proofs, designing the methodology or experiments, implementing methods or code, or interpreting experimental results, and these tasks are not applicable to this work. All AI-assisted text was reviewed, verified, and edited by the authors. We take responsibility for the final content of this work, including all text produced with the aid of generative AI.

## REFERENCES

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT but Faster. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.09461.

Yue Cao, Xiaojiang Zhou, Jiaqi Feng, Peihao Huang, Yao Xiao, Dayao Chen, and Sheng Chen. Sampling Is All You Need on Modeling Long-Term User Behaviors for CTR Prediction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pp. 2974–2983, 2022. doi: 10.1145/3511808.3557082.

Jianxin Chang, Chenbin Zhang, Zhiyi Fu, Xiaoxue Zang, Lin Guan, Jing Lu, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. TWIN: TWo-Stage Interest Network for Lifelong User Behavior Modeling in CTR Prediction at Kuaishou. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3785–3794, 2023. doi: 10.1145/3580305.3599922.

Qiwei Chen, Changhua Pei, Shanshan Lv, Chao Li, Junfeng Ge, and Wenwu Ou. End-to-end user behavior retrieval in click-through rate prediction model. *arXiv preprint arXiv:2108.04468*, 2021.

Qiwei Chen, Yue Xu, Changhua Pei, Shanshan Lv, Tao Zhuang, and Junfeng Ge. Efficient Long Sequential User Data Modeling for Click-Through Rate Prediction. *arXiv preprint arXiv:2209.12212*, 2022.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking Attention with Performers. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.14794.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4271–4282, 2020.

Huan Gui, Ruoxi Wang, Ke Yin, Long Jin, Maciej Kula, Taibai Xu, Lichan Hong, and Ed H Chi. Hiformer: Heterogeneous feature interactions learning with transformers for recommender systems. *arXiv preprint arXiv:2311.05884*, 2023.

Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, et al. Mtgr: Industrial-scale generative recommendation framework in meituan. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5731–5738, 2025.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 507–517, 2016.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers Are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 2020.

Weijiang Lai, Beihong Jin, Di Zhang, Siru Chen, Jiongyan Zhang, Yuhang Gou, Jian Dong, and Xingxing Wang. Unleashing the potential of sparse attention on long-term behaviors for CTR prediction. In *Proceedings of the ACM Web Conference 2026*, pp. 8041–8050. Association for Computing Machinery, 2026. doi: 10.1145/3774904.3792808.

Yongqi Li, Meng Liu, Jianhua Yin, Chaoran Cui, Xin-Shun Xu, and Liqiang Nie. Routing micro-videos via a temporal graph-guided recommendation system. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2019. doi: 10.1145/3343031.3350950.

Mingyang Liu, Yong Bai, Zhangming Chan, Sishuo Chen, Xiang-Rong Sheng, Han Zhu, Jian Xu, and Xinyang Chen. EST: Towards efficient scaling laws in click-through rate prediction via unified modeling. *arXiv preprint arXiv:2602.10811*, 2026.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2671–2679, 2019. doi: 10.1145/3292500.3330666.

Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. Search-Based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 2685–2692, 2020. doi: 10.1145/3340531.3412744.

Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. User Behavior Retrieval for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2347–2356, 2020. doi: 10.1145/3397271.3401440.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021.

Zihua Si, Lin Guan, ZhongXiang Sun, Xiaoxue Zang, Jing Lu, Yiqun Hui, Xingchao Cao, Zeyu Yang, Yichen Zheng, Dewei Leng, Kai Zheng, Chenbin Zhang, Yanan Niu, Yang Song, and Kun Gai. TWIN V2: Scaling ultra-long user behavior sequence modeling for enhanced ctr prediction at kuaishou. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4890–4897. Association for Computing Machinery, 2024. doi: 10.1145/3627673.3680030.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. AutoInt: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1161–1170, 2019. doi: 10.1145/3357384.3357925.

Xin Song, Xiaochen Li, Jinxin Hu, Hong Wen, Zulong Chen, Yu Zhang, Xiaoyi Zeng, and Jing Zhang. LREA: Low-rank efficient attention on modeling long-term user behaviors for CTR prediction. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2843–2847. ACM, 2025. doi: 10.1145/3726302.3730228.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *Journal of Machine Learning Research*, 27 (169):1–41, 2026. URL https://jmlr.org/papers/v27/25-0903.html.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297, 2020.

