POINT PROCESS FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Event sequences can be modeled by temporal point processes (TPPs) to capture their asynchronous and probabilistic nature. We propose an intensity-free framework that directly models the point process as a non-parametric distribution by utilizing normalizing flows. This approach is capable of capturing highly complex temporal distributions and does not rely on restrictive parametric forms. Comparisons with state-of-the-art baseline models on both synthetic and challenging real-life datasets show that the proposed framework is effective at modeling the stochasticity of discrete event sequences.

1 INTRODUCTION

Data in real-life takes various forms. Event sequences, as a special form of data, are discrete events in continuous time. This type of data is prevalent in a broad spectrum of areas, for example, patient visits to hospitals, user behavior on social media, credit card transactions, etc. In this setting, each event is discrete, and the temporal dynamics of the events are complex and asynchronous. It is crucial to understand the characteristics and dynamics of such data, so that plausible future prediction, as well as other downstream applications, such as intervention or recommendation, can be performed. Despite recent success in modeling images, videos and texts with the power of deep neural networks (DNNs), the asynchronous and probabilistic nature of event sequence data makes it challenging to utilize the power of off-the-shelf DNN-based models.

Temporal point processes (TPPs; Daley & Vere-Jones 2007) provide us with an elegant and effective mathematical framework for modeling event sequences data. A temporal point process is defined as a stochastic process whose realizations consist of a list of events with their corresponding occurring times. These occurring times can either be real numbers from an index set (defined from prior knowledge) or sampled from an intensity function. While other time-series models learn temporal patterns synchronously (with each time-step being treated as an input to the model), TPP-based frameworks directly model the time intervals between events as random variables. With such a setup, it allows for modeling long sequences without vanishing gradients or costly memory issues.

Although the temporal point process has shown to be useful in modeling events sequences, it is usually not trivial to come up with a simple yet flexible intensity function. An intensity function encodes the rate an event occurs at a specified time-step. Poisson process (Kingman, 1992) has been a popular hand-crafted design for the intensity which assumes that events are independent of each other. More sophisticated design choices are investigated in the self-exciting (Hawkes, 1971) and self-correcting process (Isham & Westcott, 1979). The key contribution of these models is to find a functional form of intensity that fits data distribution well by making various parametric assumptions on the underlying generative process of the data. Although shown effective in modeling simple synthetic datasets, parametric assumptions make such frameworks lack the flexibility to model the generative process for real-life and complex data, hindering wider adoption of TPP-based frameworks.

Recently, learning the intensity function using recurrent neural networks (RNNs) to encode the history has received an increasing amount of attention (Du et al., 2016; Zhong et al., 2018; Mei & Eisner, 2017; Jing & Smola, 2017). In this line of work, history information is encoded and exploited in learning the intensity of the point process distribution. In this case, the explicit parametric assumption on the forms of the intensity functions is relaxed. However, the maximum likelihood training criteria on these models still requires the intensity function to be simple for the likelihood to be tractable. Recent work by Mehrasa et al. (2019) proposes a probabilistic framework based on variational autoencoders for modeling point process, further facilitating the stochastic generative process. All the literature above is built upon explicit modeling of a temporal point process using the intensity function.

It is not necessary to explicitly model the intensity; a few works have tried to formulate TPP in an intensity-free manner. WGANTPP (Xiao et al., 2017; 2018) introduce an intensity-free framework for modeling the point process distribution using Wasserstein distance. The model is built upon a generative adversarial network (GAN). RLPP (Li et al., 2018) formulates this problem in a reinforcement learning framework and treats future event predictions as actions taken by an agent. Both of these models are optimized by trying to generate sequences of samples that are indistinguishable from the ground-truth sequences (by a discriminator in WGANTPP and policy learning in RLPP). Although these models are capable of generating realistic sequences, such training criteria fail to model the data distribution, resulting in intractable likelihood.

In this work, We propose a novel intensity-free point process model based on continuous normalizing flow and variational autoencoders. The proposed point process flow (PPF) model utilizes a recurrent variational autoencoder to encode the history of a given event sequence and makes probabilistic predictions on the next event. It preserves the non-parametric characteristics of point process distributions with normalizing flow. The predicted non-parametric point process distribution is capable of capturing complex time distributions of arbitrary shape, leading to more accurate modeling of event sequences. Extensive experiments are conducted on synthetic and real datasets to evaluate the performance of the proposed model. Experimental results show that our model is capable of capturing complex point process distributions as well as performing accurate stochastic forecast. The contributions are summarized as follows: (1) A novel intensity-free point process model built upon continuous normalizing flow. The proposed PPF is capable of capturing highly complex temporal distributions and does not rely on restrictive parametric forms; (2) PPF can be optimized by maximizing the exact likelihood using change of variable formula, relaxing the constraint in previous works that likelihood has to be tractable; (3) Evaluation on both synthetic and challenging real-life datasets shows improvement over the state-of-the-art point process models.

