Factorized Deep Generative Models for End-to-End Trajectory Generation with Spatiotemporal Validity Constraints

Liming Zhang
lzhang22@gmu.edu
George Mason University
Fairfax, Virginia, USA

Liang Zhao
liang.zhao@emory.edu
Emory University
Atlanta, Georgia, USA

Dieter Pfoser
dpfoser@gmu.edu
George Mason University
Fairfax, Virginia, USA

ABSTRACT

A growing number of research areas such as location-based social networks, intelligent transportation systems, and urban computing utilize large amounts of trajectory data for benchmarking data management approaches and analysis methods. Given the general lackness of available large datasets, realistic synthetic trajectory datasets become important. This work proposes deep generative models for trajectory data that can learn disentangled models for sophisticated latent patterns. Existing methods rely on predefined heuristics and cannot learn the unknown underlying generative mechanisms. The proposed novel deep generative VAE-like models factorize global and local semantics (habits vs. random routing change). We further develop new inference strategies based on variational inference and constrained optimization to encapsulate spatiotemporal validity. New deep neural network architectures are developed to implement generative and inference models with dynamic latent priors. The proposed methods represent significant quantitative and qualitative improvements over existing approaches as demonstrated by extensive experiments. The software is made publicly available [1].

CCS CONCEPTS

• **Information systems** → **Data mining**; **Location based services**.

KEYWORDS

end-to-end trajectory generation, deep generative models, spatiotemporal-validity constraint, variational autoencoders

ACM Reference Format:
Liming Zhang, Liang Zhao, and Dieter Pfoser. 2022. Factorized Deep Generative Models for End-to-End Trajectory Generation with Spatiotemporal Validity Constraints. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3557915.3560994

[1] https://github.com/tongjiyiming/TrajGen

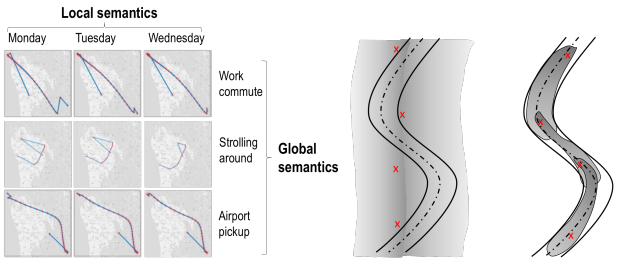
SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11.
https://doi.org/10.1145/3557915.3560994

1 INTRODUCTION

Advances in positioning technology have led to an unprecedented increase in available moving objects data such as GPS traces from vehicles or tourist check-in data from Location-based services. In each case, a series of temporally ordered and interpolated object locations can be represented as a spatiotemporal trajectory. Mining such trajectory data is important to a broad range of applications including location-based social networks, intelligent transportation systems, and urban computing [35]. Trajectory data mining involves three important tasks: 1) trajectory prediction, i.e., predicting the future locations based on past locations [18, 32], 2) trajectory representation learning, i.e., encoding trajectory data in (low-dimensional) vector space [6, 7, 20], and 3) trajectory generation [2, 3, 17, 25, 27, 28, 35]. Such approaches model the underlying distribution and mechanism of the trajectory generative process, which is crucial for tasks such as mobility simulation [3, 25], individual mobility privacy preservation [1], and data augmentation for prediction tasks [8, 41]. For example, traffic simulations for autonomous vehicles could use a generative model to generate high-fidelity temporal traffic conditions [3]. It could also be used to address privacy concerns when used to release trajectory datasets to the public [1]. The original data is replaced with generated data that follows the statistics and patterns of the original data. For data augmentation, Web-scale companies like Uber could use synthetic trajectories to obtain more data to train a large model linking users and trajectories [8]. Many relevant application contexts consider that trajectories have been quite difficult to generate due to sophisticated patterns and unknown mechanisms that are hard for traditional hand-crafted generation rules to handle. Today, thanks to the availability of large number of trajectory datasets, expressive deep generative models provide a promising way to learn the generation rules in a more end-to-end fashion. Trajectory generation approaches [3, 25, 27, 28, 35] have focused on conventional *rule-based trajectory generation* that is limited to prescribed rules and predefined distributions [21, 24, 25]. Such methods are tailored towards predefined principles and properties of movement. However, in most of the cases, the mechanism underlying movement and thus trajectory generation are too complex to be fully known a priori, or to be explicitly modeled. To address these issues, an emerging research topic is *end-to-end trajectory generation*, which extends deep generative models to trajectory data, and towards expressive generative models that are able to learn sophisticated distributions from collected trajectory data in a data-driven end-to-end fashion.

Although end-to-end strategies, such as deep learning methods have been widely used for trajectory representation learning and prediction, end-to-end methods have not been well explored when it comes to trajectory generation [35]. Few works [2, 4, 14, 23, 31] are



(a) Trajectories (b) Without constraint (c) With constraint

Figure 1: Trajectory examples (a) for different days, multiple route choices (local semantics, e.g., a traffic accident blocking a road) exist for the same origin-destination pairs (global semantics, e.g., user activities); (b) evenly distributed probability density of unconstrained generative models (red crosses are trajectory points, gray zone is probability density envelop of a generative model); (c) probability density envelope (grey zone) for a specific road constraint.

based on slightly tweaked generative models from other domains without holistically considering the major challenges for trajectory data, including **Challenge 1: Difficulty in factorizing global movement semantics from local movement semantics.** The global semantics refer to the general purpose of a trajectory, such as "Work commute", "Strolling around", or "Airport pickup", as shown in each of the three rows of Fig. 1 (a). Once the global semantics are determined, local semantics specify the detailed movement to achieve the overall purpose, such as which specific road and route to choose for "Work commute" or "airport pickup" (cf. Fig. 1 (a)). Factorizing global against local semantics is important and necessary for characterizing a trajectory in a holistic way, something existing works fall short in disentangling or modeling. **Challenge 2: Lacking an expressive prior to handle the interdependent noise of different parts of a trajectory.** Trajectories have strong interdependencies between location samples and hence priors that can characterize such interdependencies are imperative for generative models. **Challenge 3: Difficulty ensuring spatiotemporal validity of generated trajectories.** A generated trajectory is reasonable only if it satisfies necessary geometrical, physical, and social principles. Deep models are good at mimicking patterns from inputs; however, they cannot strictly avoid irrational cases that violate those principles since they assume a *continuous* distribution. For example in Fig. 1 (b), the learned smooth distributions of four positions covered the area outside of roads, although the probabilities of sampling a point away from a road are very low. Expected distributions should be limited to the geometry of the road, as shown in Fig. 1 (c) such that the probability of a location sample outside the road geometry is zero. All trajectory points (for cars) should be constrained by road geometries and the movement speed should be within a reasonable range. Therefore, although difficult, it is important to decrease the probabilistic density of invalid patterns.

To address the above issues, we propose new factorized deep generative models for trajectory generation with spatiotemporal validity constraints, namely our "End-to-End Trajectory Generation with spatiotemporal validity constraints" (EETG) framework. Through factorized latent sequential deep generative models, we can disentangle global from local semantics, while learning the representation of trajectories in an end-to-end manner. Newly-generalized dependent priors for latent sequential variables are proposed, which is in contrast to conventional independent priors in sequential models. With a novel constrained optimization solution, this reduces the probability of generating irrational and invalid vehicle trajectories. Extensive experiments including quantitative ablation studies and case studies will show the quantitative and qualitative effectiveness of the factorization latent structure, newly-generalized dependent priors, and the constrained optimization approach.

The outline of the remainder of this paper is as follows. Section 2 discusses related work. Section 3.1 gives the basic formulation of the problem. Section 3 presents our generative methods for trajectory data. The experimental evaluation of Section 4 shows the advantage of our EETG approach over existing methods. Finally, Section 5 concludes and presents directions for future work.

2 RELATED WORK

2.1 Rule-based trajectory generation/synthesis

This research domain has a long history (cf. [35]) and representative methods include GSTD [27, 34] using predefined speed, agility, direction, and clustered behavior, Oporto [30] relying on agent-based movement estimation, plausible synthesis [1] with mined semantic location similarity, and Hermoupolis [25] relying on urban points of interests. Recent approaches to mobility simulations include SUMO [21] and Ditras [24]. All of these approaches are designed with a specific purpose in mind such as spatial-temporal indexing, and inherit more or less the rule-based simulation paradigms observed in the physical world. Such rule-based simulations can become quite involved like in the case of Hermoupolis [25]. All these models are not end-to-end models, i.e., trajectory data-driven, which is what our approach strives for. The typical workflow involves generating user origin-destination pairs, or the so-called "scenarios" in SUMO in transportation planning [21]. A second step synthesizes route samples on a road network. Such approaches are hard to emulate since they use specific city/infrastructure features and require special domain knowledge.

We would like to emphasize that even though some of these efforts also take advantage of machine learning models like Hidden Markov Model [1, 24], LSTM [32], or recent spatiotemporal point process [37], deep models in each case are just a subpart of a whole pipeline. Such recently-developed models fall into the paradigm of rule-based models by pipelining different tasks, which is different from end-to-end trajectory generation discussed in this paper.

2.2 End-to-end trajectory generation/synthesis

The current emerging trend for trajectory generation is to use deep generative models in a data-driven end-to-end fashion. Deep generative models for trajectory generation are not widely explored until now as indicated by a recent survey paper in trajectory data mining [35]. As a new research domain, existing work in end-to-end trajectory generation used plain or slightly-tweaked generative models from universally adapted models. One type of work converts trajectories to images first and applies image-based GANs for generation tasks [23, 31]. Such an approach loses information including time, speed, and direction. Other efforts [2, 14] utilize a variational autoencoder [16] to generate a trajectory via a single

sequence-level variable based on a Gaussian distribution, which cannot jointly encode spatiotemporal-variant and -invariant information (traffic jams vs. work commute). Other generative models [20, 31] are used for anomaly detection with a slightly tweaked sequence-level variable that is more expressive, such as Mixture of Gaussian [20] or Infinite Gaussian Mixture [31]. This kind of randomness mixes global and local patterns into simplified black-box models without sufficient semantic disentanglement. Lastly, some approaches [20, 23] are strictly not end-to-end approaches because they map real-valued coordinates to grid cells that cannot be recovered to real-valued coordinates. Overall, deep generative models that can comprehensively take care of static and dynamic patterns in trajectories while ensuring spatiotemporal validity in continuous space are so far missing.

