MambaCTR : A efficience-based multimodal recommandation system for MM-CTR prediction

WeiCheng Duan
2252109@tongji.edu.cn
Tongji University
Shanghai,China

JunZe Zhu
2351114@tongji.edu.cn
Tongji University
Shanghai,China

SYNOPSIS

The WWW 2025 EReL@MIR Workshop Multimodal CTR Prediction Challenge (https://erel-mir.github.io/challenge) aims to improve click-through rate (CTR) prediction in recommender systems by leveraging multimodal embedding features.

This paper presents our state-of-the-art (SOTA) solution, **MambaCTR**, which optimizes both Task 1 and Task 2 in a decoupled way. MambaCTR introduces a novel two-stage user-interest extraction framework that integrates sequential modeling with feature interaction learning to effectively capture user-item relationships. By incorporating an SSM-based module design, MambaCTR significantly enhances computational efficiency while maintaining high prediction accuracy, demonstrating its potential for building high-performance recommendation systems.

We evaluate our method on the challenge dataset and conduct ablation experiments to verify the effectiveness of our approach. While the ablated transformer-based CTR model achieves an AUC score of **0.9865**, our MambaCTR not only attains comparable accuracy with an AUC of **0.9843**, but also demonstrates a **2.5× improvement in training time and 1.5× improvement in inference time**. The implementation code and configurations are available in our GitHub repository (https://github.com/tj-messi/Mamba-back-efficiency-based-recommendation-system).

KEYWORDS

Recommendation System, Multi-modality, Click-Through Rate Prediction(CTR)

ACM EngageCSEdu, July 2025.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
WeiCheng Duan and JunZe Zhu. July 2025. MambaCTR : A efficience-based multimodal recommandation system for MM-CTR prediction. In *ACM EngageCSEdu.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BACKGROUND

Click-through rate (CTR) prediction is a fundamental task in the online advertising and recommendation systems, which focus on predicting the probability of users clicking the provided advertises or videos based on video thumbnails, subtitles, and users' historical click data.The accuracy of the CTR prediction directly influence the revene of advertising platforms and the user experience.And the efficiency[1] of the CTR prediction algorithm is closely related to the resource consumption of the recommendation system and the overall operational cost of the platform.

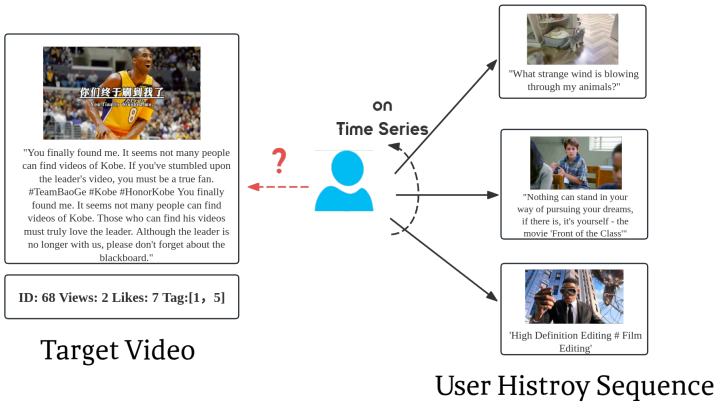

Figure 1: Multimodal CTR (MM-CTR) Prediction Task

The problem of improving the accuracy and the efficiency of the CTR prediction algorithm can be devided into two parts:
How to obtain multimodal embeddings of video thumbnails, subtitles, and users' historical click data.
How to leverage pre-extracted multimodal features for click-through rate (CTR) prediction.

All these problems are included in the Multimodal CTR Prediction (MM-CTR) Challenge at the WWW 2025 EReL@MIR Workshop[17], aiming to foster innovation in adopting multimodal embedding features for CTR prediction and improve the performance of prediction. Different from traditional ways to conduct click-through rate (CTR) prediction like factorization machines(FMs)[4, 6, 7, 9, 11] or some deep learning model[15, 18], we designed a multimodal embedding method along with a mamba based network for effectively and efficiently modeling the feature.

2 BACKGROUND

2.1 Multimodal Embedding and preprocessing

In recommender systems, Multimodal Embedding aims to integrate information from different modalities (e.g., images, text, audio, video) into a unified vector space to better understand user preferences and item characteristics. Traditional recommender systems primarily rely on user-item interaction data and item ID information. However, with the rise of multimedia services, leveraging the rich multimodal content of items has become increasingly important. Multimodal embedding not only enriches item representations but also helps alleviate data sparsity issues and improves recommendation accuracy and interpretability.

The implementation of multimodal embedding typically involves several key aspects:

2.1.1 Modality Encoders. Modality encoders are responsible for extracting high-level feature representations from raw multimodal data. Different modalities require specialized encoders to process their unique data formats and semantic information. For example:
These modality encoders transform heterogeneous multimodal data into unified dense vector representations, providing a foundation for subsequent feature interaction and recommendation models.

