Generalize for Future: Slow and Fast Trajectory Learning for CTR Prediction

Jian Zhu, Congcong Liu*,
Xue Jiang, Changping Peng, Zhangang Lin, Jingping Shao
JD.COM
lbjbx@zju.edu.cn, cliubh@connect.ust.hk, (jiangxue,pengchangping,linzhangang,shaojingping)@jd.com

Abstract

Deep neural networks (DNNs) have achieved significant advancements in click-through rate (CTR) prediction by demonstrating strong generalization on training data. However, in real-world scenarios, the assumption of independent and identically distributed (i.i.d.) conditions, which is fundamental to this problem, is often violated due to temporal distribution shifts. This violation can lead to suboptimal model performance when optimizing empirical risk without access to future data, resulting in overfitting on the training data and convergence to a single sharp minimum. To address this challenge, we propose a novel model updating framework called Slow and Fast Trajectory Learning (SFTL) network. SFTL aims to mitigate the discrepancy between past and future domains while quickly adapting to recent changes in small temporal drifts. This mechanism entails two interactions among three complementary learners: (i) the Working Learner, which updates model parameters using modern optimizers (e.g., Adam, Adagrad) and serves as the primary learner in the recommendation system, (ii) the Slow Learner, which is updated in each temporal domain by directly assigning the model weights of the working learner, and (iii) the Fast Learner, which is updated in each iteration by assigning exponentially moving average weights of the working learner. Additionally, we propose a novel rank-based trajectory loss to facilitate interaction between the working learner and trajectory learner, aiming to adapt to temporal drift and enhance performance in the current domain compared to the past. We provide theoretical understanding and conduct extensive experiments on real-world CTR prediction datasets to validate the effectiveness and efficiency of SFTL in terms of both convergence speed and model performance. The results demonstrate the superiority of SFTL over existing approaches.

Introduction

CTR prediction plays a crucial role in both research and industries, particularly in sectors such as recommendation systems and online advertising. Significant efforts have been made to develop better prediction models, with recent success achieved by deep neural networks due to their impressive capabilities in discovering personal behavior interests and complex feature interactions (Zhu et al. 2020; Liu et al. 2022a; Ivchenko et al. 2022; Liu et al. 2022b; Guo et al. 2017; Song et al. 2019; Zhu et al. 2023; Liu et al. 2023b; Zhu et al. 2021; Liu et al. 2023a). However, most of these studies primarily focus on the batch learning setting, assuming independent and identically distributed (i.i.d.) data. In this setting, neural models are trained on the entire dataset with the assumption that the input-output relationship remains constant throughout. However, this assumption is often unrealistic in real-world applications where data arrives continuously, and the input-output relationship can change over time (Gama et al. 2014).

For instance, consider a scenario where a user searches for the term "drink" in a search bar. Depending on the time of day, their intent to click on specific items may vary, such as choosing milk in the morning and cola in the afternoon from the list of available options. Another common case is when users repeatedly browse the same item over time, their interest in clicking on the item for further details may decline. These examples demonstrate that the underlying data distribution of personal interests and contextual information gradually drifts over time. In such cases, re-training the model from scratch can be time-consuming. Consequently, in practice, an alternative approach known as "online learning" or "online fine-tuning" (Rendle and Schmidt-Thieme 2008) is deployed to fine-tune the neural model using recent data.

Despite the success of online fine-tuning in real-world recommendation systems, its effectiveness relies on the assumption that real-time user behavior reflects the changing trend of personal interests. Recent learning theory has shown that optimization methods for model parameters, such as Adam (Kingma and Ba 2014) and Adagrad (Duchi, Hazan, and Singer 2011), can quickly adapt on test data with training data, even if they come from different domains (Ben-David et al. 2010). However, generalizing deep prediction models to future scenarios still remain challenging for several reasons. Firstly, there is the issue of convergence difficulty in online recommendation systems. Naively fine-tuning deep neural networks on online data streams requires a substantial number of samples and iteration to converge, which degrades real-time performance. Although recent work (Zhang et al. 2022) has shown that training models for just one epoch performs better than multiple epochs on sparse CTR datasets, it remains unclear if such models can cope when the distribution drifts. Secondly, there is the challenge of effectively

*. For inquiries, please contact Jian Zhu and Congcong Liu.

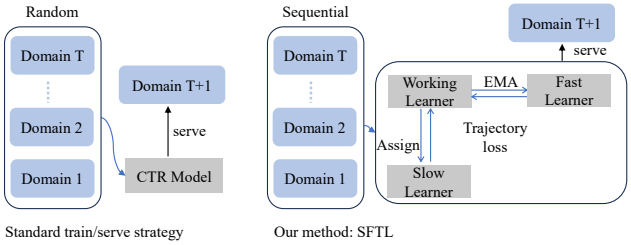

Figure 1: Overview of the SFTL framework. Compared with standard supervised learning strategy that take all of training data of different temporal domains, SFTL sequentially trains the backbone working learner and independently updates slow learner and fast learner. To enhance generalization of future domain, SFTL uses the trajectory loss to measure ranking score between working learner and trajectory learner.

modeling the gradual evolution of data distribution and its impact on the model. Despite the implicit representation provided by temporal information (e.g., day, hour), the future distribution can be drastically different at various timestamps. Ideally, if we could access the data distribution in the near future, simply performing gradient descent with this data could mitigate the domain generalization gap between recent and future data. Thirdly, there is the need for efficient temporal expressiveness for model evolution and its impact on the system. In real-world applications, a cyclic training-and-serving strategy is commonly used for the main traffic, involving complex deep neural networks. However, recent solutions (Mi, Lin, and Faltings 2020; Zhang et al. 2020; Peng et al. 2021) that attempt to dynamically evolve parameter values with recurrent and convolution networks, respectively, are often time-consuming and unsuitable for complexity-constrained conditions. Therefore, it is essential to develop an efficient model that can track model evolution.

