Generating Personalized Insulin Treatments Strategies with Conditional Generative Time Series Models

Manuel Schürch
University of Zurich
manuel.schuerch@uzh.ch

Xiang Li
University of Zurich

Ahmed Allam
University of Zurich

Giulia Hofer
University Hospital Zurich

Amina Mollaysa
University of Zurich

Claudia Cavelti-Weder
University Hospital Zurich

Michael Krauthammer
University of Zurich

Abstract

We propose a novel framework that combines deep generative time series models with decision theory for generating personalized treatment strategies. It leverages historical patient trajectory data to jointly learn the generation of realistic personalized treatment and future outcome trajectories through deep generative time series models. In particular, our framework enables the generation of novel multivariate treatment strategies tailored to the personalized patient history and trained for optimal expected future outcomes based on conditional expected utility maximization. We demonstrate our framework by generating personalized insulin treatment strategies and blood glucose predictions for hospitalized diabetes patients, showcasing the potential of our approach for generating improved personalized treatment strategies.

1 Introduction

Recent advancements in deep conditional generative models have revolutionized various domains including the generation of textual prompt-answers [Brown et al., 2020], high-quality images [Rombach et al., 2022], and molecules based on desired properties [Mollaysa et al., 2019]. However, applying these techniques for generating complex and multivariate healthcare time series poses unique challenges that have yet to be fully addressed. A fundamental issue is the lack of a clear and simple framework for leveraging conditional samples from multivariate time series. To tackle this, we explore the potential of combining deep probabilistic generative time series models [Tomczak, 2022] with decision theory, to enable optimal personalized treatment generation learned from retrospective patient data. By exploiting the power of conditional generative models, we can first learn to generate feasible personalized treatment strategies conditioned on the current personalized conditions. Second, we can also learn to generate future personalized outcome trajectories conditioned on specific treatment trajectories. Third, to bridge the gap between the conditional generation of time series and decision-making, we propose the personalized treatment generation by optimizing the expected utility of the future outcome trajectories, so that both feasible and optimal treatments can be learned.

Our proposed approach is illustrated in Figure 1 and offers several innovative contributions:

- **Framework for Time Series Samples:** We present a new framework for leveraging samples of conditional distributions over multivariate time series based on decision theory.

1st Workshop on Deep Generative Models for Health at NeurIPS 2023.

- **Generating Realistic Treatments:** We exploit deep generative models for generating novel personalized treatment strategies and future outcome trajectories.
- **Optimal Personalized Treatments:** We use goal-directed generation of personalized treatments optimized for expected future outcome utility learned from retrospective data.
- **Testing on Diabetes Patient Data:** We demonstrate the potential of our framework by generating insulin treatment and blood glucose trajectories for hospitalized diabetes patients.

2 Background

Generating personalized treatment trajectories based on historical patient data is the ultimate goal of personalized medicine with machine learning (ML). In this paper, we focus on the generation of multivariate insulin strategies that lead to optimal expected future blood glucose trajectories (Figure 1). Several ML approaches deterministically predict the blood glucose outcome [Xie and Wang, 2020, Liu et al., 2020, Noaro et al., 2020, Jaloli and Cescon, 2022] and only a few [Sergazinov et al., 2023] take into account the inherent uncertainty and multi-modal distribution of the future blood glucose. Moreover, it is not obvious how these predictions of blood glucose can be utilized for treatment generation. On the other hand, there are several reinforcement-learning approaches [Tejedor et al., 2020, Javad et al., 2019, Shifrin and Siegelmann, 2020, Zhu et al., 2020, Emerson et al., 2023] optimizing a deterministic reward function, mostly exploiting synthetic or continuously measured blood glucose data, and considering only a simplified action space, for which explicit constraints have to be introduced to obtain feasible and consistent treatments. We instead try to generate feasible multivariate treatment strategies consistent over multiple treatments and time without requiring heuristic constraints using deep generative models [Tomczak, 2022], sharing analogies with [Ajay et al., 2022]. To the best of our knowledge, our approach combining deep generative models with expected utility learning is the first to deal with the generation of entire multivariate treatment and outcome trajectories jointly learned from retrospective patient data, to achieve feasible, personalized, and optimal treatment strategies.

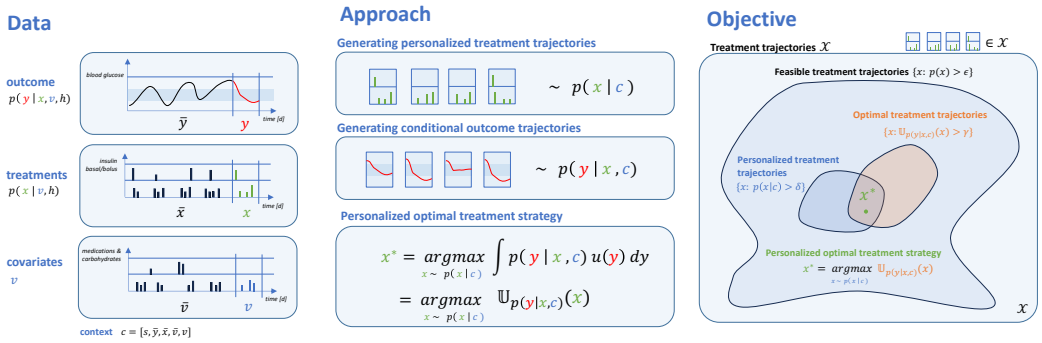


Figure 1: Overview of data, our approach, and objective for generating personalized treatments.

3 Methodology

We consider multivariate time series data involving outcome $\boldsymbol{y}_{1:T} \in \mathbb{R}^{P\times T}$, treatment $\boldsymbol{x}_{1:T} \in \mathcal{X} = \mathbb{R}^{D\times T}$, and covariate $\boldsymbol{v}_{1:T} \in \mathbb{R}^{V\times T}$ trajectories (Figure 1). For insulin treatments, we consider blood glucose as outcome, basal and bolus insulin as treatments, and other medications and carbohydrates as covariates. We divide all trajectories into past and future windows, where the latter has fixed length K, for instance $\boldsymbol{y}_{1:T} = [\bar{\boldsymbol{y}}, \boldsymbol{y}]$ with $\boldsymbol{y} \in \mathbb{R}^{P\times K}$. We define the personalized context $\boldsymbol{c} = \{\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{v}}, \boldsymbol{v}, \boldsymbol{s}\}$ with some additional non-temporal patient data $\boldsymbol{s}$. We use a dataset $\mathcal{D} = \{\boldsymbol{y}^i_{1:T_i}, \boldsymbol{x}^i_{1:T_i}, \boldsymbol{v}^i_{1:T_i}, \boldsymbol{s}^i\}_{i=1}^N$ with N historical patient time series.

3.1 Generation of Personalized Treatments

In this section, we present our approach to leverage multivariate time series samples for learning feasible and "optimal" treatment strategies based on the joint personalized distribution $p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{c}) =$

$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})p(\boldsymbol{x}|\boldsymbol{c})$. In case of insulin treatments, "optimal" is achieved when the future outcome (i.e. blood glucose values) lie within a predefined range.

Direct Approach: We could directly learn a probabilistic mapping $p(\boldsymbol{x}|\boldsymbol{c})$ from the past personalized conditions to the future treatments and take the most probable treatment strategy

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{c}).$$

However, this approach learns to replicate the treatments in the historical training data, that is problematic if these were sub-optimal regarding treatment outcomes.

Indirect Approach: In a second approach, we probabilistically predict the future outcomes $\boldsymbol{y} \in \mathbb{R}^{P \times T}$ and specify a utility function $u(\boldsymbol{y})$ describing the preferences for the future treatment outcomes (Section A.2). For insulin treatments, the utility function is highest for blood glucose values in the predefined band. We can indirectly optimize the conditional expected utility $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})u(\boldsymbol{y})d\boldsymbol{y}$ (A.2.2), that is,

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}}\, \mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}),$$

so that optimal treatments regarding future treatment outcomes are learned. However, optimizing over all possible treatments $\mathcal{X} = \mathbb{R}^{D \times T}$ yields unfeasible treatments, e.g. not adapted to the personalized conditions, and the evaluation of the learned expected utility is not accurate, compare Figure 1 (right) and subsection 3.1.

