TrajGPT: Healthcare Time-Series Representation Learning for Trajectory Prediction

Ziyang Song[1] **Qincheng Lu**[1] **He Zhu** [1] **David Buckeridge** [2] **Yue Li**[1] *

[1] School of Computer Science, McGill University [2] School of Population and Global Health, McGill University

Abstract

In many domains, such as healthcare, time-series data is irregularly sampled with varying time intervals. It creates challenges for classical time-series models that require equally spaced data. To address this, we propose a novel time-series Transformer called **Trajectory Generative Pre-trained Transformer (TrajGPT)**. It proposes a data-dependent decay mechanism that adaptively forgets irrelevant information based on clinical context. By interpreting as ordinary differential equations, our approach captures the continuous dynamics from irregular time-series data. Experimental results show that TrajGPT, with its time-specific inference approach, accurately predicts trajectories without requiring fine-tuning.

1 Introduction

Time-series representation learning plays a crucial role in various domains, as it extracts generalizable temporal patterns from large-scale, unlabeled data that can be adapted for specific tasks. A major challenge arises when dealing with irregularly-sampled time series, where observations occur at uneven intervals. This irregularity poses challenges for classical time-series models that are restricted to regular sampling [1, 26]. This issue is particularly significant in healthcare domain since longitudinal medical records are updated sporadically during outpatient visits or inpatient stays [26]. Additionally, most patients' medical histories are sparse, making it difficult to predict long-term health trajectories. Addressing these challenges requires the development of novel representation learning approaches that can learn meaningful patterns from irregular and sparse data, enabling accurate trajectory prediction.

Recent progresses in modeling irregularly-sampled time series have been achieved through specialized deep learning architectures [2, 7, 14, 16, 26], but these models lack the ability to pre-train generalizable representations. Though time-series Transformer models have gained attention [11, 24, 25], they are primarily designed for continuous data and often struggle to extrapolate over long sequences [17]. To handle both continuous and irregularly-sampled time series, TimelyGPT addresses these limitations by incorporating extrapolatable position embedding (xPos) for forecasting long sequences beyond the training length [17]. BiTimelyGPT extends the unidirectional TimelyGPT by pre-training bidirectional contextualized representations for discriminative tasks [18]. Nonetheless, these models only utilize data-independent decay mechanism, which is not content-aware and fails to capture the complex temporal dependencies in healthcare data.

In this study, we propose a **Trajectory Generative Pre-trained Transformer (TrajGPT)**, a novel architecture for healthcare time-series representation learning. TrajGPT introduces a data-dependent decay mechanism that adaptively forgets past information based on the medical context of observations. By interpreting our model as ODEs, TrajGPT effectively models continuous dynamics in irregularly-sampled time series, facilitating accurate forecasting of patient health trajectories.

*Correspondence to yue.yl.li@mcgill.ca.

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

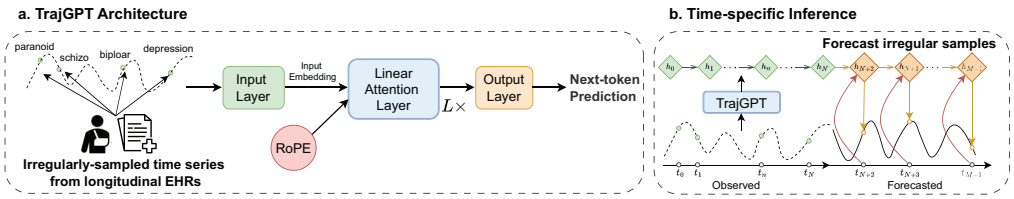

Figure 1: Overview of the TrajGPT architecture and time-specific inference. (a) TrajGPT processes time-series data by embedding an input sequence with RoPE and passing it through L linear attention layers. (b) Time-specific inference operates as neural ODEs, directly predicting irregular samples using historical hidden states and target timesteps.

2 Methodology

We denote an irregularly-sampled time series as $x = \{s_1, \ldots, s_N\}$, where N is the total number of samples. Each samples s_n is represented as a tuple (x_n, t_n), with x_n as the observation (e.g., a structured diagnosis code) and t_n as the corresponding timestep. The input sequence is then projected into embeddings $\boldsymbol{X} \in \mathbb{R}^{N\times d}$, where d denotes the embedding size, through an input projection layer.