2 Preliminaries

2.1 Temporal point process

A temporal point process (TPP; Daley & Vere-Jones 2007) is a mathematical framework for modeling asynchronous sequences of actions. It is a stochastic process whose realization is a sequence of discrete events in time $t_{1:n} = \{t_1, t_2, ..., t_n\}$ where t_n is the time when the n^{th} event occurred.

A temporal point process distribution is modeled by specifying the probability density function of the time of the next event:

$$f(t|\mathcal{H}_t) = \lambda(t|\mathcal{H}_t) \exp\left\{ -\int_{t_{n-1}}^{t} \lambda(u|\mathcal{H}_u)\ \mathrm{d}u \right\}, \tag{1}$$

where the intensity function $\lambda(t|\mathcal{H}_t)$ is the conditional intensity function. It encodes the expected rate of event happening in a small area around t and $\mathcal{H}_t = \{t_1, t_2, ..., t_{n-1}|t_{n-1} < t\}$ is the sequence of event times up to time t. Many works explored different design choices of intensity function to capture the phenomena of interest. Here we review two popular hand-crafted design choices:

Poisson Process. Poisson process (Kingman, 1992) is based on the assumption that events happen independent of each other where the intensity is a fixed positive constant $\lambda(t) = \lambda$ and $\lambda > 0$. In a more general case, λ could be a function of time $\lambda(t|\mathcal{H}_t) = \lambda(t)$ but still independent of other events, which is called inhomogeneous Poisson process.

Self-exciting Process (Hawkes process). Self-exciting process (Hawkes, 1971) assumes that occurrence of an event increases the probability of other events happening in near future. Its intensity function has the functional form of $\lambda(t|\mathcal{H}_t) = \mu + \alpha \sum_{t_i<t} \exp(-(t - t_i))$, where μ and α are positive constants and $t_i < t$ are all the events happening before time t.

2.2 Normalizing flow

Normalizing flows are generative models that allow both density estimation and sampling. They map simple distributions to complex ones using bijective functions. Specifically, if our interest is to estimate the density function $p_{\mathbf{X}}$ of a random vector $\mathbf{X} \in \mathbb{R}^d$, then normalizing flows assume $\mathbf{X} = g_\theta(\mathbf{Z})$, where $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is a bijective function, and $\mathbf{Z} \in \mathbb{R}^d$ is a random vector with a tractable density function $p_{\mathbf{Z}}$. We further denote the inverse of g_θ by f_θ. On one hand, the probability density function can be evaluated using the change of variables formula:

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f_\theta(\mathbf{x})) \left| \det \left(\frac{\partial f_\theta}{\partial \mathbf{x}} \right) \right|, \tag{2}$$

where $\partial f_\theta / \partial \mathbf{x}$ denotes the Jacobian matrix of f_θ. On the other hand, sampling from $p_{\mathbf{X}}$ can be done by first drawing a sample from the simple distribution $\mathbf{z} \sim p_{\mathbf{Z}}$, and then apply the bijection $\mathbf{x} = g_\theta(\mathbf{z})$.

Given the expressive power of deep neural networks, it is natural to construct g_θ as a neural network. However, it requires the bijection g_θ to be invertible, and the determinant of the Jacobian matrix should be efficient to compute. Several methods have been proposed along this research direction (Rezende & Mohamed, 2015; Dinh et al., 2014; 2017; Kingma et al., 2016; Kingma & Dhariwal, 2018; Papamakarios et al., 2017). An extensive overview of normalizing flow models is given by Kobyzev et al. (2019).

2.3 Continuous normalizing flow

From a dynamical systems perspective, the residual network can be regarded as the discretization of an ordinary differential equation (ODE; Haber & Ruthotto 2017; Chang et al. 2018; Lu et al. 2018). Inspired by that, Chen et al. (2018) propose neural ODE, where the continuous dynamics of hidden units is parameterized using an ordinary differential equation specified by a neural network:

$$\frac{d\boldsymbol{z}(t)}{dt} = h(\boldsymbol{z}(t), t, \theta). \tag{3}$$

The neural ODE can be used to construct a continuous normalizing flow. The invertibility is naturally guaranteed by the theorem of the existence and uniqueness of the solution of the ODE. Furthermore, using the instantaneous change of variables formula, similar to Equation 2, the log-density can be evaluated by solving the following ODE:

$$\frac{\partial \log p(z(t))}{\partial t} = -\mathrm{Tr}\left(\tfrac{\partial h}{\partial z(t)} \right). \tag{4}$$

Grathwohl et al. (2019) propose an improved version of neural ODE, named FFJORD, which has lower computational cost by using an unbiased stochastic estimation of the trace of a matrix.

