# Deep Time-Stream Framework for Click-through Rate Prediction by Tracking Interest Evolution

**Shu-Ting Shi,**[1,2] **Wenhao Zheng,**[2] **Jun Tang,**[2] **Qing-Guo Chen,**[2] **Yao Hu,**[2] **Jianke Zhu,**[3] **Ming Li**[1*]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]YouKu Cognitive and Intelligent Lab, Alibaba Group, Hangzhou, China
[3]Zhejiang University, Hangzhou, China
shist@lamda.nju.edu.cn, {zwh149850, donald.tj, qingguo.cqg, yaoohu}@alibaba-inc.com,
jkzhu@zju.edu.cn, lim@nju.edu.cn

## Abstract

Click-through rate (CTR) prediction is an essential task in industrial applications such as video recommendation. Recently, deep learning models have been proposed to learn the representation of users' overall interests, while ignoring the fact that interests may dynamically change over time. We argue that it is necessary to consider the continuous-time information in CTR models to track user interest trend from rich historical behaviors. In this paper, we propose a novel Deep Time-Stream framework (DTS) which introduces the time information by an ordinary differential equations (ODE). DTS continuously models the evolution of interests using a neural network, and thus is able to tackle the challenge of dynamically representing users' interests based on their historical behaviors. In addition, our framework can be seamlessly applied to any existing deep CTR models by leveraging the additional Time-Stream Module, while no changes are made to the original CTR models. Experiments on public dataset as well as real industry dataset with billions of samples demonstrate the effectiveness of proposed approaches, which achieve superior performance compared with existing methods.

## Introduction

Click-through rate (CTR) prediction aims to estimate the probability of a user clicking on a given item, which has drawn increasing attention in the communities of academia and industry. In the example of a video website, a CTR algorithm is deployed to provide users with videos from thousands of different categories, and thus it is crucial to precisely capture users' interests so that they will keep using the website longer and bring more revenue to the website.

To accomplish this goal, the key problem is how to model user interest based on user historical clicks which reflects user preference. To extract representation of user's interests, many models have been proposed from traditional methodologies (Friedman 2001; Rendle 2010) to deep CTR models (Guo et al. 2017; Qu et al. 2016; Lian et al. 2018). Although these models achieve great success in modeling users' overall interests, they are ignorant of the dynamic changes in users' preferences. To pursue a more precise result, RNN-based methods (Wu et al. 2017; Hidasi and Tikk 2016; Zhou et al. 2019) have been proposed to capture the dependencies in user-item interaction sequences. However, these methods only consider the order of users' behaviors and ignore the *time interval* between behaviors which is the important information on predicting users' behaviors. As an example, in Figure 1, Mike usually watches videos about Donald Trump during the day while enjoys music videos of Taylor Swift at night, according to his behaviors' timestamps. Thus, regarding Mike's playlog only as a sequence of clicked videos would neglect changes of his latent interests over time. Unfortunately, existing CTR models do not have the capability to model the pattern on the continuous-time, since most of them are unaware to the time interval.

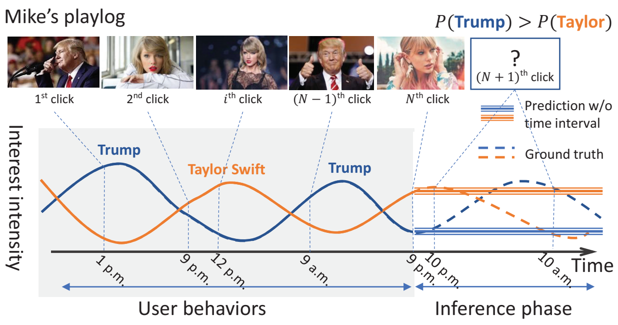

Figure 1: The user interests evolve on the time-stream. Only when considering intervals among user behaviors could the sequential pattern be captured.

Besides, at the inference phase, it is problematic that only predicting the *next* click without considering *when* the action will be performed. Incorporating the time of users' behaviors, *i.e.* modeling the effects of elapsed time interval between behaviors, is important in accurately modeling users' interest. For example, in Figure 1, if Mike watches a video of Taylor at 9 p.m., it more likely that he will watch another video of Taylor than Donald in a few hours, while the probability of watching videos of Donald should be significantly higher after half a day. However, traditional methods always

*Corresponding Author

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

get the exactly same prediction at any time.

Based on the aforementioned observations, we argue that it is crucial to consider time-stream information, *i.e.* continuous-time information, in CTR models. Therefore, we propose a novel Deep Time-Stream framework (DTS), which introduces the time-stream information into CTR model. Time-stream information could be formulated by ordinary differential equations (ODE), which refers to a function that describes the relationship between a dependent variable's derivative and the independent variable. Specifically, DTS leverages ODE to model the evolution of users' latent interests, by parameterizing the derivative of users' latent interests states with respect to time, such that the solution of ODE describes the dynamic evolution of users' interests. Moreover, DTS is equipped with the ability to unify users' historical behaviors (what have clicked) and target items (what will click) on the time-stream by clicks' timestamp, thus would make inference corresponding to the given *next time* and provide a more precises CTR prediction. To archieve the minimum model-altering cost, the ODE is packaged as a Time-Stream Module which can be applied in any deep CTR models. The contributions of this paper are summarized as follows:

- We propose a novel DTS framework that models users' latent interests evolution as an ODE, which significantly improves the expressive ability of models and can better capture the evolving characteristics of users' interests.
- DTS can generate users' feature at an arbitrary time and thus allows flexible as well as adaptive evaluations.
- The Time-Stream Module can be easily transplanted into existing CTR models without changing the original structure.