Joint Approach: We instead propose to exploit the joint distribution $p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{c}) = p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})p(\boldsymbol{x}|\boldsymbol{c})$. In particular, we first sample U feasible personalized treatment suggestions

$$\boldsymbol{x}_{|\boldsymbol{c}}^{(1)}, \ldots, \boldsymbol{x}_{|\boldsymbol{c}}^{(u)}, \ldots, \boldsymbol{x}_{|\boldsymbol{c}}^{(U)} \overset{\text{iid}}{\sim} p(\boldsymbol{x}|\boldsymbol{c})$$

and secondly, we choose the treatment with maximal conditional expected utility

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}_{|\boldsymbol{c}}^{(u)} \sim p(\boldsymbol{x}|\boldsymbol{c})}{\operatorname{argmax}}\; \mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}_{|\boldsymbol{c}}^{(u)}).$$

With this approach, we learn feasible and personalized treatment strategies $\boldsymbol{x}^{(u)} \sim p(\boldsymbol{x}|\boldsymbol{c})$ that have a high probability of resulting in good future treatment outcomes $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}^{(u)}) = \int p(\boldsymbol{y}|\boldsymbol{x}^{(u)}, \boldsymbol{c})u(\boldsymbol{y})d\boldsymbol{y}$. However, computing this integral exactly is infeasible for non-trivial distributions $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})$ (A.2.8), therefore we propose an approach with deep generative models.

Expected Utility with Generative Models: Deep conditional generative models are powerful for efficiently generating conditional samples of complex distributions over multivariate time series, for instance, personalized outcome trajectories

$$\boldsymbol{y}_{|\boldsymbol{x},\boldsymbol{c}}^{(1)}, \ldots, \boldsymbol{y}_{|\boldsymbol{x},\boldsymbol{c}}^{(s)}, \ldots, \boldsymbol{y}_{|\boldsymbol{x},\boldsymbol{c}}^{(S)} \overset{\text{iid}}{\sim} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c}).$$

These can be exploited to approximate the integral in the expected utility $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})u(\boldsymbol{y})d\boldsymbol{y}$ with Monte-Carlo samples $\hat{\mathbb{U}}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}) = \frac{1}{S}\sum_{s=1}^{S} u\left(\boldsymbol{y}_{|\boldsymbol{x},\boldsymbol{c}}^{(s)}\right)$, leading to the optimization of the sample-based expected utility, that is,

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}_{|\boldsymbol{c}}^{(u)} \sim p(\boldsymbol{x}|\boldsymbol{c})}{\operatorname{argmax}}\; \hat{\mathbb{U}}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}_{|\boldsymbol{c}}^{(u)}) = \underset{\boldsymbol{x}_{|\boldsymbol{c}}^{(u)} \sim p(\boldsymbol{x}|\boldsymbol{c})}{\operatorname{argmax}}\; \sum_{s=1}^{S} u\left(\boldsymbol{y}_{|\boldsymbol{x}^{(u)},\boldsymbol{c}}^{(s)}\right),$$

for which we refer to Section A.2.7 for more details.

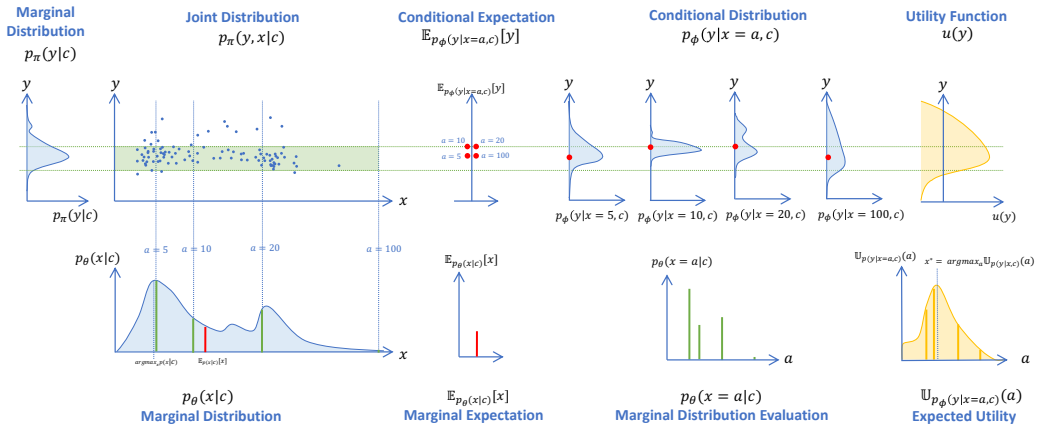
3.2 Deep Generative Model

In this section, we present our approach to learn the personalized joint distribution

$$p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{c}) = p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})p_\theta(\boldsymbol{x}|\boldsymbol{c})$$

with deep generative time series models [Tomczak, 2022, Murphy, 2022] with learnable parameters $\pi = \{\phi, \theta\}$. In particular, we focus on a probabilistic encoder-decoder transformer [Vaswani et al.,

Figure 2: Illustration of the joint personalized distribution $p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{c})$, with scalar and non-temporal $\boldsymbol{y}$ and $\boldsymbol{x}$.



2017], as illustrated in Figure 3 and further described in A.3. First, for encoding the personalized history, we use either a deterministic encoder, where the personalized history $\boldsymbol{h}$ is mapped to a fixed latent representation $\boldsymbol{z} = f_\psi(\boldsymbol{h}) \in \mathbb{R}^{L\times(T-K)}$, or a probabilistic encoder with a deep parametrized probability distribution $p_\psi(\boldsymbol{z}|\boldsymbol{h}) = \prod_{t=1}^{T-K}\prod_{l=1}^{L}\mathcal{N}\left(z_{tl}|\mu_\psi^{tl}(\boldsymbol{h}), \sigma_\psi^{tl}(\boldsymbol{h})\right)$ involving deep neural networks $\mu_\psi^{tl}(\boldsymbol{h})$ and $\sigma_\psi^{tl}(\boldsymbol{h})$. Second, the multivariate outcome trajectories $\boldsymbol{y} \in \mathbb{R}^{P\times K}$ are parametrized as $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}) = \prod_{t=1}^{K}\prod_{p=1}^{P}\mathcal{N}\left(y_{tp}|\mu_\phi^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}), \sigma_\phi^{tp}\right)$, with learnable mean $\mu_\phi^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})$ and variance σ_ϕ^{tp}. Third, for the multivariate treatment trajectories $\boldsymbol{x} \in \mathbb{R}^{D\times K}$, a Poisson likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z}) = \prod_{t=1}^{K}\prod_{d=1}^{D}\mathcal{P}\left(x_{td}|\lambda_\theta^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})\right)$ with deep parametrized mean $\lambda_\theta^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})$ is used.

3.2.1 Source of Stochasticity

Although the likelihoods above are parametric distributions, the predictive distributions of the outcome and treatment trajectories can be arbitrarily complex when using deep generative models. We compare our model with three different sources of stochasticity in the generative process (Figure A.3.4).

Parametric Mode: For a deterministic encoder without any additional stochasticity, the joint distribution $p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})$ is $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})$ with encoded history $\boldsymbol{z} = f_\psi(\boldsymbol{h})$. Training with the log-likelihood

$$\mathcal{L}_1(\pi; \mathcal{D}) = \log p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})$$

can be used as a baseline with resulting parametric predictive distributions (factorized Gaussians and Poissons) of treatments and outcomes.