2.1 TrajGPT Methodology

Our proposed TrajGPT utilizes rotary position embedding (RoPE) to encode relative positional information between tokens n and m based on their interval $n - m$ [19]:

$$\boldsymbol{Q}_n = \boldsymbol{X}_n \boldsymbol{W}_Q e^{i\theta n},\ \boldsymbol{K}_m = \boldsymbol{X}_m \boldsymbol{W}_K e^{-i\theta m},\ \boldsymbol{V}_m = \boldsymbol{X}_m \boldsymbol{W}_V \tag{1}$$

We propose the TrajGPT architecture, built on the linear Transformer framework [10]. To compute the output embedding $\boldsymbol{O}$, we introduce the recurrent form of the linear attention layer as follows:

$$\boldsymbol{S}_n = \gamma_n \boldsymbol{S}_{n-1} + \boldsymbol{K}_n^\top \boldsymbol{V}_n,\ \boldsymbol{O}_n = \boldsymbol{Q}_n \boldsymbol{S}_n,\ \text{where } \gamma_n = \text{Sigmoid}(\boldsymbol{X}_n \boldsymbol{W}_\gamma^\top)^{\frac{1}{\tau}} \tag{2}$$

where $\boldsymbol{S}_n$ and γ_n represent the state variable and the data-dependent decay rate, respectively. The temperature parameter $\tau = 16$ prevents rapid decay of past information, allowing the model to retain relevant information over long sequences. The data-dependent decay rate γ_n enables the model to adapt to clinical contexts based on observations, efficiently forgetting irrelevant information. For chronic diseases, the model assigns higher γ_n values, which slow forgetting process and capture long-term dependencies. Conversely, lower γ_n values accelerate decay to prioritize recent events, enhancing the model's responsiveness to acute conditions. The parallel forward pass of our linear attention is efficiently computed using a precomputed decay matrix $\boldsymbol{D}$:

$$\boldsymbol{O} = (\boldsymbol{Q}\boldsymbol{K}^\top \odot \boldsymbol{D})\boldsymbol{V},\ \boldsymbol{D}_{nm} = \begin{cases} \frac{b_n}{b_m}, & n \geq m \\ 0, & n < m \end{cases},\ \text{where } b_n = \prod_{t=1}^{n} \gamma_t \tag{3}$$

A detailed derivation is provided in Appendix A. This parallel formulation allows for efficient training with linear complexity, while the recurrent form ensures constant complexity during inference [10]. To capture a broader range of contexts, We extend the single-head SRA in Eq. 2 to a multi-head SRA:

$$\boldsymbol{O}_n^h = \boldsymbol{Q}_n^h \boldsymbol{S}_n^h,\ \boldsymbol{S}_n^h = \gamma_n^h \boldsymbol{S}_{n-1}^h + \boldsymbol{K}_n^{h\top} \boldsymbol{V}_n^h,\ \text{where } \gamma_n^h = \text{Sigmoid}(\boldsymbol{X}_n \mathbf{w}_\gamma^{h\top})^{\frac{1}{\tau}}, \tag{4}$$

2.2 Connection to SSM and ODE

We establish a theoretical connection between our proposed linear attention layer and state space models (SSMs) and ODEs. Due to space limitations, we provide only a high-level summary in this section, with detailed derivations available in Appendix B. At a high level, each head of the linear attention layer in Eq. 4 can be interpreted as a discrete SSM that approximates a continuous SSM [6]. This continuous SSM is mathematically formulated as an ODE, where the continuous dynamics are captured by data-dependent parameters.

We consider a neural ODE defined as $\frac{d\boldsymbol{S}(t)}{dt} = f(\boldsymbol{S}(t), t, \theta_t)$, where f is a differentiable neural network and θ_t represents data-dependent, time-varying parameters [3]. Our linear attention layer in

Eq. 4 can be approximated by the following neural ODE with a discrete step size Δ:

$$\frac{d\boldsymbol{S}(t)}{dt} = \boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{B}^\top \boldsymbol{V}(t) \Longleftrightarrow f(\boldsymbol{S}(t), t, \theta_t)$$
$$\text{where } \theta_t = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}),\ \boldsymbol{A} = \frac{\ln(\gamma_t^h)}{\Delta},\ \boldsymbol{B} = \boldsymbol{A}(e^{\Delta \boldsymbol{A}} - \boldsymbol{I})^{-1}\boldsymbol{K}_t^\top,\ \boldsymbol{C} = \boldsymbol{Q}_t \tag{5}$$

As a result, each head of TrajGPT captures unique dynamics of patient health trajectories through the neural ODE framework. Therefore, our model can forecast irregular observations in the time-series by taking into account the discretization step size Δ [3, 14]. We thus propose an effective **time-specific inference** that makes predictions at arbitrary timesteps. For each sample $s_n = (x_n, t_n)$, the model utilizes both the timestep t_n and the preceding sample $s_{n-1} = (x_{n-1}, t_{n-1})$ to predict the observation x_n based on its discrete step size $\Delta t_{n,n-1} = t_n - t_{n-1}$.

3 Experiments

3.1 Dataset and Pre-processing

The Population Health Record (PopHR) database hosts massive amounts of longitudinal claim data from the provincial government health insurer in Quebec, Canada (Régie de l'assurance maladie du Québec, RAMQ) [15, 23]. There are approximately 1.3 million participants in the PopHR database, which represents a randomly sampled 25% of the Montreal population between 1998 and 2014. We extracted irregularly-sampled time series from the PopHR database. Specifically, we converted ICD-9 diagnostic codes to phenotype codes (PheCodes) using the PheWAS [4, 5]. We selected 315 unique PheCodes, each with over 50,000 occurrences, and excluded patients with fewer than 50 PheCode records. This resulted in a dataset of 489,000 patients, with an average of 112 records per patient. The dataset was then split into training (80%), validation (10%), and testing (10%) sets.