## Background

In machine learning, it is a crucial task to efficiently conduct a class of hypothesis, linear or nonlinear, that can represent the data patterns. *Ordinary Differential Equations (ODEs)* can also be used as a hypothesis. Considering the differential equation in $R^d$: $\frac{dz}{dt} = f(z,t),\ z(0) = z_0$, the solution of $z$ at time $t$ is denoted as $z(t)$. The basic idea behind the ODE approaches to supervised learning is to tune $f$ so that the map $z(t)$ can produce nonlinear function needed to fit the data.

In fact, (Chen et al. 2018) reveals that deep neural networks can be considered as discrete ODE, and their iterative updates can be regarded as an Euler discretization of a continuous transformation. On the other hand, neural ODEs are a family of deep neural network models that can be interpreted as a continuous equivalent of Residual Networks (ResNets) or Recurrent Neural Networks (RNNs). To see this, consider the transformation of a hidden state from a layer $t$ to $t+1$ in ResNets or RNNs:

$$h_{t+1} = h_t + f_t(h_t). \tag{1}$$

In ResNets, $h_t \in R^d$ is the hidden state at layer $t$ and $f_t : R^d \to R^d$ is some differentiable function which preserves the dimension of $h_t$. In RNNs, $h_t \in R^d$ is the hidden state at $t$-th RNN cell which update throw a function $f_t : R^d \to R^d$. The difference $h_{t+1} - h_t$ can be interpreted as a discretization of the derivative $h'(t)$ with timestep $\Delta t = 1$. Letting $\Delta t \to 0$, we see that the dynamically hidden state can be parameterized by an ODE: $\lim_{\Delta t \to 0} \frac{h_{t+\Delta t} - h_t}{\Delta t} = f(h,t)$.

This solution of $z(t)$ or $h(t)$ can be solved using an ODE solver with many sophisticated numerical methods to choose, e.g. linear multi-step methods, Runge-Kutta methods (Runge 1895; Kutta 1901) and adaptive time-stepping. Above methods could be helpful on deep learning, since they could adaptively choose the layers of network. It should be noted that here our concern is not the solver itself, but rather the representation of the data. So we refered to the solver as a black-box differential equation solver:

$$z_{t_1}, \cdots, z_{t_N} = ODEsolve(z_{t_0}, f, \theta_f, t_1, \cdots, t_N) \tag{2}$$

where $\theta_f$ is the parameters of $f$.

In next section, we show how the ODEs are utilized to model the dynamics of users' interest evolution, and how to make ODEs stable while training.

## The Deep Time-Stream Framework

In this section, we describe the details of our proposed model. We firstly formalize CTR as a binary classification problem. Given a sample of data $x = (x^U, x^V, x^P) \in \mathcal{X}$, where $(x^U, x^V, x^P)$ denotes the collection of the concatenate of different fields' one-hot vectors from *User behavior*, *target Video* and *user Profiles*, respectively. Moreover, each field contains a list of click behaviors, $x^U = [(v_1, c_1); (v_2, c_2); ...; (v_N, c_N)]$, in which $x_i^U = (v_i, c_i)$ denotes the video $v_i$ and corresponding category $c_i$ at the $i$-th behaviors that happens at time $t_i$, $N$ is the number of user's history behaviors; $x^V$ denotes the target video and its category $x^V = (v_{N+1}, c_{N+1})$, and the equation is established because the target video should happen as the $(N+1)$-th user click, the predicting time of this potential click is refers to as *next time* $t_{N+1}$. Thus, we unify the user historical behaviors and target video on the time-stream by the their timestamps denoted as $t$, $t = [t_1, t_2, \cdots, t_N, t_{N+1}]$. *User Profiles* $x^P$ contains useful profile information such as *gender*, *age* and so on. Label $y \in \mathcal{Y}$ indicates whether the user click the specific video, $y = 1$ means click while $y = 0$ means not. The goal of CTR is to learn a mapping $h \in \mathcal{H}$ from $\mathcal{X}$ to $\mathcal{Y}$, where $\mathcal{H}$ denotes the hypothesis space, $h : \mathcal{X} \mapsto \mathcal{Y}$ to predict whether the user will click the video. The prediction function $h$ can be learned by minimizing the following objective function:

$$\min_h \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{L}(h(x;t), y) \tag{3}$$

where $\mathcal{L}$ is the empirical loss which will be introduced in detail in following subsections.