2.3 Spatiotemporal constraints

Studies of trajectories consider spatiotemporal validity constraints, such as vehicular motion [3], turn restrictions [33], and speed and heading rate constraints [29]. Many of these approaches adopt first-principled physics rules of moving objects, and utilize agent-based models to perform a simulation that tries to mimic the physical world. Deep learning models capture only statistical patterns, so such constraints are not trivial to be integrated in deep generative models. This presents a major challenge to the generation of realistic trajectories using neural networks.

2.4 Disentangled and factorized models

Disentangled deep generative models are a promising approach for many domains. The notion of disentanglement and factorization is to separate out the underlying disentangled factors responsible for variations of the data. The generative representations learned in this way are relatively resilient to existing complex variants [12]. They can be used to enhance interpretability, generalizability, and explainability. Additional inductive biases could be considered by using particular data properties, such as factorizing graph data into node and edge patterns [10, 11, 36], dynamic graph features [39], and factorizing video data into object and motion patterns [19]. Although such models show promising outputs for graphs, images and video data, they lack customized designs for end-to-end trajectory generation.

3 END-TO-END TRAJECTORY GENERATION

This section introduces our "End-to-End Trajectory Generation with spatiotemporal validity constraints" (EETG) framework, which addresses the aforementioned three challenges through each section of details: (1) the general problem formulation for end-to-end trajectory generation in Section 3.1, (2) the overall design of EETG through a novel Bayesian generative process of trajectories with validity constraints as introduced (Section 3.2), (3) a VAE-based formula for latent variables of the generative process and detailed inferences address Challenge 1 by learning global and local semantics in a factorized manner (Section 3.2), (4) a deep encoder and decoder architecture (Section 3.4) provides comprehensive and special neural network layers to handle stochastic latent state modeling and solves Challenge 2 involving lacking dynamic expressive priors, and (5) spatiotemporal validity constraints are transformed into a learnable reformulation that could use a gradient descent technique together with original trajectory generation objectives (cf. Section 3.5) to address Challenge 3.

3.1 Problem formulation

What follows is a description of the essential concepts and a definition of the problem addressed in this work.

Definition 3.1 (Trajectory)**.** A trajectory $s_{1:T}$ is defined as a length T sequence of spatiotemporal points $\{s_1, s_2, \cdots, s_T\}$ at time steps $1, 2, \cdots, T$, where $s_t = (x_t, y_t)$ is a point at time t with a continuous coordinate x_t, y_t in 2D space, and the time interval in between GPS samples is ϵ seconds with a sampling rate $\frac{1}{\epsilon}$. A trajectory dataset is a set of trajectories $\mathbb{S}$ with $s_{1:T} \in \mathbb{S}$.

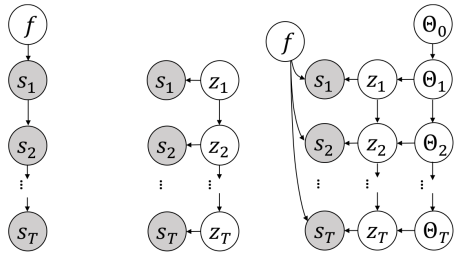
Definition 3.2 (End-to-End Trajectory Generation)**.** The problem of end-to-end trajectory generation is to generate a synthetic trajectory $\hat{s}_{1:T}$ typically using deep learning models based on the underlying distribution $\hat{s}_{1:T} \sim p(\mathbb{S})$ from the real trajectories $\mathbb{S}$, where p represents a deep model.

3.2 Overall design of the generative process

Existing works in the trajectory domain consider generative processes to include either a single global variable [2, 14, 20, 31] (shown in Fig. 2 (a)), while other domains, like video and speech, with sequential data consider additional latent variables for time stamps [5, 13], which we base our novel generative process on (shown in Fig. 2 (b)). Therefore, to address the unique challenges mentioned above, the proposed generative method (shown in Fig. 2 (c)) focuses on a novel generative process that factorizes the semantics of a trajectory into three aspects: (1) global semantics f, (2) local semantics z_t related to a specific location and time. Global semantics cover the reasons for a trip, e.g., commuting, a stroll downtown, airport pickup, and local semantics capture spatiotemporal autoregressive patterns and guide how the next location sample and point in a trajectory is dependent on the previous one, and (3) dynamic priors Θ_t introduce a more reasonable inductive bias and expressiveness by considering the dependencies of close-by trajectory points and removing the i.i.d. assumption between $\{z_1, \cdots, z_T\}$. Before introducing the details of model inference (Section 3.3) and architecture for each component (Section 3.4), we summarize the complete generative process, shown in Fig. 2 (c) as follows:

- Draw a sequence of priors $\{\Theta_1, \cdots, \Theta_T\}$ for time-step random noise Θ_0, based on conditional probability: $\Theta_t \sim p(\Theta_t|\Theta_{t-1})$, where $p(\Theta_0)$ is a predefined distribution such as a unit Gaussian.
- For each trajectory, draw a variable f as the general semantic from $p_\theta(f)$ such as a unit Gaussian;
- For each trajectory, draw the time-step random variable z_1 for the first time point $t = 1$ from Θ_1.
  - For each time point $t \geq 1$, draw the underlying time-step random variable z_t with the conditional probability $z_t \sim p(z_t|z_{t-1}, \Theta_t)$.
  - For each t, draw the observed variable $s_t \sim p(s_t|z_t, f)$, $\forall s_t \notin C$, constrained by the violation function in C.

We want to point out that given the case of Θ_0 being generated from f instead of a unit Gaussian and chain-randomness removed, our model reverts to a baseline SVAE model (cf. [14]) and z_t, Θ_t would become internal parameters and states that have are insignificant. Section 4 provides the respective ablation studies to illustrate the use of dynamic factors $z_{1:T}$ with its priors $\Theta_{0:T}$ (more details of model evolution are in Supplement A.1).



(a) Sequential variational model (b) State space model (c) Proposed EETG model

Figure 2: Plate notations of the evolution of existing and proposed deep generative models : (a) Sequential variational model [14] with one latent variable f for the entire sequence. This model is widely adopted by existing works [14, 20, 23, 31]; (b) State space model [5] capturing local semantics in sequential random variables $z_{1:T}$. This model has not been used by trajectory generation, but by other sequential data domains; (c) (Proposed) EETG model extending existing works and additionally uses dynamic priors $\Theta_{0:T}$ for better dynamic induction of z_t.

3.3 VAE-like model inference

Since the proposed generative model is intractable to infer, we propose to solve it based on variational inference used to train variational autoencoders. We initially establish a posterior $q_\phi(z_{1:T}, f|s_{1:T})$ to approximate the original distribution $p(z_{1:T}, f|s_{1:T})$. Two possible choices of q_ϕ are

$$q_\phi(f, z_{1:T}|s_{1:T}) = \begin{cases} q_\phi(f|s_{1:T})q_\phi(z_{1:T}|s_{1:T}) & (\textit{factorized}) \\ q_\phi(f|s_{1:T})q_\phi(z_{1:T}|f, s_{1:T}) & (\textit{full}) \end{cases}$$

$$z_{1:T} \sim \Theta_{0:T} \tag{1}$$

where the level of variances of $z_{1:T}$ could change depending on f in the *full* model. For example, if most roads between home and work are highways, then there is almost no variance when it comes to route choice, while the level of noise of $z_{1:T}$ in the *factorized* model does not depend on f. Such a model could reflect the different road networks in cities. Following β-VAE [12], the objective is developed as follows:

$$\min_{\psi,\phi} \mathcal{L}(p_\psi, q_\phi) = -\mathbb{E}_{q_\phi}[\log p_\psi(s_{1:T}|z_{1:T}, f, \Theta_{0:T})]$$
$$+\beta\, KL(q_\phi(z_{1:T}, f, \Theta_{0:T}|s_{1:T})||p_\psi(z_{1:T}, f, \Theta_{0:T})) \tag{2}$$

where β is a hyper-parameter to control disentanglement in β-VAE, KL is short for Kullback–Leibler divergence [12], ψ and ϕ are sets of neural network parameters. $z_{1:T}, f, \Theta_{0:T}$ are multi-variant Gaussians with their own distribution parameters of means and variances, $(\mu_{z_t}, \sigma_{z_t})$, (μ_f, σ_f), $(\mu_{\Theta_t}, \sigma_{\Theta_t})$, etc. The first term in Eq. 2 is typically used for minimizing the reconstruction loss as follows:

$$-E_{q_\phi(z_{1:T}, f|s_{1:T})}[log(p_\theta(s_{1:T}|z_{1:T}, f))] = \sum_{t=1}^{T} ||s_t - \widehat{s_t}||_2 \tag{3}$$

where $||\cdot||_2$ calculates the $L2$ norm. We can use Monte Carlo sampling to obtain $\widehat{s_t}$. The other term in the second line of Eq. 2 helps regularize the learned posterior close to the prior distributions. More specifically, the term can be expanded as:

$$\begin{aligned} &KL(q_\phi(z_{1:T}, f, \Theta_{0:T}|s_{1:T})||p_\psi(z_{1:T}, f, \Theta_{0:T})) \\ &=KL(q_\phi(f|s_{1:T})q_\phi(z_{1:T}|f, s_{1:T})||p_\psi(\Theta_{0:T})p_\psi(z_{1:T}, f, \Theta_{0:T})) \\ &=KL((q_\phi(f|s_{1:T})||\mathcal{N}(\mathbf{0}, \boldsymbol{I}))) \\ &+KL((q_\phi(z_{1:T}|f, s_{1:T})||\prod_{t=1}^{T}\mathcal{N}(\mu_{\Theta_t}(\Theta_{<t}, \psi), \sigma_{\Theta_t}(\Theta_{<t}, \psi))\mathcal{N}(\mu_{\Theta_0}, \sigma_{\Theta_0})) \\ &= -\frac{1}{2N}(1 + log(\sigma_f) - \mu_f^2 - \sigma_f) \\ &-\frac{1}{2N}\sum_{t=1}^{T}(1 - log(\sigma_{\Theta_t}) + log(\sigma_{z_t}) - \sigma_{z_t} + \frac{(\mu_{z_t} - \mu_{\Theta_t})^2}{log(\sigma_{\Theta_t})}) \end{aligned} \tag{4}$$

where $\mathcal{N}(\mu_{\Theta_0}, \sigma_{\Theta_0})$ is from the prior $p_\psi(\Theta_0)$ and follows an unit Gaussian distribution. The prior sequence generator $q_\phi(\Theta_{0:T})$ is an independent network without inputs from a real trajectory $s_{1:T}$, μ_{Θ_t} is a function that outputs the mean value of Θ_t, and σ_{Θ_t} is a function that outputs the variance value of Θ_t. In the last line, to simplify the notation, μ_{Θ_t} and σ_{Θ_t} are the actual mean and variance, while μ_{Θ_t} and σ_{Θ_t} in the $KL(||)$ represent a function.

3.4 Deep encoders and decoders architecture

We are ready to introduce the detailed architecture for our proposed EETG framework. For a trajectory $s_{1:T}$, its abstracted operations are shown in Fig. 3. They follow a VAE-like design with encoders and decoders. Corresponding to Eq. 4, our encoder $q_\phi(z_{1:T}, f, \Theta_{0:T}|s_{1:T})$ is decomposed into three sub-parts that are explained as follows:

1) *Global-semantics encoder* $q(f|s_{1:T})$. It consumes the sequence that captures the stochastic whole-sequence representation f detailed in the upper left corner of Fig. 3. Specifically, the "Coordinate encoding" module is denoted as $MLP_s(\cdot)$, which is a multi-layer perceptron that converts two-dimensional points (x_t, y_t) into a high-dimensional representation e_t. Then, a Bidirectional LSTM [9] denoted as $BiLSTM_f$ is used to encode all representations e_t with forward and backward information for a time step. The first and last cells' output vectors o_1 and o_T are concatenated and encoded as the mean μ_f and variance σ_f of variable f.

2) *Local-semantics encoder* with factorized modeling alternative $q(z_{1:T}|s_{1:T})$ and full modeling alternative $q(z_{1:T}|f, s_{1:T})$ takes each coordinate representation to step-wise generate a stochastic posterior representation z_t (detailed in the lower left corner of Fig. 3). It utilizes another Bidirectional-LSTM to capture the bi-directional information flow. It is followed by an RNN [16] to encode each time-step variable z_t. In the deep architecture of EETG shown in Fig. 3, the factorized alternatives can be obtained by removing the dashed blue lines starting from f and the $Concat()$ operations.

3) *Prior generator* takes an initial noise Θ_0 and generates prior Θ_t step by step, utilizing a variational recurrent structure of a VRNN [5] to handle dynamic sequential randomness (shown in the lower right corner of Fig. 3). Corresponding to Eq. 4, it models the dynamic priors as $\mathcal{N}(\mu_{\Theta_t}(\Theta_{<t}, \psi), \sigma_{\Theta_t}(\Theta_{<t}, \psi))))$, which means that Θ_{t+1} is generated conditioned on Θ_t instead of a i.i.d. sampling.

4) The *Joint-factor decoder* $p_\psi(s_{1:T}|z_{1:T}, f)$ combines during the training phase sampled f and z_t, to stochastically generate coordinates s_t (shown in the right part of Fig. 3), and minimizes our training loss. It first concatenates f and z_t to a new vector c_t as the high-dimensional representation of spatial information. Then, another "coordinate decoding" module is used to transform c_t to two-dimensional map coordinates. In the following, we use $MLP_*(\cdot)$ to

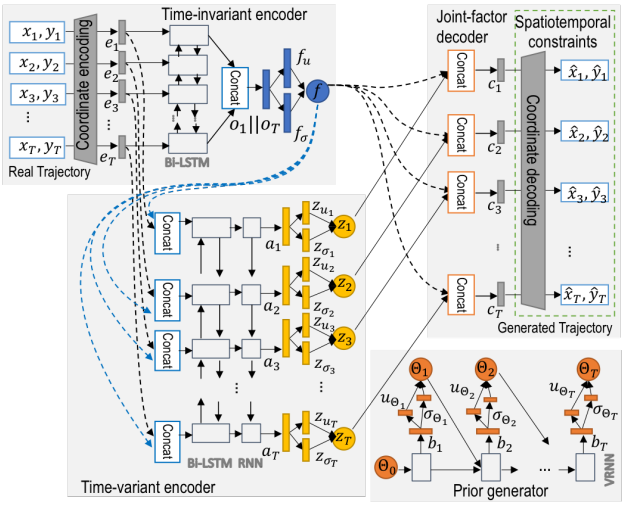


Figure 3: Deep architecture of EETG. It shows the complete architecture for DSVAE in our ablation study. Removing blue dashed lines and $Concat()$ operations becomes FDSVAE alternative.

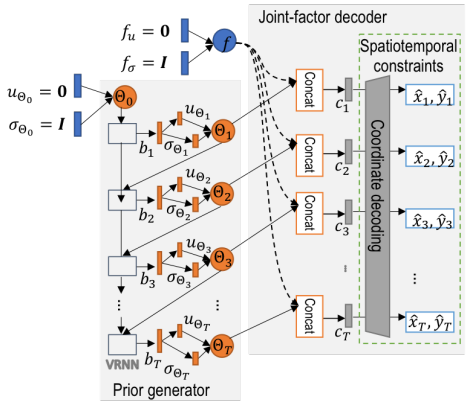


Figure 4: Generator for synthesis: a dynamic sequential generator to sample from sequential meta-priors replacing $z_{1:T}$ with its priors $\Theta_{0:T}$ with a recurrent dependence.

denote different multi-layer perceptrons for "coordinate decoding".

All operations can be summarized as follows:

Global-semantic encoder:

$$e_t = MLP_s(x_t, y_t); o_{t+1} = BiLSTM_f(e_t, o_t)$$
$$o = o_1||o_T, \mu_f = MLP_{\mu_f}(o),\ \sigma_f = MLP_{\sigma_f}(o),\ f \sim \mathcal{N}(\mu_f, \sigma_f)$$

Local-semantic encoder:

$$\begin{cases} factorized: & a_{t+1} = BiLSTM_z(e_t, a_t) \\ full: & a_{t+1} = BiLSTM_z(e_t||f, a_t) \end{cases}$$
$$a_{t+1} = RNN_z(a_t)$$
$$\mu_{z_t} = MLP_{\mu_{z_t}}(a_t), \sigma_{z_t} = MLP_{\sigma_{z_t}}(a_t), \quad z_t \sim \mathcal{N}(\mu_{z_t}, \sigma_{z_t})$$

Prior generator:

$$\Theta_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$
$$b_{t+1} = VRNN_\Theta(\Theta_t, b_t)$$
$$\mu_{\Theta_t} = MLP_{\mu_\Theta}(b_t), \sigma_{\Theta_t} = MLP_{\sigma_\Theta}(b_t), \Theta_{t+1} \sim \mathcal{N}(\mu_{\Theta_{t+1}}, \sigma_{\Theta_{t+1}})$$

Joint-factor decoder:

$$\widehat{x}_t; \widehat{y}_t = MLP_s(c_t), c_t = f||z_t$$

where $||$ is the concatenation operation of vectors, $\sim$ is the sampling operation, which uses the re-parameterization trick [16] to allow gradient back-propagation. The other output vectors of the LSTM models are omitted for simplicity reasons in all operations and in Figures 3 and 4. o_*, a_*, b_* are outputs for either the $BiLSTM_*$, RNN_*, or $VRNN_*$ modules, respectively. Omitting parts of the architecture indicated by dashed blue lines, the approach reverts to a *factorized* encoder alternative. Including those connections means introducing a concatenation operation for $e_t||f$ and we have a *full* alternative encoder. The $VRNN_*$ module is a unique recurrent network. Its input for $t+1$ cell includes a randomly sampled value Θ_t, instead of a deterministic state. For step $t+1$, it uses an LSTM cell to take previous Θ_t and hidden state m_{Θ_t} as inputs and outputs b_t. Two MLP_* modules are used to transform the b_t into a mean $\mu_{\Theta t+1}$ and variance $\sigma_{\Theta_{t+1}}$ of Θ_{t+1}. Θ_{t+1} is sampled from a Gaussian distribution $\mathcal{N}(\mu_{\Theta_{t+1}}, \sigma_{\Theta_{t+1}})$. Θ_0 is sampled from an unit Gaussian noise. **Training phase:** Utilizes the objective function of Eq. 2 and performs a gradient back-propagation with selected optimizer.