3 Proposed framework

We propose an intensity-free flow framework to model the timing of events in point process sequences. More specifically, we learn a non-parametric distribution over the timing of asynchronous event sequences by transforming a simple base probability density through continuous normalizing flow, *i.e.*, a series of invertible transformations. With our proposed framework, we are able to model complex point process distributions without making any assumption on the functional form of the distribution while being able to evaluate the likelihood of sequences under our model.

3.1 Non-parametric point process flows

Let the input be a sequence of asynchronous events $\{t_1, t_2, ...\}$, where $t_i \in \mathbb{R}_+$ represents the starting time of the i-th event. We define the inter-arrival time τ_n as the time difference between the starting time of events t_{n-1} and t_n. Our goal is to model the distribution over inter-arrival time τ_n given the past history of events inter-arrival times $\tau_{1:n-1}$, *i.e.*, learning to model the conditional distribution $p(\tau_n | \tau_{1:n-1})$.

Our approach is to construct the distribution over inter-arrival time τ_n by transforming a simple base distribution through normalizing flow transformations. At time-step n, we assume that inter-arrival time τ_n was generated by first sampling from a simple distribution $p(z_n)$ and then transforming the drawn sample z_n through an invertible transformation $g_\theta : \mathbb{R} \to \mathbb{R}_+$ parametrized by θ :

$$z_n \sim p(z_n), \quad \tau_n = g_\theta(z_n). \tag{5}$$

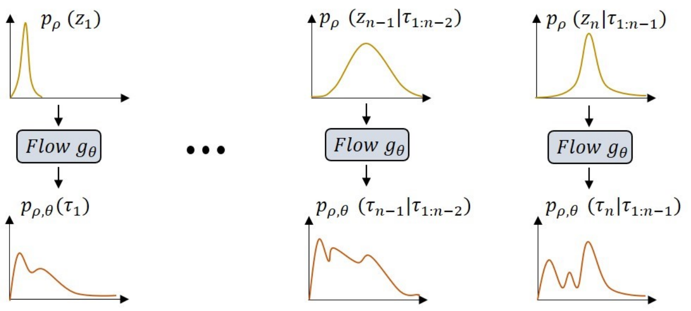
$p_\rho\,(z_1)$
$p_\rho\,(z_{n-1}|\tau_{1:n-2})$
$p_\rho\,(z_n|\tau_{1:n-1})$
Flow g_θ
Flow g_θ
Flow g_θ
$p_{\rho,\theta}(\tau_1)$
$p_{\rho,\theta}\,(\tau_{n-1}|\tau_{1:n-2})$
$p_{\rho,\theta}\,(\tau_n|\tau_{1:n-1})$

Figure 1: This figure shows the overall structure of our non-parametric point process modeling via normalizing flow. We learn a non-parametric distribution over the inter-arrival time of asynchronous event sequences by transforming a base probability density which is conditioned on the past history through normalizing flow transformations.

With this assumption and the change of variable formula discussed in subsection 2.2, we can write the distribution over inter-arrival time τ_n as:

$$p_\theta(\tau_n) = p(z_n)\left|\frac{\partial f_\theta}{\partial \tau_n}\right|, \tag{6}$$

where $f_\theta(\tau_n) = g_\theta^{-1}(\tau_n) = z_n$, and the scalar Jacobin value $df_\theta/d\tau_n$ shows the changes in the density when moving from τ_n to z_n. We dropped the determinant term in the change of variable formula, because in our case, the inter-arrival time τ_n is a one-dimensional variable. With this formulation, we are able to model the inter-arrival time distribution, without any specific assumption on the functional form of the distribution. Exact samples of the inter-arrival time distribution can be obtained by sampling from the base distribution $z_n \sim p(z_n)$ and transforming it through flow transformation $\tau_n = g_\theta(z_n)$. We are also able to compute the exact likelihood of τ_n by computing the likelihood of $f_\theta(\tau_n)$ and multiplying it with the associated Jacobian term $\left|\frac{\partial f_\theta}{\partial \tau_n}\right|$.