2.1.2 Feature Interaction. Another core challenge in multimodal embedding is how to effectively fuse features from different modalities. Since features from different modalities exist in distinct semantic spaces, and user preferences for different modalities may vary, effective feature interaction mechanisms are needed to integrate this information. Common feature interaction methods include:
Bridging: Aims to build multimodal information transfer channels to capture multimodal relationships between users and items. For example, by constructing user-item graphs, item-item graphs, or knowledge graphs, multimodal information is integrated into the graph structure and propagated and aggregated through graph neural networks, thereby enhancing user and item representations. This method helps models understand user preference differences for various modalities.
Fusion: Focuses on the relationships between different modalities within an item, combining various preferences into the modalities. Attention mechanisms are commonly used fusion methods, which can assign different weights based on the importance of different modalities, achieving coarse-grained or fine-grained fusion. Coarse-grained fusion typically integrates features from entire modalities, while fine-grained fusion focuses on local features within modalities, such as specific regions of an image or keywords in text. Other fusion methods include concatenation and gating mechanisms.
Filtration: Aims to filter out noisy information in multimodal data that is irrelevant to user preferences. Noise can exist in interaction graphs or multimodal features themselves. For example, image processing techniques (e.g., image segmentation) can remove irrelevant regions from images, or denoising mechanisms in graph neural networks can correct erroneous information in user-item interaction graphs.

2.1.3 Feature Enhancement. To further improve the quality of multimodal embeddings and recommendation performance, researchers have also explored various feature enhancement techniques, primarily including:
Disentangled Representation Learning (DRL): Aims to decompose item or user representations into independent, semantically meaningful latent factors, thereby better understanding user preferences for specific aspects of target items. DRL can help models distinguish unique and common semantic information across different modalities and alleviate feature entanglement.
Contrastive Learning (CL): Enhances representation learning through data augmentation and the design of contrastive loss functions. The goal of CL is to bring similar samples (e.g., representations of the same item across different modalities) closer in the embedding space while pushing dissimilar samples apart. CL is often used for modality alignment, ensuring that representations from different modalities remain semantically consistent.

In summary, multimodal embedding is the cornerstone of multimodal recommender systems. Through effective modality encoding, feature interaction, and feature enhancement techniques, it can fully leverage the advantages of multimodal information to improve the performance and user experience of recommender systems [10].

2.2 Click-through Rate (CTR) Prediction

Click-Through Rate (CTR) prediction is a core task in recommender systems and online advertising[16], aiming to predict the probability of a user clicking on a specific item (e.g., an advertisement, product, or video). The accuracy of CTR prediction directly impacts the revenue of advertising

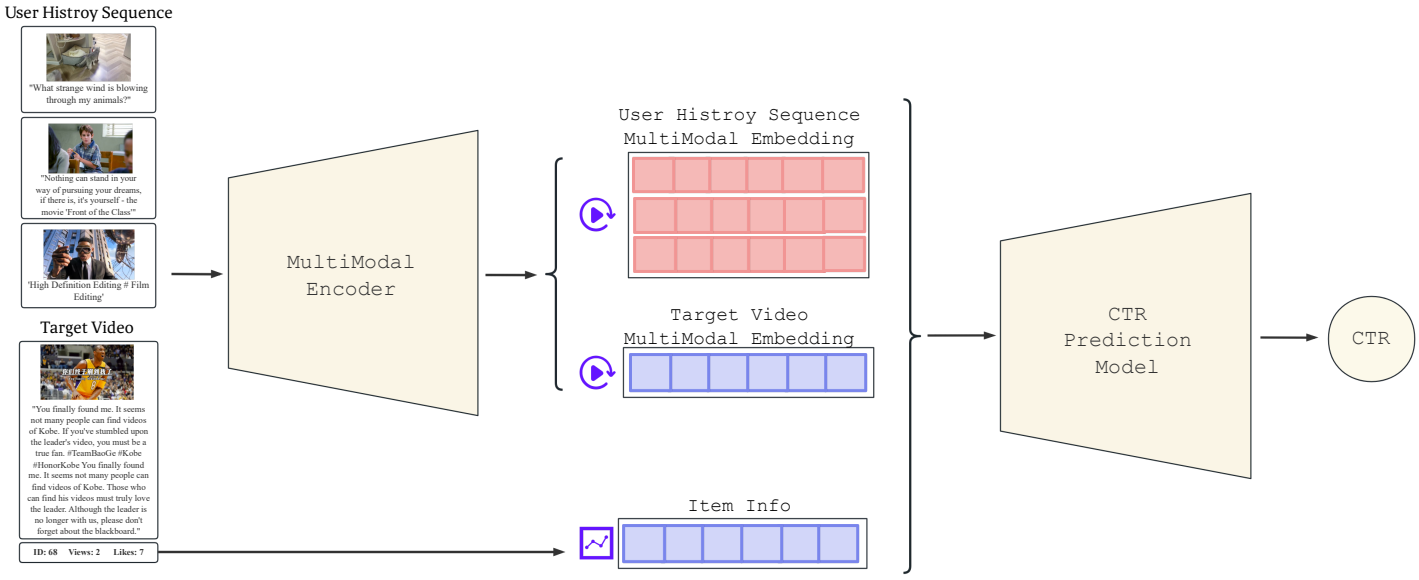

Figure 2: Model Structure of MM-CTR Challange

platforms and user experience. Typically, CTR prediction is modeled as a binary classification problem, i.e., predicting whether a user will click on an item. With the explosive growth of internet content and the increasing complexity of user behavior, CTR prediction models have continuously evolved from traditional machine learning methods to deep learning models.

