Modeling Human Actions in Time-Stamped Activity Sequences

Vinayak Gupta
IIT Delhi
vinayak.gupta@cse.iitd.ac.in

Srikanta Bedathur
IIT Delhi
srikanta@cse.iitd.ac.in

ABSTRACT

Any human activity can be represented as a temporal sequence of actions performed to achieve a certain goal. Unlike machine-made time series, these action sequences are highly disparate as the time taken to finish a similar action might vary between different persons. Therefore, understanding the dynamics of these sequences is essential for many downstream tasks such as activity length prediction, goal prediction, *etc.* Existing neural approaches that model an activity sequence are limited to visual data and completely ignore the temporal aspect associated with every action. In this paper, we revisit a recent approach, called ProActive, a neural marked temporal point process (MTPP) framework for modeling the continuous-time distribution of actions in an activity sequence. Specifically, via ProActive we address three high-impact problems – next action prediction, sequence-goal prediction, and a first-of-its-kind application of *end-to-end* sequence generation.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

marked temporal point process; continuous-time sequences; activity modeling; goal prediction; sequence generation

ACM Reference Format:
Vinayak Gupta and Srikanta Bedathur. 2022. Modeling Human Actions in Time-Stamped Activity Sequences. In *Applied Machine Learning Methods for Time Series Forecasting (AMLTS) Workshop, October 21, 2022, Atlanta, Georgia, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A majority of the data generated via human activities, *e.g.*, running, playing basketball, cooking, *etc.*, can be represented as a continuous-time action sequence (CTAS). These actions denote a step taken by a user towards achieving a certain goal and vary in their start and completion times, depending on the user and the surrounding environment[7, 8]. For *e.g.*, one person making omelets may take a longer time to cook eggs while another may prefer to cook for a short time; or in a football game, Xavi may make a faster pass than Pirlo, even though the goals and the sequence of actions are the same. Thus, action sequences vary significantly from machine-made or synthetic time series. In recent years, neural marked temporal point processes (MTPP) have shown significant promise in modeling a variety of continuous-time sequences in healthcare, finance, and social networks [1, 2, 4, 5, 9, 15, 16]. However, standard MTPP have a limited modeling ability for CTAS as: (i) they assume a homogeneity among sequences, *i.e.*, they cannot distinguish between two sequences of similar actions but with different time duration; (ii) in a CTAS, an action may finish before the start of the next action and thus, to model this empty time interval an MTPP must introduce a new action type, *e.g.*, *NULL* or *end-action*, which may lead to an unwarranted increase in the types of actions to be modeled; and (iii) they cannot encapsulate the additional features associated with an action, for *e.g.*, the minimum time for completion or can be extended to sequence generation.

Our Contribution. In this paper, we summarize a recently proposed approach, called ProActive (**P**oint P**ro**cess flows for **Activ**ity S**e**quences) [3], that is a neural MTPP framework, designed specifically to model the dynamics of a CTAS. Specifically, ProActive is capable to address three challenging tasks – (i) action prediction; (ii) goal detection; and (iii) the first-ever application for MTPP via *end-to-end* sequence generation. We learn the distribution of actions in a CTAS using temporal normalizing flows (NF) [10] conditioned on the dynamics of the sequence as well as the action features (*e.g.*, minimum completion time, *etc.*). Moreover, our model is designed for *early* detection of the sequence goal, *i.e.*, identifying the result of a CTAS without traversing the complete sequence. We achieve this by using a time-bounded optimization procedure, *i.e.*, by incrementally increasing the probability of identifying the true goal. This further provides ProActive the ability to learn the hierarchy between the sequence goal and the actions within a CTAS.