Latent Variable Mode: With a probabilistic encoder $p_\psi(\boldsymbol{z}|\boldsymbol{h})$ and corresponding prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \sigma_p^2\mathbb{I})$ of the latent variable, the joint distribution is $p_\pi(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{h}) = p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})p(\boldsymbol{z})$. Based on variational inference (A.3.4), the objective function is

$$\mathcal{L}_2(\pi; \mathcal{D}) = \mathbb{E}_{p_\psi(\boldsymbol{z}|\boldsymbol{h})}\left[\log p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})\right] + \mathbb{E}_{p_\psi(\boldsymbol{z}|\boldsymbol{h})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})\right] - KL\left[p_\psi(\boldsymbol{z}|\boldsymbol{h})||p(\boldsymbol{z})\right],$$

leading to infinite mixtures of Gaussian and Poisson predictive distributions of the outcomes and treatments.

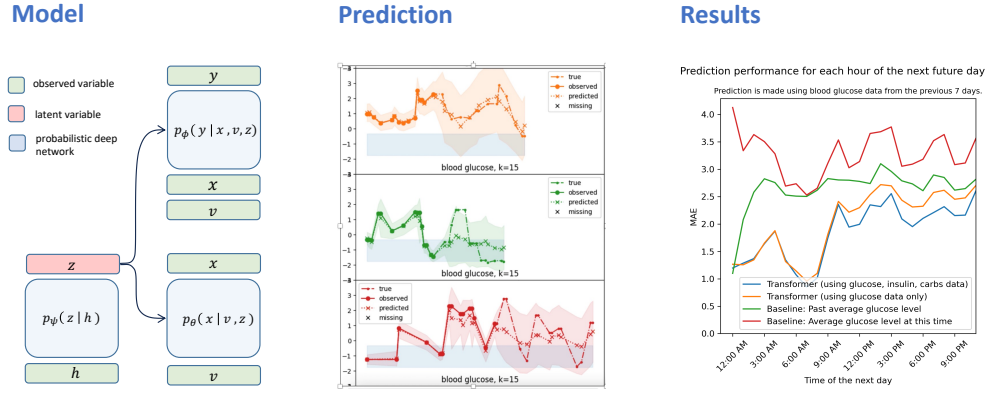
Auto-Regressive Mode: In the auto-regressive case, the joint distribution $p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})$ is decomposed as $\prod_{t=1}^{T} p_\pi(\boldsymbol{y}_t, \boldsymbol{x}_t|\boldsymbol{v}_t, \boldsymbol{z}_{1:t-1})$ with varying encoded history $\boldsymbol{z}_{1:t-1} = f_\psi([\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t-1}, \boldsymbol{v}_{1:t-1}, \boldsymbol{s}])$. Maximizing the log-likelihood

$$\mathcal{L}_3(\pi; \mathcal{D}) = \sum_{t=1}^{T} \log p\left(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{z}_{1:t-1}\right) + \log p\left(\boldsymbol{x}_t|\boldsymbol{v}_t, \boldsymbol{z}_{1:t-1}\right)$$

allows us to generate multivariate trajectories beyond infinite mixture distributions.

Implementation: We use a transformer encoder [Vaswani et al., 2017] with multi-head self-attention blocks to encode the history $p_\psi(\boldsymbol{z}|\boldsymbol{h})$. The output $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})$ and the treatment $p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})$ prediction networks are implemented using two transformer decoders with self-attention and cross-attention blocks to attend also to the encoded past $\boldsymbol{z}$ (Figure 3). We train our model with historical patient trajectories $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$ using the three objective functions $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ (A.4, A.3.4).

Figure 3: Transformer-based model (left), illustration (middle), and evaluation (right) of personalized prediction.



4 Experiments and Results

Insulin and Blood Glucose Trajectories: We test our approach using retrospective patient data from University hospital ($N = 2621$) comprised of blood glucose trajectories $\boldsymbol{y}_{1:T} \in \mathbb{R}^{1\times T}$, multivariate treatment strategies $\boldsymbol{x}_{1:T} \in \mathbb{R}^{2\times T}$ representing basal and bolus insulin injection doses, and carbohydrates as covariates $\boldsymbol{v}_{1:T} \in \mathbb{R}^{1\times T}$. For our heterogeneous patient data, we provide comprehensive descriptive data analysis in Appendix A.6. Note that this dataset presents particular challenges due to the infrequent reporting of carbohydrates and sparse blood glucose measurements, typically only 3-4 times per day.

Results for Blood Glucose Prediction: We provide preliminary results from our proposed method for generating personalized insulin treatment and future outcome blood glucose trajectories. In particular, the probabilistic prediction of the personalized outcomes trajectories $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})$ are shown in Figure 3 (middle and right), and in the Appendix in Table 1 and Figure A.6. From Figure 3 (right), we observe that our transformer model outperforms two simple baselines for predicting blood glucose 24 hours ahead. The green line refers to the baseline using the patient's past average glucose level. The red baseline is the average of glucose measurements at a particular time point from all patients. For the sake of simplicity, the prediction always starts at midnight, so that the best performance gap is around 3-6 hours in the future. In future work, we will predict the outcomes from any time point, and improve the experiments in several directions, as outlined in Appendix A.5.

5 Conclusion

In this paper, we propose a framework for leveraging conditional samples of multivariate time series generated by deep conditional generative time series models. It can be used to learn the complex interaction between outcome, treatment, and covariate trajectories from retrospective patient time series data. Moreover, feasible, personalized, and optimal treatment trajectories can be generated by combining it with conditional expected utility maximization. Our preliminary results focused on modeling insulin and blood glucose trajectories from diabetes patients, however, our framework can be generally applied to other healthcare patient trajectory data. We plan to improve our work with several experiments, as outlined in Appendix A.5. Moreover, we pursue interesting extensions for our framework following the end-to-end goal-directed generation of personalized treatment strategies, as outlined in A.2.9.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Harry Emerson, Matthew Guy, and Ryan McConville. Offline reinforcement learning for safer blood glucose control in people with type 1 diabetes. *Journal of Biomedical Informatics*, 142:104376, 2023.

Mehrad Jaloli and Marzia Cescon. Long-term prediction of blood glucose levels in type 1 diabetes using a cnn-lstm-based deep neural network. *Journal of Diabetes Science and Technology*, page 19322968221092785, 2022.

Mahsa Oroojeni Mohammad Javad, Stephen Olusegun Agboola, Kamal Jethwani, Abe Zeid, Sagar Kamarthi, et al. A reinforcement learning–based method for management of type 1 diabetes: exploratory study. *JMIR diabetes*, 4(3):e12905, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Xiran Liu, Ivana Jankovic, and Jonathan H Chen. Predicting inpatient glucose levels and insulin dosing by machine learning on electronic health records. *medRxiv*, pages 2020–03, 2020.

Amina Mollaysa, Brooks Paige, and Alexandros Kalousis. Conditional generation of molecules from disentangled representations. 2019.

Amina Mollaysa, Brooks Paige, and Alexandros Kalousis. Goal-directed generation of discrete structures with conditional generative models. *Advances in Neural Information Processing Systems*, 33:21923–21933, 2020.

Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

Giulia Noaro, Giacomo Cappon, Martina Vettoretti, Giovanni Sparacino, Simone Del Favero, and Andrea Facchinetti. Machine-learning based model to improve insulin bolus calculation in type 1 diabetes therapy. *IEEE Transactions on Biomedical Engineering*, 68(1):247–255, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Manuel Schürch, Dario Azzimonti, Alessio Benavoli, and Marco Zaffalon. Recursive estimation for sparse gaussian process regression. *Automatica*, 120:109127, 2020.

Manuel Schürch, Dario Azzimonti, Alessio Benavoli, and Marco Zaffalon. Correlated product of experts for sparse gaussian process regression. *Machine Learning*, pages 1–22, 2023.