2.2.1 Traditional CTR Prediction Models. Early CTR prediction models were primarily based on Logistic Regression (LR), Factorization Machines (FM)[2, 11], and their variants (e.g., Field-aware Factorization Machines, FFM)[7]. These models predicted CTR by modeling linear or second-order interactions between user, item, and contextual features. They showed some ability in handling sparse data and capturing feature interactions, but their capacity to model complex higher-order nonlinear feature interactions was limited.

2.2.2 Deep Learning CTR Prediction Models. In recent years, deep learning techniques have made significant progress in CTR prediction, greatly improving model performance. Deep learning models can automatically learn complex representations and higher-order nonlinear interactions of features, thereby better capturing user interests and item characteristics. Common deep learning CTR prediction models include:
MLP-based Models: Such as Wide and Deep Learning [Google] and DeepFM [Huawei][4, 9]. These models combine linear models (for memorizing broad features) with deep neural networks (for learning deep feature interactions), aiming to leverage both shallow and deep features to improve prediction effectiveness.
Attention-based Models: Such as Deep Interest Network (DIN)[18] and Deep Interest Evolution Network (DIEN)[15]. These models introduce attention mechanisms to dynamically activate relevant interests from a user's historical behavior based on the current target item, thereby more accurately capturing the diversity and dynamism of user interests.
Graph Neural Network (GNN) Models: GNNs[5] are increasingly applied in CTR prediction, especially when dealing with user-item interaction graphs and knowledge graphs[8]. GNNs can capture complex relationships between users and items and perform effective feature propagation and aggregation, thereby improving recommendation accuracy.
Sequential Models: Considering the sequential nature of user behavior, recurrent neural networks (RNNs)[13] and Transformers[12] and so on[2] are also applied in CTR prediction to capture the evolution path of user interests and contextual dependencies.
Multi-task Learning Models: Some models adopt a multi-task learning paradigm, simultaneously optimizing CTR prediction and other related tasks (e.g., conversion rate prediction) to improve the model's generalization ability and robustness.

2.2.3 Multimodal CTR Prediction. With the widespread adoption of multimedia content, multimodal information plays an increasingly important role in CTR prediction. Multimodal CTR prediction aims to integrate features from various modalities (visual, textual, audio, etc.) into CTR prediction

models to provide richer and more comprehensive user and item representations. This typically involves the extraction, fusion, and interaction of multimodal features, as discussed in the

previous section on "Multimodal Embedding." By effectively utilizing multimodal information, the accuracy of CTR prediction can be further improved, especially in scenarios with data sparsity or cold start.

In summary, CTR prediction is an active research area in recommender systems. Future developments will continue to focus on how to more effectively model user interests, item characteristics, and their complex interactions, as well as how to leverage multimodal information and more advanced deep learning techniques to improve prediction performance and efficiency [10].

3 METHOD

In this section, we first formulate the problem of multimodal recommendation and present the overall framework of proposed MambaCTR and then introduce each component in detail.

3.1 Problem Definition

We denote the user set as $\mathcal{U} = \{u\}$ and the item set as $\mathcal{I} = \{i\}$. Each user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ is associated with an ID embedding vector, denoted as $e_u \in \mathbb{R}^d$ and $e_i \in \mathbb{R}^d$, respectively, where d is the embedding dimension.

Furthermore, we incorporate the multimodal content features of items. For any item i, the raw modality-specific feature under modality $m \in \mathcal{M}$ is denoted as $e_i^m \in \mathbb{R}^{d_m}$, where $\mathcal{M}$ is the set of considered modalities, and d_m is the feature dimension corresponding to modality m. In this work, we focus on two mainstream modalities: visual (v) and textual (t), i.e., $\mathcal{M} = \{v, t\}$.

In MM-CTR prediction task, the goal is to estimate the probability that a user clicks on a given target item based on their historical interaction behavior and the multimodal content of the target item.

For each user $u \in \mathcal{U}$, we define their historical interaction sequence as:

$$\mathcal{H}_u = [h_1, h_2, \ldots, h_L],$$

where L is the sequence length, and each $h_l \in \mathbb{R}^d$ denotes the fused representation of the item the user interacted with at the l-th position. The fused representation is computed from the multimodal features of the corresponding item via a fusion function $f_{\text{fuse}}(\cdot)$:

$$h_l = f_{\text{fuse}}\left(\{e_{i_l}^m\}_{m \in \mathcal{M}}\right),$$

where i_l is the item in the l-th interaction.

Similarly, for a given target item $i^{\text{target}} \in \mathcal{I}$, we obtain its multimodal fused representation as:

$$e^{\text{target}} = f_{\text{fuse}}\left(\{e_{i^{\text{target}}}^m\}_{m \in \mathcal{M}}\right).$$

We also denoted structured side items information $s_{i^{\text{target}}}$ associated with the target item, which includes categorical or ordinal metadata(Like textTags, ViewLevel, LikeLevel): and can be embedded into vector form to enhance the item representation.

