Differentiable Adversarial Attacks for Marked Temporal Point Processes

Pritish Chakraborty*[1], Vinayak Gupta*[2], Rahul R[1], Srikanta J. Bedathur[3], Abir De[1]

[1]Indian Institute of Technology Bombay
[2]University of Washington Seattle
[3]Indian Institute of Technology Delhi
{pritish, rrahul, abir}@cse.iitb.ac.in, vinayak@cs.washington.edu, srikanta@cse.iitd.ac.in

Abstract

Marked temporal point processes (MTPPs) have been shown to be extremely effective in modeling continuous time event sequences (CTESs). In this work, we present adversarial attacks designed specifically for MTPP models. A key criterion for a good adversarial attack is its imperceptibility. For objects such as images or text, this is often achieved by bounding perturbation in some fixed L_p norm-ball. However, similarly minimizing distance norms between two CTESs in the context of MTPPs is challenging due to their sequential nature and varying time-scales and lengths. We address this challenge by first permuting the events and then incorporating the additive noise to the arrival timestamps. However, the worst case optimization of such adversarial attacks is a hard combinatorial problem, requiring exploration across a permutation space that is factorially large in the length of the input sequence. As a result, we propose a novel differentiable scheme - PERMTPP - using which we can perform adversarial attacks by learning to minimize the likelihood, while minimizing the distance between two CTESs. Our experiments on four real-world datasets demonstrate the offensive and defensive capabilities, and lower inference times of PERMTPP.

1 Introduction

In recent years, marked temporal point processes (MTPPs) have become indispensable for modeling continuous time event sequences (CTESs). They find applications in various fields, including information diffusion (Du et al. 2015; De et al. 2016; Farajtabar et al. 2017; Sharma et al. 2021), finance (Bacry, Mastromatteo, and Muzy 2015; Hawkes 2018; Omi, Ueda, and Aihara 2019; Gupta et al. 2021), epidemiology (Lorch et al. 2018; Rizoiu et al. 2018), and spatial data (Likhyani et al. 2020; Gupta and Bedathur 2021). They offer impressive predictive power; nevertheless, our experiments reveal they are also vulnerable to adversarial attacks, similar to the neural models for images (Zhang et al. 2019; Wang et al. 2019; Madry et al. 2018; Sriramanan et al. 2021) and texts (Zou et al. 2023; Jia and Liang 2017). Such vulnerabilities may compromise their reliability and robustness in several real world applications. For example, in information diffusion, they can jeopardize a viral marketing plan; in epidemiology, they can affect policy implementation or epidemic control plans, etc. Existing works on adversarial attacks predominantly focus on image (Zhang et al. 2019; Rice, Wong, and Kolter 2020; Croce and Hein 2020; Lin et al. 2022; Huang and Kong 2022; Wang, Zhang, and Zhang 2023; Vorácek and Hein 2023; Ma et al. 2023) and texts (Zou et al. 2023; Jia and Liang 2017; Wei, Haghtalab, and Steinhardt 2023; Shin et al. 2020; Wen et al. 2023; Jones et al. 2023; Zhao et al. 2023), with few works on discrete time-series (Oregi et al. 2018; Karim, Majumdar, and Darabi 2020; Wu et al. 2022; Fawaz et al. 2019; Ma et al. 2020; Yoon et al. 2022; Ding et al. 2023; Belkhouja, Yan, and Doppa 2022). However, there is still a lack of research on designing adversarial attacks for MTPPs, which if unaddressed, may leave potential vulnerabilities in the current deployed MTPP-based systems (Shchur et al. 2021).

Challenges in designing attacks for MTPPs: In this work, we focus on developing adversarial attacks specifically for MTPP models, which can help design more robust training methods. MTPPs are very different from images in terms of both representations and modeling characterizations. Hence, the existing adversarial attacks designed for images are not suitable for MTPPs. Moreover, due to the disparate inter-event arrival times, MTPPs pose the following challenges in designing both adversarial attacks and their defense.

— *Order-induced complexity of perturbation.* In the context of images, measuring adversarial perturbation between clean and perturbed objects is relatively straightforward. The Euclidean distance between pixel matrices serves as a suitable distance metric, generating zero-mean i.i.d. additive adversarial perturbations for each pixel value. The strength of the perturbation is quantified using the maximum difference between corresponding pixel values.

The concept of adversarial perturbation becomes more complex in the context of continuous time event sequences (CTESs). In contrast to images, the input data consists of an ordered set of events. Here, introducing noise to the arrival times may not manifest a simple monotone relationship with the overall perturbation of the sequence. For instance, applying a high level of noise to each event may still yield a smaller overall perturbation. This is because the perturbed sequence closely resembles the clean sequence offset with nearly constant noise, thereby maintaining the order of events unchanged. In such cases, the lack of impact on the

*These authors contributed equally.

temporal ordering weakens the overall attack. On the contrary, a small amount of noise might lead to a larger overall perturbation as it alters the order of events.

— *Non-differentiability in adversarial generation.* In the context of MTPP, even a small temporal noise can change the event order. Such a change causes an abrupt shift in the embedding vector within the MTPP model, which introduces non-differentiability in the overall model. This hinders the model's ability to accurately learn from perturbed data.

1.1 Our Contributions

We present PERMTPP, a novel trainable adversarial attack for CTES. PERMTPP handles perturbations in event ordering, event marks, and event times, *i.e.*, all associated properties of an MTPP. In detail, our contributions are:

Novel formulation of adversarial attacks: Given a CTES $\mathcal{H}$, we generate a perturbed sequence $\mathcal{H}'$ through a two-stage adversarial attack, followed by worst case likelihood minimization. In the first stage, we adversarially permute the events, while explicitly controlling the extent of shuffling. To do so, we permute the events in such a way that the total deviation in event positions from the clean sequence is small. In the second stage, we add adversarial noise to the arrival time of each event in the permuted sequence. Finally, we compute the perturbed sequence by minimizing the worst case likelihood over all possible permutations with additive noise, subject to the constraint that distance between the original and the perturbed sequence remains small.

Differentiable permutation guided event reordering: Worst case likelihood optimization over the set of all possible permutations is a hard combinatorial challenge. To tackle this challenge, we generate a soft permutation matrix through a differentiable permutation scheme, built upon a Gumbel-Sinkhorn (GS) network (Mena et al. 2018). Based on the MTPP model available to the adversary, GS takes the embeddings of the model as input and outputs a doubly stochastic matrix, which serves as a differentiable surrogate for the hard permutations.

Order-constrained additive noise injection: Having permuted the sequence (both time and mark) as above, we introduce a temporal noise to each of the event timestamps, through a noise generator. However, this additive noise might further shuffle the ordering of the events, which is already decided based on the event reordering in the previous step. We prevent such further event shuffling, by adding a hinge loss regularizer into the attacker's overall loss.

Our experiments on four real-world datasets show that PERMTPP is able to significantly degrade an MTPP model, trained on both clean and adversarially perturbed CTESs.

2 Problem Formulation

Marked temporal point processes (MTPPs): MTPPs model sequences of discrete events in continuous time. An event $e = (t, c)$ where $t \in \mathbb{R}^+$ and $c \in \mathcal{C}$ denote the arrival time and the mark, respectively, with $\mathcal{C}$ being discrete. Thus, CTES can be seen as $\mathcal{H} = \{e_i = (t_i, c_i) \,|\, t_{i-1} < t_i\}$, where e_i is the i-the event in the CTES. We denote $\mathcal{H}(t)$, as the history of events occurred until and excluding time t, and a counting process $N(t)$, which counts the number of events that occurred until time t. Then, we define the rate of events using the conditional intensity function $\lambda(t \,|\, \mathcal{H}_t)$, which is computed as: $\mathbb{E}[dN(t) \,|\, \mathcal{H}_t] = \lambda(t \,|\, \mathcal{H}_t)dt$. Here, $dN(t) \in \{0, 1\}$ is number of events arriving in the infinitesimal time interval $[t, t + dt)$, with $\lambda(t \,|\, \mathcal{H}_t)$ depends on history. We model the mark distribution using multinomial distribution $c \sim m(\bullet \,|\, \mathcal{H}_t)$, which is used to draw the mark of the event arriving at time t. Given $\mathcal{H}(t_i)$, the expected time of the *next* event is: $\mathbb{E}\left[t_{i+1} | \mathcal{H}(t_i)\right] = \int_{t_i}^{\infty} t \cdot \lambda(t \,|\, \mathcal{H}_t)\, dt$. Moreover, we sample the mark of the next event from the discrete distribution over all possible categories.