3.2 Forecasting irregularly-sampled diagnostic codes

We evaluated the long-term forecasting task using irregularly-sampled time series data from the PopHR database. We set a look-up window of 50 timestamps, using the remaining codes as the forecasting windows, which can extend to over 100 timestamps (e.g., diagnosis codes). To assess model performance, we used the top-K recall metric with $K = (5, 10, 15)$.

We compared our model against several time-series transformer baselines, including TimelyGPT [17], BiTimelyGPT [18], Informer [27], Fedformer [28], AutoFormer [22], and PatchTST [11]. Notably, TimelyGPT and BiTimelyGPT are specifically designed to handle irregularly-sampled time series through a time decay mechanism. Additionally, we evaluated models designed for irregularly-sampled time series, including mTAND [16], GRU-D [2], RAINDROP [26], and SeFT [7]. For the Transformer models, we followed established pre-training procedures [17], pre-training on the entire training data and forecasting without additional fine-tuning. In contrast, the models designed for irregular time-series were trained from scratch on the training set. We followed previous works to set Transformer model parameters to about 7.5 million, as detailed in Table 2.

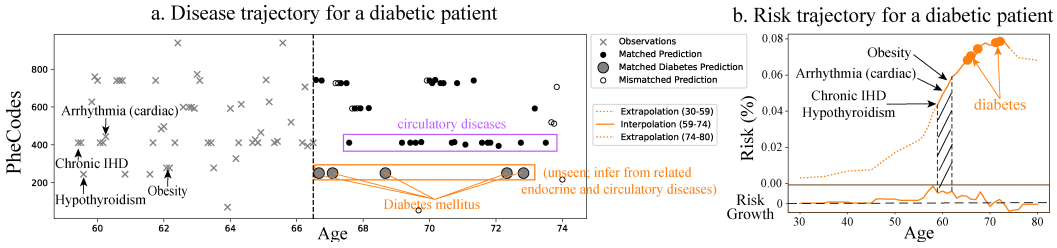

Figure 2: Health trajectories for a diabetic patient. **(a) and (b)** show the inferred disease trajectories with look-up and forecast windows. Matched predictions (solid circles) occur when the top 10 predicted PheCodes match the ground-truth. Larger solid circles indicate correctly predicted diabetes.

Table 1: Forecasting performance on PopHR's irregular-sampled time series dataset. TrajGPT achieved the highest recall at $K = 10$ and $K = 15$ and second highest at $K = 5$. The bold and underline indicate the best and second best results, respectively.

Metrics	**Recall @K (%)**		
	$K = 5$	$K = 10$	$K = 15$
TrajGPT (Time-specific inference)	<u>57.42</u>	**71.67**	**84.05**
TrajGPT (Standard inference)	53.26	65.48	77.21
TimelyGPT	**58.65**	<u>70.83</u>	82.69
BiTimelyGPT	48.17	63.26	70.53
Informer	46.37	60.14	71.24
Autoformer	42.87	57.43	68.59
Fedformer	43.31	58.34	69.60
PatchTST	48.17	65.55	73.31
MTand	52.59	70.21	<u>83.73</u>
GRU-D	54.25	69.48	80.51
RAINDROP	46.52	67.21	72.20
SeFT	49.26	68.10	79.39

3.3 Forecasting Results

TrajGPT with time-specific inference achieves the highest recall rates at $K = 10$ and $K = 15$, with scores of 71.67% and 84.05%, respectively (Table 1). Notably, the time-specific inference outperforms the standard inference approach, demonstrating the advantage of modeling continuous dynamics through the neural ODE framework. These results highlight TrajGPT's superior ability to forecast healthcare trajectories, even when dealing with sparse and irregular time-series data.

We then examined the distributions of the top-10 recall across 3 forecast windows, comparing TrajGPT with TimelyGPT, PatchTST, and mTAND. As shown in Fig. 3, all models experience a performance decline as the forecast window increased, reflecting the increased uncertainty in the long-term trajectory prediction. Despite this, TrajGPT achieves better and more stable performance within the first 100 steps. This stability is attributed to the time-specific inference, which takes into account the evolving states and query timestep over irregular intervals. As a result, our TrajGPT offers a more effective approach for forecasting patient health trajectories compared to existing methods.