### General Framework

Our proposed framework DTS could be regarded as BaseModel plus Time-Stream Module, as shown in Figure 2. BaseModel is referred to an existing deep CTR model such as DNN (Covington, Adams, and Sargin 2016), PNN (Qu et

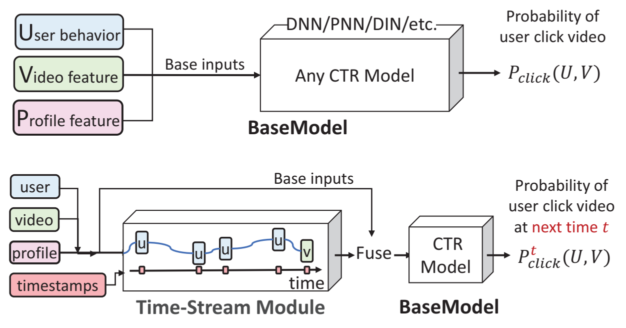

Figure 2: DTS = BaseModel + Time-Stream Module. The Time-Stream Module introduces continuous-time information and by a fuse operation the base inputs are enhanced that can be feed into basemodel to get a more precise result.

al. 2016) and DIN (Zhou et al. 2018b). Aside from the basemodel, Time-Stream Module collects the timestamps of all the events, including a user's historical click time in the past and the user's potential click time in the predicting moment. Note that the latter part are ignored in existing CTR models. Moreover, Time-Stream Module tracks the latent interest evolution by an ODE to calculate an enhanced input which introduces the continuous-time information while preserves the dimensions of base inputs. Thus, any deep CTR model can be used as the BaseModel in our DTS framework without changes made. Compared with the BaseModel that output an click probability on the event of user click item, DTS can improve the output by predicting the click probability on the event of user click item at given time.

In the following subsections, we would explain the structure of BaseModel, and introduce the Time-Stream Module that are used for capturing interests and then modeling interest evolution.

## BaseModel

Most deep CTR models are built on the basic structure of Embedding-Pooling-MLP. The basic structure is composed of several parts:

- **Embedding** Embedding layer is the common operation that transforms the sparse feature into low-dimensional dense feature. $x^U$, $x^V$ and $x^P$ are embedded as $(e_1, ..., e_N)$, $e_{N+1}$ and $e^P$, respectively. In the field of *User Profiles*, the sparse features embedded as $e^U$
- **Pooling** The embedding vectors are fed into pooling operation. $e^U = Pooling(e_1, ..., e_N)$ where $e^U$ refers to as user vector. The pooling can be sum pooling, average pooling, or specially designed attentive pooling(Zhou et al. 2018b).
- **Multilayer Perceptron (MLP)** All these pooled vectors from different categories are concatenated. Then, the concatenated vector is fed into a following MLP for final prediction.

**Target Loss** A widely used loss function in deep CTR models is negative log-likelihood function, which uses the label of target item to supervise overall prediction:

$$\mathcal{L}_{target} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log p(x_i) + (1-y_i)\log(1-p(x_i))) \tag{4}$$

where $x_i = (x_i^U, x_i^V, x_i^P) \in \mathcal{D}$, $\mathcal{D}$ is the training set of size $N$. $y_i \in \{0,1\}$ represents whether user clicks target item. $p(x)$ is the output of network, which is the predicted probability that the user clicks target item.

## Time-Stream Module

Users' interests are dynamic over time rather than static. BaseModel obtains a representation vector of user interest by a pooling operation over the clicked item feature but ignores the time information. The absence of dynamic pattern restricts the power of user behavior feature, which plays a key role in modeling user interests since the user clicked items are the expression of a user's interest at the corresponding time. For BaseModel, the lack of ability for modeling continuous pattern leads to the inaccuracy to model the dynamic user interest.

Is there an elegant way to represent a user's real-time interests and model the dynamic interest evolution pattern? The nature of continuous-time evolving inspires us to design a novel method named Time-Stream Framework that leveraging ODE to model the dynamic interest trace. ODE have been applied to a wide variety of fields such as physics, biology, chemistry, engineering and economics and if the ODE can be solved, given an initial point it is possible to determine all its future positions, a collection of points known as a trajectory or orbit. In this paper we novelly using ODEs as hypothesis class that the trajectory denotes a latent interest evolution trace. As mentioned in Eq 1, ODE can be a general form of RNNs and RNNs can be thought of as a discretization of the continuous ODE. There are several advantages with continuous ODE approach such as the flexible evaluations, which corresponds to choosing the RNN lengths adaptively. Moreover, we can also use advanced numerical methods for training, such as the multi-grid method or the parallel shooting method. Figure 3 illustrates the architecture of the Time-Stream Module.

In details, to represent the interest evolution by a latent trajectory of ODE, a differentiable function $f$ is used, $\frac{dz(t)}{dt} = f(z(t), t; \theta_f)$ denotes the interest evolution rate, where $\theta_f$ is the parameters of $f$. Thus, given a initial state $z_{t_0}$ the trajectory of ODE can be solved using a solver mentioned in Equation 2:

$$\begin{aligned} z_{t_1}, \cdots, z_{t_N}, z_{t_{N+1}} \\ = ODEsolve(z_{t_0}, f, \theta_f, t_1, \cdots, t_N, t_{N+1}), \end{aligned} \tag{5}$$

in which, where $z_{t_1}, \cdots, z_{t_N}, z_{t_{N+1}}$ is the solution of ODE which describes the latent state of dynamics $f$ at each observation time $t_1, \cdots, t_N, t_{N+1}$. Since similar people are more likely to have similar interest evolution pattern, we construct a mapping $g$ that transforms the user profile embedding $e^P$ to the latent time-stream space to obtain the initial value: $z_{t_0} = g(e^P; \theta_g)$, the mapping $g$ is a linear transformations