2 PROACTIVE MODEL

2.1 Problem Formulation

We represent an activity via a continuous-time action sequence derived from videos, *i.e.*, a series of actions undertaken by users and their corresponding time of occurrences. Specifically, we represent a CTAS as $\mathcal{S}_k = \{e_i = (c_i, t_i) | i \in [k], t_i < t_{i+1}\}$, where $t_i \in \mathbb{R}^+$ is the start time of the action, $c_i \in C$ is the discrete category or mark of the i-th action, C is the set of all categories, $\Delta_{t,i} = t_i - t_{i-1}$ as the inter-action time, and $\mathcal{S}_k$ denotes the sequence of first k actions. Each CTAS has an associated result, $g \in \mathcal{G}$, that signifies the goal of the CTAS. Here, $\mathcal{G}$ denotes the set of all possible sequence goals.

Input to ProActive. A CTAS of all actions, $\mathcal{S}_k$, consisting of categories and times of different actions that lead to a goal g.

Output by ProActive. A probabilistic prediction model with three distinct tasks – (i) to estimate the likelihood of the next action e_{k+1} along with the action category and occurrence time; (ii) to predict the goal of the CTAS being modeled, *i.e.*, $\widehat{g}$; and (iii) a generative model to sample a sequence of actions, $\widehat{\mathcal{S}}$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
AMLTS '22, October 21, 2022, Atlanta, Georgia, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.2 High Level Overview

We use an MTPP denoted by $p_\theta(\cdot)$, to learn the generative mechanism of a continuous-time action sequence. Moreover, we design the sequence modeling framework of $p_\theta(\cdot)$ using a self-attention based encoder-decoder model [12]. Specifically, we embed the actions in a CTAS, *i.e.*, $\mathcal{S}_k$, to a vector embedding, denoted by $\mathbf{s}_k$, using a weighted aggregation of all past actions. Therefore, $\mathbf{s}_k$ signifies a compact neural representation of the sequence history, *i.e.*, all actions till the k-th index and their marks and occurrence times. We use our MTPP $p_\theta(\cdot)$ to estimate the generative model for the $(k+1)$-th action conditioned on the past, *i.e.*, $p(e_{k+1})$ as:

$$p_\theta(e_{k+1}|\mathbf{s}_k) = \mathbb{P}_\theta(c_{k+1}|\mathbf{s}_k) \cdot \rho_\theta(\Delta_{t,k+1}|\mathbf{s}_k), \tag{1}$$

where, $\mathbb{P}_\theta(\cdot)$ and $\rho_\theta(\cdot)$ denote the probability distribution of marks and the density function for inter-action arrival times respectively. Note that both the functions are conditioned on $\mathbf{s}_k$ and thus ProActive requires a joint optimizing procedure for both – action time and mark prediction. Next, we describe the mechanism used in ProActive to predict the next action and goal detection in a CTAS.

Next Action Prediction. We determine the most probable mark and time of the next action, using $p_\theta(\cdot)$ via standard sampling techniques over $\mathbb{P}_\theta(\cdot)$ and $\rho_\theta(\cdot)$ respectively [1, 10].

$$\widehat{e_{k+1}} \sim p_\theta(e_{k+1}|\mathbf{s}_k), \tag{2}$$

To keep the history embedding up-to-date, we incrementally update $\mathbf{s}_k$ to $\mathbf{s}_{k+1}$ by incorporating the details of action e_{k+1}.

Goal Detection. Since the history embedding, $\mathbf{s}_k$, represents an aggregation of all past actions in a sequence, it can also be used to capture the influences between actions and thus, can be extended to detect the goal of the CTAS. Specifically, to detect the CTAS goal, we use a non-linear transformation over $\mathbf{s}_k$ as:

$$\widehat{g} \sim \mathbb{P}_{g' \in \mathcal{G}}(\Phi(s_k)), \tag{3}$$

where $\mathbb{P}_\bullet$ denotes the distribution over all sequence goals and $\Phi(\cdot)$ denotes the transformation via a fully-connected MLP layer.