Zhichen Zeng, Xiaolong Liu, Mengyue Hang, Xiaoyi Liu, Qinghai Zhou, Chaofei Yang, Yiqun Liu, Yichen Ruan, Laming Chen, Yuxin Chen, Yujia Hao, Jiaqi Xu, Jade Nie, Xi Liu, Buyun Zhang, Wei Wen, Siyang Yuan, Hang Yin, Xin Zhang, Kai Wang, Wen-Yen Chen, Yiping Han, Huayu Li, Chunzhi Yang, Bo Long, Philip S. Yu, Hanghang Tong, and Jiyan Yang. Interformer: Effective heterogeneous interaction learning for click-through rate prediction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, pp. 6225–6233. Association for Computing Machinery, 2025. ISBN 9798400720406. doi: 10.1145/3746252.3761527. URL https://doi.org/10.1145/3746252.3761527.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58484–58509. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhai24a.html.

Buyun Zhang, Liang Luo, Xi Liu, Jay Li, Zeliang Chen, Weilin Zhang, Xiaohan Wei, Yuchen Hao, Michael Tsang, Wenjun Wang, et al. Dhen: A deep and hierarchical ensemble network for large-scale click-through rate prediction. *arXiv preprint arXiv:2203.11014*, 2022.

Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Shen Li, Yanli Zhao, Yuchen Hao, Yantao Yao, Ellie Dingqiao Wen, Jongsoo Park, Maxim Naumov, and Wenlin Chen. Wukong: Towards a Scaling Law for Large-Scale Recommendation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59421–59434. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24ao.html.

Zhaoqi Zhang, Haolei Pei, Jun Guo, Tianyu Wang, Yufei Feng, Hui Sun, Shaowei Liu, and Aixin Sun. OneTrans: Unified feature interaction and sequence modeling with one transformer in industrial recommender. In *Proceedings of the ACM Web Conference 2026*, pp. 8162–8170. Association for Computing Machinery, 2026. doi: 10.1145/3774904.3792838.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1059–1068, 2018. doi: 10.1145/3219819.3219823.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep Interest Evolution Network for Click-Through Rate Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5941–5948, 2019. doi: 10.1609/aaai.v33i01.33015941.

Yifeng Zhou, Yuehong Hu, Zhixiang Feng, Junwei Pan, Kaihui Wu, Hanyong Li, Shangyu Zhang, Shudong Huang, Zhangbin Zhu, Chengguo Yin, Haijie Gu, and Jie Jiang. TokenFormer: Unify the multi-field and sequential recommendation worlds. *arXiv preprint arXiv:2604.13737*, 2026.

## A Derivation of the Closed-Form Memory

For a fixed memory head, stack keys row-wise in the design matrix $D \in \mathbb{R}^{N\times d_{k,h}}$, values in $V \in \mathbb{R}^{N\times d_{v,h}}$, and weights in the diagonal matrix $W = \mathrm{diag}(w_1, \ldots, w_N)$. Equation 12 becomes

$$\mathcal{J}(A) = \|W^{1/2}(V - DA)\|_F^2 + \lambda\|A\|_F^2. \tag{23}$$

Expanding the trace and differentiating gives

$$\nabla_A \mathcal{J}(A) = 2D^\top WDA - 2D^\top WV + 2\lambda A. \tag{24}$$

Setting the gradient to zero yields the normal equations

$$(D^\top WD + \lambda I)A = D^\top WV. \tag{25}$$

Because $\lambda > 0$, the left-hand matrix is positive definite and the unique solution is

$$A^\star = (D^\top WD + \lambda I)^{-1}D^\top WV. \tag{26}$$

The implementation forms $\Sigma = D^\top WD$ and $G = D^\top WV$ without materializing $W$, then solves $(\Sigma + \lambda I)A = G$ by Cholesky factorization. It never forms an explicit matrix inverse.

## B Detailed Complexity Comparison

Let $p = d_k/H$ and $v = d_v/H$. After key and value projection, accumulating the weighted normal and cross terms over $H$ memory heads costs

$$O\left(\frac{N(d_k^2 + d_k d_v)}{H}\right). \tag{27}$$

The Cholesky factorizations and linear solves cost

$$O\left(\frac{d_k^3 + d_k^2 d_v}{H^2}\right). \tag{28}$$

The resulting multi-head memory contains

$$S_M = \frac{d_k d_v}{H} \tag{29}$$

scalar entries. Therefore, its storage and the cost of one complete multi-head memory lookup are both independent of the history length $N$.