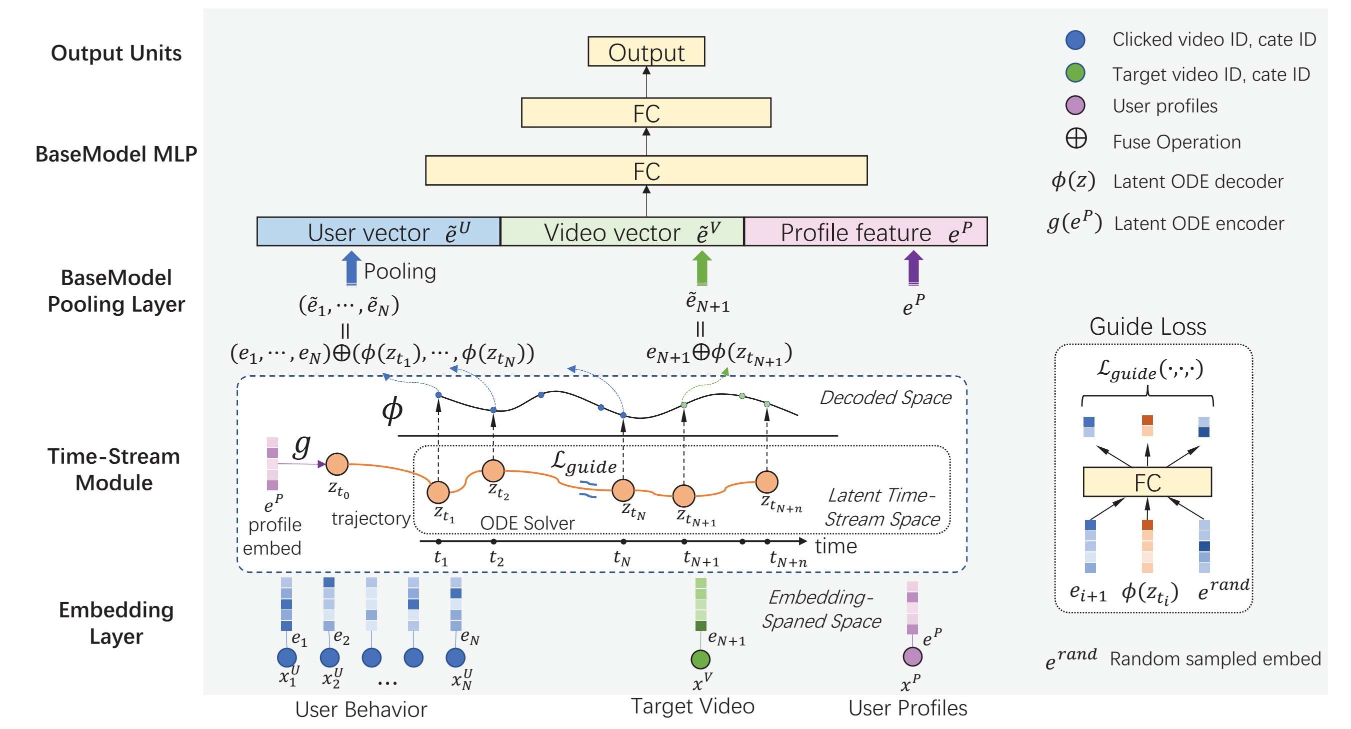
Output Units
BaseModel MLP
BaseModel Pooling Layer
Time-Stream Module
Embedding Layer
Output
FC
FC
User vector $\tilde{e}^U$
Video vector $\tilde{e}^V$
Profile feature $e^P$
Pooling
$(\tilde{e}_1, \cdots, \tilde{e}_N)$
$(e_1, \cdots, e_N) \oplus (\phi(z_{t_1}), \cdots, \phi(z_{t_N}))$
$\tilde{e}_{N+1}$
$e_{N+1} \oplus \phi(z_{t_{N+1}})$
$e^P$
Decoded Space
Latent Time-Stream Space
ODE Solver
trajectory
profile embed
time
Embedding-Spaned Space
User Behavior
Target Video
User Profiles
Clicked video ID, cate ID
Target video ID, cate ID
User profiles
Fuse Operation
$\phi(z)$ Latent ODE decoder
$g(e^P)$ Latent ODE encoder
Guide Loss
$\mathcal{L}_{guide}(\cdot,\cdot,\cdot)$
$e_{i+1}$ $\phi(z_{t_i})$ $e^{rand}$
$e^{rand}$ Random sampled embed

Figure 3: The structure of Time-Sream Module. DTS keeps the basic framework of BaseModel thus inheriting their proven performance. Moreover, DTS extends Time-Sream module that the latent time state $z_t$ is modeled as an ODE. Decoder $\phi$ mapping $z_t$ to embedded space fused with embedding to enhance the quality of embedding. Guide loss is designed to help the hidden state converge

with parameter $\theta_g$ and serves as an encoder from profile embedding space to the latent time-stream space.

