ACCELERATING WEIGHTED SCORE MATCHING FOR POINT PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Weighted score matching (WSM) avoids intractable intensity integration in the likelihood for point processes, but nested reverse-mode automatic differentiation (AD) for event-coordinate derivatives introduces a training bottleneck. To solve this issue, we propose *Taylor-mode WSM*, a plug-and-play method that replaces the nested derivative evaluator with a Taylor-mode evaluator while retaining reverse-mode AD for parameter updates. To explain and quantify the acceleration, we develop a program-level cost framework that derives the training-step speedup ratio, characterizes *when* the speedup is most pronounced, and establishes *why* the Taylor-mode evaluator is faster. Through experiments across statistical point processes, deep temporal point processes, and deep spatio-temporal point processes (STPPs), our framework explains the varying speedups across the three model families and further reveals that outer backpropagation is the dominant source of acceleration, accounting for 90.4%–99.6% of the total time reduction. Additional studies show that the speedup becomes more pronounced at higher derivative orders, while the advantage persists with increasing event-coordinate dimension. Overall, Taylor-mode WSM achieves an average speedup of $3.85\times$ and up to $5.6\times$ on deep STPPs, without sacrificing model performance.

1 INTRODUCTION

Spatio-temporal point processes (STPPs) model events that occur in continuous time and space, with applications in seismology (Ogata, 1998; 1999), finance (Bacry & Muzy, 2014; Hawkes, 2018), criminology (Mohler, 2013), and neuroscience (Linderman & Adams, 2015; Zhou et al., 2022a). Maximum likelihood estimation (MLE) is the predominant training objective for STPPs. However, likelihood evaluation requires integrating the intensity function over the spatio-temporal domain, which is generally intractable and thus relies on Monte Carlo sampling or numerical quadrature. These approximations inevitably introduce estimation error and become increasingly expensive in high-dimensional settings, substantially limiting training efficiency.

Weighted score matching (WSM) (Cao et al., 2024; 2025) avoids the intensity integration in the likelihood by replacing the MLE objective with a score-matching loss. However, WSM requires computing first- and second-order derivatives w.r.t. the event coordinates. Typical implementations obtain second-order derivatives by applying reverse-mode automatic differentiation (AD) twice, i.e., using nested reverse-mode AD, a common strategy for higher-order derivatives that can introduce redundant computation (Bettencourt et al., 2019). As a result, although WSM eliminates intensity integration, nested reverse-mode AD becomes the new computational bottleneck during training.

Forward-mode AD avoids this redundancy by propagating tangent variables directly along the primal trace (Baydin et al., 2018; Griewank & Walther, 2008). Taylor-mode AD extends this idea to higher-order derivatives by propagating Taylor coefficients through the primal computation (Griewank et al., 2000; Bischof et al., 1993). Therefore, in this work we propose to compute the event-coordinate derivatives in WSM using Taylor-mode AD. Our method is a drop-in replacement for event-coordinate derivatives computation, leaving all other components unchanged.

We further develop a program-level cost analysis that derives the theoretical training-step speedup ratio and identifies two sufficient conditions under which Taylor-mode WSM has a lower training-step cost, together with a shared-cost dilution effect that limits the speedup. Consistent with this theoretical analysis, the training-step speedup generally follows a clear model-family pattern: it is

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

modest for statistical point processes, larger for deep temporal point processes (TPPs), and most pronounced for deep STPPs, reaching up to $5.6\times$ without sacrificing model performance.

Specifically, we make the following contributions:

(1) ***A Plug-and-Play Differentiation Method for WSM.*** We propose Taylor-mode WSM, a training method for point processes with a hybrid differentiation strategy. It uses Taylor-mode AD to compute the required event-coordinate derivatives and reverse-mode AD to update the model parameters. It replaces only the event-coordinate derivative evaluator, leaving the objective, architecture, and optimizer unchanged, and can thus be directly integrated into existing WSM training pipelines.

(2) ***A Program-Level Cost Analysis of Taylor-Mode WSM Speedup.*** We develop a program-level framework that decomposes each training step into shared computation, derivative evaluation, and outer backpropagation. It yields the theoretical training-step speedup ratio and two sufficient conditions for Taylor-mode WSM to have a lower cost: a lower derivative evaluation and a lower outer backpropagation. It also characterizes the shared-cost dilution effect, whereby increasing shared computation reduces the attainable speedup.

(3) ***An Empirical Evaluation of Taylor-Mode WSM Speedup.*** The experiments reveal a model-family pattern consistent with the theoretical analysis: from statistical point processes to deep TPPs and deep STPPs, the average shared-cost ratio decreases from 97.8% to 85.0% and 66.0%, while average training-step speedup increases from $1.32\times$ to $1.43\times$ and $3.85\times$. Stage-wise measurements on deep STPPs support the two sufficient conditions and attribute 90.4%–99.6% of the time savings to the outer backpropagation. We further analyze how the derivative order and event-coordinate dimensionality affect the achieved speedup.

2 Background

In this section, we briefly introduce the background on STPPs, WSM for STPPs, and AD operators for derivative computation.

2.1 Spatio-temporal Point Processes

A STPP generates a sequence of events $\mathcal{X} = (x_1, \ldots, x_N)$, where $x_n = (t_n, \mathbf{s}_n)$, over the observation window $(0, T] \times \mathcal{S}$ (Ogata, 1998) with $\mathcal{S} \subset \mathbb{R}^{d_s}$ being the spatial domain and N the random number of observed events. The events are ordered as $0 < t_1 < \cdots < t_N \leq T$. Let $\mathcal{H}_{n-1} = \{(t_i, \mathbf{s}_i)\}_{i=1}^{n-1}$ denote the history up to the $(n-1)$-th event, and $\mathcal{H}_{t^-}$ the history up to but excluding t. A STPP can be specified by the next-event conditional density:

$$p_\theta(t_n, \mathbf{s}_n \mid \mathcal{H}_{n-1}) = f_\theta(t_n \mid \mathcal{H}_{n-1}) q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1}),$$

where f_θ and q_θ are the conditional densities of the next event time and spatial location, respectively. Equivalently, the process can be characterized by the conditional intensity function $\lambda_\theta(t, \mathbf{s} \mid \mathcal{H}_{t^-})$, often abbreviated as $\lambda_\theta^*(t, \mathbf{s})$ (superscript $*$ denoting history dependence). For a candidate event at $t > t_{n-1}$, the conditional density and the conditional intensity are related by (Daley & Vere-Jones, 2003; Rasmussen, 2018):

$$p_\theta(t, \mathbf{s} \mid \mathcal{H}_{n-1}) = \lambda_\theta^*(t, \mathbf{s}) \exp\left(- \int_{t_{n-1}}^{t} \int_{\mathcal{S}} \lambda_\theta^*(u, \mathbf{r}) \, d\mathbf{r} \, du \right).$$

Defining the temporal ground intensity as $\lambda_\theta^*(t) = \int_{\mathcal{S}} \lambda_\theta^*(t, \mathbf{r}) \, d\mathbf{r}$, we obtain the conditional spatial density $q_\theta(\mathbf{s} \mid t, \mathcal{H}_{n-1}) = \frac{\lambda_\theta^*(t, \mathbf{s})}{\lambda_\theta^*(t)}$, and the induced temporal density $f_\theta(t \mid \mathcal{H}_{n-1}) = \lambda_\theta^*(t) \exp\left(- \int_{t_{n-1}}^{t} \lambda_\theta^*(u) \, du \right)$. A standard estimator is MLE: $\mathcal{L}^{\mathrm{MLE}}(\theta) = \sum_{n=1}^{N} \log \lambda_\theta^*(t_n, \mathbf{s}_n) - \int_0^T \int_{\mathcal{S}} \lambda_\theta^*(u, \mathbf{r}) \, d\mathbf{r} \, du$. The integral is generally intractable and requires Monte Carlo estimation or numerical quadrature (Shchur et al., 2021; Zhou & Yu, 2023a;b), which introduce numerical error and suffer from the curse of dimensionality.

2.2 Weighted Score Matching for STPPs

To bypass numerical integration in MLE, score matching (SM) (Hyvärinen, 2005) minimizes the Fisher divergence between the data distribution $p(\mathbf{z})$ and model distribution $p_\theta(\mathbf{z})$: $\mathcal{L}^{\mathrm{SM}}(\theta) =$

$\frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p}\left[\|\nabla_{\mathbf{z}}\log p(\mathbf{z}) - \nabla_{\mathbf{z}}\log p_\theta(\mathbf{z})\|^2\right]$, where $\mathbf{z}$ is a generic random vector. The key advantage of SM is that it avoids computing the normalizing constant: if $p_\theta(\mathbf{z}) = \frac{\tilde{p}_\theta(\mathbf{z})}{Z(\theta)}$, $Z(\theta) = \int \tilde{p}_\theta(\mathbf{z})\, d\mathbf{z}$, then $\nabla_{\mathbf{z}}\log p_\theta(\mathbf{z}) = \nabla_{\mathbf{z}}\log \tilde{p}_\theta(\mathbf{z})$. However, the explicit objective depends on the unknown data score $\nabla_{\mathbf{z}}\log p(\mathbf{z})$. Under the boundary condition $p(\mathbf{z})\nabla_{\mathbf{z}}\log p_\theta(\mathbf{z}) \to 0$, integration by parts eliminates the data score and yields the tractable implicit objective $\mathcal{J}^{\mathrm{SM}}(\theta) = \mathbb{E}_{\mathbf{z}\sim p}\left[\frac{1}{2}\|\nabla_{\mathbf{z}}\log p_\theta(\mathbf{z})\|^2 + \mathrm{Tr}\left(\nabla^2_{\mathbf{z}}\log p_\theta(\mathbf{z})\right)\right]$, where Tr denotes the matrix trace.