We aimed to demonstrate TrajGPT's effectiveness with a case study on a diabetic patient. We visualized the observed and predicted disease trajectories for this patient. Specifically, we estimate the probabilities of the diabetes token across timesteps. We also calculated risk growth by comparing each timestep to the previous one, identifying the ages with high risk growth as well as the associated phenotypes. In Fig. 2, TrajGPT with time-specific inference achieves a top-10 recall of 90.1% for this diabetic patient. TrajGPT predicts most diseases in endocrine/metabolic and circulatory systems. Although this patient has no prior diabetes diagnosis in the observed data, TrajGPT successfully forecasts diabetes onset by identifying related metabolic and circulatory symptoms. Fig. 2.b displays predicted risk trajectory for this patient, indicating a gradual increase in diabetes risk with age. We highlight specific phenotypes that contribute to the noticeable high risk growth between ages 59 and 62, including chronic IHD, hypothyroidism, obesity, and arrhythmia. These conditions are common comorbidities of diabetes, substantially elevating the likelihood of diabetes onset over time.

4 Conclusion and Further Work

In this study, our goal was to develop a representation learning approach capable of handling irregularly-sampled healthcare time series and predicting patient health trajectories. To achieve this, we introduced TrajGPT, a novel time-series Transformer that incorporates a data-dependent decay mechanism and time-specific inference. By linking our model with the Neural ODE framework, we demonstrated its ability to capture continuous dynamics from sparse and irregular observations, achieving superior forecasting performance without task-specific fine-tuning.

To further validate the generalizability, we plan to extend our experiments to public datasets such as MIMIC [9], EHRSHOT [21], and INSPECT [8]. Additionally, we will explore the potential of foundation LLMs, such as GPT-based [20] and Llama-based [13] methods.

References

[1] Jose Roberto Ayala Solares, Francesca Elisa Diletta Raimondi, Yajie Zhu, Fatemeh Rahimian, Dexter Canoy, Jenny Tran, Ana Catarina Pinho Gomes, Amir H. Payberah, Mariagrazia Zottoli, Milad Nazarzadeh, Nathalie Conrad, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. Deep learning for electronic health records: A comparative review of multiple deep neural architectures. *Journal of Biomedical Informatics* 101 (2020), 103337. https://doi.org/10.1016/j.jbi.2019.103337

[2] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports* 8 (04 2018). https://doi.org/10.1038/s41598-018-24271-9

[3] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 6572–6583.

[4] Joshua Denny, Lisa Bastarache, Marylyn Ritchie, Robert Carroll, Raquel Zink, Jonathan Mosley, Julie Field, Jill Pulley, Andrea Ramirez, Erica Bowton, Melissa Basford, David Carrell, Peggy Peissig, Abel Kho, Jennifer Pacheco, Luke Rasmussen, David Crosslin, Paul Crane, Jyotishman Pathak, and Dan Roden. 2013. Systematic comparison of phenome-wide association study of electronic medical record data and genome-wide association study data. *Nature biotechnology* 31 (11 2013). https://doi.org/10.1038/nbt.2749

[5] Joshua C. Denny, Marylyn D. Ritchie, Melissa A. Basford, Jill M. Pulley, Lisa Bastarache, Kristin Brown-Gentry, Deede Wang, Dan R. Masys, Dan M. Roden, and Dana C. Crawford. 2010. PheWAS: demonstrating the feasibility of a phenome-wide scan to discover gene–disease associations. *Bioinformatics* 26, 9 (03 2010), 1205–1210. https://doi.org/10.1093/bioinformatics/btq126

[6] Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. arXiv:2111.00396 [cs.LG] https://arxiv.org/abs/2111.00396

[7] Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. 2020. Set Functions for Time Series. arXiv:1909.12064 [cs.LG]

[8] Shih-Cheng Huang, Zepeng Huo, Ethan Steinberg, Chia-Chun Chiang, Curtis Langlotz, Matthew P Lungren, Serena Yeung, Nigam Shah, and Jason Alan Fries. 2023. INSPECT: A Multimodal Dataset for Pulmonary Embolism Diagnosis and Prognosis. *arXiv preprint arXiv:2311.10798* (2023).

[9] Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Benjamin Moody, Brian Gow, Li wei H. Lehman, Leo Anthony Celi, and Roger G. Mark. 2023. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data* 10 (2023). https://api.semanticscholar.org/CorpusID:255439889

[10] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. arXiv:2006.16236 [cs.LG]

[11] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. arXiv:2211.14730 [cs.LG]

[12] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[13] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. 2024. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. arXiv:2310.08278 [cs.LG] https://arxiv.org/abs/2310.08278

[14] Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. 2019. *Latent ODEs for irregularly-sampled time series*. Curran Associates Inc., Red Hook, NY, USA.

