Hybrid^2 Neural ODE Causal Modeling and an Application to Glycemic Response

Bob Junyi Zou [1] **Matthew E. Levine** [2] **Dessi P. Zaharieva** [3] **Ramesh Johari** [4] **Emily B. Fox** [5 6]

Abstract

Hybrid models composing mechanistic ODE-based dynamics with flexible and expressive neural network components have grown rapidly in popularity, especially in scientific domains where such ODE-based modeling offers important interpretability and validated causal grounding (e.g., for counterfactual reasoning). The incorporation of mechanistic models also provides inductive bias in standard blackbox modeling approaches, critical when learning from small datasets or partially observed, complex systems. Unfortunately, as the hybrid models become more flexible, the causal grounding provided by the mechanistic model can quickly be lost. We address this problem by leveraging another common source of domain knowledge: *ranking* of treatment effects for a set of interventions, even if the precise treatment effect is unknown. We encode this information in a *causal loss* that we combine with the standard predictive loss to arrive at a *hybrid loss* that biases our learning towards causally valid hybrid models. We demonstrate our ability to achieve a win-win, state-of-the-art predictive performance *and* causal validity, in the challenging task of modeling glucose dynamics post-exercise in individuals with type 1 diabetes.

1. Introduction

In many scientific and clinical domains, an influx of high resolution sensing data has brought the promise of more refined and informed scientific discovery and decision-making. The motivating example we consider in this paper is type 1 diabetes (T1D) management, where continuous glucose monitoring (CGM) and smart insulin pumps are revolutionizing care by offering near-real-time insights into physiological states (Tauschmann et al., 2022). To provide data-driven management strategies, there is a pressing need for interpretable models that not only make **accurate predictions**, but also **grasp the causal mechanisms** behind physiological responses (e.g., for counterfactual reasoning over various potential interventions). The target of causally-grounded, performant models is critical in many scientific disciplines ranging from astronomy to cell biology to neuroscience; in many of these settings, we are faced with only partial or indirect observations of a complex physiological or physical process we aim to reason about.

Blackbox sequence models have demonstrated extraordinary performance in numerous fields, ranging from natural language processing to forecasting. Popular methods include recurrent neural networks (RNNs), such as long short-term memory networks (LSTMs) (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRUs) (Chung et al., 2014); (temporal) convolutional neural networks (CNNs) (Lea et al., 2016; Bai et al., 2018; Shi et al., 2023); Transformer models, including Informer (Zhou et al., 2021) and Autoformer (Wu et al., 2021); and state space sequence models, such as S4 (Gu et al., 2022b). Despite their transformative role in many fields, the application of these models to various scientific domains has encountered challenges. An obvious hurdle is the limited size of some scientific datasets. But even in the presence of lots of data, these methods are still crippled by the incomplete nature of what can be measured. This problem is exacerbated by the fact that most data is *observational*: blackbox models are adept at identifying associations leading to good predictions, but may not learn causally coherent models. For example, in our T1D setting, insulin delivery frequently coincides with a planned meal or snack, which initially causes glucose to rise; the model may incorrectly infer that insulin causes glucose to rise, when the reality is the opposite.

Sophisticated mechanistic models—specified via a set of ordinary differential equations (ODEs)—remain a preferred method in many scientific domains, as they capture lab-validated or otherwise known physical or physiological causal properties of the system. Examples include cardiac and renal modeling (Hilgemann & Noble, 1987; ten Tuss-

[1]Institute for Computational and Mathematical Engineering, Stanford University [2]Broad Institute of MIT and Harvard [3]Department of Pediatrics, Stanford University [4]Department of Management Science and Engineering, Stanford University [5]Department of Statistics and Department of Computer Science, Stanford University [6]Chan Zuckerberg Biohub – San Francisco. Correspondence to: Emily B. Fox <ebfox@stanford.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