For STPPs defined on bounded domains (e.g., $t \in (0, T]$ and $\mathbf{s} \in \mathcal{S}$), the boundary condition generally fails, so the implicit SM objective above cannot be derived directly via integration by parts. To address this issue, Cao et al. (2024; 2025) proposed WSM with temporal and spatial weight functions satisfying $h_{T,n}(t_{n-1}) = h_{T,n}(T) = 0$ and $h_{S,n}(\mathbf{s}) = 0$ for $\mathbf{s} \in \partial\mathcal{S}$. These weight functions enforce the boundary condition, enabling integration by parts and yielding the tractable objective $\mathcal{J}^{\mathrm{WSM}}_{h_T,h_S}(\theta) = \mathcal{J}^{\mathrm{WSM,T}}_{h_T}(\theta) + \mathcal{J}^{\mathrm{WSM,S}}_{h_S}(\theta)$ with the two components given by

$$\begin{aligned}
\mathcal{J}^{\mathrm{WSM,T}}_{h_T}(\theta) &= \mathbb{E}_{\mathcal{X}\sim p}\left[\sum_{n=1}^{N}\left\{\frac{1}{2}(\psi_n^T)^2 h_n^T + \dot{\psi}_n^T h_n^T + \psi_n^T \dot{h}_n^T\right\}\right], \\
\mathcal{J}^{\mathrm{WSM,S}}_{h_S}(\theta) &= \mathbb{E}_{\mathcal{X}\sim p}\left[\sum_{n=1}^{N}\left\{\frac{1}{2}\left\|\boldsymbol{\psi}_n^S\right\|^2 h_n^S + \delta_n^S h_n^S + \boldsymbol{\psi}_n^S \cdot \mathbf{g}_n^S\right\}\right],
\end{aligned} \tag{1}$$

where $\psi_n^T = \psi_{T,n,\theta}(t_n \,|\, \mathcal{H}_{n-1})$, $\dot{\psi}_n^T = \partial_t \psi_{T,n,\theta}(t \,|\, \mathcal{H}_{n-1})|_{t=t_n}$, $h_n^T = h_{T,n}(t_n)$, $\dot{h}_n^T = \partial_t h_{T,n}(t)|_{t=t_n}$, $\boldsymbol{\psi}_n^S = \boldsymbol{\psi}_{S,n,\theta}(\mathbf{s}_n \,|\, t_n, \mathcal{H}_{n-1})$, $\delta_n^S = \mathrm{Tr}(\nabla_{\mathbf{s}}\boldsymbol{\psi}_{S,n,\theta}(\mathbf{s} \,|\, t_n, \mathcal{H}_{n-1}))\big|_{\mathbf{s}=\mathbf{s}_n}$, $h_n^S = h_{S,n}(\mathbf{s}_n)$, $\mathbf{g}_n^S = \nabla_{\mathbf{s}} h_{S,n}(\mathbf{s})|_{\mathbf{s}=\mathbf{s}_n}$. Using the notation from Section 2.1, $\psi_{T,n,\theta}(t \,|\, \mathcal{H}_{n-1}) = \partial_t \log f_\theta(t \,|\, \mathcal{H}_{n-1}) = \partial_t \log \lambda^*_\theta(t) - \lambda^*_\theta(t)$ and $\boldsymbol{\psi}_{S,n,\theta}(\mathbf{s} \,|\, t, \mathcal{H}_{n-1}) = \nabla_{\mathbf{s}} \log q_\theta(\mathbf{s} \,|\, t, \mathcal{H}_{n-1})$.

Evaluating WSM requires second-order derivatives w.r.t. event coordinates, including the temporal term $\dot{\psi}_n^T = \partial_t^2 \log f_\theta(t \,|\, \mathcal{H}_{n-1})\big|_{t=t_n} = \left(\partial_t^2 \log \lambda^*_\theta(t) - \partial_t \lambda^*_\theta(t)\right)\big|_{t=t_n}$ and spatial term $\delta_n^S = \mathrm{Tr}\big(\nabla^2_{\mathbf{s}} \log q_\theta(\mathbf{s} \,|\, t_n, \mathcal{H}_{n-1})\big)\big|_{\mathbf{s}=\mathbf{s}_n} = \Delta_{\mathbf{s}} \log q_\theta(\mathbf{s} \,|\, t_n, \mathcal{H}_{n-1})|_{\mathbf{s}=\mathbf{s}_n}$. Computing these terms with nested reverse-mode AD incurs substantial computational and memory costs, making second-order derivative evaluation the main training bottleneck.

2.3 Automatic Differentiation Operators

AD evaluates derivatives by applying the chain rule to primitive operations in a computation trace (Baydin et al., 2018; Griewank & Walther, 2008). Forward-mode AD propagates tangents along the primal trace and computes Jacobian-vector products, whereas reverse-mode AD records the primal trace and propagates cotangents backward to compute vector-Jacobian products. For scalar-output objectives, reverse mode efficiently computes gradients w.r.t. many inputs, making it the standard approach for parameter update in deep learning (Rumelhart et al., 1986).

Taylor-mode AD extends forward-mode AD to higher-order derivatives by propagating truncated Taylor coefficients along the primal trace (Griewank et al., 2000; Bettencourt et al., 2019). While standard forward-mode AD computes only first-order derivatives (Jacobian-vector products) in a single forward pass, Taylor-mode AD propagates a truncated Taylor series expansion of order k, enabling the computation of all derivatives up to order k simultaneously in one forward pass.

3 Methodology

To mitigate the computational bottleneck of nested reverse-mode AD for second-order derivatives required by WSM, we turn to Taylor-mode AD as an alternative. Equation (1) decomposes the WSM objective $\mathcal{J}^{\mathrm{WSM}}$ into temporal and spatial components, $\mathcal{J}^{\mathrm{WSM,T}}$ and $\mathcal{J}^{\mathrm{WSM,S}}$, both requiring first- and second-order derivatives w.r.t. the event coordinates. Taylor-mode WSM (Taylor) and WSM with nested reverse-mode AD (Nested) differ only in their event-coordinate derivative evaluators, and we use $M \in \{\mathrm{Taylor}, \mathrm{Nested}\}$ to index the two methods.

For the n-th event and spatial coordinate j, let $\mathbf{e}_j \in \mathbb{R}^{d_s}$ be the unit vector and ϵ_s a scalar offset from $\mathbf{s}_n$ along $\mathbf{e}_j$. Define the one-dimensional restriction of the conditional spatial log-density as $F_{n,j}(\epsilon_s) = \log q_\theta(\mathbf{s}_n + \epsilon_s \mathbf{e}_j \mid t_n, \mathcal{H}_{n-1})$. Its second-order Taylor expansion at $\epsilon_s = 0$ is:

$$F_{n,j}(\epsilon_s) = F_{n,j}(0) + \epsilon_s[\boldsymbol{\psi}_n^S]_j + \frac{\epsilon_s^2}{2}\partial_{s_j}^2 \log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1}) + o(\epsilon_s^2),$$

from which the first- and second-order coefficients give the j-th spatial score component and diagonal Hessian entry, respectively. Collecting them over all d_s directions yields the spatial score $\boldsymbol{\psi}_n^S$ and its divergence $\delta_n^S = \sum_{j=1}^{d_s} \partial_{s_j}^2 \log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1})$.

The temporal derivatives are obtained analogously. For a scalar offset ϵ_t from t_n, define $G_n(\epsilon_t) = \log f_\theta(t_n + \epsilon_t \mid \mathcal{H}_{n-1})$. Its expansion at $\epsilon_t = 0$ is

$$G_n(\epsilon_t) = G_n(0) + \epsilon_t \psi_n^T + \frac{\epsilon_t^2}{2}\dot{\psi}_n^T + o(\epsilon_t^2).$$

The first- and second-order coefficients yield the temporal score ψ_n^T and its derivative $\dot{\psi}_n^T$. Proposition 1 establishes their equivalence to nested reverse-mode AD; the proof is provided in Appendix A.

After evaluating $\mathcal{J}^{\mathrm{WSM}} = \mathcal{J}^{\mathrm{WSM,T}} + \mathcal{J}^{\mathrm{WSM,S}}$, training requires its parameter gradient $\nabla_\theta \mathcal{J}^{\mathrm{WSM}}$. Let $P = \dim(\theta)$, where typically $P \gg d_s$. Taylor-mode AD would require P coordinate-wise sweeps to compute this gradient, whereas reverse-mode AD obtains all P components in one backward pass. We therefore use Taylor-mode AD for the event-coordinate derivatives and reverse-mode AD for the outer backpropagation. This hybrid strategy modifies only the inner derivative evaluator, leaving the WSM objective, model architecture, and outer differentiation mode unchanged.

4 Theoretical Analysis

In this section, we develop a program-level framework for comparing two programs induced by Taylor-mode AD and nested reverse-mode AD, and use it to derive the costs of a WSM training step under both strategies, identifying the conditions under which Taylor-mode AD is more efficient.

4.1 A Program-level Computational Cost Framework

We represent a computation as a program using an operation graph $\mathcal{G}$, whose nodes are primitive operations and whose edges encode data dependencies. Its theoretical cost is $W(\mathcal{G}) = \sum_{o \in \mathrm{Ops}(\mathcal{G})} w(o)$, where $\mathrm{Ops}(\mathcal{G})$ is the multiset of executed primitive-operation instances and $w(o)$ is the cost of operation o (Griewank & Walther, 2008). To abstract away hardware-specific factors, we adopt the unit-cost convention $w(o) = 1$, so $W(\mathcal{G})$ counts scalar primitive operations (Baur & Strassen, 1983; Bürgisser et al., 1997). This convention lets us focus on structural differences between the AD-induced programs.

Let d denote the input dimension of the relevant component ($d = d_s$ for spatial, $d = 1$ for temporal). Let $\mathcal{G}_{\mathrm{primal}}$ denote the primal program that evaluates the conditional log-density (or log-intensity) required by the WSM objective, with cost $C = W(\mathcal{G}_{\mathrm{primal}})$. Taylor-mode AD and nested reverse-mode AD transform this primal program into $\mathcal{G}_{\mathrm{tay}}$ and $\mathcal{G}_{\mathrm{nest}}$, respectively, each computing the primal output together with the required first- and second-order derivatives. Their costs are $W(\mathcal{G}_{\mathrm{tay}}) = \tau_{\mathrm{tay}} C$, $W(\mathcal{G}_{\mathrm{nest}}) = \tau_{\mathrm{nest}} C$, where τ_{tay} and τ_{nest} are the corresponding cost multipliers relative to the primal program. Thus, comparing the costs of the two evaluators reduces to comparing τ_{tay} and τ_{nest}, whose explicit forms are derived below.

4.2 Cost Decomposition of A Training Step

A complete WSM training step comprises four stages: context computation, event-coordinate derivative computation, WSM loss assembly, and the outer backpropagation for parameter update. Two WSM training methods use identical programs for context computation and loss assembly but differ in their event-coordinate derivative evaluators. Because the outer backpropagation traverses the derivative program generated by each evaluator, structural differences between the two derivative programs affect the costs of both derivative computation and the outer backpropagation.

Using the notation introduced in Section 3, let $M \in \{\text{Taylor}, \text{Nested}\}$ index the WSM training method. Combining the shared costs of context computation and loss assembly, the training-step cost of method M can be partitioned into three components:

$$C^M_{\text{step}} = C_{\text{shared}} + C^M_{\text{deriv,fwd}} + C^M_{\text{deriv,outer}}. \tag{2}$$

Here, C_{shared} comprises the forward and backward costs of context computation and WSM loss assembly, which are identical for both methods. $C^M_{\text{deriv,fwd}}$ is the forward execution cost of the derivative program for method M, while $C^M_{\text{deriv,outer}}$ is the cost of the outer backpropagation when it differentiates through that program.

4.3 Training-step Costs of Two WSM Training Methods

We now instantiate the cost decomposition in Equation (2) for the two derivative evaluators. The shared cost C_{shared} is identical for both, so we focus on the method-specific components arising from derivative evaluation and the outer backpropagation.