[15] Arash Shaban-Nejad, Maxime Lavigne, Anya Okhmatovskaia, and David Buckeridge. 2016. PopHR: a knowledge-based platform to support integration, analysis, and visualization of population health data: The Population Health Record (PopHR). *Annals of the New York Academy of Sciences* 1387 (10 2016). https://doi.org/10.1111/nyas.13271

[16] Satya Narayan Shukla and Benjamin M. Marlin. 2021. Multi-Time Attention Networks for Irregularly Sampled Time Series. arXiv:2101.10318 [cs.LG]

[17] Ziyang Song, Qincheng Lu, Hao Xu, He Zhu, David L. Buckeridge, and Yue Li. 2024. TimelyGPT: Extrapolatable Transformer Pre-training for Long-term Time-Series Forecasting in Healthcare. arXiv:2312.00817 [cs.LG] https://arxiv.org/abs/2312.00817

[18] Ziyang Song, Qincheng Lu, He Zhu, David Buckeridge, and Yue Li. 2024. Bidirectional Generative Pre-training for Improving Healthcare Time-series Representation Learning. arXiv:2402.09558 [cs.AI] https://arxiv.org/abs/2402.09558

[19] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2022. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864 [cs.CL]

[20] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. 2024. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *International Conference on Learning Representations (ICLR)*.

[21] Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason Fries, and Nigam Shah. 2023. EHRSHOT: An EHR Benchmark for Few-Shot Evaluation of Foundation Models. (2023). arXiv:2307.02028 [cs.LG]

[22] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2022. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. arXiv:2106.13008 [cs.LG]

[23] Mengru Yuan, Guido Powell, Maxime Lavigne, Anya Okhmatovskaia, and David Buckeridge. 2018. Initial Usability Evaluation of a Knowledge-Based Population Health Information System: The Population Health Record (PopHR). *AMIA ... Annual Symposium proceedings. AMIA Symposium* 2017 (04 2018), 1878–1884.

[24] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2020. A Transformer-based Framework for Multivariate Time Series Representation Learning. arXiv:2010.02803 [cs.LG]

[25] Wenrui Zhang, Ling Yang, Shijia Geng, and Shenda Hong. 2023. Self-Supervised Time Series Representation Learning via Cross Reconstruction Transformer. arXiv:2205.09928 [cs.LG]

[26] Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. 2022. Graph-Guided Network for Irregularly Sampled Multivariate Time Series. arXiv:2110.05357 [cs.LG]

[27] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. arXiv:2012.07436 [cs.LG]

[28] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. arXiv:2201.12740 [cs.LG]

A Detailed Derivation of TrajGPT

Given the recurrent form of the TrajGPT model in Eq. 2, assuming $\boldsymbol{S}_0 = 0$, we can derive the cumulative state variable S as follows:

$$
\begin{aligned}
&\boldsymbol{S}_n = \gamma_n \boldsymbol{S}_{n-1} + \boldsymbol{K}_n^\top \boldsymbol{V}_n \\
&\boldsymbol{S}_1 = \boldsymbol{K}_1^\top \boldsymbol{V}_1 \\
&\boldsymbol{S}_2 = \gamma_2 \boldsymbol{K}_1^\top \boldsymbol{V}_1 + \boldsymbol{K}_2^\top \boldsymbol{V}_2 \\
&\boldsymbol{S}_3 = \gamma_3 \gamma_2 \boldsymbol{K}_1^\top \boldsymbol{V}_1 + \gamma_2 \boldsymbol{K}_2^\top \boldsymbol{V}_2 + \boldsymbol{K}_3^T \boldsymbol{V}_3 \\
&\vdots \\
&\boldsymbol{S}_n = \sum_{m=1}^{n} \left(\prod_{t=m+1}^{n} \gamma_t \right) \boldsymbol{K}_m^\top \boldsymbol{V}_m = \sum_{m=1}^{n} \left(\frac{b_n}{b_m} \right) \boldsymbol{K}_m^\top \boldsymbol{V}_m, \text{ where } b_n = \prod_{t=1}^{n} \gamma_t
\end{aligned} \tag{6}
$$

We can compute the output $\boldsymbol{O}_n$ using the $\boldsymbol{Q}_n$:

$$
\begin{aligned}
\boldsymbol{O}_n &= \boldsymbol{Q}_n \boldsymbol{S}_n = \boldsymbol{Q}_n \sum_{m=1}^{n} \left(\frac{b_n}{b_m} \right) \boldsymbol{K}_m^\top \boldsymbol{V}_m \\
&= \boldsymbol{Q}_n \boldsymbol{K}_{m \le n}^\top \boldsymbol{D}_{m \le n} \boldsymbol{V}_{m \le n}, \text{ where } \boldsymbol{D}_{m \le n} = \frac{b_n}{b_m}
\end{aligned} \tag{7}
$$

We generalize the parallel computation for all tokens as follows:

$$
\boldsymbol{O} = (\boldsymbol{Q}\boldsymbol{K}^\top \odot \boldsymbol{D})\boldsymbol{V}, \ \boldsymbol{D}_{nm} = \begin{cases} \frac{b_n}{b_m}, & n \ge m \\ 0, & n < m \end{cases} \tag{8}
$$

The parallel computation of $\boldsymbol{O}_n$ can be rewritten as a matrix product between $(\boldsymbol{Q}_n b_n)$ and $\left(\frac{\boldsymbol{K}_m^\top}{b_m} \right)$.