Synthesis phase: With the dynamic expressive priors generator and factorized semantics, we have more control over trajectory generation. For example, we can control the global semantic f to be fixed and study how the local semantics z_t changes and vice versa. The other uniqueness is that because we propose a dynamic chain of priors, the synthesis phase in Fig. 4 is slightly different from a regular VAE model and needs special treatment. In the case of regular VAEs, samples from the prior distribution replace posterior distributions for the decoder module to generate synthetic data [16]. The prior distribution of latent variable f is a unit Gaussian noise $\mathcal{N}(\mathbf{0}, \mathbf{1})$ as is also the case for regular VAEs. However, the $z_{1:T}$ variables are not i.i.d. distributions and could not use $\mathcal{N}(\mathbf{0}, \mathbf{1})$ to sample noise over each time step. This involves two steps: (1) an initial noise for Θ_0 is sampled as $\mathcal{N}(\mathbf{0}, \mathbf{1})$; (2) the trained Prior generator module uses Θ_0 to generate the chain priors $\Theta_{1:T}$, and the generated $\Theta_{1:T}$ are used as inputs to replace $z_{1:T}$. The operations can be summarized as follows:

$$f \sim \mathcal{N}(\mathbf{0}, \mathbf{1}), \Theta_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$
$$b_{t+1} = VRNN_\Theta(\Theta_t, b_t)$$
$$\mu_{\Theta_{t+1}} = MLP_{\mu_\Theta}(b_t), \sigma_{\Theta_{t+1}} = MLP_{\sigma_\Theta}(b_t), \Theta_t \sim \mathcal{N}(\mu_{\Theta_t}, \sigma_{\Theta_t})$$
$$\widehat{x}_t; \widehat{y}_t = MLP_s(c_t), c_t = f||\Theta_t$$

3.5 Spatiotemporal-validity constraints

Although the generative model learned by Eq. 2 could effectively characterize the underlying process of trajectory generation, the trajectories sampled from the learned generative model may not resemble real-world trajectories, i.e., obey motion physics. Embedding such an inductive bias can effectively increase the model generalizability and strengthen the robustness w.r.t. noise as introduced by measurement and sampling errors (cf. [26]). The central contribution is imposing spatiotemporal validity constrains in optimizing the loss function $\mathcal{L}$ in Eq. 2 as follows:

$$\min_{\psi,\phi} \mathcal{L}(p_\psi, q_\phi),\ s.t. \forall s_{1:T} \notin C : p_\psi(s_{1:T}|z_{1:T}, f, \Theta_{0:T}) = 0 \quad (5)$$

where C denotes the set of all trajectory patterns that satisfy the spatial validity constraints, which can be specified by the user based on the practical needs. For example, if the constraint states all trajectories must be on *roads*, then $C_1 = \{[x_1, \cdots, x_T] | x_t \in \mathcal{R}\}$, with $\mathcal{R}$ denoting roads as spatial regions. The constraint could also relate to phenomena such as the *speed limits*, meaning the trajectory's

moving speed must be achievable. This could be denoted as $C_2 = \{[x_1, \cdots, x_T] || \Delta x_t| \leq \mathcal{S}\}$, with $|\Delta x_t|$ denoting the object's speed at time t and $\mathcal{S}$ is the speed limit. Another pattern could be *turn angles* between two consecutive segments of a trajectory. In many situations, it is unlikely to have many consecutive sharp turns. To avoid this behavior, we could have $C_3 = \{[x_1, \cdots, x_T] | \sum_t cos(x_{t-1} - x_t, x_{t+1} - x_t) < \lambda\}$, where $cos(x_{t-1} - x_t, x_{t+1} - x_t)$ denotes the cosine similarity of the two vectors, each representing movement in two-dimensional Euclidean space.

Moreover, the spatial constraint C can also be composed of the logical combinations among multiple rules, such as $C = C_2 \bigcap (C_1 \bigcup C_3)$. For example, we can combine speed limits and turn angles with such a logical operation. Directly solving complex constrained problems using conventional ways, such as a Lagrangian has been demonstrated to be inefficient for deep neural networks. Here, we extend a recent deep constrained optimization framework [22] to address this by reformulating Eq. 5 as follows:

$$\tilde{\mathcal{L}}(p_\psi, q_\phi, \gamma) = \mathcal{L}(p_\psi, q_\phi) + \gamma \, [\int \mathbb{1}(g(z_{1:T}, f, \Theta_{0:T}) \notin C) \cdot p_\psi(z_{1:T}, f, \Theta_{0:T}) \, \mathrm{d}z_{1:T} \, \mathrm{d}f \, \mathrm{d}\Theta_{0:T} \,]^{\frac{1}{2}}$$

where C is the set of validity functions, and $\mathbb{1}(\cdot)$ is an indicator function that outputs 1 if a generated trajectory is invalid, otherwise it outputs 0. We can reduce the integral term using Monte Carlo Sampling in VAE [16]. To allow gradient flow over the regularization term, constraint functions in C must have gradients.

The above represents the complete learning objective of our EETG framework, while the implementation details are shown in Section 4.2.

4 EXPERIMENTS

To demonstrate the superior reconstruction and constraint performance of EETG, we compare it to competing and ablation methods using quantitative and qualitative results with three real world and one synthetic dataset. All experiments were conducted on a 64-bit i9 Intel computer using an NVIDIA 1080ti GPU.

4.1 Datasets

Reference datasets The first dataset was collected by 442 taxis in Porto, Portugal capturing a whole year (from 01/07/2013 to 30/06/2014) [2]. Position samples do not have timestamps, but the data was generated using a fixed 15s sampling interval. The second dataset is the T-Drive data consisting of 10,357 taxis collected during one week. This data includes timestamps [3]. Pre-processing steps are used to clean the data, including Noise Filtering and Stay Point Detection [35]. The third data set is Gowalla check-in data [4], of which we select the Dallas metropolitan area.

Synthetic dataset We generated a Points-of-Life (POL) dataset of 10,000 students living on a university campus using a geospatial agent-based simulation [15]. The agents mimic real-world contact and check-in patterns based on predefined living and social preference settings. All datasets use a respective projected coordinate system. We split the data using a 0.9/0.1 ratio into training and testing sets for evaluation purpose.

[2] www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

[3] www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

[4] snap.stanford.edu/data/loc-Gowalla.html

4.2 Constraints setting

4.2.1 Physics-induced constraints: Since the sampling time interval ϵ is small (15s), (1) the average speed constraint $\gamma_t = \frac{||s_t - s_{t-1}||_2}{\epsilon}$ is higher than a threshold $\bar{\gamma} = 60km/h$,(2) to avoid sharp turns, we cannot observe a preceding angle η_t (in cosine value) smaller than a threshold $\bar{\eta}$ and $\eta_t = \frac{(s_t - s_{t-1}) \cdot (s_{t-1} - s_{t-2})}{||s_t - s_{t-1}||_2 ||s_{t-1} - s_{t-2}||_2}$. This regularization imposes penalties only if the angle is smaller than $\eta_t > \bar{\eta}$ and the segment is larger than $\gamma_t > \bar{\gamma}$. We show such patterns in the Porto dataset in Fig. 5 (red dashed region in second row), which is formulated as follows:

$$\frac{\lambda}{N} \sum^{J} \sum_{t=2}^{T} c(\epsilon, s_{1:T}) = \frac{\lambda}{N} \sum^{J} \sum_{t=2}^{T} (\gamma_t - \bar{\gamma})_+ (\eta_t - \bar{\eta})_+$$

where $()_+$ is the Relu function and λ is a hyper-parameter. Notice that a total of $T - 2$ constraints for each trajectory are possible.

4.2.2 Behavior-induced constraints: They are derived from human behavior and do not necessarily violate basic physics principles. For example, when driving, one does not make two u-turns within 5s, in other words, two consecutive sharp angles (less than 30 degrees) are improbable. Such a constraint can be seen in the T-Drive dataset (red dashed regions in the first row) in Fig. 5. This regularization penalizes the case that an angle is smaller than a threshold $\eta_t > \bar{\eta}$ if its preceding angle is also sharp and violates the angle threshold $\eta_{t-1} > \bar{\eta}$. The formula is as follows:

$$\frac{\lambda}{N} \sum^{J} \sum_{t=3}^{T} c(s_{1:T}) = \frac{\lambda}{N} \sum^{J} \sum_{t=3}^{T} (\eta_t - \bar{\eta})_+ (\eta_t - \bar{\eta})_+$$

where η is the cosine value of angles. Notice that a total of $T - 3$ constraints for each trajectory are possible. Other potential constraints will be discussed in future work.

4.3 Competitor methods and ablation study

Rule-based: This model utilizes Ditras [24], which uses a Hidden Markov Model for origin-destination generations. The model is based on discretized grids. Road networks are used for generating GPS points.

LSTM: a plain LSTM model that can take any start point as input without any latent variable for stochastic modeling.

IGMM-GAN: an image-like GAN-based model [31], not a sequential model, using a single variable (cf. Fig. 2 (a)).

SVAE-f: uses a single latent variable and a VAE model [14]. It corresponds to the sequential variational model of Fig. 2 (a). This method is considered in the ablation study by using f without $z_{1:T}$.

SVAE-z-naive: it corresponds to the state space model of Fig. 2 (b). It does not represent existing work in trajectory generation, but it is an intuitive ablation model to EETG.

SVAE-z: simplified EETG model without using f (cf. Fig. 2 (c)). An ablation model to demonstrate the effects of $z_{1:T}$ and $\Theta_{0:T}$.

Disentangled SVAE (DSVAE): the complete model with f, $z_{1:T}$, and $\Theta_{0:T}$ (cf. Fig. 2 (c)).

Factorized Disentangled SVAE (FDSVAE): the factorized alternative of the complete model shwon in Fig. 2 (c).

Models with spatiotemporal validity use the "-S" suffix, such as SVAE-f-S, SVAE-z-S, DSVAE-S, FDSVAE-S. Spatial constraint do not apply to check-in trajectories and respective experiments are omitted.