cher et al., 2004; Marsh et al., 2005; Passini et al., 2017), immunology and viral kinetics (Perelson et al., 1996; Canini & Perelson, 2014), epidemiology (He et al., 2020), neural circuits (Hodgkin & Huxley, 1952; Ladenbauer et al., 2019), and pharmacokinetics (Holz & Fahr, 2001). The UVA/Padova simulator (Man et al., 2014) of insulin-glucose dynamics we consider in this paper has been FDA approved as a substitute for pre-clinical trials in the development of artificial pancreas algorithms. Notably, sophisticated mechanistic models often introduce additional states to capture intricate, non-Markovian dynamics such as delays. While this approach preserves the tractability of linear ODEs, it can lead to over-parameterization. The latest version of the UVA/Padova simulator has over 30 states. Yet, these models fail to capture important dynamics observed in real-world data; moreover, their parameter inference is highly sensitive. Further limiting this class of models is the typical assumption of a fixed mechanistic structure with a static set of simulator parameters. In T1D, many unobserved or partially-observed time-varying factors affect glycemic responses, including stress, hormone cycles, sleep, and activity levels (cf., Wellen et al., 2005). Likewise, the causal mechanism governing glycemic responses can vary—e.g., during certain types of physical activity, multiple underlying processes (not modeled in UVA/Padova) are activated based on available energy sources (McArdle et al., 2006).

Given these limitations, we focus on an alternative approach that hybridizes machine learning (ML) with domain knowledge encoded in mechanistic models. These *hybrid models* have gained traction across the natural and physical sciences (Willard et al., 2022), while being given different names and interpretations such as *graybox modeling* (Rico-Martinez et al., 1994), *physics-informed machine learning* (Karniadakis et al., 2021), *universal differential equations* (Rackauckas et al., 2020) and *neural closure learning* (Gupta & Lermusiaux, 2021). The core idea of hybrid modeling is to infuse domain inductive bias such that the trained model can gain the best of both worlds—mechanistic rigor with the flexibility and expressivity of deep learning. The hope is that hybrid methods not only allow one to solve complex modeling problems with improved precision and accuracy (Pathak et al., 2018; Willard et al., 2022), but also reduce the demand on data (Rackauckas et al., 2020), enhance reliability and robustness (Didona et al., 2015), and make the ML algorithms interpretable (Karniadakis et al., 2021; Du et al., 2019). Examples of successes of hybrid modeling in healthcare can be found in Qian et al. (2021); Sottile et al. (2021); Hussain et al. (2021).

Indeed, in Fig. 1 (left), we see the important inductive bias such mechanistic models provide. On the one hand, the pure mechanistic model makes poor predictions due to its brittle nature while on the other hand, the flexible blackbox models are subject to overfitting. The hybrid models generally outperform either alternative in terms of prediction error.

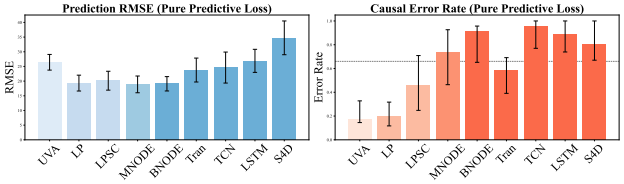


Figure 1. Prediction RMSE and standard error (*left*) vs. 10/50/90-th (lower/bar/upper) percentile classification error rate (*right*) for (1) UVA/Padova mechanistic model (UVA); (2) hybrid models: latent parameter dynamics (LP), latent parameter dynamics + state closure (LPSC), MNODE; and (3) blackbox models: neural ODE (BNODE), temporal convolutional network (TCN), LSTM, Transformer (Trans) and S4D, all trained on pure predictive loss ($\alpha = 0$). The dashed gray line in the second figure corresponds to the causal classification error rate of random guessing ($2/3$).

Unfortunately, as we also see in Fig. 1 (right), hybrid models can quickly lose their valid causal grounding as more and more flexible modeling components are deployed. When tasked with selecting the intervention with the largest treatment effect amongst a set of three counterfactual simulations (see Sec. 5), classification error for hybrid models gradually deteriorates as the model becomes more blackbox. Amongst the hybrid and blackbox models, we see that the majority have a classification error rate statistically indistinguishable from random guessing, 67%.

To address these challenges, we leverage another common source of domain knowledge: rankings of various treatment effects. While domain knowledge is often insufficient to specify expected treatment effects under counterfactually-applied interventions, we are still often able to make quantitative claims about relative magnitudes of different treatment effects. For example, while we may not know the precise treatment effect of eating a small salad versus a whole birthday cake (all else held constant), we do know the sign and relative scale of treatment effect. As this prior knowledge is challenging to encode in the hybrid model itself, we introduce a *causal loss* function that encourages our learned model to perform well in these comparison tasks. Our *hybrid loss* is a convex combination of predictive loss and causal loss. When applied to training hybrid models, we refer to the overall method as **hybrid^2 neural ODE causal modeling (H^{2}NCM)**. In a real-world task of guiding individuals with T1D on the impact of interventions so they can safely exercise, we show that our H^{2}NCM approach achieves the best of both worlds: state-of-the-art predictive accuracy with causal validity.