2.3 Neural Parameterization

We realize our MTPP, $p_\theta(\cdot)$, using a multi-layer architecture:

Input Layer. As each action $e_i \in \mathcal{S}_k$ is represented by a mark c_i and time t_i, we embed each action as a combination of both as:

$$\mathbf{y}_i = \boldsymbol{w}_{y,c} c_i + \boldsymbol{w}_{y,t} t_i + \boldsymbol{w}_{y,\Delta} \Delta_{t,i} + \mathbf{b}_y, \tag{4}$$

where $\boldsymbol{w}_{\bullet,\bullet}, \mathbf{b}_\bullet$ are trainable parameters and $\mathbf{y}_i \in \mathbb{R}^D$ denotes the vector embedding for the action e_i respectively.

Self-Attention Layer. We use a *masked* self-attention layer to embed the past actions to $\mathbf{s}_k$ and to interpret the influence between the past and the future actions. We follow the standard attention procedure [12] to calculate an attentive aggregation of all actions in the past, *i.e.*, we perform three independent linear transformations to get the *query*, *key*, and *value* embeddings, *i.e.*,

$$\mathbf{q}_i = \boldsymbol{W}^Q \mathbf{y}_i, \quad \mathbf{k}_i = \boldsymbol{W}^K \mathbf{y}_i, \quad \boldsymbol{v}_i = \boldsymbol{W}^V \mathbf{y}_i, \tag{5}$$

where, $\mathbf{q}_\bullet, \mathbf{k}_\bullet, \boldsymbol{v}_\bullet$ denote the query, key, and value vectors respectively. Following standard self-attention model, we represent $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$ and $\boldsymbol{W}^V$ as trainable *Query*, *Key*, and *Value* matrices respectively. Finally, we compute $\mathbf{s}_k$ conditioned on the history as:

$$\mathbf{s}_k = \sum_{i=1}^{k} \frac{\exp\left(\mathbf{q}_k^\top \mathbf{k}_i / \sqrt{D}\right)}{\sum_{i'=1}^{k} \exp\left(\mathbf{q}_k^\top \mathbf{k}_{i'} / \sqrt{D}\right)} \boldsymbol{v}_i, \tag{6}$$

where D denotes the number of hidden dimensions. In addition, we employ: (i) point-wide feed-forward layer, (ii) layer normalization; (iii) stacking multiple self-attention blocks; and (iv) multi-head attention. Since these are standard techniques, we omit their mathematical descriptions in this paper.

Output Layer. At every index k, ProActive outputs the next action and the most probable goal of the CTAS.

Action Prediction: We use the output of the self-attention layer, $\mathbf{s}_k$ to estimate the mark distribution and time density of the next event, *i.e.*, $\mathbb{P}_\theta(e_{k+1})$ and $\rho_\theta(e_{k+1})$ respectively. Specifically, we model the $\mathbb{P}_\theta(\cdot)$ as a softmax over all other marks as:

$$\mathbb{P}_\theta(c_{k+1}) = \frac{\exp\left(\boldsymbol{w}_{c,s}^\top \mathbf{s}_i + \mathbf{b}_{c,s}\right)}{\sum_{c'=1}^{|C|} \exp\left(\boldsymbol{w}_{c',s}^\top \mathbf{s}_i + \mathbf{b}_{c',s}\right)}, \tag{7}$$

where, $\boldsymbol{w}_{\bullet,\bullet}$ and $\mathbf{b}_{\bullet,\bullet}$ are trainable parameters.

We capture the inter-action arrival times via a *temporal* normalizing flow (NF). In detail, we use a *LogNormal* flow to model the temporal density $\rho_\theta(\Delta_{t,k+1})$. However, standard flow-based approaches [10] utilize a common NF for all events in a sequence, *i.e.*, the arrival times are determined from a single or mixture of flows trained on all sequences. Such an assumption limits the ability to model a CTAS, as unlike standard events, actions have two distinguishable characteristics – (i) a required minimum time for completion and (ii) the time taken by a user to complete an action would be similar to the times of another user. For example, the time taken to complete the action '*add-coffee*' would require a certain minimum time of completion and these times would be similar for all users. To incorporate these features in ProActive, we identify actions with similar completion times and model them via independent temporal flows. Specifically, we cluster all actions $c_i \in C$ into $\mathcal{M}$ non-overlapping clusters based on the *mean* of their times of completion and for each cluster we define a trainable embedding $\boldsymbol{z}_r \in \mathbb{R}^D \; \forall r \in \{1, \cdots, \mathcal{M}\}$. Later, we sample the start-time of the future action by conditioning our temporal flows on the cluster of the current action as:

$$\widehat{\Delta_{t,k+1}} \sim \text{LogNormal}\left(\mu_k, \sigma_k^2\right), \tag{8}$$

where, $[\mu_k, \sigma_k^2]$, denote the mean and variance of the log-normal temporal flow and are calculated via the sequence embedding and the cluster embedding as:

$$\mu_k = \sum_{r=1}^{\mathcal{M}} \mathcal{R}(e_k, r)\left(\boldsymbol{w}_\mu \left(\mathbf{s}_k \odot \boldsymbol{z}_{c,i}\right) + \mathbf{b}_\mu\right), \tag{9}$$

$$\sigma_k^2 = \sum_{r=1}^{\mathcal{M}} \mathcal{R}(e_k, r)\left(\boldsymbol{w}_\sigma \left(\mathbf{s}_k \odot \boldsymbol{z}_{c,i}\right) + \mathbf{b}_\sigma\right), \tag{10}$$

where $\boldsymbol{w}_\bullet, \mathbf{b}_\bullet$ are trainable parameters, $\mathcal{R}(e_k, r)$ is an indicator function that determines if event e_k belongs to the cluster r and $\boldsymbol{z}_r$ denotes the corresponding cluster embedding. Such a cluster-based formulation facilitates the ability of the model to assign similar completion times for events in the same cluster. To calculate the time of the next action, we add the sampled time difference to the time of the previous action e_k, *i.e.*,

$$\widehat{t_{k+1}} = t_k + \widehat{\Delta_{t,k+1}}, \tag{11}$$

where, $\widehat{t_{k+1}}$ denotes the predicted time for the action e_{k+1}.

Goal Detection: ProActive can also identify the goal of a sequence, *i.e.*, a hierarchy on top of the actions in a sequence, based on the

past sequence dynamics. To determine the goal of a CTAS, we utilize the history embedding $\mathbf{s}_k$ as it encodes the inter-action relationships of all actions in the past. Specifically, we use a non-linear transformation via a feed-forward network, denoted as $\Phi(\cdot)$ over $\mathbf{s}_k$ and apply a softmax over all possible goals.

$$\Phi(\mathbf{s}_k) = \textsc{ReLU}(\boldsymbol{w}_{\Phi,s}\mathbf{s}_k + \mathbf{b}_{\Phi,s}), \tag{12}$$

where, $\boldsymbol{w}_{\bullet,\bullet}, \mathbf{b}_{\bullet,\bullet}$ are trainable parameters. We sample the most probable goal as in Eqn. (3).

We predict the CTAS goal at each interval, though a CTAS can have only one goal. This facilitates *early* goal detection in comparison to detecting the goal after traversing the entire CTAS.

Early Goal Detection. To facilitate early detection of the goal of a CTAS in ProActive, we devise a ranking loss that forces the model to predict a *non-decreasing* detection score for the correct goal category. Specifically, the detection score of the correct goal at the k-th index of the sequence, denoted by $p_k(g|\mathbf{s}_k, \Phi)$, must be more than the scores assigned the correct goal in the past. Formally, we define the ranking loss as:

$$\mathcal{L}_{k,g} = \max\left(0, p_k^*(g) - p_k(g|\mathbf{s}_k, \Phi)\right), \tag{13}$$

where $p_k^*(g)$ denotes the maximum probability score given to the correct goal in all past predictions.

$$p_k^*(g) = \max_{j\in\{1,k-1\}} p_j(g|\mathbf{s}_j, \Phi), \tag{14}$$

where $p_k(g)$ denotes the probability score for the correct goal at index k. Intuitively, the ranking loss $\mathcal{L}_{k,g}$ would penalize the model for predicting a smaller detection score for the correct CTAS goal than any previous detection score for the same goal.