In this paper, we aim to address these challenges regarding the temporal generalization failure of recent deep neural networks in CTR prediction. To achieve this, our training algorithm should be robust to different temporal distributions and enable the decision boundary to extrapolate well when facing temporal drifts. To this end, we introduce a novel and efficient learning strategy called Slow and Fast Trajectory Learning (SFTL) strategy that enhances the deep network via utilizing the historical models' outputs. Our intuition is that, directly optimizing for future is impossible without future data while looking backward past trajectories (e.g. changing trend of model outputs) may instruct how to evolve towards future domain, thus we need to focus on those trajectories during learning. In practice, for each outputs of current inputs, we only consider the recent models' outputs from last temporal domain or last iteration that is mostly likely to reflect the temporal drift in different extent.

The key of our approach is two novel and independent neural network modules for ctr prediction named slow trajectory learner (STL) and fast trajectory learner (FTL). These two modules update themselves by utilizing model weights from past training trajectory. As illustrated in Figure 1, for every temporal domain in a sequential training strategy, the slow trajectory learner updates model weights by directly assigning parameter values from working learner, which stores past knowledge. For each training iteration, the fast trajectory learner updates model weights by exponentially moving averaging model parameter values from all of training trajectories, which stores the recent temporal evolution. Then we propose and apply a trajectory loss to improve the model performance of working learner by aligning and promoting the working learner's outputs with those of STL and FTL respectively. It measures the surrogate ranking loss for metric scores. Though we additionally introduce two networks, the training computation is efficient due to the only gradient computation of working learner. To fairly compare our methods with other temporal generalization methods or continual learning methods, we only use the outputs of trajectory learner for validation which showcase both the efficiency and effectiveness of trajectory learning.

In summary, our work makes the following contributions:

- We formulate the learning of CTR prediction models as a temporal generalization problem with gradual distribution shift. To overcome this, we propose a novel training framework, slow and fast trajectory learning (SFTL), for end-to-end adaptation to concept drift. This method allows for robust feature learning of the main network by maximizing future performance in the next-time domain, as well as efficient learning of the trajectory network through the averaging of historical parameter values.
- In our experiments, we propose using sequential training to adaptive learn the temporal drift instead of randomly shuffling the entire dataset. Additionally, we provide a theoretical understanding of the generalization gap in our proposed method. Through extensive experiments on real-world CTR datasets, we demonstrate the efficacy and superiority of our training framework.

Related Work

Temporal Domain Generalization Recent researches on domain generalization (Wang et al. 2022) aim to mitigating the domain gap by representation learning and data manipulation methods. In this area, the problem assumes that the training domain is different to test domain (i.e. image texture) while the label space is same. However, the discrete domain gap does not meet the real-world setting where domain shifts temporally. CIDA (Wang, He, and Katabi 2020) proposes to learning domain invariant feature for each temporal domain via a minimax optimization. GI (Nasery et al. 2021) propose an adversarial gradient loss to regularize the loss curvature within specific time window. This loss encourages a model to capture shared information between different timestamps. DRAIN (Bai, Ling, and Zhao 2023) proposes a drift-aware dynamic neural network where network weights are sequentially generated by a recurrent network with previous model parameters. Despite their success, none of these work are validated on large-scale datasets and more complex data distribution (e.g. recommendation system). To alleviate generalization gap, recent researchers (Foret et al. 2020; Izmailov et al. 2018) try to seek a robust and flat minima

as a generalization baseline. *Sharpness-aware minimization* (SAM)(Foret et al. 2020) proposes a novel training procedure by simultaneously minimizing the loss value and loss sharpness for the aim to lower down the training loss around the neighborhood of model weight θ. SWA (Izmailov et al. 2018) introduces a method by averaging model weight every several epochs which directly generate a flatter minima.

Continual Learning Continual learning is widely researched for alleviating the problem named as catastrophic forgetting. However, majority of those work on continual learning mainly focus on *backward transferring* (Lopez-Paz and Ranzato 2017; Buzzega et al. 2020; Rolnick et al. 2019; Chaudhry et al. 2019; Parisi et al. 2019) which may hinder the deployment on recommendation system. Recently, continual learning for CTR prediction has become a significant research problem (Mi, Lin, and Faltings 2020; Zhang et al. 2020; Peng et al. 2021; Cai et al. 2022). ADER (Mi, Lin, and Faltings 2020) employs a exemplar replay buffer with knowledge distillation loss to prevent *catastrophic forgetting*. SML (Zhang et al. 2020) and ASMG (Peng et al. 2021) both propose weight generation technique to improve incremental learning performance by MLP and RNN respectively. However, the large computation complexity hinder their deployment on real-world application. CR (Zhu et al. 2023) considers the difference between offline retraining and online serving and then develops the practical surrogate losses for maximizing the offline model performance towards online deployment. However, none of these works have addressed the problem of maximizing future performance.