$$
\boldsymbol{O}_n = \boldsymbol{Q}_n \sum_{m=1}^{n} \left(\frac{b_n}{b_m} \right) \boldsymbol{K}_m^\top \boldsymbol{V}_m = \sum_{m=1}^{n} (\boldsymbol{Q}_n b_n) \left(\frac{\boldsymbol{K}_m^\top}{b_m} \right) \odot \boldsymbol{D}' \boldsymbol{V}_m, \ \boldsymbol{D}'_{nm} = \begin{cases} 1, & n \ge m \\ 0, & n < m \end{cases} \tag{9}
$$

However, when the cumulative decay term $b_m = \prod_{t=1}^{m} \gamma_t$ becomes very small, it can lead to computational errors due to numerical instability. To address this issue, we a temperature term $\tau = 16$ in Eq. 2. This helps to mitigate the effects of rapid decay and retains long-term dependencies.

B TrajGPT as SSM and Neural ODE

The continuous SSM defines a linear mapping from an t-step input signal $\boldsymbol{X}(t)$ to output $\boldsymbol{O}(t)$ via a state variable $\boldsymbol{S}(t)$. It is formulated as an ODE:

$$
\boldsymbol{S}'(t) = \boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{B}\boldsymbol{X}(t), \ \boldsymbol{O}(t) = \boldsymbol{C}\boldsymbol{S}(t) \tag{10}
$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ denote the state matrix, input matrix, and output matrices, respectively. Since data in real-world is typically discrete instead of continuous, continuous SSMs require discretization process to align with the sample rate of the data. This transformation is commonly achieved using the zero-order hold (ZOH) rule, which is detailed in Appendix C [6]:

$$
\begin{aligned}
&\boldsymbol{S}_t = \bar{\boldsymbol{A}} \boldsymbol{S}_{t-1} + \bar{\boldsymbol{B}} \boldsymbol{X}_t, \ \boldsymbol{O}_t = \boldsymbol{C}\boldsymbol{S}_t \\
&\bar{\boldsymbol{A}} = e^{\Delta \boldsymbol{A}}, \ \bar{\boldsymbol{B}} = (e^{\Delta \boldsymbol{A}} - \boldsymbol{I})\boldsymbol{A}^{-1}\boldsymbol{B}
\end{aligned} \tag{11}
$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are the discretized state matrices and Δ is the discrete step size.

We start with the single-head linear attention proposed in Section. 2.1. We rewrite the recurrent form of our TrajGPT in Eq. 2 as follows:

$$
\boldsymbol{S}_t = \Lambda_t \boldsymbol{S}_{t-1} + \boldsymbol{K}_t^\top \boldsymbol{V}_t \tag{12}
$$

where Λ_t is a diagonal matrix, with each element equals to γ_t. This recurrent forward pass of TrajGPT is a discrete SSM [6]:

$$\begin{aligned} \boldsymbol{S}_t &= \Lambda_t \boldsymbol{S}_{t-1} + \boldsymbol{K}_t^\top \boldsymbol{V}_t \quad &\Longleftrightarrow \quad \boldsymbol{S}_t &= \bar{\boldsymbol{A}} \boldsymbol{S}_{t-1} + \bar{\boldsymbol{B}} \boldsymbol{X}_t \\ \boldsymbol{O}_t &= \boldsymbol{Q}_t \boldsymbol{S}_t &\Longleftrightarrow \quad \boldsymbol{O}_t &= \boldsymbol{C} \boldsymbol{S}_t \end{aligned} \tag{13}$$

where $(\Lambda_t, \boldsymbol{K}, \boldsymbol{Q})$ draw parallel to $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \boldsymbol{C})$. We thus derive the connection between $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ and TrajGPT's parameters:

$$\begin{aligned} \bar{\boldsymbol{A}} &= e^{\Delta \boldsymbol{A}} = \Lambda_t &&\Longrightarrow \boldsymbol{A} = \frac{\ln(\Lambda_t)}{\Delta} \\ \bar{\boldsymbol{B}} &= (e^{\Delta \boldsymbol{A}} - \boldsymbol{I}) \boldsymbol{A}^{-1} \boldsymbol{B} = \boldsymbol{K}_t^\top &&\Longrightarrow \boldsymbol{B} = \boldsymbol{A}(e^{\Delta \boldsymbol{A}} - \boldsymbol{I})^{-1} \boldsymbol{K}_t^\top \\ \boldsymbol{C} &= \boldsymbol{Q}_t &&\Longrightarrow \boldsymbol{C} = \boldsymbol{Q}_t \end{aligned} \tag{14}$$

As a result, our TrajGPT can be viewed as a discretized continuous SSM, which is represented as an ODE. Note that $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ are data-dependent with respect to the t-th observation $\boldsymbol{X}_t$. Therefore, we can interpret it as a Neural ODE as follows:

$$\frac{d\boldsymbol{S}(t)}{dt} = \boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{B}^T \boldsymbol{V}(t) = f(\boldsymbol{S}(t), t, \theta) \tag{15}$$

Consequently, our TrajGPT serves as a discretized neural ODE. In the multi-head attention proposed in Section. **??**, each head of TrajGPT functions as its own neural ODE.