2. The Hybrid Modeling Spectrum

In this section, we provide background on hybrid modeling. For this paper, it is useful to think of hybrid modeling as a spectrum from pure mechanistic whitebox models to fully

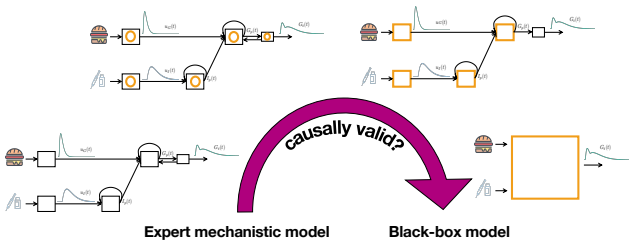


Figure 2. Visualization of hybrid modeling spectrum for a simple mechanistic model of insulin-glucose dynamics. From left to right: (1) mechanistic model; (2) latent parameter dynamics; (3) MNODE; (4) neural ODE. Dependencies between states are maintained until (4), with state dynamics increasingly flexible.

blackbox approaches, as illustrated in Fig. 2. There are two important components to the hybrid modeling spectrum: the degree to which neural networks learn the state dynamics and the degree to which the dependencies between states encoded in the mechanistic model are maintained.

The notion of a hybrid spectrum is an oversimplification as the space of possible hybrid models involves any number of combinations of hybridizations that are not easily ordered along a single axis. However, this framework is helpful for pedagogical purposes. Further, the hybrid models we focus on in our experiments *can* be ordered in terms of increasing model flexibility, which helps illustrate the risk that hybrid models can lose causal validity as we walk along this spectrum (see Fig. 1).

2.1. Mechanistic Models

On the far left-hand side of our hybrid modeling spectrum in Fig. 2 is the pure mechanistic model specified via a set of ordinary differential equations (ODE),

$$\frac{ds}{dt} = m\big(s(t);\ \beta\big), \tag{1}$$

where s is a vector representing the state, and β is a vector representing the simulator parameters. In applications where interventions or other external controls, x, are being applied to the observed dynamical system, we obtain a *controlled* ordinary differential equation (CDE),

$$\frac{ds}{dt} = m\big(s(t), x(t);\ \beta\big). \tag{2}$$

For simplicity, we still refer to such systems as ODEs, just as controlled state space models are often simply called state space models. Our focus will be on this controlled setting since our interest is in causal hybrid modeling to produce (valid) counterfactual simulations for an intervention x.[1] In Fig. 2, we introduce a cartoon version of such a (controlled) ODE for insulin-glucose dynamics.

[1]We use x to represent both possible controls or interventions, as well as exogenous covariates relevant to the state dynamics.

2.2. Neural ODE

In the absence of mechanistic domain knowledge, neural ODEs (Rico-Martinez et al., 1992; Weinan, 2017; Haber & Ruthotto, 2017; Chen et al., 2018; Kidger, 2021) have been proposed as a blackbox approach to learning such a dynamical system:

$$\frac{ds}{dt} = f(s(t), x(t);\ \theta). \tag{3}$$

Here, f denotes a neural network with parameters θ. Neural ODEs represent the far right side of the spectrum in Fig. 2. These methods have proven useful in many dynamical system modeling and simulation tasks, especially in the sciences where ODEs are a standard language for describing systems (Qian et al., 2021; Lu et al., 2021; Owoyele & Pal, 2022; Asikis et al., 2022; Li et al., 2022).