Proposed Framework

Preliminaries and Problem Formulation

From the perspective of real-world application, the ctr prediction pipeline can be decomposed into two training stages:

- **Offline training**: given old dataset $\mathcal{D}_{old} \triangleq \{\mathcal{D}_i\}_1^T$ that consists of continuous data from *T* temporal domains with corresponding input sample x and ground truth labels y, a machine learning model f with parameters θ is trained only once (Zhang et al. 2022) followed by online serving due to the computation complexity. We note the output logits with $z \triangleq h(x;\theta)$ and the corresponding probability with $f(x;\theta) \triangleq \text{sigmoid}(h(x;\theta))$.
- **Online Fine-tuning**: In this stage, it requires online data (Rendle and Schmidt-Thieme 2008) or batch data (Zhu et al. 2023) in a small time window Δ (e.g. Δ=10 minutes) for serving next-time traffic. As we get new arriving data $\mathcal{D}_{new}$, we retrain previous deployed model f_{old} to overcome the challenge of inconsistency between $\mathcal{D}_{old}$ and $\mathcal{D}_{new}$.

The standard solution to optimize f on the $\mathcal{D}$ is to minimize the average loss on training examples:

$$\min_{\theta} \mathcal{L}_{\mathcal{D}}(\theta) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, f(x;\theta))]. \tag{1}$$

Benefited from mitigation on distributional drift, the strategy of online fine-tuning with recent data can efficiently improve the generalization on the next serving stage (Zhu et al. 2023; Rendle and Schmidt-Thieme 2008). In real-world setting, we only care about the future performance on test data $\mathcal{D}_{new}$. Formally, the goal of CTR prediction is to find a model which minimize both $\mathcal{L}_{\mathcal{D}_{old}}$ and $\mathcal{L}_{\mathcal{D}_{new}}$ by only minimizing an empirical risk $\mathcal{L}_{\mathcal{D}_{old}}$ over past training data. From the perspective of generalization into future, CTR prediction can be seen as a special problem of temporal domain generalization which is ignored in previous works (Guo et al. 2017; Lian et al. 2018).

Ideally, our goal is to promote future performance by maximizing the performance of current model on future data which can be defined as:

$$\min_{\theta} \sum_{i=1}^{T} \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell(y, f(x;\theta))] \tag{2}$$

$$\text{s.t.} \quad \theta = \arg\min_{\theta} \mathcal{L}_{\mathcal{D}_{T+1}}(f(\theta)) - \mathcal{L}_{\mathcal{D}_{T+1}}(f(\theta^*)) \tag{3}$$

where θ and θ^* refer to model parameters trained on datasets $\{\mathcal{D}_1, ..., \mathcal{D}_T\}$ and best optimized on $\mathcal{D}_{T+1}$ respectively. The objective assumes that we can directly acquire test-time domain data with optimal parameter, which force θ over-fitting on $\mathcal{D}_{T+1}$. This naive training scheme forces the neural model to learn towards future-aware patterns, although we never acquire test-domain data and cannot calculate the optimal decision boundary.

A Bayes Perspective for CTR Prediction

From the perspective of real-world system, a CTR prediction model is trained over all temporal domains with a cyclic setting using ERM which can be seen as a state-space model or recursive Bayes as described in Definition 1.

Definition 1 *Given sequential domains $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{T+1}$ and a initial model θ_0, the latent weights and domain dynamics on a cyclic train-and-serve setting is assumed to evolve as: $\theta_t \sim p(\theta_t|\theta_{t-1}, t)$ and $\mathcal{D}_t \sim p(\mathcal{D}|\theta_t)$. Through using a Bayes' theorem, a state-space model can be defined as:*

$$p(\theta_t|\mathcal{D}_{<t}) = \int p(\theta_t|\theta_{t-1}, t) p(\theta_{t-1}|\mathcal{D}_{<t}) \mathrm{d}\theta_{t-1} \tag{4}$$

Proposition 1 *Given model functions $f_1, f_2, \ldots, f_T$ and its respective training domain in Definition 1. For any domain i and j, $\forall j > i$, we have :*

$$|\epsilon_i(h) - \epsilon_T(h)| \leq |\epsilon_j(h) - \epsilon_T(h)| \tag{5}$$

where h is a hypothesis and $\epsilon_s(h, f_s) \triangleq \mathbb{E}_{x\sim\mathcal{D}_s}[|h(x) - f(x)|]$ for short $\epsilon_s(h)$,

From the perspective of domain adaption theory (Blitzer et al. 2007), we cannot directly estimate the difference of divergence between target domain and multiple source domains. The key observation allowing for our proposed method to facilitate the learning of each temporal domain in the Bayes framework is the co-evolving situation where *the parameter trained on older temporal domain perform worse than recent one* (Proposition 1). We show the empirical evidence and theoretical proof in the appendix.

A reasonable proposal to generalize better on future is learning sequentially with the goal of maximizing the performance on next training domain (Proposition 2).

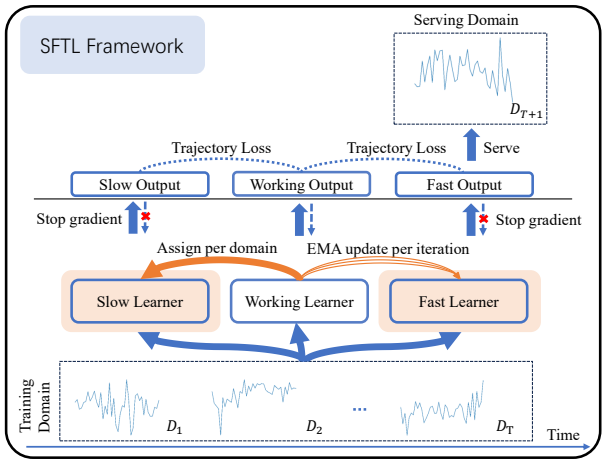

Figure 2: Illustration of slow and fast trajectory learning framework.