CTR is computed by CTR prediction model $f_{\text{CTR}}(\cdot)$ that takes as input the user's history $\mathcal{H}_u$, the fused representation of the target item e^{target}, and its side information $s_{i^{\text{target}}}$:

$$\hat{y}_{u,i^{\text{target}}} = f_{\text{CTR}}(\mathcal{H}_u,\ e^{\text{target}},\ s_{i^{\text{target}}}).$$

3.2 Detailed Structure of Framework

3.2.1 Overview. As illustrated in Figure 2, our proposed framework of MambaCTR consists of two major components: ***(i) MultiModal Embedding Encoder***, which convert original images and texts representation to a single fusion multimodal embedding; ***(ii) SSM-based CTR Prediction Model*** with a novel two stage representation extractor, which focuses not only on user's self-attention interest in a static scope, but also on interactive-interest to target item in a dynamic scope. Besides, the mamba backbone design greatly enhances the training and inference efficency.

The proposed decoupled MambaCTR framework effectively integrates visual, textual, and structured information to improve the ranking performance. Two parts can work independently in a cascaded way.

3.2.2 MultiModal Embedding Encoder. Obtaining a unified and high-quality representation across different modalities is crucial for downstream CTR prediction. We adopt the pretrained CLIP model as our foundational alignment framework. CLIP offers unified embedding space effectively bridges low-level modality gaps and facilitates cross-modal interaction. Compared to modality-specific encoders trained independently, CLIP offers better generalization and transferability due to its joint multimodal optimization.

Nevertheless, the original CLIP model exhibits limitations in modeling long textual content, which is common in recommendation scenarios where video titles often consist of complex, multi-sentence metadata. So exactly we incorporate *Jina-CLIP-v2* as a replacement for the standard CLIP-Text encoder. This variant enhances the Transformer backbone and integrates semantic-aware pooling mechanisms, enabling better extraction of key information from long-form text while preserving alignment with the visual modality.

Based on the above, we designed a ***Multimodal Embedding Encoder*** module to extract fusion representation from multimodal metadata(As shown in **Figure 3**). for a given item i, we first extract its visual feature and its corresponding title

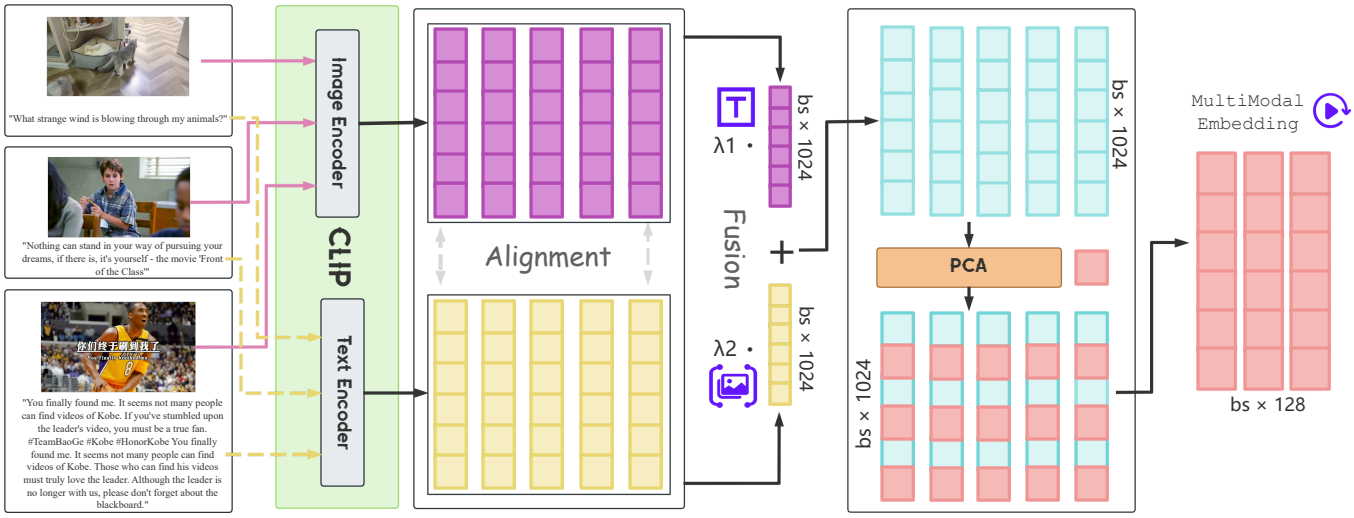

Figure 3: Multimodal Embedding Encoder

text feature in an aligned manner:

$$e_i^v = \texttt{Encoder}_{\texttt{image}}(x_i^v), \quad e_i^v \in \mathbb{R}^{1024},$$
$$e_i^t = \texttt{Encoder}_{\texttt{text}}(x_i^t), \quad e_i^t \in \mathbb{R}^{1024}.$$

To align the dimensionality and integrate both modalities, we project these representations into a shared latent space via principal component analysis (PCA).

$$e_i^{\text{fuse}} = \texttt{PCA}(\lambda_1 \cdot e_i^v + \lambda_2 \cdot e_i^t),$$

where $\lambda_1 + \lambda_2 = 1$ should be guaranteed to ensure that the value range of the fused embedding remains roughly the same as the original.

The use of PCA here serves a dual purpose: it not only reduces the dimensionality of high-dimensional features but also suppresses low-variance noise commonly found in pre-trained outputs. This ensures a compact, denoised, and interpretable embedding space.

Finally, we obtain the unified multimodal embedding for item i, whose dimension is 128. This fused embedding serves as the input to both the user behavior encoder and the CTR prediction module.