Renat Sergazinov, Mohammadreza Armandpour, and Irina Gaynanova. Gluformer: Transformer-based personalized glucose forecasting with uncertainty quantification. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

Mark Shifrin and Hava Siegelmann. Near-optimal insulin treatment for diabetes patients: a machine learning approach. *Artificial Intelligence in Medicine*, 107:101917, 2020.

Miguel Tejedor, Ashenafi Zebene Woldaregay, and Fred Godtliebsen. Reinforcement learning application in diabetes blood glucose control: A systematic review. *Artificial intelligence in medicine*, 104:101836, 2020.

Jakub M. Tomczak. Deep Generative Modeling. *Deep Generative Modeling*, pages 1–197, 1 2022. doi: 10.1007/978-3-030-93158-2.

Cécile Trottet, Manuel Schürch, Amina Mollaysa, Ahmed Allam, and Michael Krauthammer. Generative time series models with interpretable latent processes for complex disease trajectories. In *Deep Generative Models for Health Workshop NeurIPS 2023*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Jinyu Xie and Qian Wang. Benchmarking machine learning algorithms on blood glucose prediction for type i diabetes in comparison with classical time-series models. *IEEE Transactions on Biomedical Engineering*, 67(11):3101–3124, 2020.

Taiyu Zhu, Kezhi Li, Lei Kuang, Pau Herrero, and Pantelis Georgiou. An insulin bolus advisor for type 1 diabetes using deep reinforcement learning. *Sensors*, 20(18):5058, 2020.

A Details and Extension about Model

A.1 Notation

In this section, we provide more precise notation. In particular, we consider multivariate time series data, including outcome $\boldsymbol{y}_{1:T_y} \in \mathcal{Y} = \mathbb{R}^{P \times T_y}$, treatment $\boldsymbol{x}_{1:T_y} \in \mathcal{X} = \mathbb{R}^{D \times T_x}$, and covariate $\boldsymbol{v}_{1:T_v} \in \mathbb{R}^{V \times T_v}$ trajectories. For each time series, we assume irregularly sampled times $\boldsymbol{\tau}^y_{1:T_y}$, $\boldsymbol{\tau}^x_{1:T_x}$, and $\boldsymbol{\tau}^v_{1:T_v}$, respectively. For the sake of simplicity, we assume a fixed future time window of length $K \in \mathbb{N}$ (e.g. 24 hours) and divide the trajectories into past and future, that is, $\boldsymbol{y}_{1:T_y} = [\bar{\boldsymbol{y}}, \boldsymbol{y}]$, $\boldsymbol{x}_{1:T_x} = [\bar{\boldsymbol{x}}, \boldsymbol{x}]$, and $\boldsymbol{v}_{1:T_v} = [\bar{\boldsymbol{v}}, \boldsymbol{v}]$, where $\boldsymbol{y} \in \mathbb{R}^{P \times K_y}$, $\boldsymbol{x} \in \mathbb{R}^{D \times K_x}$, and $\boldsymbol{v} \in \mathbb{R}^{V \times K_v}$, compare also Figure 1 on the left. Note that, in the implementation, we consider a moving future window. We define the personalized history $\boldsymbol{h} = \{\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{v}}, \boldsymbol{s}\}$, where $\boldsymbol{s}$ are some additional non-temporal patient information. Moreover, we define the context $\boldsymbol{c} = [\boldsymbol{h}, \boldsymbol{v}]$ for the sake of convenience. We assume a dataset $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$ consisting of N historical patient time series $\mathcal{D}_i = \{\boldsymbol{y}^i_{1:T^i_y}, \boldsymbol{x}^i_{1:T^i_x}, \boldsymbol{v}^i_{1:T^i_v}, \boldsymbol{s}^i\}$, however, we omit the explicit dependency to i if it is clear from the context.

A.2 Decision Making

A.2.1 Expected Utility

Suppose we have an utility function $u(\boldsymbol{y}) : \mathcal{Y} \to \mathbb{R}$ for a random variable $\boldsymbol{y} \in \mathcal{Y}$ with density $p(\boldsymbol{y})$. We define the expected utility $\mathbb{U}_{p(\boldsymbol{y})} \in \mathbb{R}$ as

$$\mathbb{U}_{p(\boldsymbol{y})} = \mathbb{E}_{p(\boldsymbol{y})}\left[u(\boldsymbol{y})\right] = \int p(\boldsymbol{y}) u(\boldsymbol{y}) d\boldsymbol{y}.$$

A.2.2 Conditional Expected Utility

Similarly, for a conditional distribution $p(\boldsymbol{y}|\boldsymbol{x})$ conditioned on $\boldsymbol{x} \in \mathcal{X}$, we define the conditional expected utility $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x})$ as

$$\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[u(\boldsymbol{y})\right] = \int p(\boldsymbol{y}|\boldsymbol{x}) u(\boldsymbol{y}) d\boldsymbol{y}, \tag{1}$$

which can be considered as a function in $\boldsymbol{x}$.

A.2.3 Joint Expected Utility

For a joint distribution $p(\boldsymbol{y}, \boldsymbol{x})$, we define the joint expected utility $\mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x})}$ as

$$\mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x})} = \mathbb{E}_{p(\boldsymbol{y},\boldsymbol{x})}\left[u(\boldsymbol{y},\boldsymbol{x})\right] = \int p(\boldsymbol{y},\boldsymbol{x})u(\boldsymbol{y},\boldsymbol{x})d\boldsymbol{y}\, d\boldsymbol{x} = \int p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})u(\boldsymbol{y},\boldsymbol{x})d\boldsymbol{y}\, d\boldsymbol{x}.$$

Here, we introduce the joint utility function $u(\boldsymbol{y}, \boldsymbol{x}) : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$, which also expresses the utility for the conditional variable $\boldsymbol{x}$ (treatment strategies). In the case of a factorized joint utility function $u(\boldsymbol{y}, \boldsymbol{x}) = u(\boldsymbol{y})u(\boldsymbol{x})$, we can write

$$\mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x})} = \int p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})u(\boldsymbol{y})u(\boldsymbol{x})d\boldsymbol{y}\, d\boldsymbol{x} = \mathbb{E}_{p(\boldsymbol{x})}\left[u(\boldsymbol{x})\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x})\right],$$

which can be decomposed as the expectation over the conditional variables (treatments) of the weighted conditional expected utility. For a utility function $u(\boldsymbol{y}, \boldsymbol{x}) = u(\boldsymbol{y})$, this simplifies to $\mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x})} = \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x})\right]$.

A.2.4 Hierarchical Conditional Utility

We can also extend the introduced concepts to more variables, for instance, conditioning on a context variable $\boldsymbol{c}$, so that the expected utilities become $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{c})}(\boldsymbol{c})$, $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})}(\boldsymbol{x}, \boldsymbol{c})$, and $\mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{c})}(\boldsymbol{c})$, which are all functions in $\boldsymbol{c}$.

A.2.5 Multivariate Utility Functions

Suppose $\mathcal{Y} = \mathbb{R}^P$. We can define a factorized utility function $u(\boldsymbol{y}) : \mathbb{R}^P \to \mathbb{R}$

$$u(\boldsymbol{y}) = \prod_{p=1}^{P} \alpha_p u(y_p)$$

with weights α_p for each dimension. This is particularly interesting when modeling concurring outcomes, for instance, the success of treatment together with complications or side effects. This allows specifying the different utilities for multiple outcomes and enables the analysis of treatments that satisfy Pareto-optimality regarding multiple outcomes.