Proposition 2 *(Informal) Given sequential training and test domain $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_t$ and $\mathcal{D}_{t+1}$. For any two predictors $f: \mathbb{X} \to \mathbb{R}^1$ with different weights θ_i and θ_j , we have:*

$$\mathbb{E}[(\mathcal{R}_{\mathcal{D}_i}(\theta_i) - \mathcal{R}_{\mathcal{D}_{t+1}}(\theta^*))^2] \leq \mathbb{E}[\mathbb{V}(\mathcal{R}_{\mathcal{D}_j}(\theta_j) - \mathcal{R}_{\mathcal{D}_{t+1}}(\theta^*))] + \mathcal{O}(\mathbb{E}[||\mathcal{R}_{\mathcal{D}_j}(\theta_j) - \mathcal{R}_{\mathcal{D}_{t+1}}(\theta^*)||])^2 \tag{6}$$

where θ_i and θ_j are trained on $\mathcal{D}_i$ and $\mathcal{D}_j$ and $i < j$, θ^* is the posterior estimate.

Trajectory Learning Mechanism

In this section, we bridge the gap between theoretical understanding and practical algorithm by introducing a novel slow and fast trajectory learning mechanism into deep ctr prediction model, enabling effective and efficient generalization. Specifically, the proposed model consists of three neural components: a working learner f, two trajectory learner: slow and fast trajectory learner sharing same model architecture as working learner, which are parameterized by θ_w, θ_s and θ_f, respectively. The slow learner maintains the historical semantic knowledge of the encountered domains while fast learner evolves towards recent decision boundary with maintaining long-term knowledge. The working learner is interacted in order to generalize toward future domain by aligning and promoting its decision boundary with the trajectory learner. SFTL's overall design is illustrated in Fig. 2.

Slow Trajectory Learner The central idea of slow trajectory learner (STL) is to maintain historical information which accumulate and consolidate information over past temporal domains. The main difference between the slow learner and the working learner is that the slow learner is never updated by gradient decent methods instead of a direct weights assignment of the working model as it sequentially learns on different temporal domains:

$$\theta_s \longleftarrow \theta_w^t \tag{7}$$

where θ_w^t is the parameter values sequentially trained on $\mathcal{D}_{<t}$. The slow trajectory learner is updated only when temporal domain drift (i.e. from Domain 1 to Domain 2).

Fast Trajectory Learner In the above assumption of slow trajectory learner, gradual temporal drift occurs in each temporal domain. However, in a real-world setting, the temporal domain can be considered a short-term time window that is more volatile. Therefore, learning short-term temporal drift requires a robust training scheme. We proposes a continuous model updating mechanism by introducing a fast trajectory learner (FTL). Unlike STL, FTL updates the weights of the fast learner using exponential moving average (EMA) weights of the working learner:

$$\theta_f \longleftarrow \alpha\theta_f + (1-\alpha)\theta_w \tag{8}$$

Where α is the update coefficient, θ_w is the model weights of the working learner. When $\alpha \to 1$, the fast learner becomes a replica of the working learner. When $\alpha \to 0$, the fast learner becomes a replica of the slow learner. The coefficient controls the adaption capacity of the fast learner.

Trajectory Loss As we want to maximize the metric performance of our working learner, we further design a trajectory loss to force working learner to perform better than the trajectory learner. To improve the bipartite ranking performance of CTR prediction, we expect our optimization objective measures the difference of pairwise metric scores whose surrogate function can be defined as $\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}[(f_{\theta_w}(x_i^+) - f_{\theta_w}(x_j^-)) > (f_{\theta_{tra}}(x_i^+) - f_{\theta_{tra}}(x_j^-))]$ where x_i^+ and x_j^- are i-th positive sample and j-th negative sample. Thus, our proposed trajectory loss can be defined as:

$$\mathcal{L}_{tra} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} log(1 + exp^{-(u-v)}) \tag{9}$$

where u and v are $f_{\theta_w}(x_i^+) - f_{\theta_w}(x_j^-)$ and $f_{\theta_{tra}}(x_i^+) - f_{\theta_{tra}}(x_j^-)$ respectively, θ_w and θ_{tra} denote the model weights of the working learner and trajectory learner. We only calculate the gradients of the working learner.

Theorem 1 ***(Bias-Variance bound for trajectory loss)*** *Pick any convex loss ℓ. Suppose we have a teacher model p^t with corresponding empirical pairwise risk $\widehat{R}(f) = \frac{1}{N}\frac{1}{M}\sum_{n=0}^{N}\sum_{m=0}^{M}\ell(d[f(x_n^+), f(x_m^-)], d[f_t(x_n^+), f_t(x_m^-)])$ and population risk $R(f) = \mathbb{E}_x\left[\ell(d[f(x_n^+), f(x_m^-)])\right]$ where $f_t(x)$ is the teacher output and x^+, x^- are positive and negative samples respectively. For any predictor $f: \mathcal{X} \to \mathbb{R}^L$,*

$$\mathbb{E}\left[(\widehat{R}(f) - R(f))^2\right] \leq \mathbb{E}\left[(R(f_t))^2\right] \tag{10}$$

The fidelity of the trajectory risk only depends on one factor: how well the teacher model estimates approximate the true pairwise disagreement in a logistic sense. The theorem 1 can be proved by Jensen's inequality. We have stated a statistical perspective on trajectory loss, resting on the observation that trajectory learning offers a bound that always approximates Bayes probabilities based on the performance of the teacher model. Although it may be loose and unstable for real-world applications, this qualitative bound can still hold the majority of conditions in practice.