Action Hierarchy. Standard MTPP approaches assume the category of marks as independent discrete variables, *i.e.*, the probability of an upcoming mark is calculated independently [1, 9, 15, 16]. Such an assumption restricts the predictive ability while modeling CTAS, as in the latter case, there exists a hierarchy between goals and actions that lead to the specific goal. We incorporate this hierarchy in ProActive via an action-based ranking loss. In detail, we devise a loss function, denoted by $\mathcal{L}_{k,c}$, similar to Eqn. 13 where we restrict the model to assign non-decreasing probabilities to all actions leading to the goal of CTAS under scrutiny.

Optimization. We optimize the trainable parameters in ProActive using a two channels of training – action and goal prediction. Specifically, to optimize the ability of ProActive for predicting the next action, we maximize the joint likelihood for the next action and the log-normal density distribution of the temporal flows.

$$\mathscr{L} = \sum_{k=1}^{|\mathcal{S}|} \log\left(\mathbb{P}_\theta(c_{k+1}|\mathbf{s}_k) \cdot \rho_\theta(\Delta_{t,k+1}|\mathbf{s}_k)\right), \tag{15}$$

where $\mathscr{L}$ denotes the joint likelihood, which we represent as the sum of the likelihoods for all CTAS. In addition, to optimize the parameters for *early* goal detection via a temporally weighted cross entropy (CE) loss overall sequence goals. Specifically, we follow a popular reinforcement recipe of using a time-varying *discount* factor over the prediction loss as:

$$\mathcal{L}_g = \sum_{k=1}^{|\mathcal{S}|} \gamma^k \cdot \mathcal{L}_{CE}(p_k(g|\mathbf{s}_k)), \tag{16}$$

where $\gamma \in [0, 1]$, $\mathcal{L}_{CE}(p_k(g|\mathbf{s}_k))$ denote the decaying factor and a standard softmax cross-entropy loss respectively. The discount factor penalizes the model for taking longer times for detecting the CTAS goal by decreasing the gradient updates to the loss [11]. In addition, we minimize the margin losses $\mathcal{L}_{k,g}$ and $\mathcal{L}_{k,c}$.

Sequence Generation. A crucial contribution of this paper via ProActive is an end-to-end generation of action sequences. Specifically, given the CTAS goal as input, we can generate a most probable sequence of actions that may lead to that specific goal. A standard approach for training a sequence generator is to sample future actions in a sequence and then compare them with the true actions [14]. However, such a procedure has multiple drawbacks as it is susceptible to noises during training, deteriorates the scalability of the model, and cannot be used with a self-attention-based model as it requires a fixed-sized sequence as input [12]. Therefore, we resort to a two-step generation procedure that is defined below:

(1) **Pre-Training:** This step requires training all ProActive parameters for action prediction and goal detection. This step is necessary to model the relationships between actions and goals and we represent the set of optimized parameters as θ^* and the corresponding MTPP as $p_{\theta^*}(\cdot)$ respectively.
(2) **Iterative Sampling:** We iteratively sample events and update parameters via our trained MTPP till the model predicts the correct goal for the CTAS or we encounter an <EOS> or end-of-sequence action. Specifically, using $p_{\theta^*}(\cdot)$ and the first *real* action (e_1) as input, we calculate the detection score for the correct goal, *i.e.*, $p_1(g|\mathbf{s}_k)$ and while its value is highest among all probable goals, we sample the mark and time of next action using Eqn. 7 and Eqn. 8 respectively.

Such a generation procedure harnesses the fast sampling of temporal normalizing flows and simultaneously is conditioned on the action and goal relationships.