Taylor-mode AD transforms the primal program into the Taylor-lifted program $\mathcal{G}_{\text{tay}}$ by propagating Taylor coefficients up to second order independently along each input dimensions. Let α_2 denote the cost multiplier of second-order Taylor propagation along one direction relative to the primal cost C. Propagation along one direction therefore costs $\alpha_2 C$, and extending this computation to all d input directions gives $C^{\text{Taylor}}_{\text{deriv,fwd}} = W(\mathcal{G}_{\text{tay}}) = \tau_{\text{tay}} C$, $\tau_{\text{tay}} = d\alpha_2$. The outer backpropagation must subsequently traverse the Taylor-lifted program. Let ρ_{tay} denote the ratio of this outer backpropagation cost to the forward execution cost of derivative-program $C^{\text{Taylor}}_{\text{deriv,fwd}}$, such that $C^{\text{Taylor}}_{\text{deriv,outer}} = \rho_{\text{tay}} C^{\text{Taylor}}_{\text{deriv,fwd}}$. Therefore, the training-step cost of Taylor-mode WSM is:

$$C^{\text{Taylor}}_{\text{step}} = C_{\text{shared}} + (1 + \rho_{\text{tay}})\tau_{\text{tay}} C.$$

Nested reverse-mode AD constructs the nested derivative program $\mathcal{G}_{\text{nest}}$ through successive reverse-mode transformations. First, applying reverse-mode AD to the scalar primal output produces the d-dimensional first-order derivatives at an additional cost of $\rho^{(1)}C$, where $\rho^{(1)}$ is the cost ratio of this reverse pass to the primal evaluation. Together, the primal evaluation and the first reverse pass form the *first-order program*, with cost $(1 + \rho^{(1)})C$. Reverse-mode AD is then applied separately to each of the d derivative components to obtain the corresponding diagonal Hessian entries. Let $\rho^{(2)}$ denotes the cost ratio of each second reverse pass to the first-order program, these d passes cost $d\rho^{(2)}(1 + \rho^{(1)})C$. Hence, the forward execution cost of the resulting nested derivative program is $C^{\text{Nested}}_{\text{deriv,fwd}} = W(\mathcal{G}_{\text{nest}}) = \tau_{\text{nest}} C$, $\tau_{\text{nest}} = (1 + \rho^{(1)})(1 + d\rho^{(2)})$. The outer backpropagation subsequently traverses $\mathcal{G}_{\text{nest}}$. Let ρ_{nest} denote its cost relative to the forward execution cost of derivative-program, such that $C^{\text{Nested}}_{\text{deriv,outer}} = \rho_{\text{nest}} C^{\text{Nested}}_{\text{deriv,fwd}}$. The resulting training-step cost is

$$C^{\text{Nested}}_{\text{step}} = C_{\text{shared}} + (1 + \rho_{\text{nest}})\tau_{\text{nest}} C.$$

4.4 Sufficient Conditions for Taylor-Mode WSM Speedup

Based on the preceding cost analysis, we identify two sufficient conditions that jointly ensure that Taylor has a lower training-step cost than Nested.

Condition 1 (Derivative-computation cost). Derivative computation with the Taylor-lifted program has a lower cost than with the nested derivative program if $\tau_{\text{tay}} < \tau_{\text{nest}}$.

Condition 2 (Outer-backpropagation cost). Outer backpropagation through the Taylor-lifted program has a lower cost than through the nested derivative program if $\tau_{\text{tay}} < \tau_{\text{nest}}$ and $\rho_{\text{tay}} \leq \rho_{\text{nest}}$.

When both conditions are satisfied, $C^{\text{Taylor}}_{\text{step}} < C^{\text{Nest}}_{\text{step}}$. We quantify this advantage using the training-step speedup ratio

$$R = \frac{C^{\text{Nest}}_{\text{step}}}{C^{\text{Taylor}}_{\text{step}}} = \frac{C_{\text{shared}} + (1 + \rho_{\text{nest}})\tau_{\text{nest}} C}{C_{\text{shared}} + (1 + \rho_{\text{tay}})\tau_{\text{tay}} C},$$

where $R > 1$ indicates that Taylor-mode WSM has a lower per-step cost. The speedup depends on the shared cost relative to the method-specific costs. When C_{shared} is negligible, $R \approx (1 +$

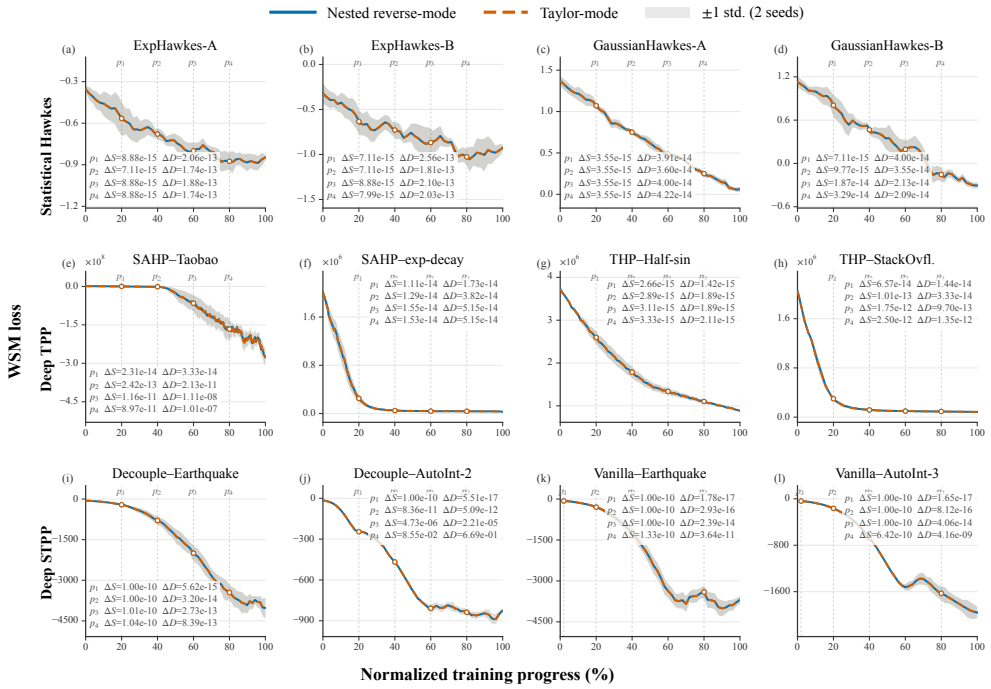


Figure 1: Consistency between TAYLOR (dashed orange) and NESTED (solid blue). The curves and shaded regions show the mean WSM training loss and one standard deviation over two random seeds. The checkpoints p_1–p_4 correspond to 20%, 40%, 60%, and 80% of training progress. At each checkpoint, ΔS and ΔD denote the maximum absolute differences between the score and divergence outputs of the two derivative evaluators.

$\rho_{\text{nest}})\tau_{\text{nest}}/((1+\rho_{\text{tay}})\tau_{\text{tay}})$ and the cost difference is pronounced. Conversely, as C_{shared} becomes dominant, $R \approx 1$, even if conditions remain satisfied. Thus, shared computation dilutes the speedup arising from differences between the two derivative programs. We call this phenomenon the *shared-cost dilution effect*. Section 5 verifies the two conditions and quantifies this effect.

5 EXPERIMENTS

We compare TAYLOR with NESTED by addressing five questions: **(1)** Does TAYLOR preserve the performance of NESTED? **(2)** How does the speedup vary across model families, and is this variation consistent with our theoretical analysis? **(3)** Do the two sufficient conditions hold empirically, and which stage contributes most to the time savings? **(4)** How does the speedup scale with derivative order? **(5)** How does it scale with event-coordinate dimensionality?

5.1 EXPERIMENTAL SETUP

We compare TAYLOR and NESTED across three point-process model families. For statistical point processes, we use four synthetic bivariate Hawkes-process datasets with **Exponential** or **Gaussian triggering kernels**. For deep TPPs, we evaluate **SAHP** (Zhang et al., 2020) and **THP** (Zuo et al., 2020) on three synthetic datasets (*conditional-logistic*, *exp-decay*, and *half-sin*) and two real-world datasets (*StackOverflow* (Leskovec & Krevl, 2014) and *Taobao* (Xue et al., 2022)). For deep STPPs, we evaluate **SMASH** (Li et al., 2024), a THP-based architecture, with Vanilla and Decouple intensity parameterizations on three synthetic datasets (*SC1*, *AutoIntHawkes-2*, and *AutoIntHawkes-3*) and the real-world *Japan Earthquake* dataset (U.S. Geological Survey, 2020). Appendix B provides details on the datasets, preprocessing procedures, and model architectures.

Table 1: Speedups of TAYLOR over NESTED for WSM forward pass, outer backpropagation, and training step across statistical Hawkes processes, deep TPPs, and deep STPPs. For $r \in \{\text{forward}, \text{outer}, \text{step}\}$, $R_r = T^r_{\text{nest}}/T^r_{\text{tay}} > 1$ favors TAYLOR. Moderate training-step speedups are shown in **bold blue** ($1.5 \le R_{\text{step}} < 3$); larger speedups are shown in **bold red** ($R_{\text{step}} \ge 3$).

Model	Dataset	R_{fwd}	R_{outer}	R_{step}	Model	Dataset	R_{fwd}	R_{outer}	R_{step}
STATISTICAL HAWKES									
Exp.	ExpHawkes-A	1.00	1.41	1.32	**Gau.**	GauHawkes-A	1.01	1.48	1.36
	ExpHawkes-B	1.00	1.35	1.28		GauHawkes-B	1.03	1.39	1.31
DEEP TPPS									
SAHP	Cond.-logit	1.07	1.98	**1.79**	**THP**	Cond.-logit	1.09	2.02	**1.83**
	Exp.-decay	1.12	2.48	**2.18**		Exp.-decay	1.08	2.44	**2.13**
	Half-sin	1.08	1.13	1.13		Half-sin	1.10	1.07	1.07
	Taobao	1.10	0.92	1.04		Taobao	1.07	1.10	1.08
	StackOvfl.	1.12	0.97	1.02		StackOvfl.	1.15	1.07	1.10
DEEP STPPS									
Vanilla	SC1	1.58	6.77	**5.59**	**Decouple**	SC1	1.58	2.20	**2.09**
	AutoInt-2	1.49	5.42	**5.28**		AutoInt-2	1.40	5.10	**4.98**
	AutoInt-3	1.47	3.73	**3.67**		AutoInt-3	1.47	3.77	**3.73**
	Earthquake	1.52	2.72	**2.70**		Earthquake	1.44	2.79	**2.77**

All experiments use a NVIDIA GeForce RTX 5090 GPU with JAX 0.6.2 and CUDA 12. Within each model–dataset configuration, TAYLOR and NESTED share the same initialization, architecture, WSM objective, optimizer, and hyperparameters. Runtime measurements exclude data loading, JAX compilation, and auxiliary operations; forward runtime covers the WSM forward pass with event-coordinate derivative evaluation, while training-step runtime additionally includes the outer backpropagation and optimizer update. Appendix C reports the full training and timing settings.