Algorithm 1: Training procedure of SFTL

Input: The working learner f_{θ_w} with weights θ_w; the slow learner f_{θ_s} with weights θ_s; the fast learner f_{θ_f} with weights θ_f; Training domains $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$; Epochs E; Fast alpha α; SFTL start domain/epoch s_{start}; Trajectory loss coefficient λ_s and λ_f; .
$\theta_s \leftarrow 0, \theta_f \leftarrow 0, s \leftarrow 0$
for e = 1 to E **do** ▷ E = 1 for one-pass learning
  for t = 1 to T **do**
    $s = t$ if E = 1 else e
    while $\mathcal{D}_t$ is not completed **do**
      sample mini-batch $\mathcal{D}_t^m$ from $\mathcal{D}_t$
      Compute $\mathcal{L}^{ce}_{\mathcal{D}_t^m}(y, f(x;\theta_w))$
      $\mathcal{L}_{tra} = 0$
      if $s > s_{start}$ **then** computing trajectory loss
        $\mathcal{L}^f_{tra} = \mathcal{L}^s_{\mathcal{D}_t^m}(f(x;\theta_w), f(x;\theta_s))$
        $\mathcal{L}^s_{tra} = \mathcal{L}^f_{\mathcal{D}_t^m}(f(x;\theta_w), f(x;\theta_f))$
        $\mathcal{L}_{tra} = \lambda_s \mathcal{L}^s_{tra} + \lambda_f \mathcal{L}^f_{tra}$
      end if
      $\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{tra}$
      $\theta_w \leftarrow \theta_w - \eta \nabla_{\theta_w} \mathcal{L}$ ▷ WL updating
      $\theta_f \leftarrow \alpha\theta_f + (1-\alpha)\theta_w$ ▷ FTL updating
    end while
    $\theta_s \leftarrow \theta_w$ ▷ STL updating
  end for
end for

Datasets	Fields	Feature size	Instances
Avazu	22	2018012	40428967
Taobao	18	1760849	26557961
CIKM2019	9	7248478	58751493

Table 1: The statistic of CTR prediction datasets

Formulation SFTL involves training a working learner f_{θ_w} on a sequential data stream from $\mathcal{D}_1$ to from $\mathcal{D}_t$ with a non-iid distribution. We first warm up the working learner and trajectory learner without computing trajectory loss for s_{start} previous temporal domains (epochs for standard supervised learning). After that, we use the logit outputs of STL and FTL to compute trajectory loss. Hence, the working learner is updated with a combination of the binary cross-entropy loss and the trajectory loss,

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_s \mathcal{L}^s_{tra} + \lambda_f \mathcal{L}^f_{tra} \tag{11}$$

We demonstrate more training details in Algorithm 1.

For inference, we use fast trajectory learner as it capture more short-term temporal information, which shows effective and efficient representation for temporal generalization.

Experiments

Experimental Setting

Datasets. We explore a wide range of click-through-rate prediction datasets. Avazu[1] is a display recommendation dataset released on Kaggle that contain 40428967 samples with 22 feature fields. Taobao[2] collects users' click data from a 8-days real-world traffic platform. Cikm2019[3] contains 62 million instances for purchase prediction. We construct public datasets by split into training/validation/test set by timestamp where the samples of last day is set for testing and penultimate day's data is set for validation and others for training. We summarize the statistic on Table 1. For more details, we refer readers to the appendix.

Baselines. (1) **Feature-Interaction** methods: Firstly, we consider the feature-interaction architectures that capature temporal information only from timestamps feature: WideDeep (Cheng et al. 2016), PNN (Qu et al. 2016), DCN (Wang et al. 2017), DeepFM (Guo et al. 2017), xDeepFM (Lian et al. 2018), DCN-Mix (Wang et al. 2020), AutoInt (Song et al. 2019). (2) **Temporal Domain Generalization**: We investigate a suite baselines of temporal generalization and flatness-aware generalization methods: CIDA (Wang, He, and Katabi 2020), GI (Nasery et al. 2021); SWA (Izmailov et al. 2018), SAM (Foret et al. 2020). (3) **Incremental Learning** methods: In the setting of incremental learning, we include **experience-replay** methods: ADER (Mi, Lin, and Faltings 2020) further enhanced by DER (Buzzega et al. 2020) in our implementation and **dynamic architecture** for forward transferring: SML (Zhang et al. 2020) and ASMG (Peng et al. 2021) that continuously generate model weights with historical parameters based on convolution network and recurrent network. The SML and ASMG can be also seen as variants of DRAIN (Bai, Ling, and Zhao 2023) that employs a dynamic neural network for temporal domain generalization.

Implementation Details. To evaluate our proposed method, we followed the experimental setup of (Zhu et al. 2023) by conducting two various phase, i.e. one-pass continual learning and standard supervised training. The details of configuration are summarized as follows. **Standard supervised phase:** To imitate the offline retraining stage, all of our baselines are trained on training set and early stop until model converges on validation set. In this setting, we evaluate the performance of feature-interaction baselines and temporal domain generalization baselines. **Continual learning phase:** To imitate the online industrial system, we limit the data access only once due to the computation efficiency. Meanwhile, we sequentially train baselines along time and evaluate results on the final day for comparing the generalization power which is different to the goal of vanilla CL. In this setting, we adopt DCN-Mix as backbone of our method.