The task of designing MTPPs reduces to developing appropriate $\lambda_{\boldsymbol{w}}(t \,|\, \mathcal{H}_t)$ and $m_{\boldsymbol{w}}(\bullet \,|\, \mathcal{H}_t)$ for the intensity and mark distribution. Du et al. (2016); Mei and Eisner (2017) use recurrent networks, whereas, Zuo et al. (2020) and Zhang et al. (2020) utilize the self-attention to capture long-term dependencies. In our experiments, we use the transformer-based MTPP model (Zuo et al. 2020).

Additional notations: Given an integer I, we denote $\pi : [I] \to [I]$ as a permutation of the integers $\{1, ..., I\}$ and $\boldsymbol{P}$ as the corresponding permutation matrix of dimension $I \times I$. Given a neural MTPP model, we will frequently use $\boldsymbol{h}_i$ as the embedding of $\mathcal{H}(t_i)$ until time t_i. Typically, they summarize the first i events into $\boldsymbol{h}_i$ using a sequence encoder and feed this as input into a final layer (sometimes, with the event e_i) which output the intensity and the logit for the mark distribution. Hence, one can equivalently write: $\lambda_{\boldsymbol{w}}(t_{i+1} \,|\, \mathcal{H}_t) = \lambda_{\boldsymbol{w}}(t_{i+1} \,|\, \boldsymbol{h}_i)$ and $m_{\boldsymbol{w}}(c_{i+1} \,|\, \mathcal{H}_t) = m_{\boldsymbol{w}}(c_{i+1} \,|\, \boldsymbol{h}_i)$. We denote $\mathcal{M}_{\widehat{\boldsymbol{w}}}$ as the learner's trained model and $\mathcal{M}_{\boldsymbol{w}}$ as the adversary's model—or, the adversary's belief about the learner's trained model.

2.1 Problem Statement

Here, we formulate the optimization problem for adversarial attacks and their defense mechanisms on MTPPs.

Adversarial attack for MTPP: Given an MTPP model $\mathcal{M}_{\boldsymbol{w}}$ with parameters $\boldsymbol{w}$ with the adversary, we seek to develop methods for adversarial attacks, which would result in significant error in terms of both time and mark prediction. Here the model $\mathcal{M}_{\boldsymbol{w}}$ may not match with the learner's true model and it is rather the adversary's belief about the learner's true model. Given a clean sequence $\mathcal{H}$, the adversary seeks to find the adversarial sequence $\mathcal{H}' = \left\{(t'_i, c'_i) \,|\, t'_i < t'_{i+1}, i \in [|\mathcal{H}|]\right\}$[1] so that the adversary's model $\mathcal{M}_{\boldsymbol{w}}$ incurs high error in predicting the correct event (t_i, c_i), when it takes $\mathcal{H}'(t_i)$ as input. Thus, we define the adversary's objective function as follows.

$$\mathcal{L}(\mathcal{H} \,|\, \mathcal{M}_{\boldsymbol{w}}, \mathcal{H}') = \sum_{e_i \in \mathcal{H}} \Bigg[\log \lambda_{\boldsymbol{w}}(t_i \,|\, \mathcal{H}'(t_i)) - \int_{t'_{i-1}}^{t_i} \lambda(\tau \,|\, \mathcal{H}'(\tau)) d\tau + \log m_{\boldsymbol{w}}(c_i \,|\, \mathcal{H}'(t_i)) \Bigg] \quad (1)$$

[1] $\{(t'_i, c'_i)\}$ is always chronologically sorted: $t'_{i+1} > t'_i$. Hence, t'_i is not necessarily the perturbed time from t_i.

Then, the adversary has the following optimization problem.

$$\underset{\mathcal{H}'}{\text{minimize}}\, \mathcal{L}(\mathcal{H} \mid \mathcal{M}_{\boldsymbol{w}}, \mathcal{H}') \text{ s.t., } \text{Dist}(\mathcal{H}, \mathcal{H}') \text{ is small.} \quad (2)$$

Hence, the adversary seeks to minimize the likelihood of the *clean events* e_i, conditioned on the perturbed events $\mathcal{H}'(t_i)$ occurred until t_i. Note that the second term in (2) is slightly different than the usual survival term in the clean likelihood ($\int_{t'_{i-1}}^{t_i} \lambda(\tau \mid \mathcal{H}(\tau))d\tau$ vs. $\int_{t_{i-1}}^{t_i} \lambda(\tau \mid \mathcal{H}(\tau))d\tau$).

2.2 Computation of Dist$(\mathcal{H}, \mathcal{H}')$

Suppose, we are given a sequence of events $\{(t_i + \epsilon_i, c_i)\}$ after adding noise ϵ_i to the timestamps. Since ϵ_i are not uniform across $i \in [|\mathcal{H}|]$, the resulting sequence is not sorted. Hence, we first sort $\{(t_i + \epsilon_i, c_i)\}$ using the following:

$$\pi = \text{argsort}(\{t_i + \epsilon_i\}_{i\in[|\mathcal{H}|]}), t'_1 = \min(\{t_i + \epsilon_i\}_{i\in[|\mathcal{H}|]})$$
$$\implies \mathcal{H}' = \{(t'_i, c'_i) \mid t'_i = t_{\pi_i}, c'_i = c_{\pi_i}\} \quad (3)$$

Therefore, π indicates the permutation of the indices after sorting the events in the increasing order of the perturbed arrival times. In most real world scenarios, MTPP datasets do not provide information about the time t_{start} from which the event monitoring started. The time t_{start} is different from the first arrival time t_1 for $\mathcal{H}$ (or t'_1 for $\mathcal{H}'$), since the latter is recorded only *after* we start the monitoring. An MTPP model assumes some initial monitoring time, mostly either $t_{\text{start}} = 0$ or the time of the first event, *i.e.*, $t_{\text{start}} = t_1$. If we choose a fixed t_{start}, agnostic to the underlying sequence (*e.g.*, $t_1 = 0$), then the distance between $\mathcal{H}'$ and $\mathcal{H}$ becomes very sensitive to t_{start}. Therefore, we use $t_{\text{start}} = t_1$ and $t_{\text{start}} = t'_1$ for $\mathcal{H}$ and $\mathcal{H}'$. This allows us to offset the arrival times of $\mathcal{H}$ and $\mathcal{H}'$ from the respective initial times t_1 and t'_1 and then measure the Wasserstein distance, Dist$(\mathcal{H}, \mathcal{H}')$, as follows (Xiao et al. 2017).

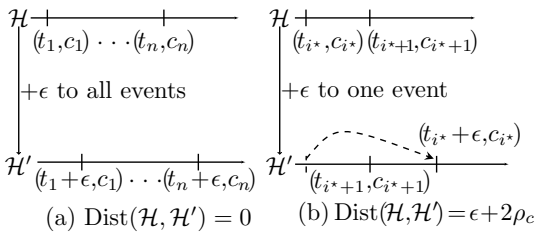
$$\text{Dist}(\mathcal{H}, \mathcal{H}') = \sum_{i\in[|\mathcal{H}|]} \Big[|(t'_i - t'_1) - (t_i - t_1)| + \rho_c \mathbb{I}[c'_{\pi_i} \neq c_i]\Big], \quad (4)$$

Here, ρ_c controls the contribution between time and mark.