5.2 NUMERICAL CONSISTENCY

Before comparing efficiency, we verify that replacing nested reverse-mode AD with Taylor-mode AD for event-coordinate derivatives preserves optimization behavior. For each model–dataset configuration, we record the WSM loss, score (first-order derivative), and divergence (second-order derivative) during training and measure the maximum absolute differences of the score and divergence outputs at multiple checkpoints. Figure 1 shows representative results across three model families. Across all configurations, TAYLOR and NESTED exhibit nearly identical loss trajectories and numerical-precision differences in score and divergence, confirming that replacing evaluators does not materially affecting optimization behavior.

5.3 ANALYSIS OF SPEEDUP SOURCES

We evaluate the practical training-step efficiency of TAYLOR and NESTED across all model–dataset configurations using wall-clock runtime. For each method M, we measure the WSM forward-pass runtime T^M_{forward} and complete training-step runtime T^M_{step}, and define $T^M_{\text{outer}} = T^M_{\text{step}} - T^M_{\text{forward}}$ as the outer backpropagation contribution. This quantity includes the optimizer update,which is shared across methods and does not affect the comparison. For $r \in \{\text{forward}, \text{outer}, \text{step}\}$, we define the empirical speedup $R_r = T^{\text{Nested}}_r / T^{\text{Taylor}}_r$, where $R_r > 1$ indicates that TAYLOR is faster. All measurements are reported in Appendix D.

Speedup patterns. Table 1 reports R_{forward}, R_{outer}, and R_{step} across the three model families. For R_{step}, statistical Hawkes processes exhibit modest gains of $1.28\times$–$1.36\times$, with a configuration-level average of $1.32\times$. Deep TPPs achieve a larger average speedup of $1.43\times$, ranging from $1.02\times$ to $2.18\times$, whereas deep STPPs exhibit the most pronounced gains, ranging from $2.09\times$ to $5.59\times$, with an average of $3.85\times$. Overall, $R_{\text{step}} > 1$ holds consistently across all model families, though the magnitude of the improvement varies substantially among them.

Shared-cost dilution. To assess whether this variation reflects the shared-cost dilution effect predicted in Section 4, we compute the shared-cost ratio $\widehat{\eta}^M_{\text{shared}} = (F^M_{\text{stage1}} + F^M_{\text{stage3}})/(F^M_{\text{stage1}} + F^M_{\text{stage2}} + F^M_{\text{stage3}})$ where Stages 1–3 denote context computation, method-specific derivatives, and loss assembly. Figure 2 shows that $\widehat{\eta}^M_{\text{shared}}$ drops from 97.8% for statistical Hawkes to 66.0% for deep STPPs, while R_{step} rises from $1.32\times$ to $3.85\times$. The decline in $\widehat{\eta}^M_{\text{shared}}$ reflects the increas-

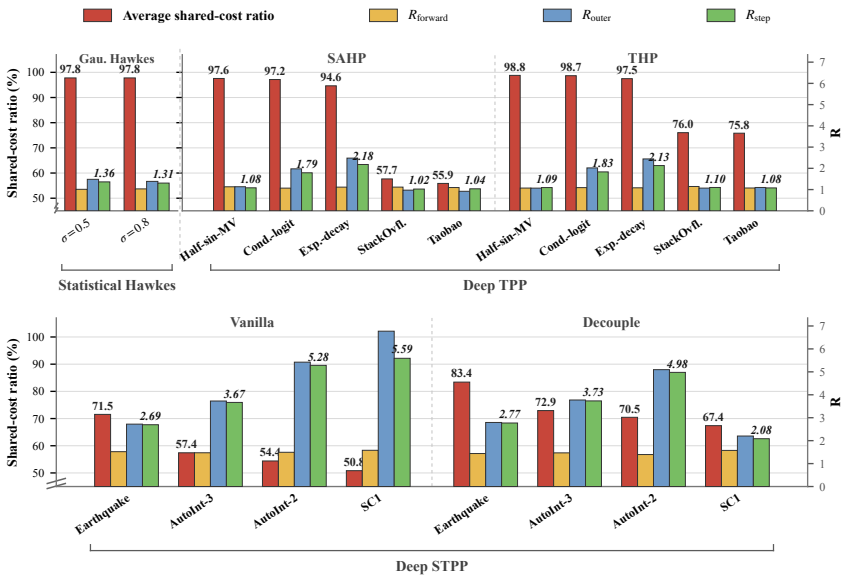


Figure 2: Family-average FLOP-based shared-cost ratio (red, left axis) and empirical runtime speedups for the WSM forward pass (R_{forward}, yellow), outer backpropagation (R_{outer}, blue), and complete training step (R_{step}, green). The shared cost includes context-computation and loss-assembly FLOPs, and runtime ratios above one favor TAYLOR. From statistical Hawkes processes to deep STPPs, the shared-cost ratio decreases while the training-step speedup increases.

Table 2: Stage-wise evidence for the sufficient conditions and speedup attribution on deep STPPs. $R_{\text{outer}}/R_{\text{forward}} \geq 1$ and $\widehat{\rho}_N^{\text{FLOPs}} \geq \widehat{\rho}_T^{\text{FLOPs}}$ support the sufficient conditions. The ordered pair $(A_{\text{outer}}, A_{\text{forward}})$ attributes the training-step time saved to the two stages.

Dataset	Vanilla			Decouple		
	$R_{\text{outer}}/R_{\text{forward}}$	$(\widehat{\rho}_N, \widehat{\rho}_T)$	$(A_{\text{outer}}, A_{\text{forward}})$	$R_{\text{outer}}/R_{\text{forward}}$	$(\widehat{\rho}_N, \widehat{\rho}_T)$	$(A_{\text{outer}}, A_{\text{forward}})$
SC1	4.310	(3.579, 2.955)	(97.1%, 2.9%)	1.392	(3.783, 1.979)	(90.4%, 9.6%)
AutoInt-2	3.671	(3.579, 2.954)	(99.5%, 0.5%)	3.691	(3.783, 1.979)	(99.6%, 0.4%)
AutoInt-3	2.544	(3.579, 2.954)	(99.4%, 0.6%)	2.585	(3.783, 1.979)	(99.5%, 0.5%)
Earthquake	1.789	(3.579, 2.954)	(99.3%, 0.7%)	1.938	(3.783, 1.979)	(99.6%, 0.4%)

ing cost of derivative computation across model families: statistical Hawkes requires only a few elementary operations, deep TPPs involve more expensive temporal derivatives due to their neural architectures, and deep STPPs incur additional costs from derivatives over both temporal and spatial coordinates. As Stage 2 expands, the shared stages occupy a smaller fraction of the training step, so TAYLOR's savings on derivative computation are less diluted and translate into a larger R_{step}, consistent with the shared-cost dilution effect.

Sufficient conditions. The preceding analysis explains when speedups are most pronounced, but not why TAYLOR is faster. We therefore verify the two sufficient conditions from Section 4.4 on deep STPPs, where the largest speedups are observed. The verification relies on three quantities: the forward speedup R_{forward}, the outer-to-forward speedup ratio $R_{\text{outer}}/R_{\text{forward}}$, and the FLOP amplification factor $\hat{\rho}_M^{\text{FLOPs}} = F^M_{\text{deriv,outer}}/F^M_{\text{deriv,forward}}$ (see Appendix E). For the first condition, Table 1 reports $R_{\text{forward}} \geq 1$ in all configurations, confirming TAYLOR's lower forward derivative cost. For the second condition, Table 2 reports $R_{\text{outer}}/R_{\text{forward}} > 1$ together with lower FLOP amplification factors for TAYLOR. Because the outer backpropagation cost equals the forward derivative cost scaled by this amplification factor, a smaller $\hat{\rho}_M^{\text{FLOPs}}$ means TAYLOR's forward advantage is further magnified in outer backpropagation. Thus, TAYLOR's speedup stems from both lower forward derivative cost and greater outer backpropagation savings.

Speedup attribution. Having verified both conditions, we next quantify their respective contributions to the overall training-step speedup. To this end, we define $\Delta T_r = T_r^{\text{Nested}} - T_r^{\text{Taylor}}$ as the run-

Table 3: Scaling of TAYLOR's speedup on deep STPPs with derivative order and event-coordinate dimensionality. **Bold** denotes $R_{\text{step}} > 3$. (a) WSM (second-order method) yields larger R_{step} than DSM (first-order method). (b) As dimension d_s increases, R_{fwd} drops below 1, while $R_{\text{step}} > 3$.

Method	Dataset	(a) Effect of derivative order				(b) Effect of event-coordinate dimensionality							
		DSM		WSM		$d_s = 4$		$d_s = 6$		$d_s = 8$		$d_s = 16$	
		R_{fwd}	R_{step}	R_{fwd}	R_{step}	R_{fwd}	R_{step}	R_{fwd}	R_{step}	R_{fwd}	R_{step}	R_{fwd}	R_{step}
Vanilla	AutoInt-2	1.099	1.700	1.490	**5.283**	0.882	**6.125**	0.670	**5.877**	0.510	**5.427**	0.283	**3.813**
	AutoInt-3	1.718	1.453	1.470	**3.667**	0.928	**3.121**	0.666	**3.777**	0.503	**3.815**	0.281	**3.535**
Decouple	AutoInt-2	1.151	1.639	1.390	**4.977**	1.056	**6.617**	0.801	**6.566**	0.670	**5.438**	0.389	**4.194**
	AutoInt-3	1.085	1.460	1.470	**3.733**	1.025	**3.432**	0.781	**4.200**	0.682	**4.094**	0.408	**4.399**

time saving at stage r for $r \in \{\text{forward}, \text{outer}, \text{step}\}$, and $A_{\text{outer}} = \Delta T_{\text{outer}}/\Delta T_{\text{step}}$ as the fraction of the total training-step saving attributable to outer backpropagation, with $A_{\text{forward}} = 1 - A_{\text{outer}}$. As shown in Table 2, outer backpropagation accounts for 90.4%–99.6% of the training-step savings, leaving the forward stage a marginal contributor. Hence, while both conditions hold, TAYLOR's acceleration is overwhelmingly driven by outer backpropagation savings.

Taken together, the shared-cost dilution effect explains *when* speedups become pronounced, while sufficient conditions and attribution analyses explain *why* they occur and *which* dominates: TAYLOR computes derivatives more efficiently, and outer backpropagation savings account for most training-step speedup.