On the other hand, $\phi$ is the decoder that transform latent time-stream feature $z_{t_i}$ to the video embedding-spaned space. $\phi(z_{t_i}; \theta_\phi)$ is the adjustment or supplementary of behavior feature which carries additional behavioral evolving patterns. For the adjustment of user behavior feature, we have: $\widetilde{e}_i = e_i + \phi(z_{t_i}; \theta_\phi)$. where $i = 1, 2, ..., N$. The fuse operation can be set as other operation such as concatenation, but in this work the add operation are used which keeps the adjustment and original feature equally contribute. For the target video feature, we have $\widetilde{e}^V = e_{N+1} + \phi(z_{t_{N+1}}; \theta_\phi)$.

The enriched behavior feature $\widetilde{e}^U = (\widetilde{e}_1, \widetilde{e}_2, ..., \widetilde{e}_N)$, video vector $\widetilde{e}^V$ and profile feature $e^P$ was then send into the rest part of Base CTR model.

Using ODEs as a generative model allows us to make predictions for arbitrary time, whether in the past or in the future since the on timeline are continuous. The output of ODE can be computed by a black-box differential equation solver, which evaluates the hidden unit dynamics wherever necessary to determine the solution with the desired accuracy.

**The choice of function** $f$ The latent function $f$ needs to be specified and different types of functions can be used to meet different requirements. Next, we will introduce several approaches that leverage different type of ODE function $f$ to model the process of interest evolution.

- **Simple form.** This is the simplest form of function $f$ under the assumption that $f$ is the function of independent variable $t$:

$$f(z,t) = \frac{dz}{dt} = A(t),\ z(t) = \int_{t_0}^{t} A(\lambda) d\lambda + C, \quad (6)$$

where $A$ is a control function, and $C$ is a constant that can be solved given a initial state. This type of problem may have an analytical solution that by computing $z(t)$ directly. If so, there is no extra cost needed to solve the ODE numerically. A special case is linear differential equation with constant coefficients $f(z,t) = A(t) = \alpha$ which means the latent state discount at rate $\alpha$. Thus $z_{t_i} = \alpha(t_i - t_0) + z_{t_0}$ holds for all $t$. The insight is that the monotonous trajectory of $f$ mimic the characteristic of user interest: is mainly influenced by recent interests, so one should reduce the impact of early interest and increase the impact of the user's recent behavior. This special case is extremely simple but achieves surprising performance shown in experiment.

- **Complex form.** The aforementioned simple form of $f$ cannot express user's diverse time-series pattern. To overcome this limitation, another choice is to parameterize the derivative of the dynamics $f$ using a neural network which improves the expressive ability of model greatly. In this paper, a two layer neural network with sigmoid activation unit is used: $f(z) = \sigma(w_2 \cdot \sigma(w_1 \cdot z + b_1) + b_2)$, where $w_1, w_2, b_1, b_2$ are the linear parameters and $\sigma(.)$ is the activate unit. It is hard to obtain an analytical solution in this form of $f$. The solution on $z_{t_1}, \cdots, z_{t_N}, z_{t_{N+1}}$ is computed using a numerical ODE solver mentioned in the Background.

**Guide Loss** The aforementioned functions can be solved on a single call to ODE toolbox and modern ODE solvers provide guarantees on the growth of approximation error. However, we have several concerns: 1) When the form of function becoming complicated, the behavior of ODE may encounter situations of explodes, converges to stationary states or exhibits chaotic behavior. This may explain some of the difficulties, e.g., the vanishing and explosion of gradients encountered in the training of deep neural networks. 2) On the other hand, since the click behavior of target item is triggered by users' interest evolution, the label only indicates the last of click behavior $z_{t_{N+1}}$, while history state $z_t$ can not obtain proper supervision.

To alleviate these problems, we propose *guide loss*, which uses behavior embedding $e_i$ to supervise the learning of latent functions. To do this, inspired by the loss of Word2Vec (Mikolov et al. 2013), we build a small network that push the decoded hidden state $\phi(z_{t_i})$ more close to the next behavior $e_{i+1}$ than a random negative sampled instance $e^{rand}$. Guide loss can be formulated as:

$$\mathcal{L}_{guide}(p, v, n) = -\frac{1}{N}\sum_i (v_i \cdot p_i + v_i \cdot n_i - \log(\frac{v_i \cdot p_i}{v_i \cdot n_i})),$$
$$p_i = FC(e_{i+1}),\ v_i = FC(\phi(z_{t_i})),\ n_i = FC(e^{rand}).$$

where $FC(x)$ is a fully connected layer with PRelu as activation. The overall loss in our model is:

$$\mathcal{L} = \mathcal{L}_{target} + \lambda \mathcal{L}_{guide} \tag{7}$$

where $\mathcal{L}$ is the overall loss function, $\mathcal{L}_{target}$ is introduced in Eqution 4 and $\lambda$ is the hyper-parameter which balances the interest representation and CTR prediction.

Overall, the introduction of guide loss has several advantages: 1) from the aspect of interest learning, the introduction of guide loss helps each hidden state of ODE represent interest expressively. 2) As for the optimization of ODE, guide loss reduces the difficulty of backpropagation when ODE models long history behavior sequence. 3) Guide loss gives more semantic information for the learning of the embedding layer, which leads to a better embedding matrix.