3.2.3 SSM-based CTR Prediction Model. The CTR Prediction Model estimates the probability that a user u will click on a given target item i^{target} based on their historical behavior context. The model is designed around a fully state-space architecture with a novel two-stage interest extractor and a lightweight residual prediction head, as shown in **Figure 4**.

Model Input Overview. For each user u, we define their input as a tuple:

$$\left(\mathcal{H}_u,\ i^{\text{target}},\ \texttt{SideInfo}(i^{\text{target}})\right),$$

where:

- $\mathcal{H}_u = [i_1, i_2, \ldots, i_L]$ denotes the user's historical interaction sequence, with each item i_l drawn from item set $\mathcal{I}$;
- i^{target} is the target item whose CTR we aim to predict;
- $\texttt{SideInfo}(i^{\text{target}})$ refers to structured item attributes such as view-level, like-level, and category tags.

Each item i_l and the target item i^{target} is associated with a fused multimodal representation e_{i_l} and $e_{i^{\text{target}}}$, respectively, both obtained from the multimodal encoder described in Section 3.2.2. These fused embeddings capture image and textual semantics.

Embedding Layer. We adopt a unified embedding module to transform structured item features into dense representations, implemented as a shared `FeatureEmbedding` class parameterized by a feature map. For each target item i^{target}, we consider a set of structured features: view-level v_i, like-level k_i, and tag-level t_i, all of which are categorical in nature and drawn from predefined vocabularies.

The embedding layer constructs a dictionary of embedding vectors by processing each feature independently. Formally, for each feature type $f \in \{v, k, t\}$, we define:

$$e_{f_i} = E_f(f_i) \in \mathbb{R}^{d_f},$$

where E_f is an embedding lookup table for feature f, initialized with random.

To accommodate differences in pretrained and target dimensions, each embedding vector can optionally be passed through a learnable linear projection layer:

$$\tilde{e}_{f_i} = W_f e_{f_i}, \quad W_f \in \mathbb{R}^{d'_f \times d_f}.$$

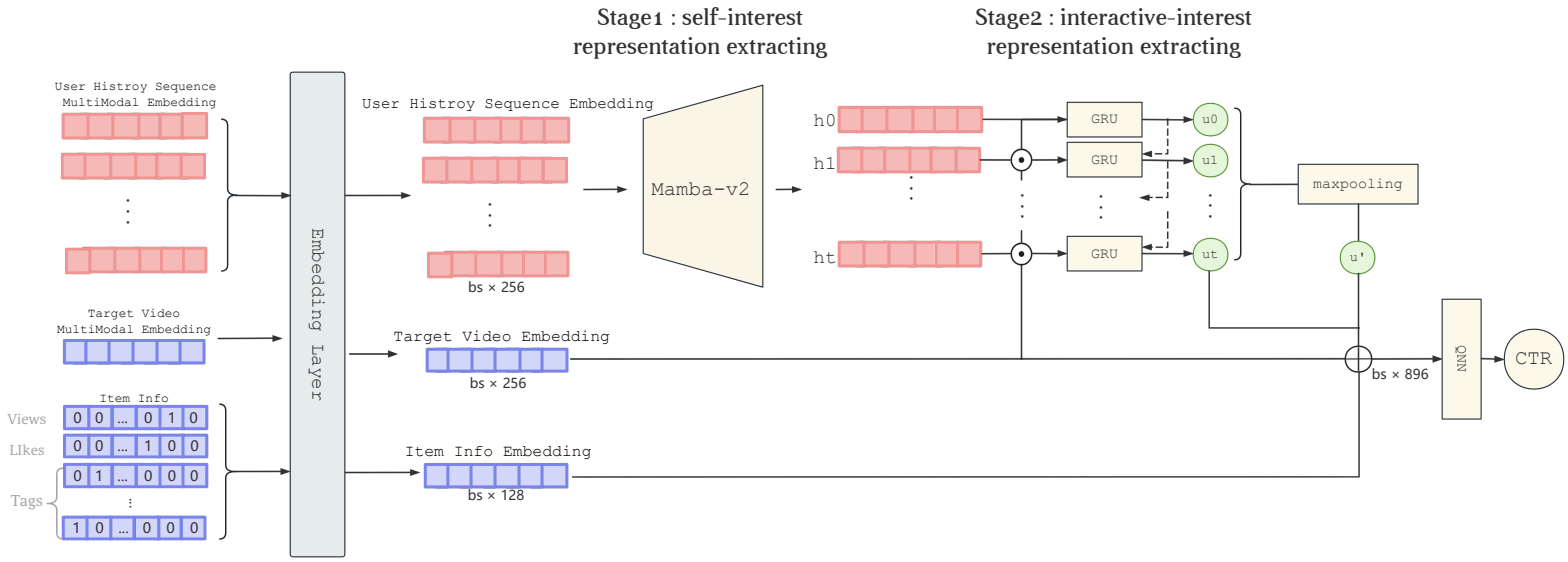

Figure 4: SSM-based CTR Prediction Model

The module supports categorical and sequence features, with 'sequence' types optionally processed using a masked sum pooling operator. Numeric features (e.g., continuous views level) are passed through a learnable linear transformation.