5.4 Effect of Derivative Order

To examine the effect of derivative order on TAYLOR's speedup, we replace WSM, requiring second-order derivatives, with denoising score matching (DSM), requiring only first-order derivatives. Following Li et al. (2023), let $\xi_{T,n} \sim \mathcal{N}(0,1)$ and $\boldsymbol{\xi}_{S,n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_s})$ be independent noises, with scales σ_T and σ_S, and denote the first-order scores at perturbed coordinates by $\tilde{\psi}_{T,n}$ and $\tilde{\boldsymbol{\psi}}_{S,n}$. The DSM objective is $\mathcal{L}^{\text{DSM}}(\theta) = \mathbb{E}_{\mathcal{X}\sim p} \sum_{n=1}^{N} \mathbb{E}_{\xi_{T,n},\boldsymbol{\xi}_{S,n}} \left[\frac{\sigma_T^2}{2} \left(\tilde{\psi}_{T,n} + \frac{\xi_{T,n}}{\sigma_T} \right)^2 + \frac{\sigma_S^2}{2} \left\| \tilde{\boldsymbol{\psi}}_{S,n} + \frac{\boldsymbol{\xi}_{S,n}}{\sigma_S} \right\|^2 \right]$. At this reduced derivative order, Panel (a) of Table 3 shows that R_{step} decreases from $3.67\times$–$5.28\times$ under WSM to $1.45\times$–$1.70\times$ under DSM, indicating that TAYLOR's advantage grows with derivative order. The gap stems from WSM's second-order derivative terms: NESTED requires d_s extra reverse passes, increasing the outer-backpropagation cost, whereas TAYLOR propagates them in a single forward pass. Removing them under DSM substantially reduces this gap.

5.5 Effect of Event-Coordinate Dimensionality

To examine scaling with event-coordinate dimension, we vary the spatial dimensionality of the WSM objective from $d_s = 4$ to 16. Panel (b) of Table 3 shows that R_{forward} decreases monotonically and falls below 1 by $d_s = 6$, yet R_{step} remains above $3\times$, consistent with the cost model. For the forward stage, NESTED's cost includes a dimension-independent term for computing the first-order score, whereas TAYLOR's scales with d_s. Thus, $R_{\text{forward}}(d_s) \approx \frac{(1+\rho^{(1)})\rho^{(2)}}{\alpha_2} + \frac{1+\rho^{(1)}}{\alpha_2 d_s}$, which decreases as the fixed term is amortized, reducing TAYLOR's forward advantage. For the training step, however, the cost ratio satisfies $\text{sign}\left(\frac{\partial R_{\text{step}}}{\partial d_s}\right) = \text{sign}\left[C_{\text{shared}}\left(A\rho^{(2)} - B\alpha_2\right) - AB\alpha_2 C\right]$, where $A = (1+\rho_{\text{nest}})(1+\rho^{(1)})$ and $B = 1 + \rho_{\text{tay}}$. Since these terms depend on model structure and hardware execution, R_{step} need not vary monotonically with d_s. The observed $R_{\text{step}} > 3\times$ thus indicates that outer-backpropagation savings outweigh the forward penalty, preserving TAYLOR's step advantage at higher dimensions.

6 Conclusion

We proposed Taylor-mode WSM, which uses Taylor-mode for event-coordinate derivatives and reverse-mode parameter update. Theory and experiments characterize its speedup conditions and identify outer backpropagation as the dominant source. Across all model families, our method achieved $1.32\times$–$3.85\times$ average and up to $5.6\times$ speedups without performance loss, with outer backpropagation contributing 90.4%–99.6% of the savings.

AI USE STATEMENT

AI-based tools were used during the preparation of this manuscript for limited purposes, including language editing, proofreading, and assisting with code implementation. All mathematical derivations, theoretical results, algorithmic designs, and scientific conclusions were developed by the authors. All AI-assisted code was manually reviewed, verified, and validated by the authors, and all experimental implementations and empirical analyses were independently checked by the authors.

REFERENCES

Emmanuel Bacry and Jean-François Muzy. Hawkes model for price and trades high-frequency dynamics. *Quantitative Finance*, 14(7):1147–1166, 2014.

Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical computer science*, 22(3):317–330, 1983.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.

Jesse Bettencourt, Matthew J Johnson, and David Duvenaud. Taylor-mode automatic differentiation for higher-order derivatives in jax. In *Program transformations for ML workshop at NeurIPS 2019*, 2019.

Christian Bischof, George Corliss, and Andreas Griewank. Structured second-and higher-order derivatives through univariate taylor series. *Optimization Methods and Software*, 2(3-4):211–232, 1993.

Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin, Heidelberg, 1997. doi: 10.1007/978-3-662-03338-8.

Haoqun Cao, Zizhuo Meng, Tianjun Ke, and Feng Zhou. Is score matching suitable for estimating point processes? *Advances in Neural Information Processing Systems*, 37:23824–23851, 2024.

Haoqun Cao, Yixuan Zhang, and Feng Zhou. Score matching for estimating finite point processes. *arXiv preprint arXiv:2512.04617*, 2025.

Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer, 2003.

Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Andreas Griewank, Jean Utke, and Andrea Walther. Evaluating higher derivative tensors by forward propagation of univariate taylor series. *Mathematics of computation*, 69(231):1117–1130, 2000.

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

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL `https://www.jmlr.org/papers/v6/hyvarinen05a.html`.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `https://snap.stanford.edu/data/`, June 2014.

Zichong Li, Yanbo Xu, Simiao Zuo, Haoming Jiang, Chao Zhang, Tuo Zhao, and Hongyuan Zha. Smurf-thp: score matching-based uncertainty quantification for transformer hawkes process. In *International Conference on Machine Learning*, pp. 20210–20220. PMLR, 2023.

Zichong Li, Qunzhi Xu, Zhenghao Xu, Yajun Mei, Tuo Zhao, and Hongyuan Zha. Beyond point prediction: Score matching-based pseudolikelihood estimation of neural marked spatio-temporal point process. In *Forty-first International Conference on Machine Learning*, 2024.

Scott W Linderman and Ryan P Adams. Scalable bayesian inference for excitatory point process networks. *arXiv preprint arXiv:1507.03228*, 2015.

George Mohler. Modeling and estimation of multi-source clustering in crime and security data. *The Annals of Applied Statistics*, 7(3), September 2013. ISSN 1932-6157. doi: 10.1214/13-aoas647. URL http://dx.doi.org/10.1214/13-AOAS647.

Yosihiko Ogata. On lewis' simulation method for point processes. *IEEE transactions on information theory*, 27(1):23–31, 1981.

Yosihiko Ogata. Space-time point-process models for earthquake occurrences. *Annals of the Institute of Statistical Mathematics*, 50(2):379–402, 1998.

Yosihiko Ogata. Seismicity analysis through point-process modeling: A review. *Pure and Applied Geophysics*, 155(2–4):471–507, 1999. doi: 10.1007/s000240050275.

Jakob Gulddahl Rasmussen. Lecture notes: Temporal point processes and the conditional intensity function. *arXiv preprint arXiv:1806.00221*, 2018.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pp. 318–362. MIT Press, 1986.

Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. *arXiv preprint arXiv:2104.03528*, 2021.

U.S. Geological Survey. Search earthquake catalog. https://earthquake.usgs.gov/earthquakes/search/, 2020. Accessed: 2020-08-21.

Siqiao Xue, Xiaoming Shi, James Zhang, and Hongyuan Mei. Hypro: A hybridly normalized probabilistic model for long-horizon prediction of event sequences. *Advances in Neural Information Processing Systems*, 35:34641–34650, 2022.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *International conference on machine learning*, pp. 11183–11193. PMLR, 2020.

Feng Zhou, Quyu Kong, Zhijie Deng, Jichao Kan, Yixuan Zhang, Cheng Feng, and Jun Zhu. Efficient inference for dynamic flexible interactions of neural populations. *Journal of Machine Learning Research*, 23(211):1–49, 2022a.

Zihao Zhou and Rose Yu. Automatic integration for fast and interpretable neural point processes. In *Learning for Dynamics and Control Conference*, pp. 573–585. PMLR, 2023a.

Zihao Zhou and Rose Yu. Automatic integration for spatiotemporal neural point processes. *Advances in Neural Information Processing Systems*, 36:50237–50253, 2023b.

Zihao Zhou, Xingyi Yang, Ryan Rossi, Handong Zhao, and Rose Yu. Neural point process for learning spatiotemporal event dynamics. In *Learning for Dynamics and Control Conference*, pp. 777–789. PMLR, 2022b.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.

A EQUIVALENCE OF THE EVENT-COORDINATE DERIVATIVE EVALUATORS

This section states the regularity conditions and proves the equivalence of the event-coordinate derivatives computed by TAYLOR and NESTED in Section 3.

A.1 REGULARITY ASSUMPTION

Assumption 1 (Event-coordinate regularity)**.** *For fixed model parameters θ, each event n, and fixed history $\mathcal{H}_{n-1}$, the map*

$$\mathbf{s} \longmapsto \log q_\theta(\mathbf{s} \mid t_n, \mathcal{H}_{n-1})$$

is twice continuously differentiable in an open neighborhood of $\mathbf{s}_n$. Likewise, the map

$$t \longmapsto \log f_\theta(t \mid \mathcal{H}_{n-1})$$

is twice continuously differentiable in an open neighborhood of t_n. Their evaluation programs are compositions of twice continuously differentiable primitive operations on these neighborhoods, with correct first- and second-order differentiation rules.

The parameters and history are held fixed when taking event-coordinate derivatives; the spatial derivatives additionally hold the event time t_n fixed. The assumption ensures that the second-order Taylor expansions and the corresponding AD computations are well defined.

A.2 EQUIVALENCE OF TAYLOR-MODE EVENT-COORDINATE DERIVATIVES

Proposition 1 (Taylor-mode equivalence)**.** *Under Assumption 1, in exact arithmetic, coordinate-wise second-order Taylor-mode AD and nested reverse-mode AD recover the same quantities*

$$\left(\log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1}), \boldsymbol{\psi}_n^S, \delta_n^S\right) \quad \textit{and} \quad \left(\log f_\theta(t_n \mid \mathcal{H}_{n-1}), \psi_n^T, \dot{\psi}_n^T\right).$$

With all other loss terms and weights held fixed, the two evaluators therefore yield the same WSM objective in Equation (1)*.*

Proof. Fix θ, an event n, and its history $\mathcal{H}_{n-1}$.

Spatial derivatives. For each coordinate $j \in \{1, \dots, d_s\}$, let $\mathbf{e}_j$ be the j-th coordinate unit vector. Following Section 3, define

$$F_{n,j}(\epsilon_s) = \log q_\theta(\mathbf{s}_n + \epsilon_s \mathbf{e}_j \mid t_n, \mathcal{H}_{n-1}).$$

Taylor's theorem gives

$$F_{n,j}(\epsilon_s) = F_{n,j}(0) + \epsilon_s F'_{n,j}(0) + \frac{\epsilon_s^2}{2} F''_{n,j}(0) + o(\epsilon_s^2). \tag{3}$$

By the chain rule,

$$F'_{n,j}(0) = \mathbf{e}_j^\top \nabla_{\mathbf{s}} \log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1}) = [\boldsymbol{\psi}_n^S]_j,$$
$$F''_{n,j}(0) = \mathbf{e}_j^\top \nabla_{\mathbf{s}}^2 \log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1}) \mathbf{e}_j = \partial_{s_j}^2 \log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1}).$$

Second-order Taylor-mode AD propagates the truncated expansion through the primitive operations of the evaluation program. Under Assumption 1, each operation preserves the expansion through second order, so their composition recovers the coefficients in Equation (3). The coefficients of ϵ_s and ϵ_s^2 are $F'_{n,j}(0)$ and $F''_{n,j}(0)/2!$; the second derivative is recovered by multiplying the latter by $2!$.