3 EXPERIMENTS

Here, we present the experimental evaluation of ProActive.

3.1 Experimental Setup

Datasets. We derive CTAS from two activity modeling datasets – Breakfast [7] with videos of different people preparing breakfast, Multi-THUMOS [13] and Activity-Net [6] that is a collective activity dataset derived from YouTube videos.

Baselines. We compare the action prediction performance of ProActive with the several state-of-the-art methods including, RMTPP [1], NHP [9], AVAE [8], SAHP [15], and THP [16].

Evaluation Criteria. We split CTAS into training and test sets based on the sequence goal. Specifically, for each goal $g \in \mathcal{G}$, we consider 80% of the sequences for training and the rest as the test set. We evaluate ProActive in terms of (i) mean absolute error (MAE) of predicted times of action and (ii) action prediction accuracy (APA). We calculate confidence intervals across 5 independent runs.

System Configuration. All our experiments were done on an NVIDIA Tesla T4 GPU with 16GB DDR6 memory.

3.2 Performance of ProActive

Action Prediction. We report on the performance of action prediction of different methods across all our datasets in Table 1. In addition, we include two variants of ProActive– (i) ProAct-c,

Table 1: Performance of all the methods in terms of action prediction accuracy (APA) and mean absolute error (MAE) across all datasets. Results marked † are statistically significant (*i.e.*, two-sided t-test with $p \leq 0.1$) over the best baseline.

	Action Prediction Accuracy		Mean Absolute Error	
	Breakfast	Activity-Net	Breakfast	Activity-Net
NHP [9]	0.528±0.024	0.684±0.034	0.411±0.019	0.796±0.045
AVAE [8]	0.533±0.028	0.678±0.036	0.417±0.021	0.803±0.049
RMTPP [1]	0.542±0.022	0.683±0.034	0.403±0.018	0.791±0.046
SAHP [15]	0.547±0.031	0.688±0.042	0.425±0.031	0.820±0.072
THP [16]	0.559±0.028	0.693±0.038	0.413±0.023	0.806±0.061
ProAct-c	0.561±0.027	0.698±0.038	0.415±0.027	0.774±0.054
ProAct-t	0.579±0.025	0.722±0.035	0.407±0.025	0.783±0.058
ProActive	**0.583±0.027**†	**0.728±0.037**†	**0.364±0.028**†	**0.742±0.059**†

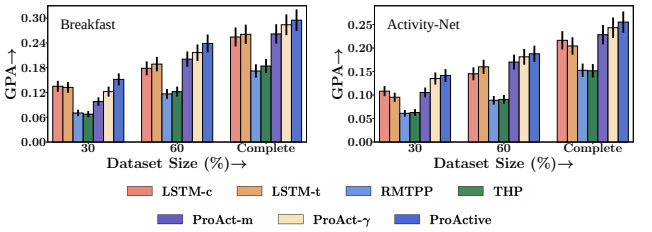

Figure 1: Sequence goal prediction performance of ProActive, its variants, and other baseline models.

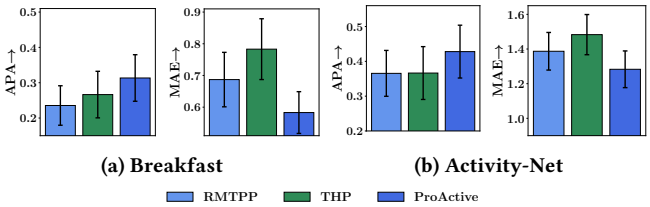

Figure 2: Sequence Generation results for ProActive and other baselines in terms of APA and MAE.

represents our model without the goal-action hierarchy loss and cluster-based flows, and (ii) ProAct-t, represents our model without cluster-based flows. From Table 1, we note that ProActive consistently yields the best prediction performance. In particular, it improves over the strongest baselines by 8-27% for time prediction and by 2-7% for action prediction. The results signify the inability of standard approaches for modeling an action sequence.