4.4 Evaluation metrics

A generative model should avoid either creating totally random data (learning useless models) or simply replicating the training data (triggering mode collapse in a generative model). Our basic metric is *Mean Distance Error* (MDE), which measures the Euclidean or Harversine Distance [14, 31]. We also utilize a dedicated metric for complex data generation evaluation called *Maximum Mean Discrepancy* (MMD). MMD is used for example used in image generation [40] and graph generation [38]. MMD is a function that measures a distribution distance similar to KL-divergence distance. Defined as $MMD(\mathbb{S}, \widehat{\mathbb{S}}) \mapsto [0, 1]$ outputs 0 for exact similarity and 1 for non-similarity, where spatial features of trajectories are used in *MMD* calculation (detailed in Supplement A.4). In many other deep generative model evaluations, another important performance factor is *novelty*, which means that generated samples should not be exact replicas of original samples. Since our models are developed for continuous-valued coordinates, it is almost impossible to sample the same continuous real value. To perform this analysis, we have to convert real-valued points to grid cells. Each trajectory is projected onto a raster. Each trajectory point is mapped to a corresponding cell and grid index (i, j). The raw trajectory $(x_1, y_1), ..., (x_t, y_t)$ is converted to a sequence $(i_1, j_1), ..., (i_t, j_t)$, where t is the time stamp. If the sequence of a generated trajectory cannot be exactly matched to the training set, it is considered a "novel" trajectory. Finally, the novelty score is calculated as N/M, where M is total number of generated trajectories and N is the number of novel trajectories. However, a complete random guess or small grid size could also result in a perfect score of 1, which shows the disadvantage of this metric.

"Violation Score" (VS) is a method proposed in this paper to evaluate the results with respect to adhering to constraints. It is the ratio of the number of violation cases to the number of all cases. A lower VS score indicates a better results in terms of spatiotemporal validity. The metric is defined as follows:

$$VS = \frac{\sum_i^N \sum_{t=t_*}^T \mathbf{1}_c(\epsilon, s_{1:T}^{(i)})}{N \times (T - t_*)}, \mathbf{1}_c(\cdot) := \begin{cases} 1 & \text{if } c_*(\cdot) \wedge \dots \\ 0 & \text{else} \end{cases}$$

where $c_*(\cdot)$ is a logic form rule that returns True if a condition is satisfied, t_* is the start step defined for $c_*(\cdot)$, for example, $t_* = 2$ for the length of the segment and $t_* = 3$ for the angles of consecutive segments. We could have a rule $c_*(\cdot)$ that checks if the angle between consecutive segments is larger than 15 degree or not. We start at $t_* = 2$ since the first segment does not have predecessors.

4.5 Quantitative results

The performance of the various methods is shown in Table 1 using the MDE metric and MMD distance scores between actual and reconstructed trajectories in terms of angle distribution, segment length distribution, total length distribution, and grid point count distribution.

Reconstruction performance: EETG models outperform other deep generative methods and rule-based methods for most metrics (cf. Table 1). The rule-based model performs worse than all deep generative models. This model performs orders of magnitude worse than EETG for all metrics. For deep generative models, the improvement of VAE-based models and IGMM-GAN over LSTM is quite significant. For metrics such as angles and grid point density in MMD, LSTM performs worse than others by orders of magnitude as well. This is caused by the lack of randomness in rule-based and LSTM models, which demonstrates the important role of latent variables. Specifically, comparing simple SVAE-f and SVAE-z to our proposed DSVAE and FDSVAE in Table 1, the MMD metric shows that DSVAE performs best for both taxi datasets. The spatial constraint versions normally improve over non-constraint ones and DSVAE-S and FDSVAE-S perform best for most metrics. An exception being the SVAE-z-S method with respect to angles for Porto, and total length and grid points for the T-Drive MMD metric. Interestingly, IGMM-GAN shows the best performance in three MMDs for the POL dataset. This might be due to the relatively small size of POL data for VAE-like models to learn well.

All the methods, including EETG (versions) and comparison methods, have very high novelty scores (typically over 0.85). This indicates that they are indeed generating instead of copying trajectories. Note that a very trivial random trajectory generator can easily achieve a novelty score very close to 1. Therefore, purely having a high novelty score does not necessarily indicate high-quality generation. Comparing to purely random generation, a good generator usually generates more realistic trajectories and hence its generation may have a better chance to "coincide" with true and observed trajectories, which can lead to a lower novelty score. Moreover, the version with spatial constraints further increases the chance of such a "coincidence" (since it precluded many unrealistic ones) and could have a slightly lower novelty score. To sum up, an extremely high novelty score is not necessarily the best outcome. The novelty score only reflects one aspect of a generator's quality on whether it is simply copying the training set or suffers from serious "mode collapse". A reasonably high score (like the above results of all the methods) can well indicate that the generator is not trivially copying and does not suffer from serious "mode collapse" (high but not perfect score).

Constraints: By comparing models without constraints to models with constraints in Table 2, the proposed spatial regularization terms help generate much fewer violation cases for all models since VS consistently decreases when adding constraints. The rule-based model has no violation cases since it follows a agent-based simulation on a network with predefined rules. The distribution of related features in Fig. 5 shows more white space (indicating a zero number of samples) in red dashed areas, which indicate violation zones following the addition of constraints.

4.6 Qualitative results

Point distributions: Fig. 6 show the density of trajectory points as heatmaps to indicate the quality of the generated trajectories. The study areas are subdivided using a regular spatial grid and all generated and reference trajectory points are counted for each cell. The darker a cell, the more points are located in it. Rule-based models perform worst and we cannot find any meaningful patterns. We can see that all VAE-derived models with/without constraints perform generally better than LSTM. IGMM-GAN models also create patterns comparable to the reference trajectories. However, IGMM-GAN and other simplified models such as VAE-y and VAE-z, and

Table 1: Experimental results using Mean Distance Error (MDE) and Maximum Mean Discrepancy (MMD).

dataset	Metrics / Method	MDE+	Novelty	Angles in MMD	Segment lengths in MMD	Total lengths in MMD	Grid point density in MMD
Porto	Rule-based	NaN	0.975	0.68	0.03	0.11	0.252
	LSTM	13.6525	0.9022	0.5243	0.4976	0.4136	0.1135
	IGMM-GAN	11.1488	0.9348	0.0772	1.0	0.3779	0.0429
	SVAE-f	1.2422	0.8758	0.0430	0.0034	0.0655	0.0244
	SVAE-z-naive	1.7474	0.8993	0.0222	0.0144	0.3669	0.0384
	SVAE-z	1.73782	0.9122	0.0081	0.0069	0.3303	0.0279
	DSVAE	0.9018	0.9076	0.0649	0.0041	0.2439	0.0208
	FDSVAE	1.7415	0.9187	0.0380	0.0019	0.0497	0.0294
	SVAE-f-S	1.9755	0.8721	0.0795	0.1009	0.6947	0.0647
	SVAE-z-S	1.1896	0.8409	**0.0054**	0.0055	0.2593	0.0106
	DSVAE-S	**0.8059**	0.8953	0.0628	0.0032	0.2133	0.0208
	FDSVAE-S	1.2281	0.9072	0.0273	**0.0004**	**0.0495**	**0.0105**
T-Drive	Rule-based	NaN	0.9934	0.71	0.03	0.11	0.253
	LSTM	16.0594	0.9422	0.4005	0.6447	0.4080	0.3717
	IGMM-GAN	0.9577	0.9134	0.0556	0.7280	0.1052	0.003
	SVAE-f	0.6073	0.9215	0.3844	0.3563	0.1409	0.1360
	SVAE-z-naive	1.2258	0.8945	0.0317	0.0298	0.1285	0.0007
	SVAE-z	0.9849	0.8644	0.0178	0.0074	0.0437	0.0005
	DSVAE	0.5916	0.8931	0.1310	0.0788	0.1448	0.1040
	FDSVAE	1.1136	0.9103	0.3635	0.0076	0.7308	0.1594
	SVAE-f-S	1.5179	0.9379	0.4316	0.3456	0.1288	0.1207
	SVAE-z-S	0.9138	0.8857	0.0200	0.0079	**0.0436**	**0.0002**
	DSVAE-S	**0.5130**	0.8910	**0.0047**	0.0087	0.0852	0.0088
	FDSVAE-S	0.6118	0.8954	0.0088	**0.0018**	0.0610	0.0660
POL	Rule-based	NaN	0.9577	0.2407	0.0033	0.002	0.0121
	LSTM	34.5681	0.9608	0.5921	0.9292	0.9915	0.9835
	IGMM-GAN	0.8872	0.9027	**0.1957**	**0.0003**	**0.0004**	0.0023
	SVAE-f	10.6391	0.9428	0.2107	0.6874	0.0067	0.4426
	SVAE-z-naive	1.7419	0.8975	0.2958	0.0394	0.0277	0.0030
	SVAE-z	1.4154	0.8921	0.2403	0.0129	0.0044	0.0041
	DSVAE	0.3811	0.8376	0.2259	0.0042	0.0016	**0.0001**
	FDSVAE	**0.3487**	0.8310	0.2348	0.0193	0.0142	0.0002
Gowalla	Rule-based	NaN	0.9910	0.0181	**0.0026**	0.0011	0.0033
	LSTM	629.07	1.000	0.0325	NaN	NaN	NaN
	IGMM-GAN	101.32	0.9820	0.0351	0.1015	0.0431	0.0163
	SVAE-f	90.292	0.9233	0.0111	0.0430	0.0017	0.0030
	SVAE-z-naive	101.60	0.9761	0.0742	0.0654	0.0028	0.0026
	SVAE-z	40.2241	0.9016	**0.0067**	0.0328	0.0002	0.0026
	DSVAE	2.7645	0.8631	0.0065	0.0196	0.0002	0.0036
	FDSVAE	**1.7714**	0.8511	0.0102	0.0144	**0.0001**	**0.0025**

*NaN: models could not finish in acceptable time or not doable for MMD computing.
+: MDE unit are different here. Porto and T-Drive are kilometer. POL and Govalla are meter.