2.3 Key Technical Challenges

Difficulty in generating controlled perturbation $\mathcal{H}'$: A fundamental constraint of the adversarial perturbation is that the amount of perturbation, *i.e.*, the distance between $\mathcal{H}$ and $\mathcal{H}'$, has to be small enough to be undetectable. In the context of images or multivariate time series (Goodfellow, Shlens, and Szegedy 2014), the amount of perturbation simply corresponds to the difference between the underlying feature vectors. Consequently, in these applications, the task of generating adversarial objects is straightforward, as it involves adding controlled magnitude adversarial noise to the existing data. Unlike images, simply controlling the magnitude of the noise may not control Dist$(\mathcal{H}, \mathcal{H}')$. This is because the value of ϵ_i has a non-monotone relationship with Dist$(\mathcal{H}, \mathcal{H}')$, which manifests through the argsort operation.

For instance, consider $c'_i = c_i$, $\epsilon_i = \epsilon$. Despite the significant amount of perturbation introduced in $\mathcal{H}'$, both $\mathcal{H}'$ and $\mathcal{H}$ share the same order of events. Consequently, the distance between the mark sequences becomes zero. Moreover, we have $(t_i + \epsilon_i - t'_{\min}) - (t_i - t_1) = 0$ for all i. Hence, Dist$(\mathcal{H}, \mathcal{H}') = 0$ (Figure 1 (a)). On the contrary, we can construct $\mathcal{H}'$ by adding a small amount of noise to a single event, which swaps it with its successive event. Specifically, assume that for some $i = i^\star$, we have $c_{i^\star} \neq c_{i^\star+1}$. We choose $\epsilon_{i^\star} = \epsilon$ such that $t_{i^\star+1} - t_{i^\star} < \epsilon < t_{i^\star+2} - t_{i^\star}$ and add it to $t_{i^\star}$ (Figure 1 (b)). For all $i \neq i^\star$, we have $\epsilon_i = 0$. This leads $e_{i^\star}$ and $e_{i^\star+1}$ exchanging their positions in the resulting sequence. Moreover, we have $t'_1 = t_1 < t'_2 = t_2 < ... < t'_{i^\star} = t_{i^\star+1} < t'_{i^\star+1} = t_{i^\star} + \epsilon < t'_{i^\star+2} = t_{i^\star+2} < ... < t'_n = t_n$, resulting in perturbation of the event ordering. A trivial calculation shows that:



(a) Dist$(\mathcal{H}, \mathcal{H}') = 0$ (b) Dist$(\mathcal{H},\mathcal{H}') = \epsilon + 2\rho_c$

Figure 1: More noise may not increase Dist$(\mathcal{H}, \mathcal{H}')$.

$$\begin{aligned}\text{Dist}(\mathcal{H}, \mathcal{H}') &= \textstyle\sum_{i\in[|\mathcal{H}|]} \Big[|(t'_i - t'_1) - (t_i - t_1)| \\ &\quad + \rho_c \mathbb{I}[c'_{\pi_i} \neq c_i]\Big] \quad (5)\\ &= |t_{i^\star+1} - t_{i^\star}| + |t_{i^\star} + \epsilon - t_{i^\star+1}| \\ &\quad + \rho_c \mathbb{I}[c_{i^\star+1} \neq c_{i^\star}] + \rho_c \mathbb{I}[c_{i^\star} \neq c_{i^\star+1}] \quad (6)\\ &= (t_{i^\star+1} - t_{i^\star}) + \epsilon - (t_{i^\star+1} - t_{i^\star}) + 2\rho_c \\ &= \epsilon + 2\rho_c. \quad (7)\end{aligned}$$

Hence, we have a non-zero distance between the both time and mark sequences. Thus, using additive noise to control the distance between $\mathcal{H}$ and $\mathcal{H}'$ is a challenging task.

Non-differentiability of adversarial loss: The objective of adversarial perturbation is to induce a significant prediction error, which leads to intrinsic non-differentiability in the perturbed model. However, in other applications, this effect is less pronounced in the initial layers of the model due to the smoothness in the feature perturbation process— a small-magnitude noise causes only minor perturbations of the initial features. However, such a smoothness does not apply to perturbations on CTESs, resulting in increased non-differentiability for perturbed MTPP models.

To illustrate this point, consider the previous example where $\mathcal{H}' = \{(t'_i = t_i + \epsilon_i, c_i)\}$ and $\epsilon_{i^\star} > t_{i^\star+1} - t_{i^\star}$ for some $i = i^\star$, and $\epsilon_i = 0$ for all $i \neq i^\star$. In this scenario, due to the shift in event ordering, $\mathcal{H}'$ significantly differs from the original $\mathcal{H}$, even with the addition of small noise. Consequently, this leads to an abrupt jump in the embedding $\boldsymbol{h}'_{i^\star}$, resulting in a high value of $||\boldsymbol{h}_{i^\star} - \boldsymbol{h}'_{i^\star}||$.

3 PERMTPP Model

In this section, we describe PERMTPP, our proposed adversarial attack model and its different components.

3.1 Proposed Adversarial Attack Model

Overview of our method: The key hurdle behind controlling the extent of perturbation in the sequence $\mathcal{H}'$ is the involvement of the argsort operation in distance computation in Eq. (3), which changes the event ordering. To tackle this problem we first decide on π, the chronology of the events before adding the underlying noise. Then, we add the noise $\{\epsilon_i\}$ in an adversarial manner, subject to the constraint that the resulting chronological order after adding the noise does not change from π. In the following, we build PERMTPP, Permutation Guided Adversarial Attack in two stages.

Combinatorial formulation of two staged adversarial attack: We note that the difference between two CTESs $\mathcal{H}$ and $\mathcal{H}'$ is guided by three factors: (i) the discordance between the ordering of the events; (ii) the difference between the arrival times; and (iii) the mismatch between the marks. Therefore, we generate adversarial perturbations in two stages. First, we perform an adversarial permutation on the events to generate a new perturbed order. This corresponds to factors (i) and (iii) above. Then, we add suitable noise to the timestamps of those events, subject to the above order. This corresponds to factor (ii).

— *Adversarial permutation.* Consider a clean CTES $\mathcal{H}$, which consists of n events. In the first stage, we explicitly apply a permutation $\pi : [n] \to [n]$ on $\mathcal{H}$ before adding any noise. This results in a new permuted sequence denoted as $\mathcal{H}_\pi = \{(t_{\pi_1}, c_{\pi_1}), (t_{\pi_2}, c_{\pi_2}), \ldots, (t_{\pi_n}, c_{\pi_n})\}$. Note that unlike $\mathcal{H}$, the ordered set $\mathcal{H}_\pi$ may not maintain a chronological order, and thus, the condition $t_{\pi_i} < t_{\pi_{i+1}}$ may not hold.