In the implementation, $F$ global fields query the memory in each of $L$ reader blocks, giving an exact memory-lookup cost of $O(LFS_M)$ per candidate. Because $F$, $L$, and all representation dimensions are fixed model hyperparameters, Table 5 focuses on scaling with the history length $N$ and the number of candidates $K$. Non-sequential field interaction, projection, FFN, and prediction-head costs are excluded consistently for all methods.

Table 5: History storage and access complexity. $N$ is the history length, $K$ is the number of candidates, $R$ is the number of retrieved behaviors, $C_{\mathrm{ret}}(N)$ is the system-dependent retrieval cost, and $S_M = d_k d_v / H$ is the fixed WordCTR memory size. Fixed backbone computation is omitted for all methods.

| Method | User state | Per candidate | $K$ candidates |
|---|---|---|---|
| Full attention | $O(Nd)$ | $O(Nd)$ | $O(KNd)$ |
| Retrieve–attend | $O(Nd)$ + index | $O(C_{\mathrm{ret}}(N) + Rd)$ | $O(K(C_{\mathrm{ret}}(N) + Rd))$ |
| Fixed vector | $O(d)$ | $O(d)$ | $O(Kd)$ |
| **WordCTR** cached | $O(S_M)$ | $O(S_M)$ | $O(KS_M)$ |

Table 6: Uniform-write ablation compared with full WordCTR. Full-model values are repeated from Table 2. Bold marks the better result for each metric within a dataset. Higher AUC and GAUC and lower NE are better.

| Setting | AUC↑ | GAUC↑ | NE↓ |
|---|---|---|---|
| *Amazon Electronics* | | | |
| Full WordCTR | **0.8790** | **0.8779** | **0.6292** |
| Uniform writes | 0.8744 | 0.8726 | 0.6399 |
| *TaobaoAds* | | | |
| Full WordCTR | **0.6520** | **0.5687** | **0.9672** |
| Uniform writes | 0.6504 | 0.5681 | 0.9691 |
| *KuaiVideo* | | | |
| Full WordCTR | **0.7464** | **0.6658** | **0.8752** |
| Uniform writes | 0.7437 | 0.6615 | 0.8792 |

## C Write-Weight Ablation

We replace the write weight in Equation 11 with $w_t = \mathbb{I}[t \text{ is valid}]$, assigning unit weight to every valid behavior and zero weight to padding. This removes the content gate and explicit temporal decay while retaining the sequence encoder, ridge-regression associative memory, candidate-conditioned reader, and age embeddings in the value path. Table 6 places the uniform-write results alongside the full WordCTR values from Table 2. Uniform writes reduce AUC and GAUC by 0.0046 and 0.0053 on Amazon Electronics, while increasing NE by 0.0107. On TaobaoAds, AUC and GAUC decrease by 0.0016 and 0.0006, and NE increases by 0.0019. On KuaiVideo, AUC and GAUC decrease by 0.0027 and 0.0043, and NE increases by 0.0040. The uniform-write variant has LogLoss 0.4435, 0.1928, and 0.4318 on Amazon Electronics, TaobaoAds, and KuaiVideo, respectively. These results support using content- and recency-dependent write strengths on all three datasets.

## D Hyperparameters and training setup

For all four datasets—Amazon Electronics, TaobaoAds, KuaiVideo, and InHouse—we use an embedding dimension of 64, three memory-read blocks, four memory heads, key and value dimensions of 64, nine temporal-age buckets, and a convolution kernel size of 4. The ridge coefficient is initialized to 1.0, while the learnable decay rate is initialized in the range $[10^{-3}, 10^{-1}]$. The dropout rate is 0.1.

We train the model with AdamW using a learning rate of $10^{-4}$ for dense parameters and $10^{-2}$ for sparse embeddings. The dense and embedding weight decays are set to 0 and $10^{-4}$, respectively. Gradients are clipped to a maximum norm of 10. Training is performed for at most 100 epochs with early stopping after three epochs without improvement in validation AUC. We use random seed 42.