Current formulation models the inter-arrival distribution of each time-step independent of past history. The future event timing might depend on previous events in a very complex way, so its important to take the history information into consideration while modeling future events. To capture this dependency, we adapt our formulation to construct the point process distribution by learning normalizing flow parameters conditioned on the history. More specifically, we learn the parameters of flow base distribution by a time-dependent model parametrized by ρ that encodes history and provides the conditional base distribution $p_\rho(z_n|\tau_{1:n-1})$ at each time-step. The overal procedure can be seen in Figure 1. In our framework, the base distribution is assumed to follow a Gaussian distribution:

$$p_\rho(z_n|\tau_{1:n-1}) = \mathcal{N}(\mu_{\rho_n}, \sigma^2_{\rho_n}), \tag{7}$$

where $(\mu_{\rho_n}, \sigma^2_{\rho_n})$ are the parameters of the base distribution. These parameters can be obtained by encoding the history $\tau_{1:n-1}$ using various approaches. In this work, we propose two approaches to construct the base distribution of the flow: 1) Base distribution with deterministic parameters. 2) Base distribution with stochastic parameters. In the following sections, we describe each approach in more details.

3.2 Base distribution with deterministic parameters

As discussed in subsection 3.1, we aim to model the conditional distribution $p(\tau_n|\tau_{1:n-1})$ using history information in learning the base distribution parameters. Recurrent neural networks (RNNs) have shown to be powerful deterministic models in capturing temporal dependencies. Recent works adapt RNNs as a non-linear mapping of the history to the intensity function to define temporal point process distributions (Du et al., 2016; Zhong et al., 2018; Mei & Eisner, 2017). As a first attempt, we use RNNs to learn the parameters of the base distribution of the flow using the history information.

Figure 2 part (a) illustrates the overall structure of our model. To construct the conditional distribution $p_{\theta,\rho}(\tau_n|\tau_{1:n-1})$, the RNN takes the history of the past inter-arrival times $\tau_{1:n-1}$ and produces

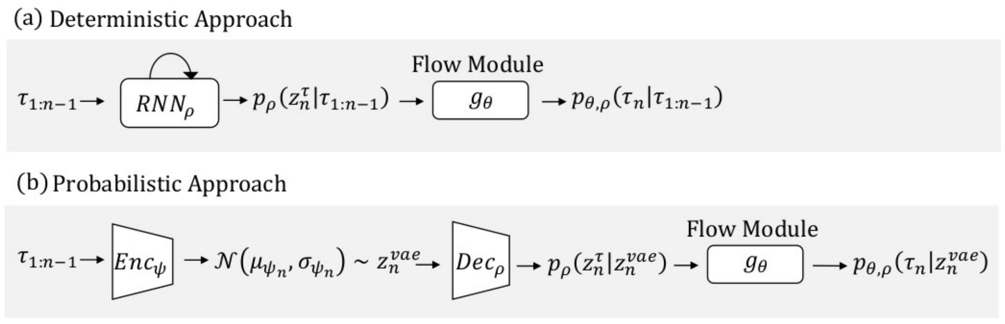
(a) Deterministic Approach
$\tau_{1:n-1}$
RNN_ρ
$p_\rho(z_n^\tau|\tau_{1:n-1})$
Flow Module
g_θ
$p_{\theta,\rho}(\tau_n|\tau_{1:n-1})$
(b) Probabilistic Approach
$\tau_{1:n-1}$
Enc_ψ
$\mathcal{N}(\mu_{\psi_n}, \sigma_{\psi_n}) \sim z_n^{vae}$
Dec_ρ
$p_\rho(z_n^\tau|z_n^{vae})$
Flow Module
g_θ
$p_{\theta,\rho}(\tau_n|z_n^{vae})$

Figure 2: Part (a) shows the deterministic approach of utilizing RNNs for predicting conditional distribution $p_{\theta,\rho}(\tau_i|\tau_{1:i-1})$. RNN encodes history into the base distribution, then it gets transformed to the target distribution by flow transformation g_θ. Part (b) shows the generation phase of incorporating the flow module in a probabilistic framework. During generation, the prior network gets the history and output the latent space distribution for the next time-step. Then a sample of this distribution is passed to the decoder which generates the non-parametric distribution over the inter-arrival time of next time-step by incorporating the flow module.

the conditional base distribution $p_\rho(z_n|\tau_{1:n-1}) = \mathcal{N}(\mu_{\rho_n}, \sigma^2_{\rho_n})$ for the next time-step. Then, the base distribution is transformed into the conditional inter-arrival time distribution over τ_n through normalizing flow transformations g_θ. The RNN is jointly optimized with the flow module by maximizing the log-likelihood of observed sequence $\tau_{1:N}$ under the predicted distribution:

$$\mathcal{L}_{\theta,\rho}(\tau_{1:N}) = \sum_{i=1}^{N} \log p_{\theta,\rho}(\tau_i|\tau_{1:i-1}) = \sum_{i=0}^{N} \log p_\rho(z_i|\tau_{1:i-1}) + \log\left|\frac{\partial f_\theta}{\partial \tau_i}\right|. \tag{8}$$