A.2.6 Temporal Expected Utility

Suppose $\mathcal{Y} = \mathbb{R}^{P\times T}$, that is a temporal indexed multivariate random variable $\boldsymbol{y} = \boldsymbol{y}_{1:T} \in \mathbb{R}^{P\times T}$. We can directly define the utility function over the whole matrix. For instance,

$$u(\boldsymbol{y}_{1:T}) = \sum_{t=1}^{T} \gamma^t u(\boldsymbol{y}_t) = \sum_{t=1}^{T} \gamma^t \prod_{p=1}^{P} \alpha_p u(y_{pt})$$

with time decaying parameter $0 < \gamma \leq 1$. In this case, the expected utility becomes

$$\mathbb{U}_{p(\boldsymbol{y}_{1:T})} = \mathbb{E}_{p(\boldsymbol{y}_{1:T})}\left[u(\boldsymbol{y}_{1:T})\right] = \int p(\boldsymbol{y}_{1:T})u(\boldsymbol{y}_{1:T})d\boldsymbol{y}_{1:T} = \int p(\boldsymbol{y}_{1:T}) \sum_{t=1}^{T} \gamma^t u(\boldsymbol{y}_t)d\boldsymbol{y}_{1:T}. \quad (2)$$

If we assume temporal independence, that is, $p(\boldsymbol{y}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t)$, this leads to

$$\mathbb{U}_{p(\boldsymbol{y}_{1:T})} = \int \sum_{t=1}^{T} p(\boldsymbol{y}_t)\gamma^t u(\boldsymbol{y}_t)d\boldsymbol{y}_t = \sum_{t=1}^{T} \gamma^t \mathbb{U}_{p(\boldsymbol{y}_t)}.$$

A.2.7 Approximation by Generative Models

All these integrals are often hard to compute for interesting objects such as complex multivariate time series. On the other hand, deep generative probabilistic models are powerful for generating conditional samples from very complex distributions over objects including multivariate time series efficiently. Thus, we propose to generate multiple conditional samples

$$\boldsymbol{y}_{|\boldsymbol{x}}^{(1)}, \ldots, \boldsymbol{y}_{|\boldsymbol{x}}^{(s)}, \ldots, \boldsymbol{y}_{|\boldsymbol{x}}^{(S)} \sim p(\boldsymbol{y}|\boldsymbol{x})$$

with deep generative models to obtain a Monte-Carlo-based approximation for the expected utilities. For instance, the expected utility $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x}) = \int p(\boldsymbol{y}|\boldsymbol{x})u(\boldsymbol{y})d\boldsymbol{y}$ in Equation 1 is approximated by

$$\hat{\mathbb{U}}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x}) = \frac{1}{S}\sum_{s=1}^{S} u(\boldsymbol{y}_{|\boldsymbol{x}}^{(s)}).$$

In the case of multivariate time series $\boldsymbol{y} = \boldsymbol{y}_{1:T} \in \mathbb{R}^{P\times T}$, we jointly sample entire trajectories $[\boldsymbol{y}_1^{(s)}, \ldots, \boldsymbol{y}_T^{(s)}] = \boldsymbol{y}_{1:T}^{(s)} \sim p(\boldsymbol{y}_{1:T})$, so that the sampling version of Equation 2 becomes

$$\hat{\mathbb{U}}_{p(\boldsymbol{y}_{1:T})} = \frac{1}{S}\sum_{s=1}^{S} u(\boldsymbol{y}_{1:T}^{(s)}) = \frac{1}{S}\sum_{s=1}^{S}\sum_{t=1}^{T} \gamma^t u(\boldsymbol{y}_t^{(s)}).$$

Similarly, we can estimate $\hat{\mathbb{U}}_{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:T})}$ for conditioning on a whole treatment trajectory $\boldsymbol{x} = \boldsymbol{x}_{1:T} \in \mathbb{R}^{D\times T}$.

Moreover, we want to emphasize that we can take gradients with respect to the parameters of the deep parametrized distributions based on end-to-end optimization. Sampling can be included via the re-parametrization trick.

A.2.8 Exact Computation

Under certain circumstances, the expected utility $\mathbb{U}_{p(\boldsymbol{y}|\boldsymbol{x})}(\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[u(\boldsymbol{y})\right] = \int p(\boldsymbol{y}|\boldsymbol{x})u(\boldsymbol{y})d\boldsymbol{y}$ in Equation (1) can be computed exactly. It can be used as baseline to compare the approximation with deep generative models as discussed in the previous section. For instance, if we assume a joint Gaussian distribution or a Gaussian Process (GP) [Williams and Rasmussen, 2006, Schürch et al., 2020, 2023] for $p(\boldsymbol{y}|\boldsymbol{x})$ and a simple utility function $u(\boldsymbol{y})$ (for instance affine, exponential, periodic, Gaussian, log-Gaussian, ...), then the expected utility can be computed exactly. This has the benefit that the variance from the two-stage sampling of the response and treatment can be reduced by only sampling the treatments.

A.2.9 Improved Treatment Generation

In order to directly learn a better distribution $p_\theta(\boldsymbol{x}|\boldsymbol{c})$ which take into account desired properties regarding optimal treatment responses, we propose to learn the

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{c})} = \underset{\theta}{\operatorname{argmax}} \int p_\theta(\boldsymbol{x}|\boldsymbol{c})p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})u(\boldsymbol{y})d\boldsymbol{y}\, d\boldsymbol{x},$$

so that we can then directly take the most probable treatment

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p_{\theta^*}(\boldsymbol{x}|\boldsymbol{c})$$

or a few high-probability samples from $p_{\theta^*}(\boldsymbol{x}|\boldsymbol{c})$, which are now coupled so that they satisfy also the utility constraints. This idea is based on the work of Mollaysa et al. [2020].

This approach can also be combined with log-likelihood-based inference, so that the objective function becomes

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}}\, \alpha\mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{c})} + (1-\alpha)\mathcal{L}(\theta;\mathcal{D}) \\ &= \underset{\theta}{\operatorname{argmax}}\, \alpha \int p_\theta(\boldsymbol{x}|\boldsymbol{c})p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})u(\boldsymbol{y})d\boldsymbol{y}\, d\boldsymbol{x} + (1-\alpha)\log p_\theta(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{v},f(\boldsymbol{h})),\end{aligned}$$

where we showed the parametric mode for the sake of simplicity.

A.2.10 Learning Utility for x

Alternatively, we could also learn the parameters π of the utility function $u_\pi(\boldsymbol{x})$ over the treatments $\boldsymbol{x}$ with deep learning, that is,

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{U}_{p(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{c})} = \underset{\pi}{\operatorname{argmax}} \int p_\theta(\boldsymbol{x}|\boldsymbol{c})p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{c})u(\boldsymbol{y})u_\pi(\boldsymbol{x})d\boldsymbol{y}\, d\boldsymbol{x},$$

which constitutes a pragmatic approach for generating efficient treatment suggestions that result in favorable outcomes.

A.3 Deep Generative Model

In this section, we present more details of our approach to learn the conditional joint distribution

$$p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{c}) = p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c}) p_\theta(\boldsymbol{x}|\boldsymbol{c})$$

with deep generative time series models [Tomczak, 2022, Murphy, 2022] from retrospective patient trajectories data, where we introduce the learnable parameters $\pi = \{\phi, \theta\}$. Among the many choices for deep generative models, a common property is to implicitly learning the joint distribution with deep neural networks by providing a mechanism to generate samples from these distributions. In particular, we focus on an encoder-decoder architecture based on a transformer [Vaswani et al., 2017] as illustrated in Figure 3 on the left.

A.3.1 Encoder of Personalized History

We consider either a deterministic or probabilistic encoder $p_\psi(\boldsymbol{z}|\boldsymbol{h})$. In the former, the personalized history $\boldsymbol{h}$ is mapped to a fixed latent representation $\boldsymbol{z} = f_\psi(\boldsymbol{h}) \in \mathbb{R}^{L \times (T-K))}$, whereas in the latter we learn a deep parametrized probability distribution

$$p_\psi(\boldsymbol{z}|\boldsymbol{h}) = \prod_{t=1}^{T-K} \prod_{l=1}^{L} \mathcal{N}\left(z_{tl}|\mu_\psi^{tl}(\boldsymbol{h}), \sigma_\psi^{tl}(\boldsymbol{h})\right),$$

with deep neural networks $\mu_\psi^{tl}(\boldsymbol{h})$ and $\sigma_\psi^{tl}(\boldsymbol{h})$.