For each target item i^{target}, we gather side information (views, likes) and generate a `Target-SideEmbed` via concatenation:

$$e^{\texttt{Target-SideEmbed}} = \tilde{e}_{v_i} \oplus \tilde{e}_{k_i} \in \mathbb{R}^{128}.$$

In addition to side information, both the multimodal embeddings of the historical items and the target item are also passed through a shared embedding transformation to align them for downstream modeling. Each multimodal embedding e_{i_l} and $e_{i^{\text{target}}}$ is processed as:

$$\tilde{e}_{i_l} = W_{\text{mm}} \cdot (e_{i_l} \oplus \tilde{e}_{t_l}), \quad \tilde{e}_{i^{\text{target}}} = W_{\text{mm}} \cdot (e_{i^{\text{target}}} \oplus \tilde{e}_{t^{\text{target}}}), \quad W_{\text{mm}} \in \mathbb{R}^{256 \times d}.$$

This produces the history embedding sequence:

$$\hat{\mathcal{H}}_u = [\tilde{e}_{i_1}, \tilde{e}_{i_2}, \ldots, \tilde{e}_{i_L}] \subset \mathbb{R}^{L \times 256},$$

and the projected target embedding:

$$e^{\texttt{Target-MMEmbed}} = \tilde{e}_{i^{\text{target}}} \in \mathbb{R}^{256}.$$

Thus, the total embedding input to the CTR prediction model includes the transformed sequence $\hat{\mathcal{H}}_u$, the side information embedding $e^{\texttt{Target-SideEmbed}}$, and the target multimodal embedding $e^{\texttt{Target-MMEmbed}}$.

Stage I: Self-Interest Representation Extracting via Mamba. We first process the enriched sequence $\hat{\mathcal{H}}_u$ using a stacked Mamba architecture to capture temporal user preferences. Mamba is a recently proposed state-space sequence model (SSM) that efficiently captures long-range dependencies through implicit recurrence and selective gating.

Let $\hat{e}_{i_l}$ be the input at step l, and h_l be the hidden representation. Mamba updates its state via:

$$x_l = \text{LN}(\hat{e}_{i_l}),$$
$$\tilde{h}_l = \text{SSM}(x_l; \theta),$$
$$h_l = h_{l-1} + \tilde{h}_l,$$

where LN denotes layer normalization, and $\text{SSM}(\cdot)$ is the linear time recurrence operator of Mamba with parameters θ. The residual connection allows stable propagation of sequential context.

We obtain a sequence of hidden states $[h_1, h_2, \ldots, h_L]$, and then input sequence embedding to the Stage II module.

Stage II: Interactive-Interest Representation Extracting via Attention-Weighted GRU Block. To model user interests in the context of the current target item, we introduce a lightweight interaction mechanism that computes the relevance between each historical behavior and the current target. This interaction is followed by a gated sequence model to capture dynamic user preferences.

For each hidden state h_l (produced from Stage I), we compute a similarity score with the target embedding:

$$\alpha_l = \frac{\langle h_l, e^{\texttt{Target-MMEmbed}} \rangle}{\|h_l\| \cdot \|e^{\texttt{Target-MMEmbed}}\|},$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, and the denominator ensures cosine normalization. The resulting scalar α_l represents the interactive relevance between the l-th historical item and the target item.

We emphasize that this attention-like similarity mechanism is distinct from full self-attention used in Transformer models. Here, the computation involves only a single target embedding and a sequence of history vectors, resulting in a linear complexity of $O(L)$, where L is the sequence length. This design preserves the lightweight nature of our fully state-space architecture while capturing pairwise dependencies with the target.

Each hidden state is then reweighted:

$$z_l = \alpha_l \cdot h_l,$$

and the resulting attended sequence $[z_1, z_2, \ldots, z_L]$ is passed into a Gated Recurrent Unit (GRU) to model the time-sensitive evolution of target-aware preferences.

The GRU updates its state as follows:

$$\begin{aligned} r_l &= \sigma(W_r z_l + U_r g_{l-1}), \\ u_l &= \sigma(W_u z_l + U_u g_{l-1}), \\ \tilde{h}_l &= \tanh(W_h z_l + U_h(r_l \odot g_{l-1})), \\ g_l &= u_l \odot g_{l-1} + (1 - u_l) \odot \tilde{h}_l, \end{aligned}$$

where r_l, u_l are the reset and update gates, $\tilde{h}_l$ is the candidate state, and g_l is the updated output at step l. $\odot$ denotes element-wise multiplication.

Finally, the user's interactive interest representation is generated by concatenating the max-pooled features with the output from the final GRU timestep:

$$u^{\texttt{pooling}} = \texttt{MaxPool}([g_1, g_2, \ldots, g_L]).$$

$$u^{\texttt{Interactive}} = u^{\texttt{pooling}} \oplus g_L \in \mathbb{R}^{512}.$$

Prediction via Residual QNN.. The static and dynamic user representations are concatenated with the target item embedding to form the final prediction input:

$$z = u^{\texttt{Interactive}} \oplus e^{\texttt{Target-SideEmbed}} \oplus \hat{e}^{\texttt{Target-MMEmbed}} \in \mathbb{R}^{896}.$$

We then feed z into a residual Quantized Neural Network (QNN), which includes two residual blocks with non-linear transformations:

$$\begin{aligned} h^{(1)} &= \phi(W_1 z + b_1) + z, \\ h^{(2)} &= \phi(W_2 h^{(1)} + b_2) + h^{(1)}, \\ \hat{y}_{u,i^{\text{target}}} &= \sigma(W_3 h^{(2)} + b_3), \end{aligned}$$

where $\phi(\cdot)$ is an activation function (e.g., GELU), and $\sigma(\cdot)$ is the sigmoid function that outputs the final CTR probability.