Table 2: Constraints: Violation Score (VS) results.

dataset	Methods	VS	Constrained Methods	VS
Porto	Rule-based	0.0	-	-
	LSTM	0.045219	-	-
	IGMM-GAN	0.02624	-	-
	SVAE-f	0.034881	SVAE-f-S	0.004960
	SVAE-z	0.018214	SVAE-z-S	0.002749
	DSVAE	0.027971	DSVAE-S	0.003682
	FDSVAE	0.021269	FDSVAE-S	0.001180
	Raw data	0.001718	-	-
T-Drive	Rule-based	0.0	-	-
	LSTM	0.004753	-	-
	IGMM-GAN	0.055332	-	-
	SVAE-f	0.008197	SVAE-f-S	0.033250
	SVAE-z	0.009581	SVAE-z-S	0.006895
	DSVAE	0.010779	DSVAE-S	0.003263
	FDSVAE	0.054197	FDSVAE-S	0.007847
	Raw data	0.003395	-	-

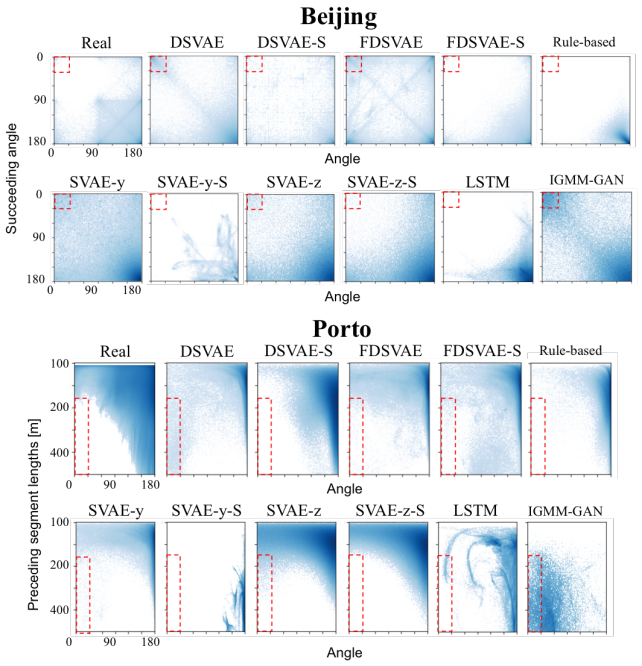


Figure 5: Constraints: results for T-Drive and Porto datasets with red-dashed areas indicating violation zones.

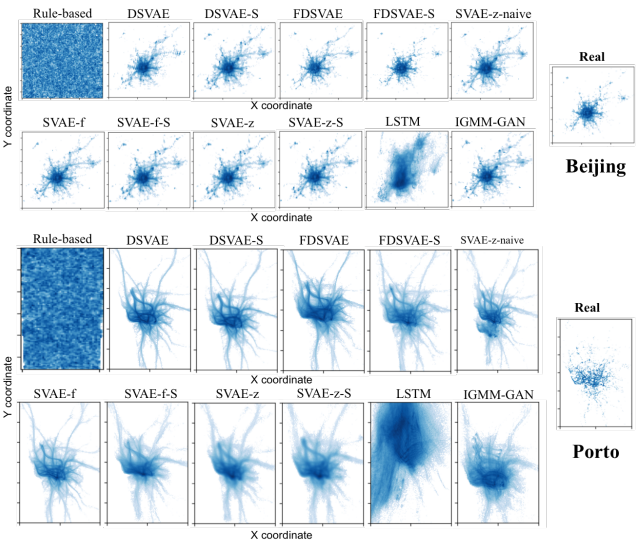


Figure 6: Point count heatmap comparison for GPS trajectories.

VAE-z-naive are not as "clean" as DSVAE(-S) and FDSVAE(-S) models. This indicates that these models do not capture small variances in data distributions.

The grid point count heatmaps for check-in trajectories in Fig. 7 also confirm that DSVAE, FDSVAE, and SVAE-z capture the patterns from the reference trajectories. Notice that generating POL data is a comparatively easier task since it is only based on 1,000 users. For rule-based methods, slightly clearer patterns of concentrations can be found since the underlying distributions are simpler. Most deep models perform well except LSTM and SVAE-f. However, for Gowalla data, LSTM almost produces no data at all since the generated points are not within the required geographic bounds. The wired nature of rule-based models is due to converting data from the *WGS84* reference system to a projected coordinate system. IGMM-GAN and other simplified VAE models capture the overall spatial trend in the whole region, but they failed to model the small variances of different sub-regions. Our proposed DSVAE and FDSVAE models generate point distributions that are very similar to the reference trajectories.

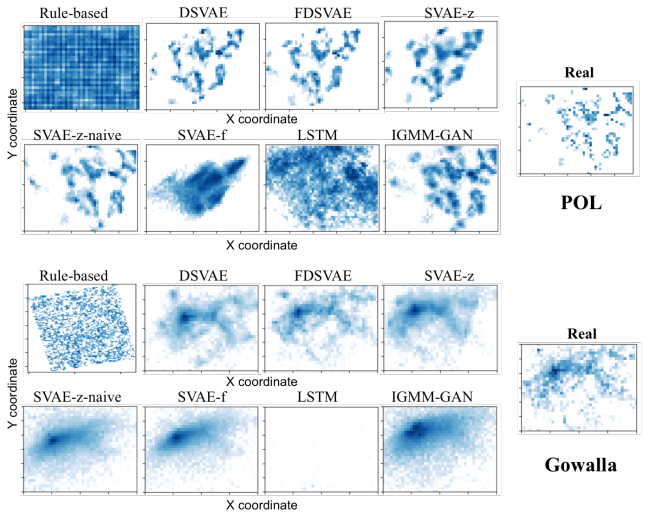
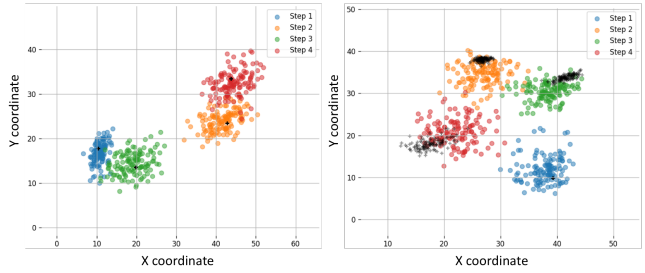


Figure 7: Point count heatmap comparison for check-in data.



(a) SVAE-z-naïve (b) SVAE-z

Figure 8: Comparing the effect of a chain of priors $\Theta_{1:T}$ to the dynamic generation of trajectories by two simplified models: (a) SVAE-z-naive model; (b) SVAE-z model. Different colors indicate different time stamps. Black crosses indicate a point generated using the Gaussian distribution's mean for each time stamp.

Dynamic priors and their effect: This experiment illustrates the effect of the chain of priors constraint $\Theta_{1:T}$ on the spatiotemporal dynamic pattern generation by comparing SVAE-z-naive (shown in the Fig. 8 (a)) and SVAE-z (shown in the Fig. 8 (b)). Each figure shows the 128 trajectories' first four points as colored dots. All 128 trajectories use the same fixed initial noise (inputs of Θ_0 are fixed). The black crosses in the two figures are the mean coordinates generated with the mean vectors of generated $z_{1:4}$ variables. The reason for the black crosses not to be directly located in the center of the point clouds is due to the non-linear decoding through different MLP_* layers. The points of the same color relate to the same time stamp. The blue, orange, green, and red colors correspond to time stamps 1, 2, 3, and 4, respectively. We can see that the SVAE-z-naive model learned a point distribution with a fixed mean since all black crosses are exactly at the same location. However, SVAE-z with chain-randomness learned a chain of dynamic patterns with slightly different means for each generated trajectory. This is the advantage of a more powerful stochastic process modeled by the chain of priors Θ_t. In Table 1, the SVAE-z model also performed best for a range of metrics, such as the MDE of 1.4154 compared to the 1.7419 for the SVAE-z-naive model.

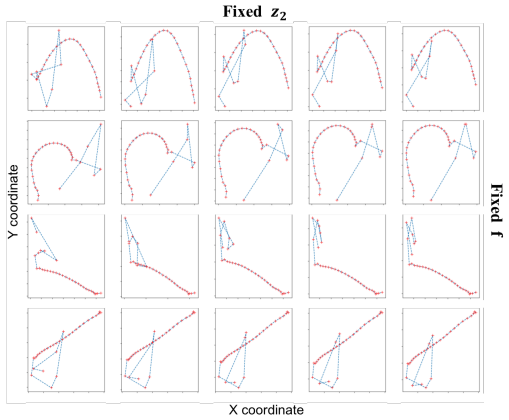


Figure 9: Disentangled factors studies. Using the Porto dataset with FDVAE-S model, each row shares the same spatiotemporal-invariant f factor, and different rows have a slightly different f. Each column shares the same spatiotemporal-variant z_t factor, and different z_t are used in different columns.

Disentanglement analysis: This experiment provides a qualitative analysis showing that separating spatiotemporal-variant and spatiotemporal-invariant factors achieves better interpretability as shown in Fig. 9. We use the FDVAE-S model for the Porto dataset as an example. The x-axis shows the sampled $z_{1:T}$ vectors' second dimension replaced with values ranging from 1 to 9. Along the y-axis, we randomly sampled nine different f vectors from the distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$. We can see that f controls the overall trend of trajectories since trajectories in the same row show very similar patterns. Trajectories in the same row show small variances due to different noise (z_t). For the same column, it shows that z_t could control high-dimensional geometric dynamics, though it is hard to visually conclude any specific geometric factor that z_t controls.