3.3 Base distribution with probabilistic parameters

It is known that there is a trade-off between the complexity of the bijective transformation and the form of base distribution (Jaini et al., 2019). With the complexity of the bijective transformation fixed, a more flexible base distribution will lead to a more expressive model. Motivated by this, our second move is to combine our flow module with a variational auto-encoder (VAE; Kingma & Welling 2014) framework in order to achieve more flexible base distributions.

To avoid confusion, at time-step n, we use the notation z_n^τ for the random variable of the normalizing flow base distribution and z_n^{vae} to refer to the VAE latent space. We start by explaining the generation phase, *i.e.*, how the distributions over inter-arrival time τ_n are generated by stacking the normalizing flow module on top of the VAE backbone and then describing the training process.

Generation. Figure 2 part (b) shows an overview of the generation process. Here, we adapt a recurrent VAE framework consisting of a time-variant prior network parametrized by ψ which takes the history of past actions $\tau_{1:n-1}$ and provides the latent distribution $p_\psi(z_n^{vae}|\tau_{1:n-1})$. Then, a sample of this distribution is passed to the VAE's decoder which produces a non-parametric distribution over the inter-arrival time τ_n by first generating the normalizing flow base distribution $p_\rho(z_n^\tau|z_n^{vae})$ and then transforming it through flow transformation g_θ. By applying the change of variable formula discussed in Equation 6, we can write the distribution over inter-arrival time τ_n as:

$$p_{\theta,\rho}(\tau_n|z_n^{vae}) = p_\rho(z_n^\tau|z_n^{vae})\left|\frac{\partial f_\theta}{\partial \tau_n}\right|. \tag{9}$$

Training. At time-step n of training, the VAE module takes the sequence of inter-arrival times $\tau_{1:n}$ to approximate the true distribution over the latent space z_n^{vae} via the help of the recurrent inference network $q_\phi(z_n^{vae}|\tau_{1:n})$ which is parametrized with ϕ. A time-dependent prior network is also adapted to help the model to take use of history information in generation phase $p_\psi(z_n^{vae}|\tau_{1:n-1})$. Both prior and posterior distributions are assumed to follow conditional multivariate Gaussian distributions with diagonal covariance:

$$p_\psi(z_n^{vae}|\tau_{1:n-1}) = \mathcal{N}(\mu_{\psi_n}, \Sigma_{\psi_n}), \tag{10}$$
$$q_\phi(z_n^{vae}|\tau_{1:n}) = \mathcal{N}(\mu_{\phi_n}, \Sigma_{\phi_n}). \tag{11}$$

At each time-step during training, a latent code z_n^{vae} is taken from the posterior and is passed to the decoder which aims to generate the conditional distribution $p_{\theta,\rho}(\tau_n|z_n^{vae})$. The VAE backbone is jointly trained with the flow module by optimizing the variational lower bound using the re-parameterization trick (Kingma & Welling, 2014):

$$\begin{aligned}\mathcal{L}_{\theta,\phi,\psi,\rho}(\tau_{1:N}) = \sum_{n=1}^{N}(&\mathbb{E}_{q_\phi(z_n^{vae}|\tau_{1:n})}[\log p_{\theta,\rho}(\tau_n|z_n^{vae})] \\ &- D_{KL}(q_\phi(z_n^{vae}|\tau_{1:n})||p_\psi(z_n^{vae}|\tau_{1:n-1}))),\end{aligned} \tag{12}$$

where we compute the log-likelihood term $\log p_{\theta,\rho}(\tau_n|z_n^{vae})$ by applying the change of variable formula of Equation 9.

4 EVALUATION

To show the effectiveness of our non-parametric approach, we evaluate the performance of our model on synthetic and real-world datasets and compare it with the state-of-the-art point process models. Please refer to Appendix A for architecture and implementation details.