The residual design improves gradient flow and robustness during training, while the quantized structure (optional during inference) ensures fast deployment with minimal latency.

Loss Function. The total loss function consists of the binary cross-entropy loss and a regularization term:

$$\mathcal{L} = \mathcal{L}_{bce} + \alpha \mathcal{L}_{reg}$$

where $\mathcal{L}_{bce}$ is the binary cross-entropy loss:

$$\mathcal{L}_{bce} = -\frac{1}{|\mathcal{D}_{tr}|} \sum_{i=1}^{|\mathcal{D}_{tr}|} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right],$$

$|\mathcal{D}_{tr}|$ is the number of samples in the training set, and $\mathcal{L}_{reg}$ is the regularization term introduced to prevent overfitting by penalizing large parameter values, defined as:

$$\begin{aligned} \mathcal{L}_{reg} = &\sum_{j \in \text{embeddings}} \sum_{(p,\lambda) \in \text{emb\_reg}} \frac{\lambda}{p} \|\theta_j\|_p^p \\ &+ \sum_{k \in \text{network}} \sum_{(p,\lambda) \in \text{net\_reg}} \frac{\lambda}{p} \|\theta_k\|_p^p, \end{aligned}$$

where θ_j represents embedding parameters, θ_k represents other network parameters, and (p, λ) denotes the norm order and regularization strength pairs for embedding and network parameters respectively. The hyperparameter λ controls the overall regularization strength.

4 EXPERIMENT

4.1 Datasets

Dataset Overview We employ the MicroLens-1M benchmark dataset from Westlake University [**?**], comprising:

- 1 million unique users
- 91,000 multimedia items
- Rich multimodal features (titles, cover images, metadata)

Data Partitioning The dataset is strategically divided to simulate real-world scenarios:

- **Training set**: 800K interactions (70%) - earliest chronological data
- **Validation set**: 150K interactions (15%) - intermediate period
- **Test set**: 150K interactions (15%) - most recent interactions

Challenge Protocol The MM-CTR Challenge adopts rigorous evaluation:

- Public data (training + validation) for model development
- Private holdout test set for final evaluation
- Temporal splitting prevents data leakage

4.2 Baselines

We compare our proposed MambaCTR with the following two groups of recommendation baselines, including (1) RNN-based baseline: **DIN** (2) Transformer-based baseline: **QIN**, **Transformer-DCN**.

4.3 Parameter Settings

Our model employs Mamba blocks with $d_{state} = 16$ and $d_{conv} = 4$ for sequence processing, combined with a 3-layer QNN for feature interactions. The training uses Adam optimizer (lr = 5×10^{-4}) with batch size 128 and dropout 0.2 for regularization. Other hyperparameters including the complete architecture details are specified in Table 4.

Table 1: SSM-based CTR Prediction Model Hyperparameters Settings

Component	Parameters
Base Configuration	
Batch size	128
Learning rate	5×10^{-4}
Embedding dimension	64
Dropout rate	0.2
Early stopping patience	5 epochs
Mamba Block	
input dim	256
SSM state	16
conv width	4
Expansion factor	2
Parameters/block	≈393K
QNN Components	
Input dimension	256
Number of layers	3
Number of rows	2
Training	
Optimizer	Adam
Loss function	Binary crossentropy
Metrics	[Logloss, AUC]

4.4 Evaluation Environments

All Experiment were conducted on a RTX3090 GPU.The version of CUDA and PyTorch are 11.8 and 2.3.1

4.5 Evaluation Results

In the MM-CTR task, both **prediction accuracy** and **computational time consumption** are crucial factors. The former determines the model quality in real-world scenarios and exhibits absolute correlation with business outcomes, while the latter governs model efficiency, being closely tied to computational costs and user experience. Therefore, we propose two distinct evaluation metrics based on these aspects to facilitate more comprehensive and reasonable comparisons.

Metric 1: AUC. Our model is evaluated using the area under the ROC curve (AUC) respectively on valid set and test set. AUC metrics are widely used in CTR prediction tasks. AUC measures the model's ability to distinguish between positive and negative samples. The higher the AUC, the better the model's performance.

Metric 2: Training & Inference Time. Besides, we also compare the training Time (s/epoch) Inference time (s/item) metrics. The less the Training & Inference Time, the higher the model's efficiency. Detailed evaluation is shown in **Table 2** and **Table 3**.

Table 2: Model Performance Comparison

Methods	AUC (Valid)	AUC (Test)
DIN	0.8655	0.8699
QIN	0.9701	0.9758
Transformer-DCN	0.9776	0.9839
MambaCTR (*origin*)	0.9605	0.9835
MambaCTR (*ablated*)	**0.9803**	**0.9865**

Table 3: Computational Efficiency Comparison

Methods	Training (s/epoch)	Inference (s/item)
DIN	**711**	0.036
QIN	1226	0.039
Transformer-DCN	1471	0.029
MambaCTR (*origin*)	739	**0.024**
MambaCTR (*ablated*)	1692	0.031

Compared with baseline methods and the ablated architecture with backbone replacement (detailed configurations in **Section 6** ablation studies), the original MambaCTR achieves significant efficiency improvements while exhibiting only marginal performance degradation relative to the SOTA variant (ablated MambaCTR). Notably, although MambaCTR requires slightly longer training time than DIN, the latter employs a lightweight architecture with substantially fewer parameters and significantly inferior performance. **Our comprehensive evaluation on both AUC and efficiency metrics demonstrates that MambaCTR effectively balances these two critical dimensions.**

5 ANALYSIS

We conduct **ablation** experiments for better understanding the contributions of different components in our proposed MambaCTR.