A.3.2 Outcome Predictor

The future multivariate outcome trajectory $\boldsymbol{y} \in \mathbb{R}^{P \times K}$ is parametrized as

$$p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}) = \prod_{t=1}^{K} \prod_{p=1}^{P} \mathcal{N}\left(y_{tp}|\mu_\phi^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}), \sigma_\phi^{tp}\right),$$

where the mean $\mu_\phi^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})$ and variance σ_ϕ^{tp} are learned with neural networks. Although this likelihood is a parametric distribution, the final distribution of the outcomes can be arbitrarily complex, see Section 3.2.1.

A.3.3 Treatment Predictor

For the treatment strategies $\boldsymbol{x} \in \mathbb{R}^{D \times K}$, we assume a Poisson likelihood

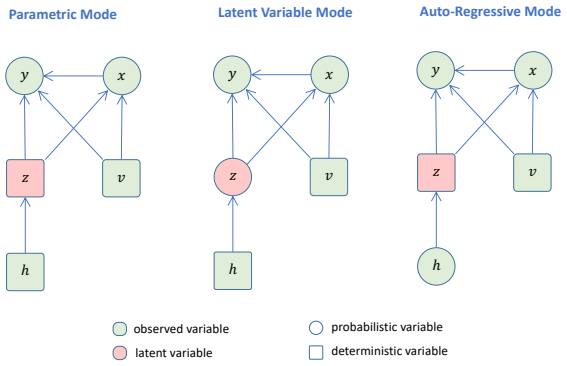
$$p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z}) = \prod_{t=1}^{K} \prod_{d=1}^{D} \mathcal{P}\left(x_{td}|\lambda_\theta^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})\right),$$

with deep parametrized mean function $\lambda_\theta^{tp}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})$. Note that the final distribution of the deep generative model is non-parametric, see next section.

A.3.4 Source of Stochasticity

We compare our model in three different modes as illustrated in Figure A.3.4 for modeling the source of the stochasticity in the generation of outcome $\boldsymbol{y}$ and treatment $\boldsymbol{x}$ trajectories. As a baseline method, we use a deterministic encoder and a simple parametric distribution for the outcomes $\boldsymbol{y}$ and treatments $\boldsymbol{x}$ without any additional stochasticity in the model, so that the the predictive distributions are actually Gaussian and Poisson distributed as defined in Sections A.3.2 and A.3.3. However, these distributions are too restrictive and do not match real-world data (Appendix A.6). As a second mode, we use a latent variable model with a probabilistic encoder (A.3.1), so that when sampling from the latent representation, we get an infinite mixture of Gaussian or a mixture of Poisson distributions, allowing to generate rather complex distributions. The third mode is based on auto-regressive learning and sampling, where we use $K = 1$ and plug back the previous values to the history $\boldsymbol{h}$, which is then the source of stochasticity in the sampling. This allows to sample multivariate trajectories beyond infinite mixture distributions.

Figure 4: Probabilistic Modes of Generative Model.



Parametric Mode: For a deterministic encoder, we have the joint distribution

$$p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{v}, f_\psi(\boldsymbol{h})) = p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, f_\psi(\boldsymbol{h}))p_\theta(\boldsymbol{x}|\boldsymbol{v}, f_\psi(\boldsymbol{h}))$$

and the objective function is the log-likelihood

$$\mathcal{L}_1(\pi; \mathcal{D}) = \log p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{v}, f_\psi(\boldsymbol{h})).$$

Latent Variable Mode: In the latent variable mode, we have the joint distribution

$$p_\pi(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{h}) = p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})p(\boldsymbol{z})$$

with prior distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \sigma_p^2\mathbb{I})$. The ideal objective function would be the *marginal* log-likelihood $\mathcal{M}(\pi; \mathcal{D}) = \log \int p_\pi(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{h})d\boldsymbol{z}$, which is not feasible to compute exactly. Therefore, we propose to use an amortized variational inference approach [Kingma and Welling, 2013, Blei et al., 2017], which maximizes a lower bound $\mathcal{L}_2(\pi; \mathcal{D}) \leq \mathcal{M}; \mathcal{D}(\pi)$, that is

$$\mathcal{L}_2(\pi; \mathcal{D}) = \mathbb{E}_{p_\psi(\boldsymbol{z}|\boldsymbol{h})}\left[\log p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})\right] + \mathbb{E}_{p_\psi(\boldsymbol{z}|\boldsymbol{h})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})\right] - KL\left[p_\psi(\boldsymbol{z}|\boldsymbol{h})||p(\boldsymbol{z})\right].$$

Auto-Regressive Mode: In the auto-regressive mode, we set the window size to $K = 1$ and we decompose the joint distribution as

$$\begin{aligned} p_\pi(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{v}, f_\psi(\boldsymbol{h})) &= \prod_{t=1}^{T} p_\pi(\boldsymbol{y}_t, \boldsymbol{x}_t|\boldsymbol{v}_t, f_\psi(\boldsymbol{h}_{1:t-1})) \\ &= \prod_{t=1}^{T} p_\phi\left(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{v}_t, f_\psi(\boldsymbol{h}_{1:t-1})\right) p_\theta\left(\boldsymbol{x}_t|\boldsymbol{v}_t, f_\psi(\boldsymbol{h}_{1:t-1})\right), \end{aligned}$$

where we define the temporal varying personalized history $\boldsymbol{h}_{1:t-1} = [\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t-1}, \boldsymbol{v}_{1:t-1}, \boldsymbol{s}]$. By maximizing the log-likelihood, we get the objective

$$\begin{aligned} \mathcal{L}_3(\pi; \mathcal{D}) &= \log \prod_{t=1}^{T} p_\pi(\boldsymbol{y}_t, \boldsymbol{x}_t|\boldsymbol{v}_t, f_\psi(\boldsymbol{h}_{1:t-1})) \\ &= \sum_{t=1}^{T} \log p_\phi\left(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{v}_t, f_\psi(\boldsymbol{h}_{1:t-1})\right) + \log p_\theta\left(\boldsymbol{x}_t|\boldsymbol{v}_t, f_\psi(\boldsymbol{h}_{1:t-1})\right). \end{aligned}$$

In the training, we can use the actually observed values $\boldsymbol{h}_{1:t-1} = [\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t-1}, \boldsymbol{v}_{1:t-1}, \boldsymbol{s}]$ without any stochasticity, which is called *teacher mode*. Only for the generation of novel trajectories, we auto-regressively sample from the model.

Optimization: In all three modes $\mathcal{L}_1(\pi; \mathcal{D})$, $\mathcal{L}_2(\pi; \mathcal{D})$, $\mathcal{L}_3(\pi; \mathcal{D})$, we optimize the parameters with a datasets $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$ by

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{i=1}^{N} \sum_{k=0}^{T_i - K} \mathcal{L}^i(\pi; \mathcal{D}_i^k),$$

which is computed with stochastic optimization using mini-batches of patients. Here, $\mathcal{D}_i^k$ refers to the data of patient i until time point k.

A.4 Implementation Details

A.4.1 Data Pre-processing

Since the time series is irregularly sampled, we convert it to an hourly sampled time series via imputation. The glucose level $\boldsymbol{x}$ is linearly imputed. The treatment $\boldsymbol{x}$ (i.e., basal and bolus insulin dose) is imputed with zero. We linearly scale $\boldsymbol{y}$ to have mean 0 and standard deviation 1. We also linearly scale $\boldsymbol{x}$ to have standard deviation 1.