For all of methods, we share the same embedding rank size (i.e. 16), MLP network (i.e. 1024-512-256), weight decay ratio (i.e. 1e-5) and use the Adam optimizer (Kingma and Ba 2014) to optimize the network. We search the best hyper-parameters (e.g. batch size, layers of explicit feature-interaction network, steps of adversarial training for GI and etc.) for all of methods on variant datasets. We refer readers to the appendix for more details due to the limited space.

1. https://www.kaggle.com/c/avazu-ctr-prediction
2. https://tianchi.aliyun.com/dataset/56
3. https://goo.su/H5HlB

Method	Avazu		Taobao		CIKM2019	
	AUC_{std}	Imp(%)	AUC_{std}	Imp(%)	AUC_{std}	Imp(%)
LR	$0.7422_{(0.0002)}$	↓0.73%	$0.5963_{(0.0001)}$	↓4.14%	$0.7308_{(0.0000)}$	↓3.58%
WideDeep	$0.7471_{(0.0017)}$	↓0.24%	$0.6353_{(0.0003)}$	↓0.24%	$0.7649_{(0.0014)}$	↓0.17%
DeepFM	$0.7481_{(0.0018)}$	↓0.14%	$0.6350_{(0.0004)}$	↓0.27%	$0.7659_{(0.0007)}$	↓0.22%
DCN	$0.7490_{(0.0009)}$	↓0.05%	$0.6372_{(0.0002)}$	↓0.05%	$0.7661_{(0.0009)}$	↓0.05%
PNN	$0.7498_{(0.0009)}$	↑0.03%	$0.6371_{(0.0003)}$	↓0.06%	$0.7656_{(0.0005)}$	↓0.10%
xDeepFM	$0.7461_{(0.0013)}$	↓0.34%	$0.6346_{(0.0005)}$	↓0.31%	$0.7645_{(0.0011)}$	↓0.21%
AutoInt++	$0.7495_{(0.0017)}$	↑0.00%	$0.6311_{(0.0024)}$	↓0.66%	$0.7624_{(0.0021)}$	↓0.42%
DCN-Mix	$0.7495_{(0.0005)}$	↑0.00%	$0.6377_{(0.0003)}$	↑0.00%	$0.7666_{(0.0009)}$	↑0.00%
SAM	$0.7517_{(0.0004)}$	↑0.22%	$0.6377_{(0.0003)}$	↑0.00%	$0.7681_{(0.0003)}$	↑0.15%
SWA	$0.7512_{(0.0008)}$	↑0.17%	$0.6378_{(0.0005)}$	↑0.01%	$0.7678_{(0.0005)}$	↑0.12%
CIDA	$0.7502_{(0.0017)}$	↑0.07%	$0.6371_{(0.0004)}$	↓0.06%	$0.7654_{(0.0009)}$	↓0.12%
GI	$0.7508_{(0.0004)}$	↑0.13%	$0.6372_{(0.0002)}$	↓0.05%	$0.7618_{(0.0021)}$	↓0.48%
SFTL	$0.7561_{(0.0006)}$	↑0.56%	$0.6388_{(0.0003)}$	↑0.11%	$0.7833_{(0.0005)}$	↑1.67%

Table 2: Comparison of final-day performance for Taobao, Avazu and Cikm2019 dataset on the standard supervised setting.

Method	Avazu		Taobao		CIKM2019	
	AUC_{std}	Imp(%)	AUC_{std}	Imp(%)	AUC_{std}	Imp(%)
IncFinetune	$0.7437_{(0.0012)}$	↑0.00%	$0.6206_{(0.0005)}$	↑0.00%	$0.7503_{(0.0003)}$	↑0.00%
SAM	$0.7443_{(0.0004)}$	↑0.06%	$0.6241_{(0.0013)}$	↑0.35%	$0.7508_{(0.0004)}$	↑0.05%
CIDA	$0.7442_{(0.0017)}$	↑0.05%	$0.6182_{(0.0014)}$	↓0.24%	$0.7490_{(0.0007)}$	↓0.13%
GI	$0.7451_{(0.0005)}$	↑0.22%	$0.6225_{(0.0012)}$	↑0.19%	$0.7497_{(0.0005)}$	↓0.06%
ADER++	$0.7468_{(0.0016)}$	↑0.29%	$0.6208_{(0.0011)}$	↑0.02%	$0.7517_{(0.0001)}$	↑0.14%
SML_{mlp}	$0.7415_{(0.0023)}$	↓0.22%	$0.6177_{(0.0009)}$	↓0.29%	$0.7492_{(0.0005)}$	↓0.11%
ASMG_{mlp}	$0.7461_{(0.0005)}$	↑0.24%	$0.6240_{(0.0006)}$	↑0.34%	$0.7497_{(0.0004)}$	↓0.06%
SFTL	$0.7510_{(0.0005)}$	↑0.73%	$0.6320_{(0.0002)}$	↑1.14%	$0.7633_{(0.0004)}$	↑1.30%

Table 3: Comparison of final-day performance on the one-pass continual learning setting.

Evaluation Metrics We used Area Under ROC (AUC) as evaluation metrics. It is widely used for recommendation and advertising. In addition, we use *Imp* metric to measure absolute improvement over models. For ctr prediction, the improvement over 0.1% is significant on large-scale datasets (Wang et al. 2020; Wang et al. 2017; Lian et al. 2018).