Specifically, our ablation study substitutes the Mamba block in MambaCTR with a vanilla Transformer block while preserving identical I/O structure. The former has linear complexity $O(n)$ while the latter has quadratic complexity $O(n^2)$.Detailed paramters of transformer block is shown in **Table 4**.

Table 4: Transformer-DCN Model Parameters Setting

Parameter	Value
Embedding dimension	64
Transformer layers	2
Attention heads	1
Attention dropout	0.1
Transformer dropout	0.2

Firstly, we validate the importance of multimodal representations for MM-CTR prediction by systematically ablating the multimodal embeddings component. Comparative experiments between architectures employing Mamba blocks versus Transformer blocks reveal that multimodal features consistently enhance performance for both frameworks (see **Table 5**).

These results demonstrate the effectiveness of our proposed MultiModal Embedding Encoder module across different architectural paradigms.

Table 5: Comparison of different models with and without multimodal embeddings on validation and test sets.

Model	w/mm_embed	AUC	Loss
Mamba	×	0.9672(1.685% ↓)	0.037882↑
	✓	**0.9835**	**0.035866**
Transformer	×	0.9756(1.117% ↓)	0.030327↑
	✓	**0.9865**	**0.029225**

Next, to further investigate the **efficiency** of our proposed backbone based on mamba-block[3], we design the following experiment. Keeping other components of the MambaCTR model unchanged, we replaced the backbone module responsible for modeling user interaction history from mamba block to transformer block.

Experimental results(see **Table 6** and **Table 7**) show that employing Mamba as the backbone leads to substantial improvements in both training and inference efficiency over Transformer-based counterparts.While considering efficiency, we also need to take into account the model's AUC and loss values when using different backbones for modeling user interaction history. Experimental results show that when using the Mamba block as the backbone, the drop in AUC is less than 0.5%, indicating that classification performance is only slightly affected despite the significant improvements in training and inference speed.

Table 6: Efficiency of different backbones.

Backbone	Training Time (s/epoch)	Inference time (s/item)
`Transformers[14]`	1692	0.031
`Mamba[3]`	**739(56.3% ↓)**	**0.024(22.6% ↓)**

Table 7: Accuracy of different backbones.

Backbone	AUC	Loss
`Transformers[14]`	**98.65**	**0.029225**
`Mamba[3]`	98.35(0.304% ↓)	0.033239(12.109% ↓)

6 SUMMARY

Although model ensemble is not allowed in the challenge, the superior performance could still be presented by the simple yet effective model architecture. FuxiCTR provides efficient configuration for tuning our model, saving us a lot of time.

Along with improving the embedding model, we also designed our MambaCTR model to preprocess the feature and make prediction.Performing much better than baseline in the scope of AUC and demonstrating a significantly faster approach for model training and inference.

REFERENCES

[1] Wazib Ansar, Saptarsi Goswami, and Amlan Chakrabarti. 2024. A Survey on Transformers in NLP with Focus on Efficiency. *arXiv preprint arXiv:2406.16893* (2024).

[2] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S Yu. 2022. Sequential recommendation via stochastic self-attention. In *Proceedings of the ACM web conference 2022*. 2036–2047.

[3] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 1725—1731.

[5] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. 2022. Vision gnn: An image is worth graph of nodes. *Advances in neural information processing systems* 35 (2022), 8291–8303.

[6] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development*

in Information Retrieval (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 355—364. https://doi.org/10.1145/3077136.3080777

[7] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 43—50. https://doi.org/10.1145/2959100.2959134

[8] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 539–548.

[9] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1754—1763. https://doi.org/10.1145/3219819.3220023

[10] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, et al. 2025. How can recommender systems benefit from large language models: A survey. *ACM Transactions on Information Systems* 43, 2 (2025), 1–47.

[11] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, USA, 995—1000. https://doi.org/10.1109/ICDM.2010.127

[12] Robin M. Schmidt. 2019. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. *CoRR* abs/1912.05911 (2019). arXiv:1912.05911 http://arxiv.org/abs/1912.05911

[13] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[15] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1785–1797. https://doi.org/10.1145/3442381.3450078

[16] Penghui Wei, Xuanhua Yang, Shaoguo Liu, Liang Wang, and Bo Zheng. 2022. Creater: ctr-driven advertising text generation with controlled pre-training and contrastive fine-tuning. *arXiv preprint arXiv:2205.08943* (2022).

[17] Junwei Xu, Zehao Zhao, Xiaoyu Hu, and Zhenjie Song. 2025. 1{st} Place Solution of WWW 2025 EReL@ MIR Workshop Multimodal CTR Prediction Challenge. *arXiv preprint arXiv:2505.03543* (2025).

[18] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1059–1068. https://doi.org/10.1145/3219819.3219823