— *Introducing temporal noise.* In the second stage, we proceed to add adversarial noise $\{\epsilon_i\}_{i\in[n]}$ to the timestamps of the permuted sequence $\mathcal{H}_\pi$. This results in a perturbed sequence $\mathcal{H}_{\pi,\boldsymbol{\epsilon}} = \{(t'_i, c'_i)\}$, where $t'_i = t_{\pi_i} + \epsilon_i$ and $c'_i = c_{\pi_i}$. We select the additive noise ϵ_i in a manner that the perturbed times become chronologically ordered and aligned with the permutation π, which was already computed in the first stage. This allows imposing constraints on ϵ_i such that:

$$t_{\pi_i} + \epsilon_i < t_{\pi_{i+1}} + \epsilon_{i+1} \text{ for all } i < n, \quad t_{\pi_1} + \epsilon_1 > 0. \quad (8)$$

Let us define $\boldsymbol{\epsilon} = [\epsilon_1; \ldots; \epsilon_n]$ and $\boldsymbol{t}_\pi = [t_{\pi_1}; \ldots; t_{\pi_n}]$. Constraints in Eq. (8) are linear constraints with respect to $\boldsymbol{t}_\pi$ and $\boldsymbol{\epsilon}$. Therefore, using suitable coefficient matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, they can be re-written as follows:

$$\boldsymbol{A}\boldsymbol{\epsilon} < \boldsymbol{B}\boldsymbol{t}_\pi \quad (9)$$

Here, $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$, with $\boldsymbol{A}[i,i] = 1$, $\boldsymbol{A}[i,i+1] = -1$, $\boldsymbol{B}[i,i] = -1$, $\boldsymbol{B}[i,i+1] = 1$ for $i \in [n-1]$ and $\boldsymbol{A}[n,1] = -1$, $\boldsymbol{A}[n,j] = 0$, $\boldsymbol{B}[n,1] = 1$, $\boldsymbol{B}[n,j] = 0$ for $1 < j \leq n$. Additionally, $\boldsymbol{A}[i,j] = 0, \boldsymbol{B}[i,j] = 0\ \forall j \neq i$ and $j \neq i+1, i \in [n-1]$. This formulation allows us to ensure that the perturbed timestamps maintain the desired chronological order, aligning with the predefined permutation π obtained in the first stage. Hence, we have our adversarial sequence as $\mathcal{H}' = \mathcal{H}_{\pi,\boldsymbol{\epsilon}} = \left\{(t'_i, c'_i) \,|\, t'_i < t'_{i+1}\right\}$ where $t'_i = t_{\pi_i} + \epsilon_i$.

Optimization of $\pi, \boldsymbol{\epsilon}$: We first re-write the distance measure $\text{Dist}(\mathcal{H}, \mathcal{H}_{\pi,\boldsymbol{\epsilon}})$ (4) as follows:

$$\text{Dist}(\mathcal{H}, \mathcal{H}_{\pi,\boldsymbol{\epsilon}}) = \textstyle\sum_{i\in[|\mathcal{H}|]} |t_{\pi_i} + \epsilon_{\pi_i} - t_i| + \rho_c \left(1 - m_{\boldsymbol{w}}(c'_{\pi_i} = c_i \,|\, \mathcal{H}_{\pi,\boldsymbol{\epsilon}})\right) \quad (10)$$

Here the first term measures the deviation due to adversarial permutation the additive noise and the second term measures the expected number of times for which the adversarially generated mark is different than the true mark, *i.e.*, $\sum_i \mathbb{E}[\mathbb{I}[c'_i \neq c_i]]$ where $\mathbb{I}[\cdot]$ is the indicator function. Given the above measure of the distance in Eq. (10) and constraints (9), we write the problem defined in Eq. (2) as a combinatorial optimization task, with respect to the combinatorial variable π and the continuous variable $\boldsymbol{\epsilon}$ as follows:

$$\begin{aligned} &\underset{\pi,\boldsymbol{\epsilon}}{\text{minimize}}\ \mathcal{L}(\mathcal{H} \,|\, \mathcal{M}_{\boldsymbol{w}}, \mathcal{H}_{\pi,\boldsymbol{\epsilon}}) + \rho_D \,\text{Dist}(\mathcal{H}, \mathcal{H}_{\pi,\boldsymbol{\epsilon}}) \\ &\text{subject to, } \boldsymbol{A}\boldsymbol{\epsilon} \leq \boldsymbol{B}\boldsymbol{t}_\pi \end{aligned} \quad (11)$$

We encode the constraint $\boldsymbol{A}\boldsymbol{\epsilon} \leq \boldsymbol{B}\boldsymbol{t}_\pi$ into a hinge loss $\sum_{i\in[|\mathcal{H}|]}[\boldsymbol{A}\boldsymbol{\epsilon} - \boldsymbol{B}\boldsymbol{t}_\pi]_+[i]$ and add it to the objective (11) to solve the following regularized optmization task, with hyperparameters ρ_D and $\rho_{A,B}$

$$\begin{aligned} \underset{\pi,\boldsymbol{\epsilon}}{\text{minimize}}\ &\mathcal{L}(\mathcal{H} \,|\, \mathcal{M}_{\boldsymbol{w}}, \mathcal{H}_{\pi,\boldsymbol{\epsilon}}) + \rho_D \,\text{Dist}(\mathcal{H}, \mathcal{H}_{\pi,\boldsymbol{\epsilon}}) \\ &+ \rho_{A,B} \textstyle\sum_{i\in[|\mathcal{H}|]}[\boldsymbol{A}\boldsymbol{\epsilon} - \boldsymbol{B}\boldsymbol{t}_\pi]_+[i] \end{aligned} \quad (12)$$

3.2 Differentiable Neural Networks for π, ϵ

Solving the optimization problem defined in Eq. (12) is hard due to the large number of optimization variables of order $O(\sum_{\mathcal{H}\in\mathcal{D}} |\mathcal{H}|)$ as well as the involvement of the combinatorial variables $\{\pi\}$. To address this challenge, we develop a neural network $\text{PERMTPP}_{\theta,\phi}$ which would model π and $\boldsymbol{\epsilon}$, based on $\mathcal{H}$ used as input to the neural network. Training this network to solve the optimization (12) will provide the necessary inductive bias into $\text{PERMTPP}_{\theta,\phi}$ to compute the optimal $\mathcal{H}'$ for an unseen $\mathcal{H}$. Thus, $\text{PERMTPP}_{\theta,\phi}$ has two components, as follows:

(1) GS_θ**:** We feed the embeddings $\{\boldsymbol{h}_i\}$ from $\mathcal{M}_{\boldsymbol{w}}$, that summarizes the sequence $\mathcal{H}$ into a Gumbel-Sinkhorn permutation generator GS_θ and outputs $\boldsymbol{P}$, a soft permutation matrix, which relaxes π.

(2) ATTN_ϕ: This network takes $\boldsymbol{P}$ and $\mathcal{H}$ as input and outputs the additive noise $\boldsymbol{\epsilon}$.

Neural model for π (GS_θ): For batched learning, we appropriately pad each sequence $\mathcal{H} \in \mathcal{D}$ to obtain the same length n. One can replace π with a hard permutation matrix $\boldsymbol{P}$ of dimensions $n \times n$, where $\boldsymbol{P}[i,j] = 1$ iff $j = \pi[i]$. To ensure end-to-end continuous optimization, we relax this hard permutation matrix using a soft permutation matrix (a doubly stochastic matrix), which is learned using Gumbel-Sinkhorn (Mena et al. 2018) network GS_θ. The neural TPP models (Du et al. 2016; Mei and Eisner 2017) provide embeddings $\boldsymbol{h}_i$, for each event $e_i \in \mathcal{H}$, which represent the history $\mathcal{H}(t_i)$ upto time t_i. Given the TPP model $\mathcal{M}_{\boldsymbol{w}}$ available to the adversary (*i.e.*, adversary's belief about the learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$), we feed these embeddings $\boldsymbol{H} = \{\boldsymbol{h}_i \,|\, i \in [n]\}$