**Training and inference** At the training phase, our model is equipped with the ability to reload the parameters of the BaseModel. Then all the weights are finetuned to get a quick convergence. We would achieve a *safe-start* by initializing the parameters of $f$ and initial value to zeros, such that the trajectory of ODE is a constant of zero. Thus, at the start of training, the overall model stay the same as the original CTR base model.

At the inference phase, we could predict the user interest evolution at the arbitrary recommendation time $t_{N+1}$, since we leverage ODE solver to integrate the function $f$ at next time $t_{N+1}$. In industrial, DTS would be efficient: When predicting multiple CTR at $t_{N+1}, t_{N+2}$ and $t_{N+n}$, there is no need to compute the hidden trajectory from scratch. It is easy to integrate the function $f$ from $t_N$ to $t_{N+n}$ that is cheap for computation.

## Experiments

To evaluate the effectiveness of the Deep Time-Stream framework, we conduct experiments on public datasets and industrial dataset. In these experiments, the proposed Time-Stream Module is applied on multiple BaseModels. Note that DTS is a framwork that inheriting the base CTR model then extending the time-stream module. Therefore the effectiveness of DTS should be reflected in the improvement compared with BaseModel. The comparison metrics will be introduced below.

### Datasets and Experimental Settings

We use both public and industrial datasets to verify the effect of the Time-Stream Module.

**Public Dataset** Amazon Dataset contains product reviews and metadata from Amazon, which is used as benchmark dataset in many works (He and McAuley 2016; McAuley et al. 2015). We conduct experiments on a subset named *Electronic*, which contains 192,403 users, 63,001 goods, 801 categories and 1,689,188 samples. For *User Behavior*, features include user-reviewed goods_id list, category_id list, and corresponding timestamp list. For *Target Item*, features include target goods_id, category_id and the next time that denote when this click prediction is made. Dataset collects user behaviors that happens at time $t_1, t_2, \cdots, t_N$, where $N$ is the number of users history behaviors. DTS can naturally handle different N, which corresponds to choosing the RNN lengths adaptively. Since some baselines contain RNN, for all dataset, the max N is set as 100 to keep the comparison fair. The task is to predict the $(k+1)$-th reviewed goods by making use of the first $k$ reviewed goods. Training dataset is generated with $k = 1, 2, \cdots, N-2$ for each user. In the test set, we predict the $N$-th good given the first $N-1$ reviewed goods.

**Industrial Dataset** The industrial dataset is constructed by user playlog and profile information from a video platform. Similar to the public dataset, we collect features including video_id, cate_id, user-watched video_id list and cate_id list. Overall, 1.7 billion samples has been collect including 1.4 million users, 6.3 million videos, 278604 categories. For *Profile features*, user profiles such as gender, age, activity score are used. Training dataset is generated with $k = 1, 2, \cdots, N-2$ for each user. In the test set, we predict the $N$-th video given the first $N-1$ watched videos.

**Compared Methods** We set BaseModels as some mainstream CTR prediction methods to evaluate the effectiveness of the Time-Stream framework. The BaseModels are used as:

- **DNN** (Covington, Adams, and Sargin 2016) DNN takes the setting of Embedding&Pooling&MLP and sum pooling operation was used to integrate behavior embeddings.
- **Wide&Deep** (Cheng et al. 2016) Wide&Deep consists of two parts: its deep model is the same as DNN, and its wide model is a linear model.
- **PNN** (Qu et al. 2016) PNN uses a product layer to capture interactive patterns between interfield categories.
- **DIN** (Zhou et al. 2018b) DIN uses the mechanism of attention to activate related user behaviors, which can be regarded as an attentive Pooling.
- **DIEN** (Zhou et al. 2019) DIEN uses GRU with attentional update gate to model the user interest pattern.

**Metrics** In CTR prediction field, AUC is a widely used metric (Fawcett 2006). It measures the items of order by ranking all these with predicted CTR, including intra-user and inter-user orders. A variation of user weighted AUC is introduced in (Zhu et al. 2017; He and McAuley 2016) which measures the items of intra-user order by averaging AUC over users and is shown to be more relevant to online performance on CTR prediction. We adopt this metric in our experiments. For simplicity, we still refer to it as AUC. Although there are other metrics are widely used in recommender system such as MRR@k or Recall@k, but CTR task with AUC metrics is our prior concern, i.e., $k = 1$. Since in industrial video recommendation, there are some key positions that requires high CTR, for example, the first recommended video take over most of the user attention and straight directly impact user retention. The metric is calculated as follows:

$$AUC = \frac{\sum_{i=1}^{n} \#impression_i \times AUC_i}{\sum_{i=1}^{n} \#impression_i}$$

where $n$ is the number of users, $\#impression_i$ and $AUC_i$ are the number of impressions and AUC corresponding to the $i$-th user. Besides, $RelaImpr$ metric is used to measure relative improvement over models, and $RelaImpr$ is defined as below:

$$RelaImpr = (\frac{AUC(measured\ model)}{AUC(base\ model)} - 1) \times 100\%$$

**Experiment Settings** The embedding size of video and category are both 18, which then been concated as an embedding of 36. The dimension of user profile embedding is 36. Mapping $g$ is a linear transformation with transparent matrix of size $36 \times 36$, and mapping $\phi$ is a two fully connect layers with size 72 and 36. The dimension of FC in guide loss is set to 18. $\lambda$ is set to 0.5. The Runge-Kutta methods is used as ODE solver. We follow the setting of BaseModels as their suggest. We train DTS on a GTX 1080ti for 5 epochs, with batch size set to 128.