Goal Prediction. Here, we evaluate the goal detection performance of ProActive. To highlight the *early* goal detection ability of our model, we report the results across different variants of the training set, *i.e.*, with the initial 30% and 60% of the actions in the CTAS in terms of goal prediction accuracy (GPA). We use two novel baselines, LSTM-c, and LSTM-t, that detect the CTAS goal using just the types and the times of actions respectively. In addition, we introduce two new variants of our model – (i) ProActive-m, represents our model without the goal-based margin loss given; and (ii) ProActive-γ, is our model without the discount-factor weight. The results for goal detection in Figure 1, show that the complete design of ProActive achieves the best performance among all other models. We also note that the performance of MTPP-based models deteriorates significantly for this new task which shows the unilateral nature of the prediction prowess of MTPP models.

Sequence Generation/ Forecasting. Here, we evaluate the sequence generation ability of ProActive. Since there may be differences in lengths of the generated and true sequences, we compare the actions in the true sequence with the initial $|\mathcal{S}|$ generated actions. Such an evaluation procedure provides us the flexibility of comparing with other MTPP models such as RMTPP [1] and THP [16]. As these models cannot be used for end-to-end sequence generation, we alter their underlying model for *forecasting* future actions given the first action and then incrementally update and sample from the MTPP parameters. We report the results in terms of APA and MAE for action and time prediction in Figure 2. The results show that ProActive can better capture the generative dynamics of a CTAS than other MTPP models.

Length Comparison. We also report the results for length comparison of the generated sequence and the true sequence. Specifically, we identify the count of instances where ProActive was able to effectively capture the generative mechanism of a sequence as, $\text{CL} = \frac{1}{N}\sum_{\forall \mathcal{S}} \#(|\mathcal{S}| = |\widehat{\mathcal{S}}|)$, where CL denotes the *Correct-Length* ratio with values 0.21, 0.11, and 0.16 for datasets Breakfast, Multi-THUMOS, and Activity-Net respectively.

4 CONCLUSION

Standard deep-learning models are not designed for modeling sequences of actions localized in continuous time. In this paper, we present ProActive, a point process framework for modeling the dynamics of a CTAS. ProActive solves the problems associated with action prediction, goal prediction, and for the first time, we extend MTPP for end-to-end CTAS generation. Experiments on large-scale diverse datasets highlight the predictive prowess of ProActive.

REFERENCES

[1] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*.
[2] Vinayak Gupta and Srikanta Bedathur. 2021. Region Invariant Normalizing Flows for Mobility Transfer. In *CIKM*.
[3] Vinayak Gupta and Srikanta Bedathur. 2022. ProActive: Self-Attentive Temporal Point Process Flows for Activity Sequence. In *KDD*.
[4] Vinayak Gupta, Srikanta Bedathur, Sourangshu Bhattacharya, and Abir De. 2021. Learning Temporal Point Processes with Intermittent Observations. In *AISTATS*.
[5] Vinayak Gupta, Srikanta Bedathur, and Abir De. 2022. Learning Temporal Point Processes for Efficient Retrieval of Continuous Time Event Sequences. In *AAAI*.
[6] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *CVPR*.
[7] H. Kuehne, A. B. Arslan, and T. Serre. 2014. The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities. In *CVPR*.
[8] Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. 2019. A Variational Auto-Encoder Model for Stochastic Point Processes. In *CVPR*.
[9] Hongyuan Mei and Jason M Eisner. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*.
[10] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. 2020. Intensity-Free Learning of Temporal Point Processes. In *ICLR*.
[11] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*.
[12] Ashish Vaswani et al. 2017. Attention is all you need. In *NeurIPS*.
[13] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. 2015. Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos. *arXiv preprint arXiv:1507.05738* (2015).
[14] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In *NeurIPS*.
[15] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. 2020. Self-attentive Hawkes processes. In *ICML*.
[16] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. 2020. Transformer Hawkes Process. In *ICML*.