5 CONCLUSIONS AND FUTURE WORK

We propose a novel "End-to-End Trajectory Generation with spatiotemporal validity constraints" (EETG) framework for trajectory synthesis. The experimentation shows that our framework achieves superior performance not only for a conventional Mean Distance Error metric that computes the error between a pair of raw and generated trajectories, but also over feature distributions (MMD metrics), including angle distribution (angles between segments), and others for real-world and synthetic reference trajectory data. In terms of future work, an interesting direction is to increase explainability by incorporating deep layers for real-world factors, such as Point-of-Interests and road networks in an end-to-end manner. Another direction is that EETG can also be used for mobility mining tasks, e.g., map construction, and in other domains such as animal migration, ant movement, and sports analytics.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) Grant No. 1755850, No. 1841520, No. 2007716, No. 2007976, No. 1942594, No. 1907805, No. 1637541, 2109647, 2127901, Department of Defense grant HM02101410004, a Jeffress Memorial Trust Award,

Amazon Research Award, NVIDIA GPU Grant, and Design Knowledge Company (subcontract number: 10827.002.120.04).

REFERENCES

[1] Vincent Bindschaedler and Reza Shokri. 2016. Synthesizing plausible privacy-preserving location traces. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 546–563.

[2] Xinyu Chen, Jiajie Xu, Rui Zhou, Wei Chen, Junhua Fang, and Chengfei Liu. 2021. TrajVAE: A Variational AutoEncoder model for trajectory generation. *Neurocomputing* 428 (2021), 332–339.

[3] Ronald Choe, Javier Puig, Venanzio Cichella, Enric Xargay, and Naira Hovakimyan. 2015. Trajectory generation using spatial Pythagorean Hodograph Bézier curves. In *AIAA Guidance, Navigation, and Control Conference*. 0597.

[4] Seongjin Choi, Jiwon Kim, and Hwasoo Yeo. 2020. Trajgail: Generating urban trajectories using generative adversarial imitation learning. *arXiv preprint arXiv:2007.14189* (2020).

[5] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*. 2980–2988.

[6] Chongming Gao, Zhong Zhang, Chen Huang, Hongzhi Yin, Qinli Yang, and Junming Shao. 2020. Semantic trajectory representation and retrieval via hierarchical embedding. *Information Sciences* 538 (2020), 176–192.

[7] Chongming Gao, Yi Zhao, Ruizhi Wu, Qinli Yang, and Junming Shao. 2019. Semantic trajectory compression via multi-resolution synchronization-based clustering. *Knowledge-Based Systems* 174 (2019), 177–193.

[8] Qiang Gao, Goce Trajcevski, Fan Zhou, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2019. DeepTrip: Adversarially Understanding Human Mobility for Trip Recommendation. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 444–447.

[9] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks* 18, 5-6 (2005), 602–610.

[10] Xiaojie Guo and Liang Zhao. 2020. A systematic survey on deep generative models for graph generation. *arXiv preprint arXiv:2007.06686* (2020).

[11] Xiaojie Guo, Liang Zhao, Zhao Qin, Lingfei Wu, Amarda Shehu, and Yanfang Ye. 2020. Interpretable Deep Graph Generation with Node-Edge Co-Disentanglement. *arXiv preprint arXiv:2006.05385* (2020).

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. (2016).

[13] Wei-Ning Hsu, Yu Zhang, and James Glass. 2017. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*. 1878–1889.

[14] Dou Huang, Xuan Song, Zipei Fan, Renhe Jiang, Ryosuke Shibasaki, Yu Zhang, Haizhong Wang, and Yugo Kato. 2019. A Variational Autoencoder Based Generative Model of Urban Human Mobility. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 425–430.

[15] Joon-Seok Kim, Hyunjee Jin, Hamdi Kavak, Ovi Chris Rouly, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. 2020. Location-based Social Network Data Generation Based on Patterns of Life. In *IEEE International Conference on Mobile Data Management (MDM'20)(to appear). IEEE*.

[16] Diederik P Kingma and Max Welling. 2019. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691* (2019).

[17] Qingzhe Li, Jessica Lin, Liang Zhao, and Huzefa Rangwala. 2017. A uniform representation for trajectory learning tasks. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–4.

[18] Xiucheng Li, Gao Cong, and Yun Cheng. 2020. Spatial transition learning on road networks with deep probabilistic models. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 349–360.

[19] Yingzhen Li and Stephan Mandt. 2018. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991* (2018).

[20] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. 2020. Online anomalous trajectory detection with deep generative sequence modeling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 949–960.

[21] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2575–2582.

[22] Tengfei Ma, Jie Chen, and Cao Xiao. 2018. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *Advances in Neural Information Processing Systems*. 7113–7124.

[23] Kun Ouyang, Reza Shokri, David S Rosenblum, and Wenzhuo Yang. 2018. A Non-Parametric Generative Model for Human Trajectories.. In *IJCAI*. 3812–3817.

[24] Luca Pappalardo and Filippo Simini. 2018. Data-driven generation of spatio-temporal routines in human mobility. *Data Mining and Knowledge Discovery* 32, 3 (2018), 787–829.

[25] Nikos Pelekis, Christos Ntrigkogias, Panagiotis Tampakis, Stylianos Sideridis, and Yannis Theodoridis. 2013. Hermoupolis: a trajectory generator for simulating generalized mobility patterns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 659–662.

[26] Dieter Pfoser and Christian S. Jensen. 1999. Capturing the Uncertainty of Moving-Object Representations. In *SSD '99: Proceedings of the 6th International Symposium on Advances in Spatial Databases*. Springer-Verlag, 111–132.

[27] Dieter Pfoser and Yannis Theodoridis. 2003. Generating semantics-based trajectories of moving objects. *Computers, Environment and Urban Systems* 27, 3 (2003), 243–263.

[28] Jinmeng Rao, Song Gao, Yuhao Kang, and Qunying Huang. 2020. LSTM-TrajGAN: A Deep Learning Approach to Trajectory Privacy Protection. *arXiv preprint arXiv:2006.10521* (2020).

[29] Wei Ren and Randy W Beard. 2004. Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. *IEEE Transactions on Control Systems Technology* 12, 5 (2004), 706–716.

[30] Jean-Marc Saglio and Jose Moreira. 2001. Oporto: A realistic scenario generator for moving objects. *GeoInformatica* 5, 1 (2001), 71–93.

[31] Daniel Smolyak, Kathryn Gray, Sarkhan Badirli, and George Mohler. 2020. Coupled IGMM-GANs with Applications to Anomaly Detection in Human Mobility Data. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 6, 4 (2020), 1–14.

[32] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. 2016. Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2618–2624.

[33] Shawn Stephens, Satyanarayana G Manyam, David W Casbeer, Venanzio Cichella, and Donald L Kunz. 2019. Randomized Continuous Monitoring of a Target by Agents with Turn Radius Constraints. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 588–595.

[34] Yannis Theodoridis and Mario A Nascimento. 2000. Generating spatiotemporal datasets on the WWW. *ACM SIGMOD Record* 29, 3 (2000), 39–43.

[35] Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. 2020. A Survey on Trajectory Data Management, Analytics, and Learning. *arXiv preprint arXiv:2003.11547* (2020).

[36] Shiyu Wang, Yuanqi Du, Xiaojie Guo, Bo Pan, and Liang Zhao. 2022. Controllable Data Generation by Deep Learning: A Review. *arXiv preprint arXiv:2207.09542* (2022).

[37] Guolei Yang, Ying Cai, and Chandan K Reddy. 2018. Recurrent spatio-temporal point process for check-in time prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2203–2211.

[38] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773* (2018).

[39] Wenbin Zhang, Liming Zhang, Dieter Pfoser, and Liang Zhao. 2021. Disentangled dynamic graph deep generation. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 738–746.

[40] Wenju Zhang, Xiang Zhang, Long Lan, and Zhigang Luo. 2020. Maximum Mean and Covariance Discrepancy for Unsupervised Domain Adaptation. *Neural Processing Letters* 51, 1 (2020), 347–366.

[41] Fan Zhou, Qiang Gao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2018. Trajectory-User Linking via Variational AutoEncoder.. In *IJCAI*. 3212–3218.

A APPENDIX

A.1 Evolution of VAE-like models

To better illustrate the novelty of this work, we first present a discussion of the evolution of generative models for trajectory data and why they can or cannot solve the three challenges motivating this work.

Existing work fails to address the first challenge because of limitations of the model shown in Fig. 2 (a)) and used by either VAE-based models [2, 14, 20] or GAN-based models [23, 28, 31] in the trajectory domain. This sequential variational model applies a Markovian-treatment of the sequence with a single latent variable f that captures mixed global and local semantics for the entire sequence. In other domains, such as for videos [19] and graphs [39], models consider the latent state variables $z_{1:T}$ of temporal samples, called "state space model" [5], as illustrated in Fig. 2 (b). They consider each time step to be dependent on a corresponding latent state z_t. Combining f and $z_{1:T}$ models could solve the first challenge and be more powerful than existing efforts since f and $z_{1:T}$ capture global semantics and local semantics, respectively. Such a model, i.e., each trajectory point is determined jointly by a current latent state z_t and the prior f could factorize global semantics (e.g., origin-destination pair types) with local semantics (e.g., road condition changes). f utilizes a prior distribution, such as an isotropic Gaussian, while all z_t share another independent single prior distribution, such as another isotropic Gaussian.

The second challenge relates to lacking a dynamic inductive bias and a basic state space model still fails to inject randomness when it generates z_{t+1} from z_t. For example, using a typical LSTM model as a state space model, we get the same latent state matrix for z_t with the same latent state z_{t+1} for the next time step. To allow for better dynamic induction, we expect that the latent state in z_{t+1} could be stochastically sampled from the same input z_t. With such a dynamic inductive bias that is reflected in Θ_t, we can expand the modeling power of spatiotemporal dynamics. Such an architecture has been introduced in [5], however, it still requires a special deep neural network architecture design, which we propose to achieve end-to-end trajectory generation.