Results on Supervised Learning Benchmarks

Table 2 reports the final-day performance of AUC and improvement for deep feature-interaction models and temporal generalization methods. The reported numbers are averaged over three runs. On all benchmarks, we sequentially train models along time with multiple epochs. Firstly, we observe that all of feature-interaction models are strong competitors without explicitly modeling temporal drift. However, such feature-interaction still cannot work well under gradual temporal domain drift. To better compare our proposed method with other baselines, we adopt DCN-Mix as backbone network. We find flatness-aware generalization methods can outperform the baselines on all of the datasets while the drift-aware methods only work on Avazu. We conjecture that in real-world application, personal interest varies in an unknown trend and range over time zones of various length which exacerbates the difficulty of modeling. The recent proposed temporal generalization methods are only validated on small-scale dataset with explicit domain drift. On the other hand, Our proposed SFTL shows promising results on all datasets and outperforms all of competing baselines in most cases. Moreover, the significant improvements on the Cikm dataset indicate SFTL's ability to quickly adapt to the non-stationary environment even without enough positive labels.

Results on One-Pass CL Benchmarks

In real-world ctr prediction scenarios, training deep prediction networks with multiple epochs is unacceptable due to large complexity and prone to over-fitting. Therefore, a practical recommendation system should be able to improve its temporal generalization without accessing same instance twice. Table 3 reports the evaluation metrics on three public datasets, where we omit other feature-interaction methods' performance on the one-pass continual learning setting since we adopt the DCN-Mix as our backbone network. We observe that memory-enhanced ADER++ and sharpness-aware minimization can work well on all datasets compared to the

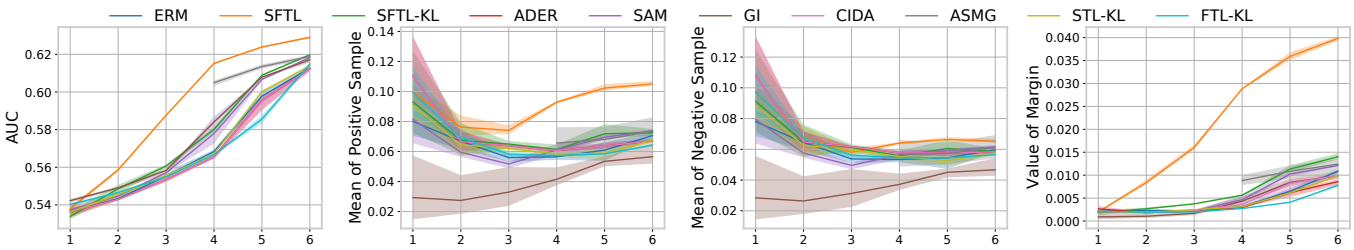

Figure 3: Domain Auc, prediction mean of negative and positive samples, sample margin on Taobao.

Method	Taobao	CIKM2019
Baseline	$0.6206_{(0.0005)}$	$0.7503_{(0.0003)}$
STL_{kl}	$0.6225_{(0.0008)}$	$0.7561_{(0.0006)}$
STL_{tra}	$0.6310_{(0.0003)}$	$0.7628_{(0.0003)}$
FTL_{kl}	$0.6243_{(0.0005)}$	$0.7610_{(0.0003)}$
FTL_{tra}	$0.6317_{(0.0002)}$	$0.7627_{(0.0004)}$

Table 4: Ablation study of different trajectory learner and loss function on one-pass continual learning.

other baseline. Similar to the results of supervised learning benchmark, we find the baselines of temporal domain generalization perform worse on Cikm2019 dataset. We conjecture that the temporal drift is hard to model due to the sparse positive label in this dataset. In addition, we also observe the dynamic architecture (SML, ASMG) even cannot compete with the incremental fintune baseline. We conjecture that reason has two fold: First, the main benefit in the original implementation mainly from the feature embedding while it can only work on dataset with small feature numbers due to the issue of out-of-memory; Second, ASMG and SML use coordinate-wise weight generation technique that can only be used in small MLP network which limits generalization capacity. On the other hand, our SFTL architecture can efficiently and effectively leverage the long-term and short-term historical knowledge to improve its performance and outperforms other baselines. This result demonstrate SFTL's potential to work in a real-world recommendation system.

Qualitative and Sensitivity Analysis

Effectiveness of trajectory learner and loss function We now study the effect of different trajectory learner and loss function to the final performance of SFTL.

- **Loss function**: To make the working learner aware of the past trajectory, we apply two variant loss to force the working learner modeling the temporal dynamics. KL-divergence regularizes the working learner's outputs via trajectory learner's for preventing large deviation between past domain and future domain while bipartite ranking loss forces the working learner scoring better than trajectory learner in different metrics. We report the results of different loss function in Table 4. We find both loss functions on different trajectory learner achieve significant improvement over incremental fine-tuning baseline. However, bipartite ranking loss always get better performance.
- **Trajectory Learner:** We independently investigate the model performance of different trajectory learner. In Table 4, we find both learner can outperform the baseline. However, the FTL consistently yields better performance due to the flat minima brought by self-ensemble model update strategy. Overall, this result shows that our SFTL's design is general enough and can work well on the combination of different temporal knowledge.

Methods	ERM	SAM	SFTL	ASMG	SML	GI
Train	1.0x	1.5x	2.1x	>8x	>6x	1.8x
Serve	1.0x	1.0x	1.0x	1.0x	1.0x	1.5x

Table 5: Comparison of training speed on last domain with same backbone whose MLP hidden layers are 1024-512-256.

Convergence Visualization In Figure 3, we show how AUC, sample margin, and prediction means the value of positive and negative samples vary over different temporal domain. The our proposed SFTL outperforms all the other methods by a large margin. We can observe SFTL both decrease the negative mean and increases the positive mean leading to the best bipartite ranking performance and fastest converge speed among all baseline methods in Table 3.