A.4.2 Transformer architecture

We use the transformer encoder, decoder, and positional encoding from Vaswani et al. [2017]. We train with learning rate 10^{-3}, batch size 8, one decoder and encoder layer, 64 features, 16 heads, and a single-hidden-layer embedding of size 100 for the feedforward network of the encoder/decoder as well as for the embedding of the input variables. When training the outcome predictor $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z})$, we only compute the loss for the non-imputed measured future $\boldsymbol{y}$ values.

A.5 Planned Experiments

In this Section, we provide several planned experiments for our method.

In general, in order to assess the quality of our approach, we will compare the prediction accuracy and reliability (uncertainty) of the outcome and treatments against the true future in the retrospective data. Moreover, to assess the quality of the generated samples, we will implement a simple classifier to distinguish whether it is a real or generated sample.

A.5.1 Outcome Trajectory Generation

In our model, we will examine how the personalized history $\boldsymbol{h}$, the covariates $\boldsymbol{v}$ and the treatment $\boldsymbol{x}$ affect the prediction performance for $\boldsymbol{y}$. In particular, we will compare different conditional sets such as $p_\phi(\boldsymbol{y})$, $p_\phi(\boldsymbol{y}|\boldsymbol{\tau_y})$, $p_\phi(\boldsymbol{y}|\bar{\boldsymbol{y}})$, $p_\phi(\boldsymbol{y}|\bar{\boldsymbol{x}})$, $p_\phi(\boldsymbol{y}|\boldsymbol{h})$, $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{h})$, $p_\phi(\boldsymbol{y}|\boldsymbol{v}, \boldsymbol{h})$, and $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{h})$.

A.5.2 Treatments Strategy Generation

Similarly, we compare the samples of different distributions of treatment trajectories, that is $p_\theta(\boldsymbol{x})$, $p_\theta(\boldsymbol{x}|\boldsymbol{v})$, $p_\theta(\boldsymbol{x}|\boldsymbol{h})$, and $p_\theta(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{h})$. It is particularly interesting to see the effect of the past history $\boldsymbol{h}$ as well as the effect from the carbohydrates $\boldsymbol{v}$.

A.5.3 Comparison of Modes

We will compare the three different modes of stochasticity in the generative model, as explained in Section 3.2.1. It will be interesting to see whether the parametric, the latent variable, or the auto-regressive approach leads to the best performance. Besides comparing the performance, it will be particularly interesting to compare the quality of the generated multivariate samples of the trajectories. Moreover, we will examine the latent space of the latent variable approach. In particular, we try to visualize the learned latent space of the treatment trajectories and check if the interpolation property is satisfied. Moreover, we check if we can generate samples around particular interesting points in the latent space.

A.5.4 Comparison with other Approaches

We will compare our approach with other prediction models for the outcomes and treatments, such as deterministic deep neural networks (RNN, CNN). Moreover, it would be interesting to find a setting in which we can compare our generated treatment with approaches from reinforcement learning and compare their quality. In particular, it would be interesting to compare how realistic and consistent our multivariate samples are.

A.5.5 Generalization to other Applications

We plan to test our general framework for generating personalized and optimal treatment strategies to other diseases. For instance, we plan to apply it to cancer treatment data as well as rheumatic disease data [Trottet et al., 2023].

A.6 Descriptive Data Analysis

In the following, we describe the blood glucose, carbohydrate intake, basal insulin, and bolus insulin data. We plot their typical values/doses (throughout the day) as well as the number of observations/insulin injections per day. We would like to empathize that the shown results show global distributions involving all patients, whereas the personalized distributions of the trajectories of treatment outcomes are mostly multi-modal, often due to the latent, unobserved variables such as carbohydrates or medications.

Figure 5: Blood glucose is mostly measured around 3-4 times per day and usually takes values between 5 and 15.

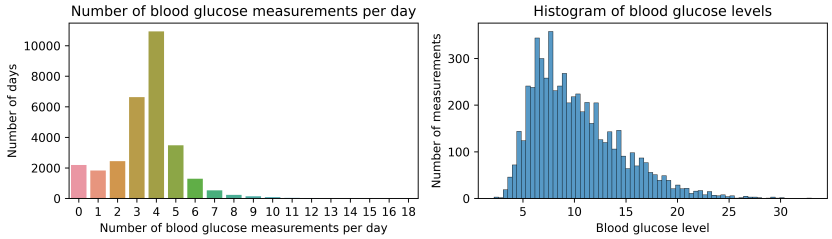


Figure 6: Blood glucose is usually measured at 7-8 a.m., 12 p.m., 6 p.m., or 10 p.m..

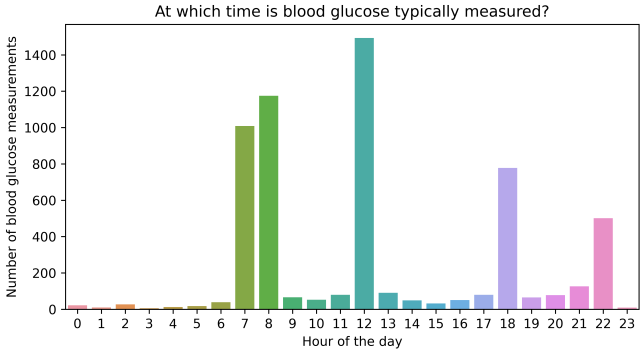


Figure 7: Blood glucose levels exhibit fluctuations throughout the day, with their lowest point typically occurring around 7 a.m.

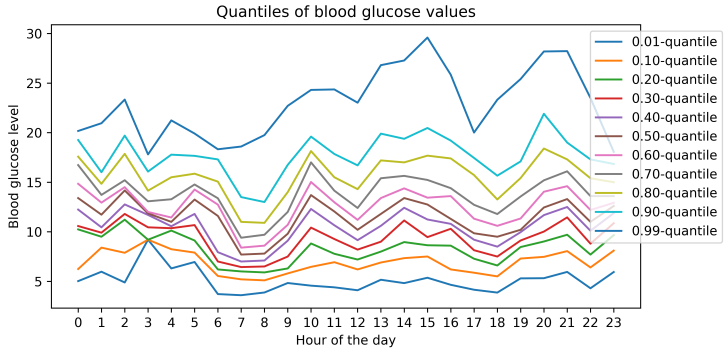


Figure 8: Carbohydrate intake data is usually not provided. If it is provided, then usually 3 times per day. Carbohydrate consumption per meal mostly ranges from 20 to 70.

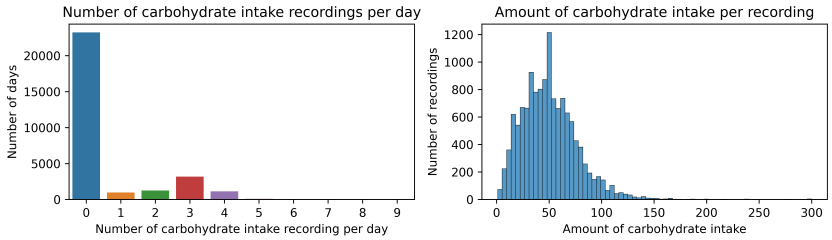


Figure 9: Carbohyrdates are usually consumed at 7-8 a.m., 12 p.m., or 6 p.m..

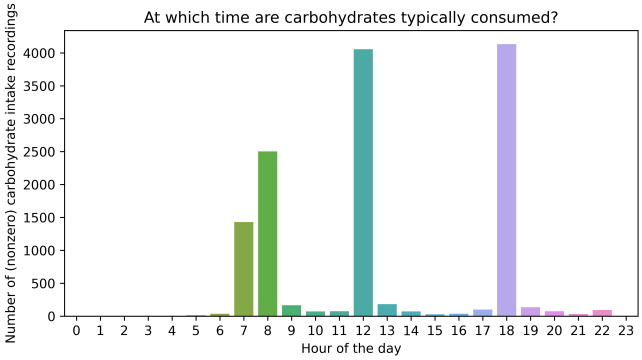


Figure 10: Carbohydrate intake throughout the day. If carbohydrates are consumed and this is reported, then the amount of carbohydrates does not depend on the time of the day.