## Results on Public Dataset

Remind that *Deep Time-Stream framwork = BaseModel + Time-Stream Module*. Thus, the effectiveness of the improvement brought by the Time-Stream Module should be verified.

Table 1: Model Comparison with AUC on Amazon Dataset.

| | BaseModel | DTS | RelaImpr |
|---|---|---|---|
| DNN | 0.7686 | 0.7789 | 1.34% |
| PNN | 0.7799 | 0.8304 | 6.48% |
| Wide&Deep | 0.7735 | 0.8390 | 8.47% |
| DIN | 0.7880 | 0.8508 | 7.97% |
| DIEN | 0.8453 | 0.8981 | 6.25% |

As shown in Table 1, there exists some facts that: (1) our proposed DTS clearly outperforms all the raw model on five BaseModel, which confirms the capacity of our model in learning impact of time. (2) for the BaseModels, PNN could capture interactive patterns between inter-field categories, which beats DNN and Wide&Deep. However, above three BaseModels use average pooling to compress user features into a fixed-length vector, which brings a difficulty to capture user's diverse interests effectively from rich historical behaviors. Moreover, DTS could generate the better representation of user behaviors by considering time-stream information, and achieves improvements up to 8.47%, which beats the performance of raw DIN. (3) DIN with Time-Stream Module outperforms DIEN. DIN leverages attention mechanism and improves the expressive ability of the model greatly. The follow-up work DIEN based on DIN further tries to capture the interest evolving process. Compared with DIEN, DIN with Time-Stream Module considers continuous time-stream information, and it could help CTR model to learn more powerful user representation compared with previous works.

## Results on Industrial Dataset

We further conduct experiments on the dataset of the real short video platform. In practice, the max length of history behaviors is set as 100.

Table 2: Model Comparison with AUC on Industrial Dataset.

| | BaseModel | DTS | RelaImpr |
|---|---|---|---|
| DNN | 0.6385 | 0.6628 | 3.81% |
| PNN | 0.6601 | 0.6763 | 2.45% |
| Wide&Deep | 0.6478 | 0.7010 | 8.21% |
| DIN | 0.7008 | 0.7268 | 3.72% |
| DIEN | 0.7023 | 0.7412 | 5.54% |

As shown in Table 2, our Time-Stream framework could improve all of the BasedModels. The BaseModels of DNN, PNN, and Wide&Deep are widely used in industry and build on large scale distributed system, and the change of model would make a great effort. Our DTS could easily apply on these model with no changes made to the original architecture, and it brings at least 3% improvement. It suggests that our DTS has great value of practical application. Similar to Amazon Dataset, DIN with Time-Stream Module outperforms raw DIEN, which confirms the capacity of our model.

Table 3: Ablation studies with AUC on the industrial dataset. “w” is the short for “with” and “w/o” is the short for “without”.

| | BaseModel | w/o adaptive step (RNN) | w simple form | w/o guide loss | DTS |
|---|---|---|---|---|---|
| DNN | 0.6385 | 0.6532 | 0.6389 | 0.6441 | **0.6628** |
| PNN | 0.6601 | 0.6703 | 0.6721 | **0.7095** | 0.6763 |
| Wide&Deep | 0.6478 | 0.6948 | 0.6802 | 0.7007 | **0.701** |
| DIN | 0.7008 | 0.7002 | 0.7012 | 0.7096 | **0.7268** |
| DIEN | 0.7023 | 0.7021 | 0.7045 | 0.7116 | **0.7412** |

### Model analysis

In this subsection, we will show the effect of adaptive step, function form and guide loss, respectively.

**Effect of adaptive step** To verify whether adaptive step helps to construct better representation, we conduct ablation study on fixing the step of hidden dynamics to demonstrate the effectiveness of adaptive step. From Table 3, without adaptive step would perform worse than origin. The DTS with fixed step is equivalent with RNNs, when $\Delta t_i \equiv t_{i+1} - t_i, i = 1, 2, ..., N$ are all constants. The time interval are not considered by RNN. Thus, compared with fixed step, adaptive step could evaluate the step of $f$ whenever neccessary. Besides, it could handle incorporate data which arrives at arbitrary time. Hence, our DTS could learn more accuracy users’ interests, such that achieving better performance.

**Effect of function form** When we use simple form of function $f$ discussed in General Framework, the performance is better than BaseModel as shown in Table 3. However, the improvement is still limited compares with the complex form that has more powerful expression.

**Effect of guide loss** Moreover, we further explore the effect of guide loss. It uses non-click items as negative instances for enhancing discrimination. As shown in Table 3, guide loss brings great improvements most of all BaseModels, which reflects the importance of supervision information when learning the representation of latent user interest.