Train/Serve speed In Table 5, we compare the training and serve speed of different methods. Despite the additional two network, the training speed of our method is only twice that of the baseline method while the dynamic architectures suffer from computation complexity.

Conclusion

This paper identify the key challenges of temporal domain generalization in CTR prediction across continuously temporal domains propose a novel model training strategy to address it with a method that can achieve strong performance without directly access future data. SFTL introduces the trajectory-based learning to CTR prediction and propose two novel neural modules to fast adapt on concept and domain drift with different extents, successfully improving the capacity of forward transferring. The resulting method significantly outperforms the previous SOTA on temporal domain generalization problems of CTR prediction.

References

Bai, G.; Ling, C.; and Zhao, L. 2023. Temporal Domain Generalization with Drift-Aware Dynamic Neural Networks. In *The Eleventh International Conference on Learning Representations*.

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine learning*, 79: 151–175.

Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Wortman, J. 2007. Learning bounds for domain adaptation. *Advances in neural information processing systems*, 20.

Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D.; and Calderara, S. 2020. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33: 15920–15930.

Cai, G.; Zhu, J.; Dai, Q.; Dong, Z.; He, X.; Tang, R.; and Zhang, R. 2022. ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems. *arXiv preprint arXiv:2204.11165*.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H.; and Ranzato, M. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

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.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

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

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*.

Ivchenko, D.; Van Der Staay, D.; Taylor, C.; Liu, X.; Feng, W.; Kindi, R.; Sudarshan, A.; and Sefati, S. 2022. TorchRec: a PyTorch Domain Library for Recommendation Systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*, 482–483.

Izmailov, P.; Podoprikhin, D.; Garipov, T.; Vetrov, D.; and Wilson, A. G. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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.

Liu, C.; Li, Y.; Zhao, X.; Peng, C.; Lin, Z.; and Shao, J. 2022a. Concept Drift Adaptation for CTR Prediction in Online Advertising Systems. *arXiv preprint arXiv:2204.05101*.

Liu, C.; Li, Y.; Zhu, J.; Teng, F.; Zhao, X.; Peng, C.; Lin, Z.; and Shao, J. 2022b. Position Awareness Modeling with Knowledge Distillation for CTR Prediction. In *Proceedings of the 16th ACM Conference on Recommender Systems*, 562–566.

Liu, C.; Shi, L.; Wang, P.; Teng, F.; Jiang, X.; Peng, C.; Lin, Z.; and Shao, J. 2023a. Loss Harmonizing for Multi-Scenario CTR Prediction. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 195–199.

Liu, C.; Teng, F.; Zhao, X.; Lin, Z.; Hu, J.; and Shao, J. 2023b. Always Strengthen Your Strengths: A Drift-Aware Incremental Learning Framework for CTR Prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, 1806–1810. New York, NY, USA: Association for Computing Machinery. ISBN 9781450394086.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Mi, F.; Lin, X.; and Faltings, B. 2020. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *ACM Conference on Recommender Systems*, 408–413.

Nasery, A.; Thakur, S.; Piratla, V.; De, A.; and Sarawagi, S. 2021. Training for the future: A simple gradient interpolation loss to generalize along time. *Advances in Neural Information Processing Systems*, 34: 19198–19209.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113: 54–71.

Peng, D.; Pan, S. J.; Zhang, J.; and Zeng, A. 2021. Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems. *Proceedings of the 15th ACM Conference on Recommender Systems*.

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 *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 1149–1154.

Rendle, S.; and Schmidt-Thieme, L. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, 251–258.

Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T.; and Wayne, G. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1161–1170.

Wang, H.; He, H.; and Katabi, D. 2020. Continuously indexed domain adaptation. *arXiv preprint arXiv:2007.01807*.

Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Qin, T.; Lu, W.; Chen, Y.; Zeng, W.; and Yu, P. 2022. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*.

Wang, R.; Fu, B.; Fu, G.; and Wang, M. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*, 12.

Wang, R.; Shivanna, R.; Cheng, D. Z.; Jain, S.; Lin, D.; Hong, L.; and Chi, E. H. 2020. DCN-M: Improved Deep & Cross Network for Feature Cross Learning in Web-scale Learning to Rank Systems. *arXiv preprint arXiv:2008.13535*.

Zhang, Y.; Feng, F.; Wang, C.; He, X.; Wang, M.; Li, Y.; and Zhang, Y. 2020. How to Retrain Recommender System?: A Sequential Meta-Learning Method. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhang, Z.-Y.; Sheng, X.-R.; Zhang, Y.; Jiang, B.; Han, S.; Deng, H.; and Zheng, B. 2022. Towards Understanding the Overfitting Phenomenon of Deep Click-Through Rate Prediction Models. arXiv:2209.06053.

Zhu, J.; Liu, C.; Wang, P.; Zhao, X.; Chen, G.; Jin, J.; Peng, C.; Lin, Z.; and Shao, J. 2021. Dynamic Parameterized Network for CTR Prediction. *arXiv preprint arXiv:2111.04983*.

Zhu, J.; Liu, C.; Wang, P.; Zhao, X.; Lin, Z.; and Shao, J. 2023. Confidence Ranking for CTR Prediction. In *Companion Proceedings of the ACM Web Conference 2023*, 437–441.

Zhu, J.; Liu, J.; Yang, S.; Zhang, Q.; and He, X. 2020. Fuxictr: An open benchmark for click-through rate prediction. *arXiv preprint arXiv:2009.05794*.