4.1 DATASETS AND BASELINES

Synthetic Datasets. We create three types of synthetic datasets as follow: **(I) Inhomogeneous Poisson Process (IP)** defines the intensity as a function of time but independent of the history. We simulate sequences of IP process with $\lambda(t) = \sum_{i=1}^{k} \alpha_i (2\pi\sigma_i^2)^{-1/2} \exp(-(t-c_i)^2/\sigma_i^2)$ where $k = 6$, $\alpha = (14, 18, 13, 17, 10, 13)$, $c = (3, 6, 9, 12, 15, 18)$ and $\sigma = (5, 5, 5, 5, 5, 5)$. **(II) Self-exciting Process (SE)** assumes that the occurrence of an event increases the probability of other events happening in the near future. It is characterized by $\lambda(t) = \mu + \beta \sum_{t_i<t} g(t - t_i)$, where in our case $g(t) = \exp(-t)$, $\mu = 1.0$, and $\beta = 0.8$. **(III) IP + SE** is created by combining the simulated data from the self-exciting process and the inhomogeneous process. For each of IP and SE, we generate 20000 sequences, where the length of each is 60, and split the sequences into train, validation and test sets with proportions of 0.7, 0.1, 0.2, respectively.

Real-world Datasets. We also evaluate our models on real datasets that cover the areas of social media, healthcare, and human activity as follow: **(I) LinkedIn** data is collected from over 3000 LinkedIn accounts and contains their job-hopping records with information including the time and company. Our model predicts the time-interval before a user's next job-hopping. After pruning users with only one job-hopping record, we collect 2439 sequences. **(II) MIMIC** (Medical Information Mart for Intensive Care III; Johnson et al. 2016; Pollard 2016) is a publicly available, large-scale dataset which contains the medical records of more than 40000 anonymous patients. Our method models the inter-arrival time of patients' admissions to hospital. We keep the record of patients who have at least three visits to hospitals and collected 2377 sequences. **(III) Breakfast** dataset (Kuehne et al., 2014) contains 1712 videos with 48 action classes related to breakfast preparation. On this dataset, we model the inter-arrival times of the actions as well as actions categories. We explain this extension in more detail in subsection 4.2. For the **LinkedIn** and **MIMIC** dataset, we also split the dataset into train, validation and test sets with proportions of 0.7, 0.1, 0.2, respectively. For the **Breakfast** dataset, we use the standard train and test split proposed by Kuehne et al. (2014).

Baselines. We compare our proposed flow-based approach with the state-of-the-art point process models: **(I) APP-LSTM**[1] is an LSTM that takes the history of past events and predicts the inter-arrival time distribution for the next time-step by mapping the history into the intensity of a point process distribution. We train it by maximizing the likelihood of observed sequences under the predicted distribution. This deterministic baseline has comparable performance to Du et al. (2016). It only differs in the way that intensity is defined; unlike Du et al. (2016), its intensity doesn't

[1]This baseline has comparable performance to Mei & Eisner (2017); Du et al. (2016).

Dataset	Model			
	APP-LSTM	PPF-D	APP-VAE	PPF-P
IP	-2.942	-1.857	$\geqslant 0.408$	$\geqslant \mathbf{0.499}$
SE	-2.990	-1.615	$\geqslant 0.562$	$\geqslant \mathbf{0.636}$
IP+SE	-2.978	-1.507	$\geqslant 0.476$	$\geqslant \mathbf{0.566}$
LinkedIn	-0.795	0.973	$\geqslant -1.713$	$\geqslant \mathbf{2.678}$
MIMIC	-1.962	-0.498	$\geqslant -1.200$	$\geqslant \mathbf{1.696}$

Table 1: **Log-likelihood Comparison.** LL is reported for synthetic and real datasets.

explicitly depend on time. Zhong et al. (2018) compare these two design choices, and implicit dependence was shown to be more effective in modeling point process distribution.

(II) APP-VAE (Mehrasa et al., 2019) is a latent variable framework for modeling marked temporal point process. The model makes conditional predictions by learning a conditional latent space. Given a history of past actions, APP-VAE generates two distributions for the next action: one over its timing (by predicting the conditional intensity and using it to define point process distributions) and one over its category. We use their original setup to compare APP-VAE with our model on breakfast dataset, but for the rest of the datasets, we modify their approach to predict the time distribution only.[2]

4.2 Results

Log-likelihood Comparison. We report log-likelihood (LL) of test sequences across all models. For our PPF model with the probabilistic approach in learning base distribution parameters (PPF-P; introduced in subsection 3.3) and APP-VAE baseline, we report the importance weighted autoencoder (IWAE) bound, which is a lower bound of the real log-likelihood. To compute IWAE, at each time-step, we draw 1500 samples from the VAE's posterior distribution and follow the standard procedure for computing IWAE. We report the average of log-likelihood along all the time-steps of all sequences in the test dataset. The experimental results are shown in Table 1. The results indicate the better capability of our normalizing flow-based approaches at modeling point process sequence data, especially the real-world data with complicated underlying distributions. Our probabilistic PPF-P approach robustly outperforms state-of-the-art intensity-based baselines across all the datasets. Without any assumption on the functional form of intensity, we are able to model the point process distribution effectively. Our probabilistic approach also has a better performance in comparison to our deterministic approach introduced in subsection 3.2 (PPF-D). This demonstrates the advantages of using a more flexible base distribution in the flow in the probabilistic approach.