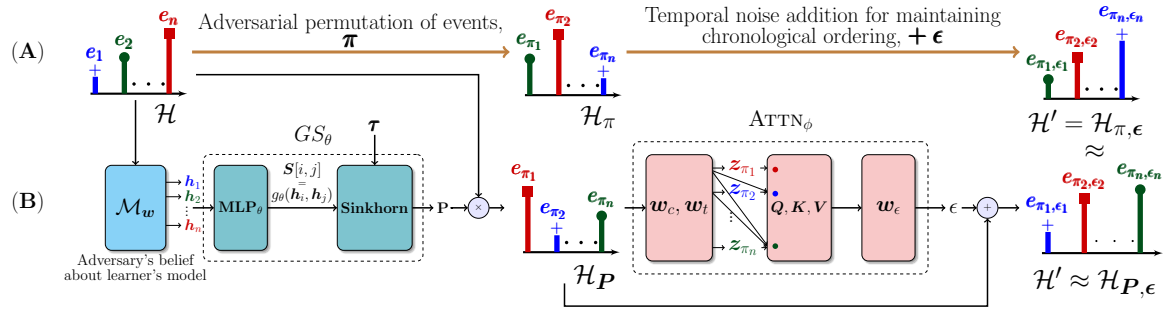


Figure 2: Overview of PERMTPP. **(A) Combinatorial perspective of two staged adversarial attack**: It describes combinatorial computation of the adversarial CTES $\mathcal{H}' = \mathcal{H}_{\pi,\boldsymbol{\epsilon}}$ from the clean CTES $\mathcal{H}$. We first apply a (hard) permutation map π on $\mathcal{H}$ to obtain the permuted sequence $\mathcal{H}_\pi$. The map π was intended to fix the chronological order of the events of the final perturbed CTES $\mathcal{H}'$. However $\mathcal{H}_\pi$ is not chronologically sorted. Hence, we add temporal noise $\boldsymbol{\epsilon} = \{\epsilon_1, ..\epsilon_n\}$ to the timestamps in $\mathcal{H}_\pi$ with a hinge regularizer (last term in Eq. (12)), so that $\mathcal{H}' = \mathcal{H}_{\pi,\boldsymbol{\epsilon}}$ is chronologically ordered and is aligned w.r.t π. **(B) Differentiable approximation of $\pi, \boldsymbol{\epsilon}$**: The CTES $\mathcal{H} = \{e_i \mid i \in [n]\}$ is fed to $\mathcal{M}_{\boldsymbol{w}}$ (adversary's belief about the learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$) to obtain the embeddings $\boldsymbol{H} = \{\boldsymbol{h}_i \mid i \in [n]\}$. They are fed to a Gumbel-Sinkhorn network GS_θ to obtain a soft permutation matrix $\boldsymbol{P}$. Within GS_θ, first an MLP network g_θ generates the seed matrix $\boldsymbol{S}$ with $\boldsymbol{S}[i,j] = g_\theta(\boldsymbol{h}_i, \boldsymbol{h}_j)$ and then $\boldsymbol{S}$ goes through a number of Sinkhorn iterations with temperature τ, to output $\boldsymbol{P}$; $\boldsymbol{P}$ is then applied on $\mathcal{H}$ to obtain the soft permuted sequence $\mathcal{H}_{\boldsymbol{P}} \approx \mathcal{H}_\pi$. Next, $\mathcal{H}_{\boldsymbol{P}}$ is fed into ATTN$_\phi$, which consists of of linear, attention and linear layers, to obtain the additive noise $\boldsymbol{\epsilon}$, which finally provides $\mathcal{H}_{\boldsymbol{P},\boldsymbol{\epsilon}} \approx \mathcal{H}_{\pi,\boldsymbol{\epsilon}}$.

into the Gumbel-Sinkhorn permutation generator GS_θ to generate the soft permutation matrix $\boldsymbol{P}$. We write that:

$$\boldsymbol{P} = GS_\theta(\boldsymbol{H}), \quad \text{where: } \boldsymbol{H} = \{\boldsymbol{h}_i \mid i \in [n]\} \tag{13}$$

GS_θ consists of a multi-layer perceptron (MLP) g_θ and a Sinkhorn iteration layer. We feed $\boldsymbol{H}$ (13) into g_θ to obtain a matrix $\boldsymbol{S}$ such that $\boldsymbol{S}[i,j] = g_\theta(\boldsymbol{h}_i, \boldsymbol{h}_j)$. This matrix $\boldsymbol{S}$ works as a seed matrix to generate $\boldsymbol{P}$ through successive Gumbel-Sinkhorn iterations from $\ell = 1, ..., L$. Specifically, we start with $\boldsymbol{A}_0$ as $\boldsymbol{A}_0 = \exp(\boldsymbol{S}/\tau)$. Then, given $\boldsymbol{A}_{\ell-1}$ computed after iteration $\ell - 1$ for $\ell > 0$, we compute $\boldsymbol{A}_\ell$. For this, we normalize each row of $\boldsymbol{A}_{\ell-1}$ to obtain $\boldsymbol{A}_\ell^{(r)}$ and then we normalize each column of $\boldsymbol{A}_l^{(r)}$ to obtain $\boldsymbol{A}_\ell$.

$$\boldsymbol{A}_\ell^{(r)}[i,j] = \frac{\boldsymbol{A}_{\ell-1}[i,j]}{\sum_{j'=1}^n \boldsymbol{A}_{\ell-1}[i,j']}; \; \boldsymbol{A}_\ell[i,j] = \frac{\boldsymbol{A}_\ell^{(r)}[i,j]}{\sum_{i'=1}^n \boldsymbol{A}_\ell^{(r)}[i',j]} \quad \text{for all } i \in [n], j \in [n] \tag{14}$$

Then we set $\boldsymbol{P} = \boldsymbol{A}_L$. Note that, $\boldsymbol{P}$ also approximately solves the following optimization problem:

$$\max_{\boldsymbol{P}} \sum_{i,j} \boldsymbol{S}[i,j]\boldsymbol{P}[i,j] - \tau \cdot \sum_{i,j} \boldsymbol{P}[i,j] \log \boldsymbol{P}[i,j], \tag{15}$$
$$\text{such that } \boldsymbol{P} \geq 0, \; \boldsymbol{P}\mathbf{1} = \boldsymbol{P}^\top \mathbf{1} = \mathbf{1}.$$

As $\tau \to 0$, $\boldsymbol{P}$ becomes close to hard permutation matrix (Mena et al. 2018; Cuturi 2013). However, for a very low value of τ, the gradient becomes almost zero. As a result, the learner cannot leverage any feedback from the loss. Finally, we apply $\boldsymbol{P}$ on the $\mathcal{H}$ to obtain the new (soft) permuted sequence as: $\mathcal{H}_{\boldsymbol{P}} = [\boldsymbol{Pt}, \boldsymbol{Pc}]$, where $\boldsymbol{t} = [t_1, .., t_n]$ and $\boldsymbol{c} = [c_1, ..., c_n]$.

Neural model for $\boldsymbol{\epsilon}$ (ATTN$_\phi$): The next stage of attack involves generating continuous noise to add to the permuted sequence $\mathcal{H}_{\boldsymbol{P}} = [\boldsymbol{Pt}, \boldsymbol{Pc}]$. The continuous noise is constrained by Eq (9) to ensure that order of the events remain the same as decided by $\boldsymbol{P}$ computed as above in Eq. (14). To determine the noise for each event $(t_{\pi_i}, c_{\pi_i}) \approx (\boldsymbol{P}[i,:]\boldsymbol{t}, \boldsymbol{P}[i,:]\boldsymbol{c})$, we employ an attention based network ATTN$_\phi$, which takes the permuted mark and time sequences, *i.e.* $\mathcal{H}_{\boldsymbol{P}}$ as inputs, and outputs an intermediate embedding $\boldsymbol{s_i}$ for each event e_i, which is later projected to obtain ϵ_i. We add ϵ_i to $t_{\pi_i} \approx \boldsymbol{P}[i,:]\boldsymbol{t}$ to ensure the CTES follows chronological order subject to inequality (9). ATTN$_\phi$ consists of three layers.
— *Input layer.* From the first stage, we use $\mathcal{H}_{\boldsymbol{P}}$ to obtain trainable embedding for every event ($\boldsymbol{z}_\pi$) using the mark and time as: $\boldsymbol{z}_{\pi_i} = \boldsymbol{w}_c c_{\pi_i} + \boldsymbol{w}_t t_{\pi_i} \approx \boldsymbol{w}_c \boldsymbol{P}[i,:]\boldsymbol{c} + \boldsymbol{w}_t \boldsymbol{P}[i,:]\boldsymbol{t}$. We also add a *continuous-time* positional encoding to the event embedding (Zuo et al. 2020; Gupta and Bedathur 2022).
— *Attention layer.* We use a *masked* self-attention layer to aggregate and represent $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ as trainable *Query*, *Key*, and *Value* matrices respectively. Finally, we compute self-attention output as:

$$\boldsymbol{s}_i = \sum_{j=1}^{i} \frac{\exp\left((\boldsymbol{Q}\boldsymbol{z}_i)^T(\boldsymbol{K}\boldsymbol{z}_j)/\sqrt{D}\right)}{\sum_{j'=1}^{i} \exp\left((\boldsymbol{Q}\boldsymbol{z}_i)^T(\boldsymbol{K}\boldsymbol{z}_{j'})/\sqrt{D}\right)} \boldsymbol{V}\boldsymbol{z}_j, \tag{16}$$

where D denotes the number of hidden dimensions. Here, we compute the attention weights via a softmax function and introduce the necessary non-linearity to the model to obtain the final output $\overline{\boldsymbol{s}}_i$.
— *Output layer.* To obtain continuous-time noise for the event e_{π_i}, *i.e.* ϵ_i, we project the hidden representation to a scalar value as $\epsilon_i \leftarrow \boldsymbol{w}_\epsilon \overline{\boldsymbol{s}}_i + \boldsymbol{b}_\epsilon$. Here, $\boldsymbol{w}_\epsilon, \boldsymbol{b}_\epsilon$ are

Dataset	Taobao	Twitter	Electricity	Health
#Events $\sum_{\mathcal{H}\in\mathcal{D}} \lvert\mathcal{H}\rvert$	0.5M	2.1M	60M	60M
#Sequences $\lvert\mathcal{D}\rvert$	2000	22000	10000	10000
#Marks $\lvert\mathcal{C}\rvert$	17	3	5	5
Mean Length	51	41	300	500
Mark imbalance	25.72	22.02	65.09	41.50

Table 1: Dataset statistics. Here, mark imbalance means the average of the ratios of the highest frequent mark and the lowest frequent mark across different $\mathcal{H} \in \mathcal{D}$.

trainable parameters. The total set of parameters is $\phi = \{\boldsymbol{w}_\bullet, \boldsymbol{b}_\bullet, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\}$.

Differentiable training objective: By replacing π with the soft permutation matrix $\boldsymbol{P}$ and introducing the neural networks GS_θ and ATTN_ϕ in the optimization (12), and aggregating the objective for all $\mathcal{H} \in \mathcal{D}$, we have:

$$\underset{\theta,\phi}{\text{minimize}} \sum_{\mathcal{H}\in\mathcal{D}} \big(\mathcal{L}(\mathcal{H} \,|\, \mathcal{M}_{\boldsymbol{w}}, \mathcal{H}_{\boldsymbol{P},\boldsymbol{\epsilon}}) + \rho_D \,\text{Dist}(\mathcal{H}, \mathcal{H}_{\boldsymbol{P},\boldsymbol{\epsilon}}) + \rho_{A,B} \textstyle\sum_{i\in[|\mathcal{H}|]} [\boldsymbol{A\epsilon} - \boldsymbol{BPt}]_+[i]\big), \quad (17)$$

where, $\boldsymbol{P} = GS_\theta(\boldsymbol{H})$ with $\boldsymbol{H} = \mathcal{M}_{\boldsymbol{w}}(\mathcal{H})$; and $\boldsymbol{\epsilon}$ s.t. $\boldsymbol{\epsilon} = \text{ATTN}_\phi(\boldsymbol{P}, \mathcal{H})$.

In contrast to the optimization (12), which seeks to solve π for each $\mathcal{H}$ separately, the training of θ, ϕ will provide the inductive biases of π and $\boldsymbol{\epsilon}$ into GS_θ and ATTN_ϕ. With enough expressive power, they would be able to ensure $\text{Dist}(\mathcal{H}, \mathcal{H}_{\boldsymbol{P},\boldsymbol{\epsilon}})$ to be small and $\boldsymbol{A\epsilon} \leq \boldsymbol{Bt_p}$ for every $\mathcal{H}$ drawn from the same distribution $\mathcal{D}$.

Adversarial training for TPP: Given a dataset of sequences $\mathcal{D} = \{\mathcal{H}\}$, our adversarial training follows the following robust optimization problem:

$$\max_{\boldsymbol{w}} \min_{\theta,\phi} \sum_{\mathcal{H}\in\mathcal{D}} \big(\mathcal{L}(\mathcal{H} \,|\, \mathcal{M}_{\boldsymbol{w}}, \mathcal{H}_{\boldsymbol{P},\boldsymbol{\epsilon}}) + \rho_D \,\text{Dist}(\mathcal{H}, \mathcal{H}_{\boldsymbol{P},\boldsymbol{\epsilon}}) + \rho_{A,B} \textstyle\sum_{i\in[|\mathcal{H}|]} [\boldsymbol{A\epsilon} - \boldsymbol{BPt}]_+[i]\big). \quad (18)$$

4 Experiments

Here, we evaluate PERMTPP by its ability to reduce the performance of learner's MTPP $\mathcal{M}_{\widehat{\boldsymbol{w}}}$. Specifically, we ask the following research questions. **(RQ1)** How much performance deterioration can PERMTPP cause when $\mathcal{M}_{\widehat{\boldsymbol{w}}}$ is trained on clean examples? **(RQ2)** How much deterioration can PERMTPP cause when $\mathcal{M}_{\widehat{\boldsymbol{w}}}$ is trained on adversarial examples? **(RQ3)** How does PERMTPP perform in the context of adversarial defense? Does it provide better robustness than other adversarial strategies? All codes for PERMTPP are available at: https://github.com/data-iitd/advtpp.

Datasets: We use Taobao (Chen et al. 2019), Twitter (Mei and Eisner 2017), Electricity (Murray, Stankovic, and Stankovic 2017), and Health (Gupta, Bedathur, and De 2022; Baim et al. 1986) datasets. We provide the details for each dataset in Table 1.

Baselines: In our knowledge, PERMTPP is the first adversarial attack on the MTPPs. As a result, we resort to the adversarial methods designed specifically for other domains and adapt them for MTPPs. We use two attack strategies from the domain of computer vision, *viz.*, (1) PGD (Madry et al. 2018), (2) MI-FGSM (Dong et al. 2018); and two attack strategies which were used for discrete time-series models, *viz.*, (3) RTS-D and (4) RTS-P. (Liu et al. 2023).

White box and black box models: Depending on how much information about the learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$ is available to the adversary, the adversary can use two types of $\mathcal{M}_{\boldsymbol{w}}$: (1) Black box (BB): Here, $\mathcal{M}_{\boldsymbol{w}} \neq \mathcal{M}_{\widehat{\boldsymbol{w}}}$, *i.e.*, the adversary uses another version of the MTPP model $\mathcal{M}_{\boldsymbol{w}}$ that is not same as same as the learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$. There can be several examples of such black box models $\mathcal{M}_{\boldsymbol{w}}$. Predominantly, we assume that the adversary uses a model which is trained on the same training set with different seed. (2) White box (WB): Here, $\mathcal{M}_{\boldsymbol{w}} = \mathcal{M}_{\widehat{\boldsymbol{w}}}$, *i.e.*, the adversary has access to the learner's model, which is going to be deployed during test; and the adversary uses this exact trained model to compute the perturbation.