Collecting the first derivatives over all d_s coordinate directions gives $\boldsymbol{\psi}_n^S$. Summing the second derivatives gives

$$\sum_{j=1}^{d_s} F''_{n,j}(0) = \mathrm{Tr}\big(\nabla_{\mathbf{s}}^2 \log q_\theta(\mathbf{s}_n \mid t_n, \mathcal{H}_{n-1})\big) = \delta_n^S.$$

Thus, the spatial WSM term requires only the diagonal Hessian entries, not the mixed spatial derivatives.

Table 4: Parameters of the four synthetic statistical Hawkes datasets.

Dataset	$\boldsymbol{\mu}$	$\boldsymbol{A}$	**Kernel parameter**
ExpHawkes-A	$[0.50, 0.70]$	$\left[\begin{smallmatrix} 0.80 & 0.20 \\ 0.15 & 0.60 \end{smallmatrix}\right]$	$\beta = 1.50$
ExpHawkes-B	$[0.30, 0.40]$	$\left[\begin{smallmatrix} 1.20 & 0.35 \\ 0.25 & 0.90 \end{smallmatrix}\right]$	$\beta = 2.00$
GaussianHawkes-A	$[0.50, 0.70]$	$\left[\begin{smallmatrix} 0.80 & 0.20 \\ 0.15 & 0.60 \end{smallmatrix}\right]$	$\sigma = 0.50$
GaussianHawkes-B	$[0.30, 0.40]$	$\left[\begin{smallmatrix} 1.20 & 0.35 \\ 0.25 & 0.90 \end{smallmatrix}\right]$	$\sigma = 0.80$

Temporal derivatives. For the temporal restriction from Section 3,

$$G_n(\epsilon_t) = \log f_\theta(t_n + \epsilon_t \mid \mathcal{H}_{n-1}),$$

the corresponding expansion is

$$G_n(\epsilon_t) = G_n(0) + \epsilon_t G_n'(0) + \frac{\epsilon_t^2}{2} G_n''(0) + o(\epsilon_t^2),$$

with

$$G_n'(0) = \partial_t \log f_\theta(t_n \mid \mathcal{H}_{n-1}) = \psi_n^T, \qquad G_n''(0) = \partial_t^2 \log f_\theta(t_n \mid \mathcal{H}_{n-1}) = \dot{\psi}_n^T.$$

The same coefficient propagation and factorial conversion therefore recover both temporal quantities.

Equality of the WSM objectives. Nested reverse-mode AD obtains the spatial gradient and diagonal Hessian entries, and the first and second temporal derivatives, by successive applications of the chain rule to the same functions. It therefore recovers the same quantities as Taylor-mode AD. Substituting these quantities into Equation (1), with the remaining loss terms and weights unchanged, gives identical WSM objective values. □

B Experimental Details

This appendix describes the datasets, preprocessing procedures, and model architectures used in our experiments. The contents are organized by model family. For each family, we specify the model parameterization or architecture, the data-generating mechanisms or data sources, and the preprocessing applied before training.

B.1 Statistical Hawkes Processes

B.1.1 Model Parameterization

The statistical experiments directly fit the parameters of a bivariate Hawkes process. The conditional intensity of event type $i \in \{1, 2\}$ is

$$\lambda_i(t \mid \mathcal{H}_t) = \mu_i + \sum_{j=1}^{2} \sum_{t_n^{(j)} < t} A_{ij}\, g(t - t_n^{(j)}), \tag{4}$$

where $\boldsymbol{\mu} \in \mathbb{R}_+^2$ is the background intensity and $\boldsymbol{A} \in \mathbb{R}_+^{2\times 2}$ is the excitation matrix. For the exponential-decay parameterization, $g(\tau) = \exp(-\beta\tau)$; for the Gaussian-kernel parameterization, $g(\tau) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\tau^2}{2\sigma^2}\right)$.

B.1.2 Datasets and Preprocessing

We construct four synthetic bivariate Hawkes-process datasets. Each configuration contains $2{,}000$ sequences observed on $[0, 100]$, generated using Ogata's thinning algorithm (Ogata, 1981). Sequences are padded to the maximum length of each dataset, with padded events excluded from the WSM objective using an explicit validity mask. Table 4 lists the data-generating parameters.

Table 5: Model architecture settings for the deep TPP experiments.

Setting	SAHP	THP
Model dimension d_{model}	16	16
Inner dimension d_{inner}	8	8
Attention heads	2	2
Encoder layers	1	1
Key/value dimensions	–	$d_k = d_v = 8$
Dropout	0.10	0.10
Position buffer length	5,000	–

B.2 Deep Temporal Point Processes

B.2.1 Model Architectures

We evaluate the Self-Attentive Hawkes Process (SAHP) (Zhang et al., 2020) and the Transformer Hawkes Process (THP) (Zuo et al., 2020). SAHP uses one self-attention block, while THP uses one Transformer encoder layer. Table 5 summarizes their configurations. The inner dimension refers to the survival MLP in SAHP and the Transformer feed-forward network in THP.

B.2.2 Datasets and Preprocessing

We evaluate both models on three synthetic and two real-world datasets.

Conditional-logistic. This univariate process has intensity $c_0 = 2$ before the first event and $\lambda(t \mid \mathcal{H}_t) = c/[1 + \beta \exp(c(t - t_{\text{last}}))]$ thereafter, where t_{last} is the most recent event time, $c = 2$, and $\beta = 0.02$.

Exp-decay and half-sin. Both are bivariate Hawkes processes with excitation matrix $\boldsymbol{A} = \left[\begin{smallmatrix} 0.33 & 0.10 \\ 0.05 & 0.33 \end{smallmatrix}\right]$. For *exp-decay*, $g_{ij}(\tau) = A_{ij} \exp(-5\tau)$ and $\boldsymbol{\mu} = [2.00, 2.50]$; for *half-sin*, $g_{ij}(\tau) = A_{ij} \sin(\tau) \mathbf{1}_{\{0<\tau<\pi\}}$ and $\boldsymbol{\mu} = [1.00, 1.50]$.

Real-world datasets. StackOverflow contains user badge sequences with 22 event types (Leskovec & Krevl, 2014), while *Taobao* contains user activity sequences with 17 event types (Cao et al., 2024). We use the provided training, validation, and test splits for all five datasets. Synthetic event times retain their original scales. Each real-world sequence is shifted so that its first event occurs at time zero, after which a dummy event at time zero is prepended; no additional time rescaling is applied.

B.3 Deep Spatio-Temporal Point Processes

B.3.1 Model Architecture

We adopt the THP-based neural spatio-temporal point process (STPP) architecture from SMASH (Li et al., 2024). Its temporal, spatial, and joint Transformer streams each contain four layers and produce the corresponding representations of $\mathcal{H}_{n-1}$, denoted by $\mathbf{c}_n^T$, $\mathbf{c}_n^S$, and $\mathbf{c}_n^J$, respectively. Let $\Delta t = t - t_{n-1}$.

Shared formulation. Using the temporal ground intensity and conditional spatial density defined in Section 2.1, each parameterization specifies $\lambda_\theta^*(t)$ and an unnormalized spatial density $\widetilde{q}_\theta$. We normalize the latter as

$$q_\theta(\mathbf{s} \mid t, \mathcal{H}_{n-1}) = \frac{\widetilde{q}_\theta(\mathbf{s} \mid t, \mathcal{H}_{n-1})}{\int_{\mathcal{S}} \widetilde{q}_\theta(\mathbf{r} \mid t, \mathcal{H}_{n-1})\, d\mathbf{r}},$$

where the integral is approximated by the mean of $\widetilde{q}_\theta$ on a 10×10 uniform spatial grid, multiplied by the area of $\mathcal{S}$.

Vanilla. The temporal and spatial streams apply an initial affine projection followed by three history-conditioned updates:

$$\begin{aligned}
\mathbf{v}^T_{n,0} &= \Phi^T_0(\Delta t),\\
\mathbf{v}^T_{n,\ell} &= \Phi^T_\ell\big(\mathrm{softplus}\big(\mathbf{v}^T_{n,\ell-1} + \Phi^T_{\ell,T}(\mathbf{c}^T_n) + \Phi^T_{\ell,J}(\mathbf{c}^J_n)\big)\big)\,,\\
\mathbf{v}^S_{n,0} &= \Phi^S_0(\mathbf{s}),\\
\mathbf{v}^S_{n,\ell} &= \Phi^S_\ell\big(\mathrm{softplus}\big(\mathbf{v}^S_{n,\ell-1} + \Phi^S_{\ell,S}(\mathbf{c}^S_n) + \Phi^S_{\ell,J}(\mathbf{c}^J_n)\big)\big)\,, \qquad \ell = 1,2,3,
\end{aligned}$$

where each Φ is a learned affine map. The temporal intensity is

$$\lambda^*_{\theta,\mathrm{Vanilla}}(t) = \Gamma^{\mathrm{V}}_T(\mathbf{v}^T_{n,3}),$$

and the spatial density is

$$\widetilde{q}_{\theta,\mathrm{Vanilla}}(\mathbf{s} \mid t, \mathcal{H}_{n-1}) = \Gamma^{\mathrm{V}}_S\big(\mathrm{concat}\big[\omega^T_{n,1}\mathbf{v}^T_{n,3} + \omega^T_{n,2}\mathbf{v}^S_{n,3},\, \omega^S_{n,1}\mathbf{v}^T_{n,3} + \omega^S_{n,2}\mathbf{v}^S_{n,3}\big]\big)\,.$$

The weights are $\boldsymbol{\omega}^T_n = \mathrm{softmax}(\Omega_T(\mathbf{u}_n))$ and $\boldsymbol{\omega}^S_n = \mathrm{softmax}(\Omega_S(\mathbf{u}_n))$, where

$$\mathbf{u}_n = C\big(\mathrm{concat}[\mathbf{c}^T_n, \mathbf{c}^S_n, \mathbf{c}^J_n]\big)\,.$$

The shared network C contains two affine layers with an intermediate ReLU, while each Ω contains three affine layers with ReLU activations. Both output heads Γ^{V}_T and Γ^{V}_S use affine–softplus–affine–softplus.