Point Estimate Comparison. We also report the mean absolute error (MAE) to evaluate the performance of our model in estimating future events timing. The MAE between the samples of predicted time distribution and the ground-truth is reported. To compute MAE for PPF-P and APP-VAE, at time-step i, we have two stages of sampling: (1) First, we draw samples from the prior distribution $z_i^{vae} \sim p_\psi(z_i^{vae}|\tau_{1:i-1})$. Then, we pass each to the decoder and (2) draw samples from each predicted distribution $\tau_i \sim p_\rho(\tau_i|z_i^{vae})$. The MAE computation at time-step i is as follow:

$$\mathbb{E}_{z_i^{vae} \sim p_\psi(z_i^{vae}|\tau_{1:i-1})}\Big[\mathbb{E}_{\tau_i \sim p(\tau_i|z_i^{vae})}\big(|\tau_i - \tau_i^*|\big)\Big], \tag{13}$$

where τ_i^* is the ground-truth inter-arrival at time-step i. We follow a similar procedure for computing the MAE for the deterministic approaches:

$$\mathbb{E}_{\tau_i \sim p(\tau_i|\tau_{1:i-1})}\big(|\tau_i - \tau_i^*|\big). \tag{14}$$

For PPF-P and APP-VAE, to estimate the expected error, we draw 100 samples from prior distribution $p_\psi(z_i^{vae}|\tau_{1:i-1})$ and 15 samples from each predicted base distribution $p_\theta(\tau_i|z_i^{vae})$. For PPF-D and APP-LSTM, we sample 1500 predictions from the output distributions $p(\tau_i|\tau_{1:i-1})$ at each step. For our PPF approaches (both deterministic and probabilistic), the corresponding samples

[2]We drop the use of mark data as input and accordingly omit the likelihood calculation of action category distribution from the optimization term.

Dataset	Model			
	APP-LSTM	PPF-D	APP-VAE	PPF-P
IP	6.765	4.7759	0.279	**0.278**
SE	7.360	4.4205	**0.290**	0.297
IP+SE	7.163	4.0525	**0.288**	0.299
LinkedIn	2.522	2.048	2.495	**1.799**
MIMIC	23.531	**17.709**	27.479	26.047

Table 2: **Mean Absolute Error Comparison.** MAE is reported for synthetic and real datasets.

Model	LL	MAE	Accuracy
APP-LSTM	-8.099	239.624	59.594
PPF-D	-7.637	251.337	61.174
APP-VAE	$\geqslant -6.463$	244.019	62.190
PPF-P	$\geqslant \mathbf{-6.342}$	**204.913**	**62.528**

Table 3: Comparison of log-likelihood, MAE, and accuracy on Breakfast dataset.

of predicted inter-arrival time distribution are obtained using Equation 5. We report the average of MAE along all the time-steps of all sequences in the test dataset. Table 2 shows the experimental results for MAE metric. We can see that our PPF-P approach is comparable to the APP-VAE baseline on the synthetic datasets. On the more challenging real datasets, our PPF-based frameworks consistenlty outperforms baseline models (PPF-D on MIMIC, and PPF-P on LinkedIn). The better log-likelihood estimations is also conformed by lower/competitive MAE which reflects the better quality of generated samples from our PPF approaches.

Extension to the Marked Temporal Point Process. On Breakfast dataset, for our model to learn a more powerful encoding of history information, we extend our approach to marked point process which models both the inter-arrival time distribution of future event and also the distribution over its category. At time-step i, given the history of past events including both time and mark information, in addition to modeling the time distribution of event at time-step $i + 1$, we also model its category distribution. Here, we assume that action category follows a multinomial distribution and accordingly, the log-likelihood of action category distribution modeling is added to training objective and evaluation criterion of our deterministic and probabilistic PPF models.[3] APP-LSTM is also extended to the marked case similar to Du et al. (2016). For the experiments on Breakfast dataset, in addition to MAE, we also report the accuracy of predicting the next action category. To compute accuracy at time-step i, for probabilistic approaches, we draw 100 samples from the prior distribution $p_\psi(z_i^{vae}|\tau_{1:i-1})$ and for each predicted category distribution, we select the action category with maximum probability as the predicted class. For each time-step, the most frequently predicted type is reported as the model's prediction. Table 3 shows the experimental results of comparing log-likelihood, MAE, and accuracy on Breakfast dataset. This dataset is much more challenging in comparison to LinkedIn and Mimic datasets, because of it contains various types of actions. We can see that our probabilistic approach has a better performance in all the metrics which shows the effectiveness of our proposed model in capturing the underlying point process distribution.