## Related Work

By virtue of the strong ability of deep learning on feature presentation and combination recent CTR models transform from traditional linear or nonlinear models(Friedman 2001; Rendle 2010) to deep models. Most deep models follow the basic paradigm of Embedding, Pooling and Multi-layer Perceptron (MLP) (Covington, Adams, and Sargin 2016). Based on this paradigm, many models pay attention to the interaction between features: Wide&Deep (Cheng et al. 2016) combines low-order and high-order features to improve the power of expression; PNN (Qu et al. 2016) proposes a product layer to capture interactive patterns between interfiled categories. DIN (Zhou et al. 2018b) introduces the mechanism of attention to activate the user historical behaviors w.r.t. given target item locally, and captures the diversity characteristic of user interests successfully.

Beyond that, several methods (Wu et al. 2017; Hidasi et al. 2015; Yuan et al. 2019) are proposed for capturing dynamic information, since user behaviors are usually dynamic instead isolated. The dynamic information also could be regarded as a kind of context information in recommendation system, which is distinguishable from features describing the underlying activity undertaken by the user within the context (Dourish 2004). Moreover, the sequential models usually get the better performance on capturing dynamic information. These works regard user-item interactions as a sequence and try to represent sequential user behaviors. (Yu et al. 2016) uses the structure of the recurrent neural network (RNN) to investigate the dynamic representation of each user and the global sequential behaviors of item purchase history. Some methods use RNNs to capture dynamic information in recommendation system, e.g. RRN (Wu et al. 2017), GRU4REC (Hidasi et al. 2015), NextItNet (Yuan et al. 2019). (Zhou et al. 2018a) uses an attention-based sequential framework to model heterogeneous behaviors. DIEN (Zhou et al. 2019) leverages GRU and designs an attentional update gate (AUGRU) to model the dependency between behaviors . Although improving the performance compared to non-sequential approaches, these RNN based methods still are no enough to represent the user interest evolution without considering time-stream information.

Recent years, some studies (Chen et al. 2018; Weinan, Han, and Li 2019) go further to explore the possibility of producing nonlinear functions using continuous ODEs, pushing the discrete approach to an infinitesimal limit. They introduce the numerical differential equations to the design of deep neural network. (Weinan, Han, and Li 2019) shows many effective networks, such as ResNet, PolyNet, FractalNet and RevNet, can be interpreted as different numerical discretizations of differential equations. Compared with deep neural networks, there are several advantages with a continuous approach, including to flexible choose the number of evaluations on recurrent networks, well-studied and computationally-cheap numerical ODE solvers. However, to the best of our knowledge, there not exists previous works that leveraging ODEs to represent user interest evolution on CTR model.

## Conclusions

In this paper we propose a novel Time-Stream framework, that adaptively mines the users’ continuously evolving interests from rich historical behavior, by leveraging neural ODE that parameterizes the derivative of the hidden state using a neural network. Unlike recurrent neural networks, which require discretizing observation and emission intervals, continuously-defined dynamics can naturally incorporate data which arrives at arbitrary time. We also propose guide loss to control the error of ODE solver. Extensive experiments show that our model can generate a more precise user feature at an arbitrary time.

## Acknowledgments

This research was partially supported by National Key Research and Development Program (2017YFB1001903) and Alibaba Group through Alibaba Innovative Research Program.

## References

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in neural information processing systems,* 6571–6583.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, 191–198.

Dourish, P. 2004. What we talk about when we talk about context. *Personal and ubiquitous computing* 8(1):19–30.

Fawcett, T. 2006. An introduction to roc analysis. *Pattern recognition letters* 27(8):861–874.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* 1189–1232.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence.*, 2782–2788.

He, R., and McAuley, J. 2016. 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*, 507–517.

Hidasi, B., and Tikk, D. 2016. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery* 30(2):342–371.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Kutta, W. 1901. Beitrag zur näherungsweisen integration totaler differentialgleichungen. *Zeitschrift für Mathematik und Physik* 46:435–453.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 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*, 1754–1763.

McAuley, J.; Targett, C.; Shi, Q.; and Van Den Hengel, A. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 43–52.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems,* 3111–3119.

Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *Proceddings of the 16th International Conference on Data Mining*, 1149–1154.

Rendle, S. 2010. Factorization machines. In *IEEE International Conference on Data Mining*, 995–1000.

Runge, C. 1895. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen* 46(2):167–178.

Weinan, E.; Han, J.; and Li, Q. 2019. A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences* 6(1):10.

Wu, C.-Y.; Ahmed, A.; Beutel, A.; Smola, A. J.; and Jing, H. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM international conference on web search and data mining*, 495–503.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International conference on Research and Development in Information Retrieval*, 729–732.

Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J. M.; and He, X. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 582–590. ACM.

Zhou, C.; Bai, J.; Song, J.; Liu, X.; Zhao, Z.; Chen, X.; and Gao, J. 2018a. Atrank: An attention-based user behavior modeling framework for recommendation. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence*, 4564–4571.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018b. Deep interest network for click-through rate prediction. In *Proceedings of the 24th International Conference on Knowledge Discovery & Data Mining*, 1059–1068.

Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5941–5948.

Zhu, H.; Jin, J.; Tan, C.; Pan, F.; Zeng, Y.; Li, H.; and Gai, K. 2017. Optimized cost per click in taobao display advertising. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2191–2200.