Decouple. The temporal and spatial components use separate projection networks and output heads:

$$\begin{aligned}
\lambda^*_{\theta,\mathrm{Decouple}}(t) &= \Gamma^{\mathrm{D}}_T\big(\Phi^T_{\Delta t}(\Delta t) + \Phi^T_T(\mathbf{c}^T_n) + \Phi^T_S(\mathbf{c}^S_n) + \Phi^T_J(\mathbf{c}^J_n)\big)\,,\\
\widetilde{q}_{\theta,\mathrm{Decouple}}(\mathbf{s} \mid t, \mathcal{H}_{n-1}) &= \Gamma^{\mathrm{D}}_S\big(\Phi^S_{\mathbf{s}}(\mathbf{s}) + \Phi^S_{\Delta t}(\Delta t) + \Phi^S_T(\mathbf{c}^T_n) + \Phi^S_S(\mathbf{c}^S_n) + \Phi^S_J(\mathbf{c}^J_n)\big)\,.
\end{aligned}$$

Each Φ contains two affine layers with an intermediate SiLU. Each Γ^{D} applies SiLU–affine–SiLU–affine–softplus, followed by a 10^{-8} offset. Both components condition on all three history representations through separate networks.

B.3.2 Datasets and Preprocessing

We evaluate the two parameterizations on three synthetic datasets and one real-world dataset.

Synthetic datasets. All synthetic datasets are defined on $\mathcal{S} = [0,1]^2$. Let

$$g_0(\boldsymbol{s}) = \mathcal{N}_2\big(\boldsymbol{s}; \boldsymbol{m}_0, \sigma_0^2 \boldsymbol{I}\big)\,, \qquad \boldsymbol{m}_0 = (0.5, 0.5),$$

denote the background Gaussian kernel, and let

$$g_2(\boldsymbol{s}, \boldsymbol{s}_i) = \mathcal{N}_2\big(\boldsymbol{s}; \boldsymbol{s}_i, \sigma_2^2 \boldsymbol{I}\big)$$

denote a spatial kernel centered at a historical event location. These Gaussian kernels are evaluated on $\mathcal{S}$ without renormalizing their mass within the domain.

SC1 is a Gaussian self-correcting process with intensity

$$\lambda^*(\boldsymbol{s}, t) = \mu \exp\left(\beta t g_0(\boldsymbol{s}) - \alpha \sum_{t_i < t} g_2(\boldsymbol{s}, \boldsymbol{s}_i)\right).$$

AutoIntHawkes-2 and *AutoIntHawkes-3* are Gaussian spatio-temporal Hawkes processes with intensity

$$\lambda^*(\boldsymbol{s}, t) = \mu g_0(\boldsymbol{s}) + \sum_{t_i < t} \alpha \exp[-\beta(t - t_i)] g_2(\boldsymbol{s}, \boldsymbol{s}_i).$$

Table 6 lists the intensity parameters.

Real-world dataset. The *Japan Earthquake* dataset contains earthquake times and locations in Japan (Zhou et al., 2022b; Zhou & Yu, 2023a). We use the provided training, validation, and test splits. For every deep-STPP dataset, time and each spatial coordinate are min–max normalized to $[0,1]$, using statistics computed separately within each split.

Table 6: Parameters of the three synthetic deep STPP datasets.

Dataset	μ	α	β	σ_0	σ_2
SC1	1.00	0.40	0.20	0.25	0.20
AutoIntHawkes-2	0.50	0.80	1.00	0.20	0.50
AutoIntHawkes-3	1.00	1.00	2.00	0.20	0.10

Table 7: Training settings for the statistical Hawkes, deep TPP, and deep STPP experiments.

Hyperparameter	**Statistical Hawkes**	**Deep TPP**	**Deep STPP**
Objective	WSM	WSM	WSM
Training duration	50 steps	100 epochs	300 epochs
Batch size	16	64	64
Optimizer	Adam	Adam	AdamW
Initial learning rate	10^{-2}	10^{-3}	10^{-3}

C Training and Timing Settings

This appendix reports the training configurations and timing protocols used in the efficiency experiments. For every model–dataset–seed configuration, TAYLOR and NESTED use identical data ordering, parameter initialization, WSM objective, optimizer, and training hyperparameters. The sole difference between the two methods is the event-coordinate derivative evaluator.

C.1 Hardware, Software, and Timing Protocol

Experiments run on a single NVIDIA GeForce RTX 5090 GPU with JAX 0.6.2, CUDA 12, and 64-bit computation. Wall-clock time is measured using `time.perf_counter`, with returned arrays synchronized before the timer is stopped.

We separately measure the WSM forward computation and the complete training step. The forward measurement includes history encoding, event-coordinate derivatives, and loss assembly, while the training-step measurement additionally includes parameter gradients and the optimizer update. Data loading, logging, evaluation, and file output are excluded.

Each probe discards 20 warm-up executions. The deep-STPP forward probe then repeats a fixed batch 100 times, while the remaining probes traverse training batches. We report the arithmetic mean of the retained timings. For the statistical Hawkes experiments, samples outside the 1.5-IQR bounds are additionally removed.

C.2 Training Settings

Table 7 summarizes the training settings common to all three experimental families. Deep-TPP settings are shared by SAHP and THP, while deep-STPP settings are shared by Vanilla and Decouple. Family-specific settings are provided below.

Statistical Hawkes. We use a maximum global gradient norm of 1.00, the temporal weight function `one_side_opt` with $\alpha = 1.00$, and run-specific batch shuffling. The initial raw background, excitation, exponential-decay, and Gaussian-bandwidth parameters are 1.00, 1.00, 3.00, and 1.00, respectively, with trainable kernel parameters.

Deep TPP. The learning rate is multiplied by 0.5 every 10 epochs. The classification, survival, and negative-term weights are 10.00, 10.00, and 50.00, respectively. The survival term is enabled, and evaluation uses an integration grid of size 100.

Deep STPP. AdamW uses $(\beta_1, \beta_2) = (0.90, 0.99)$, with a maximum gradient norm of 1.00. The spatial-loss and classification weights are $\alpha_s = 5.00$ and $\alpha_{\mathrm{CE}} = 1.00$, respectively, and the survival

Table 8: Mean wall-clock runtimes (seconds) of NESTED and TAYLOR. The outer-stage runtime is computed as $T_{\text{outer}} = T_{\text{step}} - T_{\text{forward}}$.

Model	Dataset	Forward		Outer stage		Training step	
		Nested	Taylor	Nested	Taylor	Nested	Taylor
Statistical Hawkes							
Exponential Hawkes	$\beta = 1.5$	9.811	9.847	49.337	35.072	59.148	44.919
	$\beta = 2.0$	10.095	10.100	49.679	36.746	59.774	46.846
Gaussian Hawkes	$\sigma = 0.5$	8.628	8.523	36.702	24.763	45.330	33.286
	$\sigma = 0.8$	10.778	10.464	48.355	34.899	59.133	45.363
Deep TPP							
SAHP	Cond.-logit	1.210	1.133	8.764	4.435	9.974	5.568
	Exp.-decay	1.260	1.130	9.802	3.951	11.062	5.081
	Half-sin	0.183	0.169	1.077	0.950	1.260	1.119
	Taobao	0.176	0.160	0.079	0.086	0.255	0.246
	StackOvfl.	0.092	0.082	0.141	0.145	0.233	0.227
THP	Cond.-logit	1.279	1.171	9.332	4.620	10.611	5.791
	Exp.-decay	1.302	1.203	9.890	4.049	11.192	5.252
	Half-sin	0.183	0.167	1.118	1.047	1.301	1.214
	Taobao	0.177	0.165	0.089	0.081	0.266	0.246
	StackOvfl.	0.096	0.084	0.155	0.145	0.251	0.229
Deep STPP							
Vanilla	SC1	0.477	0.302	6.807	1.005	7.284	1.307
	AutoInt-2	0.455	0.305	33.712	6.217	34.167	6.522
	AutoInt-3	0.455	0.310	33.415	8.962	33.870	9.272
	Earthquake	0.466	0.306	38.233	14.054	38.699	14.360
Decouple	SC1	0.273	0.173	1.722	0.782	1.995	0.955
	AutoInt-2	0.258	0.185	22.095	4.336	22.353	4.521
	AutoInt-3	0.250	0.170	22.781	6.038	23.031	6.208
	Earthquake	0.251	0.175	26.756	9.579	27.007	9.754

term is disabled. Temporal and spatial grid sizes are both 10. Evaluation is performed every 10 epochs, training-step timing is measured every batch, and the early-stopping patience is 10^6.

D COMPLETE RUNTIME MEASUREMENTS

Table 8 reports the detailed wall-clock runtimes for all model–dataset configurations. For each method and seed, forward and training-step runtimes are obtained by averaging the retained timing measurements. We define the outer-stage runtime as $T_{\text{outer}} = T_{\text{step}} - T_{\text{forward}}$, which includes parameter backpropagation and the optimizer update. The reported values are the arithmetic means of the resulting per-seed runtimes.

E METRIC DEFINITIONS AND INTERPRETATION

This section explains how the metrics in Section 5.3 assess forward computation cost, outer backpropagation amplification, and speedup attribution. We use $M \in \{\text{Taylor}, \text{Nested}\}$ for the two methods and $m \in \{\text{tay}, \text{nest}\}$ for their theoretical subscripts; T and N abbreviate the method names in Table 2. The connections below distinguish the measured runtimes and FLOPs from the unit-cost operation counts in Section 4.4.

E.1 Stage-Wise Runtime and FLOP Measurements

For $M \in \{\text{Taylor}, \text{Nested}\}$, let $m = \text{tay}$ or nest, respectively. The theoretical derivative-program costs are

$$C^M_{\text{deriv,fwd}} = \tau_m C, \qquad C^M_{\text{deriv,outer}} = \rho_m \tau_m C, \tag{5}$$

where $C > 0$ is the common primal cost, τ_m is the forward cost multiplier, and ρ_m is the outer-backpropagation amplification factor.

We measure the WSM forward-pass runtime T^M_{forward} and complete training-step runtime T^M_{step}, and define

$$T^M_{\text{outer}} := T^M_{\text{step}} - T^M_{\text{forward}}. \tag{6}$$

The forward pass includes context computation, event-coordinate derivative computation, and WSM loss assembly. The outer-stage contribution is estimated from separate forward and training-step probes and includes the optimizer update. For $r \in \{\text{forward}, \text{outer}, \text{step}\}$, the empirical speedup is

$$R_r = \frac{T^{\text{Nested}}_r}{T^{\text{Taylor}}_r}. \tag{7}$$

We also report $R_{\text{outer}}/R_{\text{forward}}$; ratios involving the outer contribution are interpreted only when both outer estimates are positive.

FLOP profiling isolates the forward computation of the derivative program and outer backpropagation through it, excluding shared computation and the optimizer update. The FLOP amplification factor is

$$\widehat{\rho}^{\text{FLOPs}}_M = \frac{F^M_{\text{deriv,outer}}}{F^M_{\text{deriv,forward}}}. \tag{8}$$

The subscripts N and T in Table 2 abbreviate Nested and Taylor.