C Proof of SSM Discretization via ZOH rule

To discretize the continuous model SSM, it has to compute the cumulative updates of the state $\boldsymbol{S}(t)$ over a discrete step size. For the continuous ODE in Eq. 10, we have a continuous-time integral as follows:

$$\begin{aligned} &\boldsymbol{S}'(t) = \boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{B}\boldsymbol{X}(t) \\ &\boldsymbol{S}(t+1) = \boldsymbol{S}(t) + \int_t^{t+1} \left(\boldsymbol{A}\boldsymbol{S}(\tau) + \boldsymbol{B}\boldsymbol{X}(\tau)\right) d\tau \end{aligned} \tag{16}$$

In the discrete-time system, we need to rewrite the integral as we cannot obtain all values of $\boldsymbol{X}(\tau)$ over a continuous interval $t \to t+1$:

$$\boldsymbol{S}(t+1) = \boldsymbol{S}(t) + \sum_t^{t+1} (\boldsymbol{A}\boldsymbol{S}(\tau) + \boldsymbol{B}\boldsymbol{X}(\tau)\Delta\tau \tag{17}$$

We replace $\boldsymbol{X}(t)$ in the time derivative $\boldsymbol{S}'(t)$ as follows:

$$\begin{aligned} &\boldsymbol{S}'(t) = \boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{B}\boldsymbol{X}(t) \\ &\boldsymbol{S}'(t) - \boldsymbol{A}\boldsymbol{S}(t) = \boldsymbol{B}\boldsymbol{X}(t) \\ &e^{-\boldsymbol{A}t}\boldsymbol{S}'(t) - e^{-\boldsymbol{A}t}\boldsymbol{A}\boldsymbol{S}(t) = e^{-\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{X}(t) \\ &\frac{d}{dt}\left(e^{-\boldsymbol{A}t}\boldsymbol{S}(t)\right) = e^{-\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{X}(t) \\ &e^{-\boldsymbol{A}t}\boldsymbol{S}(t) = \boldsymbol{S}(0) + \int_0^t e^{-\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{X}(\tau)d\tau \\ &\boldsymbol{S}(t) = e^{\boldsymbol{A}t}\boldsymbol{S}(0) + \int_0^t e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{X}(\tau)d\tau \end{aligned} \tag{18}$$

By introducing a discrete step size $\Delta = t_{k+1} - t_k$, we transform the above equation to a discrete-time system as follows.

$$\boldsymbol{S}(t_{k+1}) = e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{S}(t_k) + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-\tau)}\boldsymbol{B}\boldsymbol{X}(\tau)d\tau$$

$$\boldsymbol{S}(t_{k+1}) = e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{S}(t_k) + \left(\int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-\tau)}d\tau\right)\boldsymbol{B}\boldsymbol{X}(t_k) \text{ (assuming } x(\tau) \approx x(t_k) \text{ over the interval)}$$

$$\boldsymbol{S}(t_{k+1}) = e^{\Delta\boldsymbol{A}}\boldsymbol{S}(t_k) + \boldsymbol{B}\boldsymbol{X}(t_k)\int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-\tau)}d\tau$$

$$\boldsymbol{S}(t_{k+1}) = e^{\Delta\boldsymbol{A}}\boldsymbol{S}(t_k) + \boldsymbol{B}\boldsymbol{X}(t_k)\int_{0}^{\Delta} e^{\boldsymbol{A}\tau'}d\tau' \text{ (letting } \tau' = t_{k+1} - \tau)$$

$$\boldsymbol{S}(t_{k+1}) = e^{\Delta\boldsymbol{A}}\boldsymbol{S}(t_k) + \boldsymbol{B}\boldsymbol{X}(t_k)\int_{0}^{\Delta} e^{\boldsymbol{A}\tau}d\tau$$

$$\boldsymbol{S}(t_{k+1}) = e^{\Delta\boldsymbol{A}}\boldsymbol{S}(t_k) + \boldsymbol{B}\boldsymbol{X}(t_k)\left(e^{\Delta\boldsymbol{A}} - \boldsymbol{I}\right)\boldsymbol{A}^{-1} \text{ (integral of matrix exponential function)}$$

$$\boldsymbol{S}_{k+1} = \boldsymbol{A}\boldsymbol{S}_k + \boldsymbol{B}\boldsymbol{X}_k \tag{19}$$