Evaluation Metrics: Given a dataset of CTESs $\mathcal{D} = \{\mathcal{H}\}$, we split $\mathcal{D}$ into 70% training, 10% validation and 20% test set. Given the adversary's model $\mathcal{M}_{\boldsymbol{w}}$, we compute the adversarial perturbations by solving the optimization problem (17) for our method. Similarly, for baseline methods, we compute the worst case perturbations, by minimizing the log-likelihood using their respective algorithms. Then, we feed the perturbed test CTESs to the learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$ and report the results in terms of mean absolute error (MAE) between predicted and current time and mark prediction accuracy (MPA) to assess time and mark prediction ability respectively. Mathematically, $\text{MPA} = \frac{1}{|\mathcal{H}|} \sum_{e_i\in\mathcal{H}} \mathbb{I}(c_i = \widehat{c}_i)$ and $\text{MAE} = \frac{1}{|\mathcal{H}|} \sum_{e_i\in\mathcal{H}} [|t_i - \widehat{t}_i|]$. For the evaluation of adversarial attacks on the same learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$, a lower MPA and higher MAE correspond to a better attack framework, given that all other conditions are fixed, whereas, for the evaluation of defences a higher MPA and lower MAE correspond to a better defense framework.

4.1 Experimental Results

Effect of adversarial attacks on models trained on clean examples: Here, we compare the extent of the performance degradation of the learner model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$, caused by PERMTPP against the baselines, when $\mathcal{M}_{\widehat{\boldsymbol{w}}}$ is trained on clean CTESs (**RQ1**). We observed that the adversarial perturbation $\text{Dist}(\mathcal{H}, \mathcal{H}')$ is extremely difficult to control for the baselines, since there is no explicit "knob" in their model, which can regulate $\text{Dist}(\mathcal{H}, \mathcal{H}')$. As a result, we could not equalize this distance across different methods. However, we ensured that this distance for all the adversarial attack strategies remains within 10% deviation. We present the results in Table 2. It shows that PERMTPP degrades the predictive performance of the learner's model $\mathcal{M}_{\widehat{\boldsymbol{w}}}$, significantly compared to the baselines, for fourteen out of sixteen cases. **In particular, for Electricity dataset, only we achieve less than 90% MPA performance. In Taobao, PERMTPP outperforms the second best baseline by 36.5% MAE and 16.8% MPA for black box attack and by 3% MAE**

Method	MAE				MPA			
	Taobao	Twitter	Electricity	Health	Taobao	Twitter	Electricity	Health
No Attack	0.973	0.082	0.005	0.036	43.70	60.45	92.95	63.68
PGD (BB)	0.721	0.107	0.038	0.051	19.20	59.81	92.98	63.59
MI-FGSM (BB)	0.725	0.110	0.039	0.052	19.66	59.79	92.97	63.58
RTS-D (BB)	0.920	0.115	0.051	0.054	29.01	59.69	92.91	63.48
RTS-P (BB)	1.013	**0.191**	**0.063**	0.056	36.77	58.78	92.86	62.90
PERMTPP (BB)	**1.383**	0.115	0.053	**0.123**	**15.97**	**56.56**	**91.87**	**61.67**
PGD (WB)	0.708	0.101	0.015	0.047	18.43	59.86	92.90	63.63
MI-FGSM (WB)	0.730	0.103	0.048	0.048	19.52	59.84	92.91	63.61
RTS-D (WB)	0.923	0.108	0.072	0.051	28.94	59.74	92.78	63.52
RTS-P (WB)	1.309	0.187	0.070	0.155	42.52	58.90	92.83	62.87
PERMTPP (WB)	**1.350**	**0.120**	**0.078**	**0.230**	**9.21**	**55.84**	**87.96**	**42.64**

Table 2: Performance deterioration between adversarial attack strategies, where the attack is executed on the learner's MTPP model $\mathcal{M}_{\widehat{w}}$ trained on clean CTESs. We use THP (Zuo et al. 2020) as $\mathcal{M}_{\bullet}$ for both learner and adversary. BB and WB refer to black-box and white-box variants of the attacks. Across all cases, we ensure that $\text{Dist}(\mathcal{H}, \mathcal{H}')$ is approximately same. Higher the MAE and lower the MPA, more successful is the attack. **Bold** (underlined) indicate the best and the second best model.

	Taobao						Health					
Attack Method ↓	MAE			MPA			MAE			MPA		
Adv. Method →	PGD	MIF	RTS-D	PGD	MIF	RTS-D	PGD	MIF	RTS-D	PGD	MIF	RTS-D
No Attack	0.908	0.905	0.860	46.31	46.53	47.09	0.034	0.034	0.033	64.37	64.36	64.34
PGD (BB)	0.666	0.678	0.712	42.09	42.21	44.07	0.034	0.034	0.034	64.40	64.38	64.36
MI-FGSM (BB)	0.671	0.682	0.718	42.22	42.37	44.14	0.034	0.034	0.034	64.40	64.39	64.36
RTS-D (BB)	0.812	0.861	0.881	43.38	44.32	45.54	0.034	0.034	0.033	64.39	64.38	64.35
RTS-P (BB)	0.941	0.937	0.923	46.64	46.73	47.81	0.036	**0.035**	0.034	64.31	64.36	64.29
PERMTPP (BB)	**1.163**	**1.138**	**1.061**	**41.52**	**41.42**	**41.58**	**0.049**	0.034	**0.035**	**49.83**	**61.65**	**60.29**
PGD (WB)	0.670	0.681	0.717	42.00	42.12	43.97	0.034	0.034	0.034	64.39	64.39	64.36
MI-FGSM (WB)	0.674	0.686	0.723	42.17	42.26	44.06	0.034	0.034	0.034	64.39	64.39	64.36
RTS-D (WB)	0.816	0.864	0.884	43.56	44.47	45.68	0.034	0.034	0.033	64.39	64.39	64.35
RTS-P (WB)	1.147	1.138	1.060	47.31	47.50	48.00	0.034	0.035	0.034	64.29	64.11	64.30
PERMTPP (WB)	**1.265**	**1.295**	**1.228**	**18.56**	**19.94**	**21.12**	**0.042**	**0.048**	**0.041**	**40.80**	**53.67**	**49.14**

Table 3: Performance on different adversarial attack strategies, where the attack is executed on an MTPP model $\mathcal{M}_{\widehat{w}}$ learned using adversarial training methods – PGD, MI-FGSM (MIF) and RTS-D. We ensure that $\text{Dist}(\mathcal{H}, \mathcal{H}')$ is approximately same.

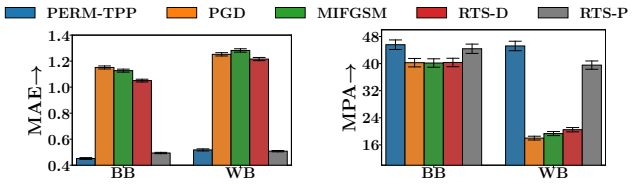


Figure 3: Adversarial Robustness of $\mathcal{M}_{\widehat{w}}$, trained using different adversarial training methods and PERMTPP attack for Taobao dataset. Here, lower the MAE and higher the MPA, more successful is the defense model.

and 50% MPA in the white-box setting.

Effect of adversarial attacks on models trained on adversarial examples: Here, we address **RQ2** by evaluating how successfully PERMTPP can deteriorate the performance of the learner's model $\mathcal{M}_{\widehat{w}}$, when it is trained on adversarially perturbed sequences. Table 3 shows the results across two datasets, for three adversarial training methods: PGD, MI-FGSM and RTS-D, which were used to train $\mathcal{M}_{\widehat{w}}$. We observe that (1) PERMTPP degrades the attacks more severely than the baselines in all cases, except for MAE in black-box setting for the Health dataset, when using the MI-FGSM adversarial training setup. (2) In many cases, adversarial training improved predictive performance under adversarial attack, from training on clean sequences (Table 2 vs Table 3).

Adversarial defense: Here, we address **RQ3** by comparing the adversarial robustness of the learner's model $\mathcal{M}_{\widehat{w}}$, across different adversarial training methods. Figure 3 shows the results for PERMTPP attack in Taobao dataset, which highlight that: (1) PERMTPP achieves the highest resilience towards attacks in terms of MPA; (2) For MAE, PERMTPP performs competitively with RTS-P; and wins over RTS-P for black box setting.