To connect these measurements to the theoretical costs, we use an additive runtime model with shared contributions common to the two methods:

$$T^M_{\text{forward}} = T_{\text{shared,forward}} + T^M_{\text{deriv,forward}}, \tag{9}$$

$$T^M_{\text{outer}} = T_{\text{shared,outer}} + T^M_{\text{deriv,outer}}. \tag{10}$$

Here, $T^M_{\text{deriv,forward}}$ and $T^M_{\text{deriv,outer}}$ denote the derivative-program contributions; the shared outer term includes the optimizer update. Equality of the shared runtimes is an assumption of this model, not a consequence of using identical shared programs.

E.2 Empirical Evidence for Condition 1

Condition 1 states that the Taylor-lifted derivative program has a lower forward cost. Under equal shared forward runtimes,

$$\tau_{\text{tay}} < \tau_{\text{nest}} \iff C^{\text{Taylor}}_{\text{deriv,fwd}} < C^{\text{Nested}}_{\text{deriv,fwd}}, \tag{11}$$

$$T^{\text{Nested}}_{\text{forward}} - T^{\text{Taylor}}_{\text{forward}} = T^{\text{Nested}}_{\text{deriv,forward}} - T^{\text{Taylor}}_{\text{deriv,forward}}, \tag{12}$$

$$R_{\text{forward}} > 1 \iff T^{\text{Taylor}}_{\text{deriv,forward}} < T^{\text{Nested}}_{\text{deriv,forward}}. \tag{13}$$

Thus, a common shared forward contribution does not change the direction of the method-specific comparison.

To make the connection to the theoretical multipliers explicit, suppose that, under fixed hardware and software conditions, the runtime of the method-specific forward program is locally proportional to its program-level cost:

$$T^M_{\text{deriv,forward}} = \kappa_{\text{forward}} \tau_m C, \qquad \kappa_{\text{forward}} > 0. \tag{14}$$

Here, κ_{forward} is a positive stage-specific conversion factor from the unit-cost program measure to wall-clock runtime, assumed common to TAYLOR and NESTED within the same model–dataset configuration. Under this proportionality, Equation (13) is equivalent to $\tau_{\text{tay}} < \tau_{\text{nest}}$.

In practice, wall-clock runtime also depends on memory access, compiler transformations, execution scheduling, and hardware utilization. Accordingly, we interpret $R_{\text{forward}} > 1$ as wall-clock evidence supporting Condition 1 rather than as a proof of the unit-cost inequality.

E.3 Empirical Evidence for Condition 2

Condition 2 combines Condition 1 with a comparison of the outer backpropagation amplification factors:

$$\tau_{\text{tay}} < \tau_{\text{nest}}, \qquad \rho_{\text{tay}} \leq \rho_{\text{nest}}, \tag{15}$$

$$\rho_{\text{tay}}\tau_{\text{tay}}C < \rho_{\text{nest}}\tau_{\text{nest}}C. \tag{16}$$

The second relation follows when the amplification factors are positive, implying a lower program-level outer-backpropagation cost for the Taylor-lifted derivative program.

The FLOP amplification factor in Equation (8) provides the corresponding computational metric:

$$\widehat{\rho}^{\text{FLOPs}}_{\text{Taylor}} \leq \widehat{\rho}^{\text{FLOPs}}_{\text{Nested}}. \tag{17}$$

This is the measured analogue of $\rho_{\text{tay}} \leq \rho_{\text{nest}}$. Together with $R_{\text{forward}} > 1$, it provides complementary empirical evidence for the two inequalities in Condition 2.

The runtime relation becomes explicit after isolating the method-specific derivative program. Define

$$\widetilde{R}_{\text{forward}} = \frac{T^{\text{Nested}}_{\text{deriv,forward}}}{T^{\text{Taylor}}_{\text{deriv,forward}}}, \qquad \widetilde{R}_{\text{outer}} = \frac{T^{\text{Nested}}_{\text{deriv,outer}}}{T^{\text{Taylor}}_{\text{deriv,outer}}}. \tag{18}$$

Under the stage-wise runtime correspondence

$$T^{M}_{\text{deriv,forward}} = \kappa_{\text{forward}}\tau_m C, \qquad T^{M}_{\text{deriv,outer}} = \kappa_{\text{outer}}\rho_m\tau_m C,$$

with positive constants shared across the two methods,

$$\frac{\widetilde{R}_{\text{outer}}}{\widetilde{R}_{\text{forward}}} = \frac{\rho_{\text{nest}}}{\rho_{\text{tay}}}. \tag{19}$$

Hence,

$$\rho_{\text{tay}} \leq \rho_{\text{nest}} \quad \Longleftrightarrow \quad \widetilde{R}_{\text{outer}} \geq \widetilde{R}_{\text{forward}}. \tag{20}$$

Together with $\widetilde{R}_{\text{forward}} > 1$, the runtime counterpart of the two theoretical inequalities is

$$\widetilde{R}_{\text{outer}} \geq \widetilde{R}_{\text{forward}} > 1.$$

The reported R_{forward} and R_{outer}, however, use the full-stage runtimes in Equations (9) and (10). When the shared contributions are removed, or are sufficiently small relative to the method-specific derivative costs, these ratios reduce to or closely approximate their tilded counterparts. Otherwise, $R_{\text{outer}}/R_{\text{forward}}$ is not algebraically identical to $\rho_{\text{nest}}/\rho_{\text{tay}}$, because the WSM forward pass and outer stage contain different additive shared terms.

Accordingly,

$$R_{\text{outer}} \geq R_{\text{forward}} > 1$$

is used as complementary stage-level runtime evidence, whereas Equation (17) is the FLOP-based counterpart of the outer-backpropagation amplification inequality. This distinction prevents wall-clock measurements from being interpreted as an exact proof of the unit-cost condition.

E.4 Speedup Attribution

The complete training-step metric directly determines whether an actual wall-clock speedup occurs:

$$R_{\text{step}} > 1 \quad \Longleftrightarrow \quad T^{\text{Taylor}}_{\text{step}} < T^{\text{Nested}}_{\text{step}}. \tag{21}$$

Unlike the stage-wise metrics, this relation does not rely on a mapping from runtime to the unit-cost model and therefore provides direct empirical evidence for end-to-end training-step speedup.

To identify where the measured time saving arises, define

$$\Delta T_r = T^{\text{Nested}}_r - T^{\text{Taylor}}_r, \qquad r \in \{\text{forward}, \text{outer}, \text{step}\},$$
$$\Delta T_{\text{step}} = \Delta T_{\text{forward}} + \Delta T_{\text{outer}}. \tag{22}$$

For configurations with $\Delta T_{\text{step}} > 0$, define

$$A_{\text{outer}} = \frac{\Delta T_{\text{outer}}}{\Delta T_{\text{step}}}, \qquad A_{\text{forward}} = \frac{\Delta T_{\text{forward}}}{\Delta T_{\text{step}}} = 1 - A_{\text{outer}}. \tag{23}$$

The pair $(A_{\text{outer}}, A_{\text{forward}})$ attributes the training-step time saving to the estimated outer-stage contribution and the WSM forward pass, respectively. The fractions lie in $[0, 1]$ when both contributions are nonnegative; otherwise, a negative contribution represents a slowdown offset by the other stage. These fractions do not test either structural condition: Condition 1 concerns the derivative-program forward cost, whereas Condition 2 concerns the outer-backpropagation cost and its amplification factor.

Finally, the shared-cost ratio explains variation in the magnitude of the speedup rather than establishing whether a speedup occurs. Let

$$K_M = (1 + \rho_m)\tau_m C$$

denote the method-specific training-step cost. When Conditions 1 and 2 imply $K_{\text{Nested}} > K_{\text{Taylor}}$, the theoretical speedup as a function of the common shared cost $S = C_{\text{shared}}$ is

$$R(S) = \frac{S + K_{\text{Nested}}}{S + K_{\text{Taylor}}}.$$

Holding the method-specific costs fixed,

$$\frac{\mathrm{d}R(S)}{\mathrm{d}S} = \frac{K_{\text{Taylor}} - K_{\text{Nested}}}{(S + K_{\text{Taylor}})^2} < 0, \qquad \lim_{S \to \infty} R(S) = 1. \tag{24}$$

Thus, increasing the shared cost moves the theoretical speedup toward 1 without changing the ordering of the two method-specific costs. The empirical shared-cost ratio is therefore used to assess whether cross-configuration variation in R_{step} is consistent with this shared-cost dilution effect.

In summary, the empirical evidence has three distinct roles:

- $R_{\text{step}} > 1$ directly establishes an observed wall-clock speedup;
- the shared-cost ratio explains variation in its magnitude; and
- the stage-wise runtime and FLOP metrics provide mechanism evidence for the two sufficient conditions on the evaluated deep-STPP configurations.

None of the stage-wise empirical relations alone proves the unit-cost conditions or establishes wall-clock speedup outside the evaluated configurations.

F Mechanistic Interpretation of the Ablation Studies

We interpret the ablation results in Table 3 through the training-step cost decomposition in Equation (2), where changes to the derivative program affect both its forward computation and outer backpropagation.

F.1 Effect of Derivative Order

The training-step speedup increases from $1.45\times$–$1.70\times$ under DSM to $3.67\times$–$5.28\times$ under WSM. This increase is not explained by the forward speedup alone: for Vanilla–AutoInt-3, R_{forward} decreases from 1.718 to 1.470, while R_{step} increases from 1.453 to 3.667.

DSM requires first-order scores, whereas WSM additionally requires diagonal second-order derivatives. Under NESTED, these terms introduce further reverse-mode transformations, enlarging the derivative program traversed by outer backpropagation. TAYLOR obtains both derivative orders by propagating coefficients along each coordinate direction. Removing the second-order terms under DSM eliminates these additional transformations while retaining outer backpropagation through the first-order program. This structural difference is consistent with the larger WSM speedups observed in the evaluated configurations.

F.2 Effect of Event-Coordinate Dimensionality

The spatial dimension varies over $d_s \in \{4, 6, 8, 16\}$, with one temporal coordinate retained. The forward speedup R_{forward} decreases and falls below 1 for all configurations at $d_s = 6$, while the training-step speedup remains in $[3.12, 6.62]$.

Both derivative programs have costs that grow with spatial dimension. NESTED additionally contains a dimension-independent reverse pass that computes all first-order score components, whereas TAYLOR performs coefficient propagation along each coordinate direction. With the cost multipliers held fixed, their derivative-program forward cost ratio is

$$\frac{\tau_{\text{nest}}}{\tau_{\text{tay}}} = \frac{(1 + \rho^{(1)})\rho^{(2)}}{\alpha_2} + \frac{1 + \rho^{(1)}}{\alpha_2 d_s}.$$

The second term decreases with d_s, providing a cost-model explanation for the declining forward speedup.

The training-step advantage can persist despite a forward slowdown. By the runtime definitions in Appendix E,

$$\Delta T_{\text{outer}} = \Delta T_{\text{step}} - \Delta T_{\text{forward}}.$$

Thus, when $R_{\text{forward}} < 1$ but $R_{\text{step}} > 1$, the estimated outer-stage saving exceeds the added forward cost, sustaining the training-step speedup even when TAYLOR loses its forward advantage.