where the discretized state matrices $\bar{\boldsymbol{A}} = e^{\Delta\boldsymbol{A}}$ and $\bar{\boldsymbol{B}} = (e^{\Delta\boldsymbol{A}} - \boldsymbol{I})\boldsymbol{A}^{-1}\boldsymbol{B}$. Note that we apply the ZOH approach considering that $x(\tau)$ is constant between t_k and t_{k+1}, we can rewrite the Eq. 19 by assuming $\boldsymbol{X}(\tau) \approx \boldsymbol{X}(t_k + 1)$:

$$\boldsymbol{S}(t_{k+1}) = e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{S}(t_k) + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-\tau)}\boldsymbol{B}\boldsymbol{X}(\tau)d\tau$$

$$\boldsymbol{S}(t_{k+1}) = e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{S}(t_k) + \left(\int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-\tau)}d\tau\right)\boldsymbol{B}\boldsymbol{X}(t_{k+1})$$

$$\boldsymbol{S}_{k+1} = \bar{\boldsymbol{A}}\boldsymbol{S}_k + \bar{\boldsymbol{B}}\boldsymbol{X}_{k+1} \tag{20}$$

The resulting equation is the discrete SSM using ZOH discretization in eq. 11.

Derivation of $\bar{\boldsymbol{B}}$. We use the equation $e^{\boldsymbol{A}\tau} = I + \boldsymbol{A}\tau + \frac{1}{2!}\boldsymbol{A}^2\tau^2 + \cdots$, we have this integral of exponential function of $\boldsymbol{A}$:

$$\begin{aligned}
\bar{\boldsymbol{B}} &= \int_0^{\Delta} e^{\boldsymbol{A}\tau}\boldsymbol{B}d\tau \\
&= \int_0^{\Delta}\left(\boldsymbol{I} + \boldsymbol{A}\tau + \frac{1}{2!}\boldsymbol{A}^2\tau^2 + \cdots\right)d\tau\boldsymbol{B} \\
&= \left(\boldsymbol{I}\Delta + \frac{1}{2}\boldsymbol{A}\Delta^2 + \frac{1}{3!}\boldsymbol{A}^2\Delta^3 + \cdots\right)\boldsymbol{B} \\
&= \left(e^{\Delta\boldsymbol{A}} - \boldsymbol{I}\right)\boldsymbol{A}^{-1}\boldsymbol{B}
\end{aligned} \tag{21}$$

D Details of Experiment

D.1 Pre-training and fine-tuning

During pre-training, TrajGPT utilizes a next-token prediction task to learn general temporal representations from unlabeled data [12]. Given a sequence with a [SOS] token, TrajGPT predicts the subsequent tokens by shifting the sequence to the right. At the last layer, each token's output representation is fed into a linear layer for next-token prediction. The pre-training loss is cross-entropy for discrete diagnosis codes.

Among other Transformer baselines, PatchTST adopted a masking-based approach, masking 40% of its patches as zeros [11]. For the Transformer models without established pre-training methods, we used a masking-based method by randomly masking 40% of timesteps [24].For downstream forecasting tasks, we employ end-to-end fine-tuning on the entire model. The final linear layer is utilized for making the forecasts. All Transformer models performed 20 epochs of pre-training with cross-entropy loss, followed by 5 epochs of end-to-end fine-tuning.

Table 2: Configurations of TrajGPT and other transformer baselines on the PopHR dataset.

Data Size (timesteps)	54.9M
Model Parameters	7.5M
TrajGPT	
Decoder Layers	8
Heads	4
Dim ($\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, FF)	200,200,400,400
Transformer baselines including Encoder-decoder and Encoder-only models	
Enc-Dec Layers	4 & 4
Encoder Layers	8
Decoder Layers	8
Heads	4
Dim ($\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, FF)	200,200,200,400

D.2 Results

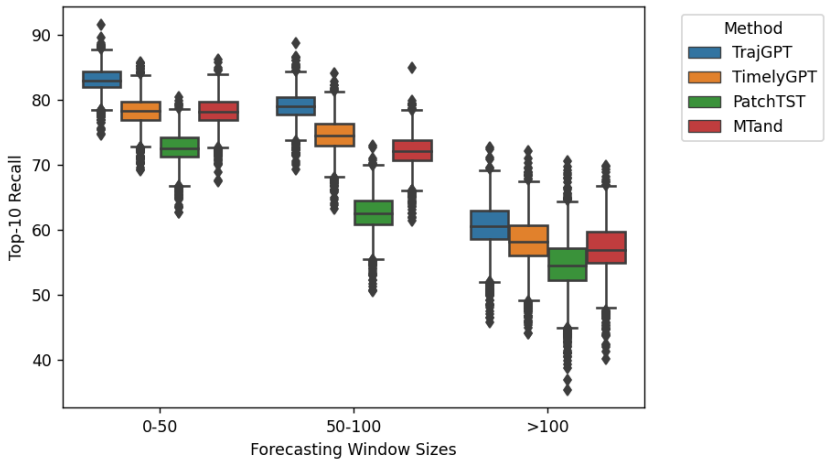

Figure 3: Top-10 recall rates for TrajGPT and baseline methods across three forecasting windows.