5 Conclusion

In this paper, we propose Point Process Flows (PPF), an intensity-free framework that directly models the point process as a non-parametric distribution by utilizing normalizing flows. The proposed model is capable of capturing arbitrary complex time distributions as well as performing stochastic future prediction. The proposed PPF can be optimized by maximizing the exact likelihood using change of variable formula, relaxing the strict tractable likelihood constraint in previous works. Extensive evaluation on both synthetic and challenging real-like datasets shows significant improvement over baseline models.

[3]For our deterministic/probabilistic approach, we assume that at each time-step given the history/latent-code, time and category are independent.

REFERENCES

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations (ICLR)*, 2017.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1555–1564. ACM, 2016.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJxgknCcK7`.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 1971.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and their Applications*, 1979.

Priyank Jaini, Ivan Kobyzev, Marcus Brubaker, and Yaoliang Yu. Tails of triangular flows. *arXiv preprint arXiv:1907.04481*, 2019.

How Jing and Alexander J Smola. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 515–524. ACM, 2017.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Machine Learning*, 2015.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

J.F.C. Kingman. *Poisson Processes*. Oxford Studies in Probability. Clarendon Press, 1992. ISBN 9780191591242.

Ivan Kobyzev, Simon Prince, and Marcus A Brubaker. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*, 2019.

Hilde Kuehne, Ali Arslan, and Thomas Serre. The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Shuang Li, Shuai Xiao, Shixiang Zhu, Nan Du, Yao Xie, and Le Song. Learning temporal point processes via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pp. 3282–3291, 2018.

Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A Variational Auto-Encoder Model for Stochastic Point Processes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Hongyuan Mei and Jason Eisner. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Alistair EW Pollard, Tom J abd Johnson. The mimic-iii clinical database. `http://dx.doi.org/10.13026/C2XW26`, 2016.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Shuai Xiao, Hongteng Xu, Junchi Yan, Mehrdad Farajtabar, Xiaokang Yang, Le Song, and Hongyuan Zha. Learning conditional generative models for temporal point processes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Y. Zhong, B. Xu, G.-T. Zhou, L. Bornn, and G. Mori. Time Perception Machine: Temporal Point Processes for the When, Where and What of Activity Prediction. In *arXiv 1808.04063*, 2018.

A Appendix

A.1 Architecture

The overall architectures are illustrated in Figure 2. For the deterministic approach introduced in subsection 3.2, a long short-term memory (LSTM; Hochreiter & Schmidhuber 1997) network is used to model the conditional distribution $p_\rho(z_n|\tau_{1:n-1}) = \mathcal{N}(\mu_{\rho_n}, \sigma^2_{\rho_n})$. The flow module g_θ is a neural ODE model described in subsection 2.3, where the derivative function $h(\cdot)$ is modeled by a multilayer perceptron (MLP).

For the probabilistic approach in subsection 3.3, both the prior distribution $p_\psi(z^{vae}_n|\tau_{1:n-1})$ and the approximate posterior distribution $q_\phi(z^{vae}_n|\tau_{1:n})$ are modeled by LSTMs. The log-likelihood term $\log p_{\theta,\rho}(\tau_n|z^{vae}_n)$ is computed in two steps. First, a decoder network maps the latent variable z^{vae}_n to a base distribution $p_\rho(z^\tau_n|z^{vae}_n)$, which is also a normal distribution. The decoder network is a MLP that outputs the parameters of the base distribution. After that, the flow module g_θ generates the distribution of the inter-arrival time $p_{\theta,\rho}(\tau_n|z^{vae}_n)$. The architecture of g_θ is the same as in the deterministic approach.

A.2 Implementation Details

For the LSTM cells, we choose hidden size to be 128 for the synthetic data and **Breakfast** dataset, 64 for **LinkedIn** and **MIMIC** datasets. The dimension of the latent space of VAE models is set to be 256 for **Breakfast** and synthetic datasets and 64 for **LinkedIn** and **MIMIC**. For PPF-P model, the latent code was decoded into the mean and variance of the base distribution by two separate decoders, each with two hidden layers of size 256. For all continuous normalizing modules, we use one block of network with 3 hidden layers of 64 dimension. We use Adam optimizer (Kingma & Ba, 2015) for all models with a learning rate of 0.001.