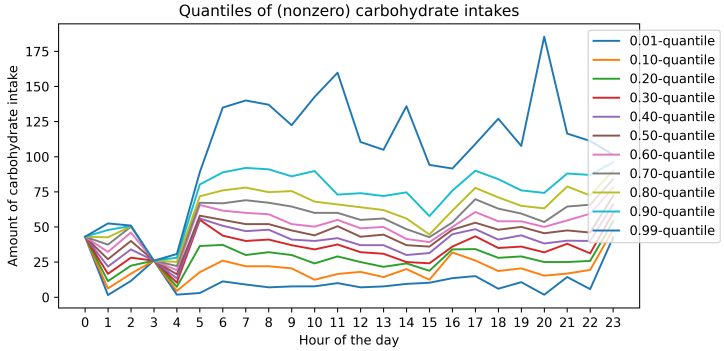


Figure 11: Basal insulin is usually injected once per day or not at all. The dose per injection is around 2-20.

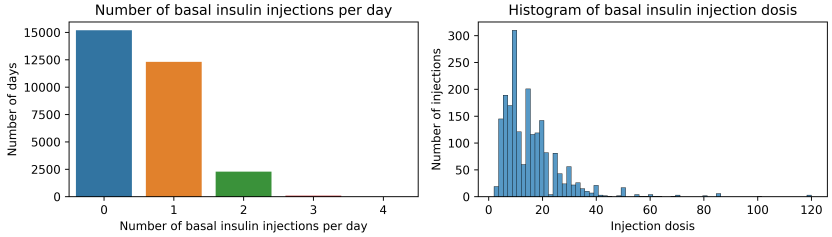


Figure 12: Basal insulin injection almost exclusively occurs at 7–8 a.m. or 6 p.m.

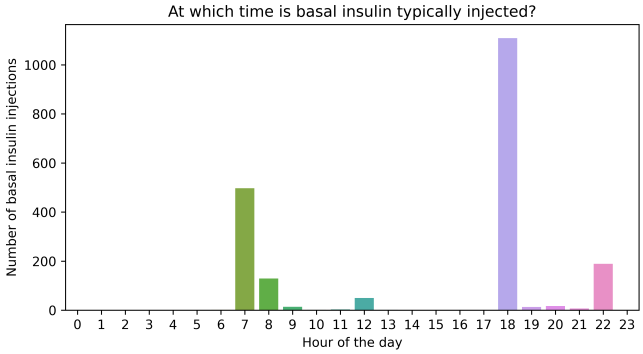


Figure 13: The basal insulin injection dose is independent of the time of the day.

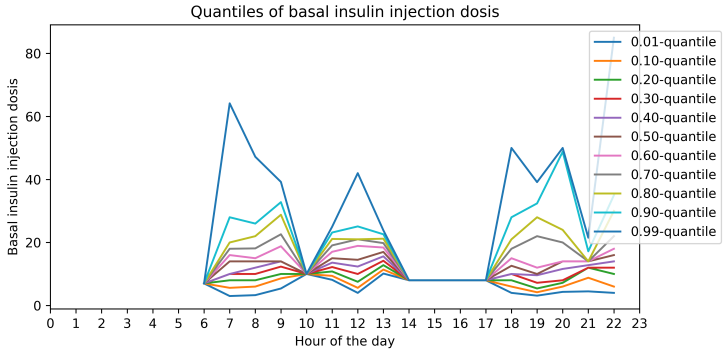


Figure 14: Bolus insulin is usually injected 0–4 times per day. The dose per injection is around 0–30.

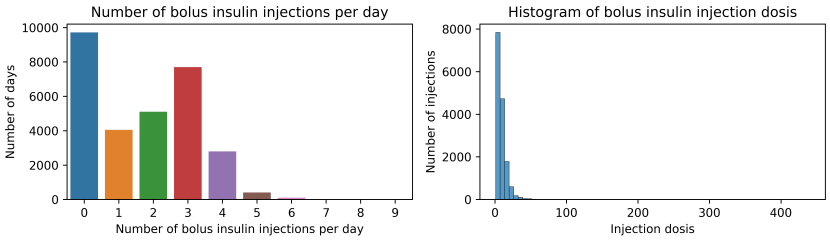


Figure 15: If bolus insulin is injected, then this occurs at 7–8 a.m., 12 a.m., 6 p.m., and, less frequently, at 10 p.m.

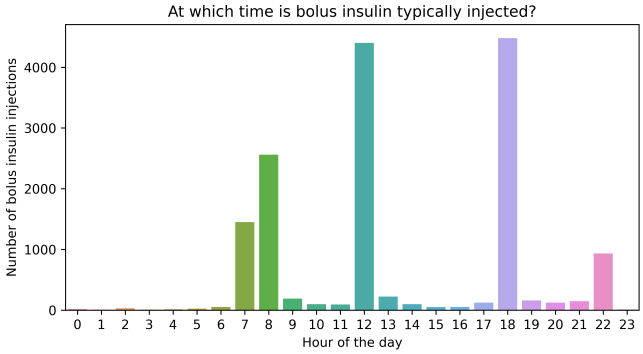


Figure 16: Bolus insulin injection doses throughout the day. Injections at 10 p.m. almost always have the same low dose. The injection dose at other times is higher and varies more.

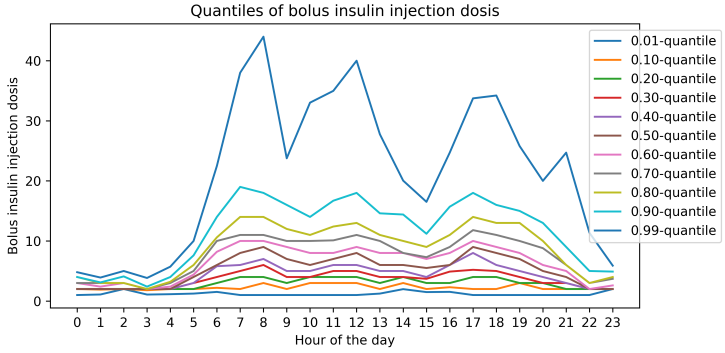
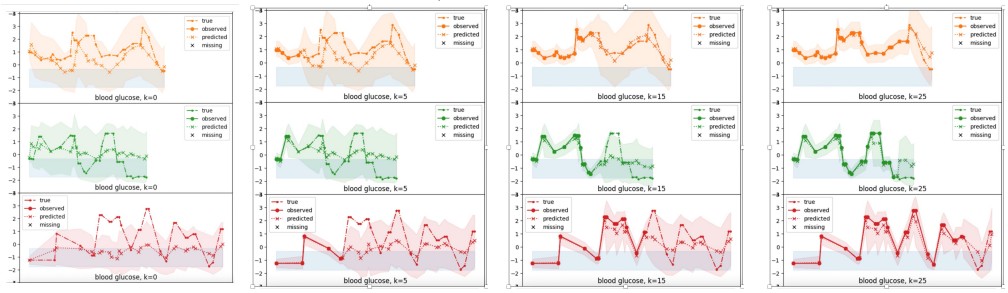


Figure 17: Probabilistic online prediction $p_\phi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})$ for different splits of past and future windows.



Model	**MAE**	**RMSE**
Baseline: Past average glucose level of the patient	2.671	3.671
Baseline: Average glucose level of all patients at this time	3.049	3.927
Transformer (using glucose data only)	1.840	2.625
Transformer (using glucose, insulin, carbs data)	1.789	2.616

Table 1: Mean performance of the y-prediction network across 30 random 50/50 train-validation splits. Absolute Error (MAE) and Root Mean Squared Error (RMSE) for baseline and transformer models.