5 Conclusion

We propose a trainable adversarial attack specifically designed for MTPP. Our novel two-stage attack strategy, using differentiable neural surrogates for permutation and noise generation, gives us enhanced degradation capabilities on diverse datasets. PERMTPP opens up several future directions, for e.g., generating differentially private CTES examples.

Acknowledgements

Srikanta acknowledges DS Chair of AI Fellowship, Abir acknowledges a Trust Lab Grant from IITB.

References

Bacry, E.; Mastromatteo, I.; and Muzy, J.-F. 2015. Hawkes processes in finance. *arXiv preprint arXiv:1502.04592*.

Baim, D. S.; Colucci, W. S.; Monrad, E. S.; Smith, H. S.; Wright, R. F.; Lanoue, A.; Gauthier, D. F.; Ransil, B. J.; Grossman, W.; and Braunwald, E. 1986. Survival of patients with severe congestive heart failure treated with oral milrinone. *Journal of the American College of Cardiology*, 7(3): 661–670.

Belkhouja, T.; Yan, Y.; and Doppa, J. R. 2022. Dynamic time warping based adversarial framework for time-series domain. *IEEE transactions on pattern analysis and machine intelligence*.

Chen, Q.; Zhao, H.; Li, W.; Huang, P.; and Ou, W. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. *arXiv preprint arXiv:1905.06874*.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*.

De, A.; Valera, I.; Ganguly, N.; Bhattacharya, S.; and Gomez-Rodriguez, M. 2016. Learning and Forecasting Opinion Dynamics in Social Networks. In *NeurIPS*.

Ding, D.; Zhang, M.; Feng, F.; Huang, Y.; Jiang, E.; and Yang, M. 2023. Black-box adversarial attack on time series classification. In *AAAI*.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *CVPR*.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*.

Du, N.; Farajtabar, M.; Ahmed, A.; Smola, A. J.; and Song, L. 2015. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *KDD*.

Farajtabar, M.; Yang, J.; Ye, X.; Xu, H.; Trivedi, R.; Khalil, E.; Li, S.; Song, L.; and Zha, H. 2017. Fake news mitigation via point process based intervention. In *ICML*.

Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Adversarial attacks on deep neural networks for time series classification. In *IJCNN*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gupta, V.; and Bedathur, S. 2021. Region Invariant Normalizing Flows for Mobility Transfer. In *CIKM*.

Gupta, V.; and Bedathur, S. 2022. ProActive: Self-Attentive Temporal Point Process Flows for Activity Sequences. In *KDD*.

Gupta, V.; Bedathur, S.; Bhattacharya, S.; and De, A. 2021. Learning Temporal Point Processes with Intermittent Observations. In *AISTATS*.

Gupta, V.; Bedathur, S.; and De, A. 2022. Learning Temporal Point Processes for Efficient Retrieval of Continuous Time Event Sequences. In *AAAI*.

Hawkes, A. G. 2018. Hawkes processes and their applications to finance: a review. *Quantitative Finance*, 18(2): 193–198.

Huang, Y.; and Kong, A. W.-K. 2022. Transferable adversarial attack based on integrated gradients. In *ICLR*.

Jia, R.; and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Jones, E.; Dragan, A.; Raghunathan, A.; and Steinhardt, J. 2023. Automatically auditing large language models via discrete optimization. In *ICML*.

Karim, F.; Majumdar, S.; and Darabi, H. 2020. Adversarial attacks on time series. *IEEE transactions on pattern analysis and machine intelligence*, 43(10).

Likhyani, A.; Gupta, V.; Srijith, P.; Deepak, P.; and Bedathur, S. 2020. Modeling Implicit Communities from Geotagged Event Traces using Spatio-Temporal Point Processes. In *WISE*.

Lin, W.; Lucas, K.; Bauer, L.; Reiter, M. K.; and Sharif, M. 2022. Constrained Gradient Descent: A Powerful and Principled Evasion Attack Against Neural Networks. In *ICML*.

Liu, L.; Park, Y.; Hoang, T. N.; Hasson, H.; and Huan, J. 2023. Robust multivariate time-series forecasting: Adversarial attacks and defense mechanisms. In *ICLR*.

Lorch, L.; De, A.; Bhatt, S.; Trouleau, W.; Upadhyay, U.; and Gomez-Rodriguez, M. 2018. Stochastic Optimal Control of Epidemic Processes in Networks. *arXiv preprint arXiv:1810.13043*.

Ma, Q.; Zhuang, W.; Li, S.; Huang, D.; and Cottrell, G. 2020. Adversarial dynamic shapelet networks. In *AAAI*.

Ma, W.; Li, Y.; Jia, X.; and Xu, W. 2023. Transferable adversarial attack for both vision transformers and convolutional networks via momentum integrated gradients. In *ICCV*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Mei, H.; and Eisner, J. M. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*.

Mena, G.; Belanger, D.; Linderman, S.; and Snoek, J. 2018. Learning Latent Permutations with Gumbel-Sinkhorn Networks. In *ICLR*.

Murray, D.; Stankovic, L.; and Stankovic, V. 2017. An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study. *Scientific Data*, 4(1).

Omi, T.; Ueda, N.; and Aihara, K. 2019. Fully Neural Network based Model for General Temporal Point Processes. In *NeurIPS*.

Oregi, I.; Del Ser, J.; Perez, A.; and Lozano, J. A. 2018. Adversarial sample crafting for time series classification with elastic similarity measures. In *Intelligent Distributed Computing XII*, 26–39. Springer.

Rice, L.; Wong, E.; and Kolter, Z. 2020. Overfitting in adversarially robust deep learning. In *ICML*.

Rizoiu, M.-A.; Mishra, S.; Kong, Q.; Carman, M.; and Xie, L. 2018. SIR-Hawkes: on the Relationship Between Epidemic Models and Hawkes Point Processes. In *WWW*.

Sharma, K.; Zhang, Y.; Ferrara, E.; and Liu, Y. 2021. Identifying coordinated accounts on social media through hidden influence and group behaviours. In *KDD*.

Shchur, O.; Türkmen, A. C.; Januschowski, T.; and Günnemann, S. 2021. Neural Temporal Point Processes: A Review. In *IJCAI*.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*.

Sriramanan, G.; Addepalli, S.; Baburaj, A.; and Radhakrishnan, V. B. 2021. Towards efficient and effective adversarial training. In *NeurIPS*.

Vorácek, V.; and Hein, M. 2023. Improving l1-certified robustness via randomized smoothing by leveraging box constraints. In *ICML*.

Wang, X.; Zhang, Z.; and Zhang, J. 2023. Structure invariant transformation for better adversarial transferability. In *ICCV*.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2019. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How does llm safety training fail? In *NeurIPS*.

Wen, Y.; Jain, N.; Kirchenbauer, J.; Goldblum, M.; Geiping, J.; and Goldstein, T. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In *NeurIPS*.

Wu, T.; Wang, X.; Qiao, S.; Xian, X.; Liu, Y.; and Zhang, L. 2022. Small perturbations are enough: Adversarial attacks on time series prediction. *Information Sciences*, 587: 794–812.

Xiao, S.; Farajtabar, M.; Ye, X.; Yan, J.; Song, L.; and Zha, H. 2017. Wasserstein Learning of Deep Generative Point Process Models. In *NeurIPS*.

Yoon, T.; Park, Y.; Ryu, E. K.; and Wang, Y. 2022. Robust probabilistic time series forecasting. In *AISTATS*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *ICML*.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020. Self-attentive Hawkes processes. In *ICML*.

Zhao, Y.; Pang, T.; Du, C.; Yang, X.; Li, C.; Cheung, N.-M. M.; and Lin, M. 2023. On evaluating adversarial robustness of large vision-language models. In *NeurIPS*.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer Hawkes Process. In *ICML*.