The third challenge of jointly ensuring spatiotemporal validity is quite unique for trajectory data. Deep generative models only model a continuous distribution (e.g. the Gaussian distribution) in a space. This does guarantee the validity even for a simple hard constraint like a truncated Gaussian, which is a much simpler constraint than the spatiotemporal rules we propose for trajectories. Current deep learning models lack a framework to handle such complicated hard constraints.

A.2 Neural network details

For the Porto and T-Drive dataset, the $MLP_s(\cdot)$ uses two layers of structures with 48 and 16 neurons, respectively. For all $BiLSTM_*$ and RNN_* modules, the dimensions of the hidden states (not shown in our paper) are set to 512. For the static pattern $BiLSTM_f$, the dimension of recurrent input o_t is 16. The MLP_{μ_f} and MLP_{σ_f} have inputs of 512×2 dimensions, and the output dimension is 256. For the dynamic pattern $BiLSTM_z$, the dimension of input $\tilde{a}^t$ is 16. The second RNN_z module takes forward and backward hidden states of dimension 512×2 as input. The hidden state of RNN has 512 dimensions. $MLP_{\mu_{z_t}}$ and $MLP_{\sigma_{z_t}}$ is set to have one layer of 64 neurons. The priors Θ include μ_t and σ_t, whose decoding modules $BiLSTM_\mu$ and $BiLSTM_\sigma$ have the same design as $BiLSTM_z$. The difference is that its input is a 16 dimensional $\mathbf{0}$ vector. MLP_μ and MLP_σ have each 64 neurons. The $f||z_t$ input's dimension is $256+64$ for the decoder module $BiLSTM_s$. MLP_s has one internal layer of 128 neurons and a last layer of two neurons for two coordinate values in s_t.

There are a few differences for POL and Gowalla data. The $MLP_s(\cdot)$ contains two layers with $[48, 32]$ neurons, respectively. For all $BiLSTM_*$ and RNN_* modules, the dimensions of the hidden states (not shown in our paper) are set to 512. For the static pattern $BiLSTM_f$, the dimension of recurrent input o_t is 32. The MLP_{μ_f} and MLP_{σ_f} input dimension is 512×2, and the output dimension is 256. For the dynamic pattern $BiLSTM_z$, the input dimension $\tilde{a}^t$ is 32. The second RNN_z module takes forward and backward hidden states of dimension 512×2 as input. The hidden state of RNN has 512 dimensions. $MLP_{\mu_{z_t}}$ and $MLP_{\sigma_{z_t}}$ are set to have one layer of 32 neurons. The priors Θ includes μ_t and σ_t, whose decoding modules $BiLSTM_\mu$ and $BiLSTM_\sigma$ have the same design as $BiLSTM_z$. The difference is that its input is a $\mathbf{0}$ vector with a dimension of 32. The MLP_μ and MLP_σ have the same number of 32 neurons. The $f||z_t$ input's dimension is $256+32$ for the decoder module $BiLSTM_s$. The MLP_s has two internal layer of $[64, 32]$ neurons and a last layer of two neurons for two coordinate values in s_t.

A.3 Additional parameter tuning

Besides the different neural network architectures, there are several hyper-parameters to be tuned. (1) β parameter for β-VAE to enhance disentangling. We tested the values $[1, 10, 100]$. 100 is the choice for the model in our paper. (2) γ parameter for regularization of constraints. We tested the values $[1, 10, 100]$. 1 is the choice for the presented model in the paper. (3) Other conventional parameters. Learning rate is set to be 0.0002 for Porto, 0.0002 for T-Drive, 0.0002 for POL, and 0.002 for Gowalla. The training epochs are all set to 100. The batch size is set to 128 for all datasets and models. All the ablation competing methods use parameter tuning for β, γ and learning rates of $[0.02, 0.002, 0.0002]$ to achieve the best performance. IGMM-GAN models are also tuned for different learning rates of $[0.01, 0.001, 0.0001]$ and training epochs of $[5000, 10000]$.

A.4 Mean Distance Error (MDE) metric details

Mean Distance Error (MDE) for Haversine Distance or Euclidean distance is the most used metric in existing end-to-end trajectory generation work [14, 31]. It is defined as $\frac{1}{n}\sum_i^N ||s_t - \hat{s}_t||_2$, where $||\cdot||_2$ denotes the L2 norm. This metric could not be used by Ruled-based models since there is no mapping from real inputs s_t to its synthetic output $\hat{s}_t$. Since MDE only computes a reconstruction loss for the distribution mean, we propose to directly evaluate the whole distributions of spatial features using "Maximum Mean Discrepancy (MMD)", a sophisticated evaluation framework used in recent approaches for different complex high-dimensional data generation problem, such as image [40] and graph [38] generation. It is a recently-developed alternative to replace conventional metrics like KL-divergence, which computes the function $MMD(D, \tilde{D}) \mapsto [0, 1]$. It outputs 1 for no similarity and 0 for matching distributions, where the rows of matrix D are spatial features of real trajectories

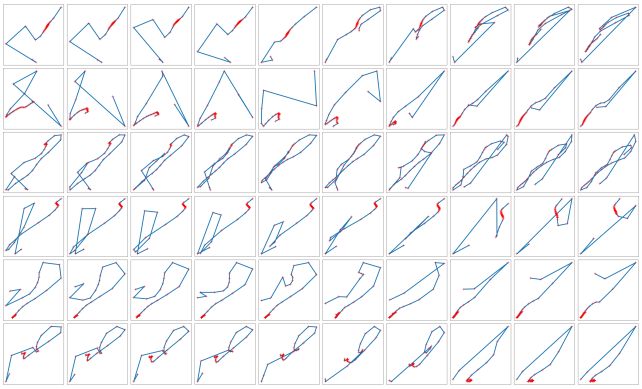


Figure 11: Effects of f (rows) and z_t (columns) over T-Drive dataset

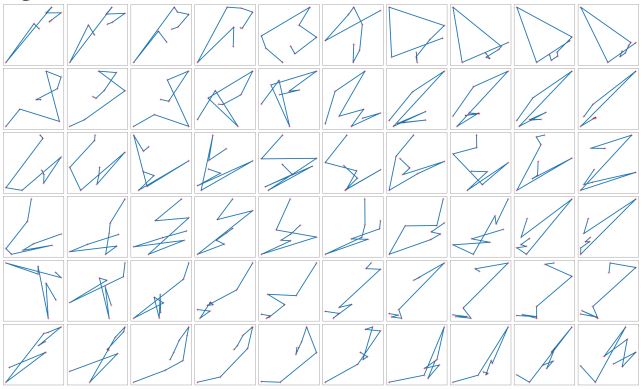


Figure 12: Effects of f (rows) and z_t (columns) over POL dataset

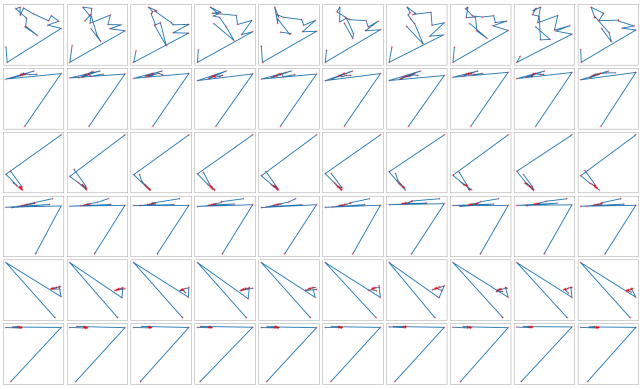


Figure 13: Effects of f (rows) and z_t (columns) over Gowalla dataset

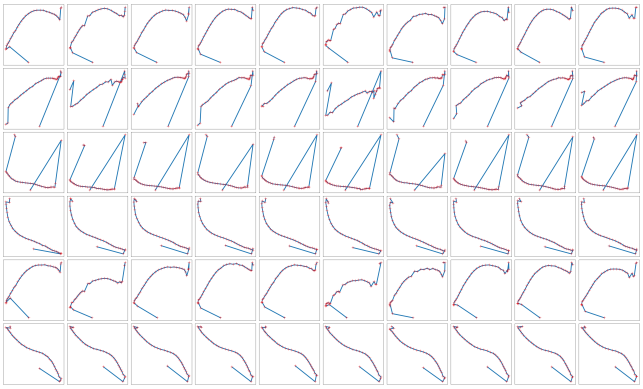


Figure 10: Effects of f (rows) and z_t (columns) over Porto dataset

and the rows of matrix $\tilde{D}$ are corresponding spatial features of generated trajectories. Alternative methods to MMD include Jensen-Shannon Divergence [23] and KL-divergence, a less sophisticated method belongs to the same family as MMD. Taking an example of spatial features, a trajectory has a feature vector $d_{1:T-1}$ representing lengths of two consecutive points in a trajectory sequence sample $s_{1:T}$. The original length set is represented as $D \in \mathbf{R}^{N\times(T-1)}$ for a total of N trajectory sequences and $T-1$ segments, and $\tilde{D} \in \mathbf{R}^{\tilde{N}\times(T-1)}$ for $\tilde{N}$ generated trajectories. For this paper, the chosen spatial features include segment lengths of two consecutive points, angles of two consecutive segments, total segment lengths, and point density on a spatial grid. Point density is a grid-based metric to compare trajectories visually [23].

A.5 Additional experimental results

We conducted an extensive case study for different datasets to illustrate the effectiveness of our factorization approaches. Each row is generated with a fixed f, and each column is generated with a fixed $z_{1:T}$ sequence. We can see that for both taxi trajectories and check-in trajectories, f controls a static pattern (similar patterns in each row), while z_t controls the variances for each trajectory in such a row. We can see that for both GPS trajectories and Check-in trajectories our proposed EETG framework could factorize a consistent general semantic because each row shows the same overall trend but a dynamic ad-hoc semantic. Different cells show small variances within the same row. However, for the local semantics across different columns, it is hard to tell the meaning of a spatiotemporal dynamic emerging from the z_t variable. ,