2.3. Hybrid Models

For our purposes, we introduce a general hybrid model that can be used to represent several different sub-methods:

$$\begin{aligned}\frac{ds}{dt} &= c_1 m\big(s(t), x(t);\ \beta(t)\big) + c_2 f_1\big(s(t), x(t), c_3 z(t);\ \theta\big)\\ \frac{dz}{dt} &= f_2\big(x(t), z(t);\ \theta\big)\\ \beta(t) &= \beta + c_4 f_3\big(x(t), z(t);\ \theta\big).\end{aligned} \tag{4}$$

This class of hybrid models can improve upon the nominal mechanistic model in Eq. (2) by introducing (i) time-varying parameters $\beta(t)$; (ii) a flexible correction term to the mechanistic ODE given by f_1; and (iii) a time-dependent latent state vector $z(t)$. We introduce constants $c_1, c_2, c_3, c_4 \in \{0, 1\}$ to "switch" between different sub-methods. If $c_1 = 1$ and $c_2 = c_3 = c_4 = 0$, we recover a fully mechanistic model. On the other hand, if $c_2 = 1$ and $c_1 = c_3 = c_4 = 0$, we obtain a blackbox neural ODE.

For other choices of these constants, we obtain various previous hybrid methods. When $c_1 = c_4 = 1$ and $c_2 = c_3 = 0$, we have a method that addresses infidelities of the mechanistic model solely via time-dependent parameters governed by latent dynamics z; we refer to this hybrid approach as *latent parameter dynamics*. In the context of glycemic modeling, Miller et al. (2020) use a deep state space model to capture time- and context-dependence of the simulator parameters. When $c_1 = c_2 = 1$, we learn a flexible correction f_1 to the mechanistic model in Eq. (2). We refer to this hybrid approach as *state closure*. If we also have $c_3 = 1$, we learn a correction f_1 that is additionally dependent on latent states $z(t)$. If instead we have $c_3 = c_4 = 0$, we learn a simple closure model f_1 which only depends on the mechanistic states $s(t)$ (Rico-Martinez et al., 1994).

Yin et al. (2021), Levine & Stuart (2022), and Karniadakis et al. (2021) provide related frameworks for defining the

space of hybrid models. The above schemes, and those further described in Levine & Stuart (2022); Karniadakis et al. (2021); Yin et al. (2021), are not mutually exclusive and are often combined (Qian et al., 2021; Wu et al., 2022; Wang et al., 2022).

Another important component of the mechanistic model that has been ignored to this point is the set of dependencies between states encoded by the mechanistic model. To make these dependencies explicit, and to explore hybrid models that—at a dynamical modeling level—maintain the same dependencies, we introduce the notion of connectivity of the state dynamics via a set of adjacency matrices A_s, A_x:

$$\begin{aligned}\frac{ds}{dt} &= c_1 m\big(s(t), x(t);\ A_s, A_x, \beta(t)\big) \\ &\quad + c_2 f_1\big(s(t), x(t), c_3 z(t);\ A_s, A_x, \theta\big) \\ \frac{dz}{dt} &= f_2\big(x(t), z(t);\ \theta\big) \\ \beta(t) &= \beta + c_4 f_3\big(x(t), z(t);\ \theta\big).\end{aligned} \tag{5}$$

Here the (i, j) entry of A_s is 1 if ds_i/dt is allowed to depend on $s_j(t)$, and similarly for A_x. In this way, the adjacency matrices A_s, A_x encode a structural equation model that constrains which state variables in s, x are allowed to influence each component of $\frac{ds}{dt}$. When $c_1 = c_2 = 1$, in contrast to the state closure model of Eq. (4), here the additive neural network correction is also forced to respect the dependencies between states.

Note that in Eq. (5), when $c_1 = 0$ but $c_2 = 1$, we maintain the dependencies between states (specified via A_s and A_x), but allow the resulting state dynamics to be fully learned via neural networks. We refer to this model as a *mechanistic neural ODE* (MNODE). MNODE represents the third illustrated model in the hybrid spectrum of Fig. 2.

3. Hybrid^2 Modeling

Hybrid models offer significant promise in many scientific domains where ODE-based mechanistic models are commonly deployed: they can provide important inductive bias, interpretability, and causal grounding as compared to their black-box alternatives; and, compared to pure mechanistic modeling, hybrid models can reduce bias through the flexibility of neural network components. However, as one walks along the hybrid modeling spectrum—from pure mechanistic to pure blackbox—these touted advantages can quickly disappear, as illustrated in Fig. 1. In particular, there is nothing constraining a hybrid model—trained on a predictive performance objective—to maintain the causal structure and interpretability of the original mechanistic model.

We address this challenge in Sec. 3.2 by introducing a *hybrid loss* that mixes predictive performance with causal validity. For the latter, we again lean on domain knowledge (as we did in our use of a mechanistic model), but this time cast as knowledge about the *direction* of treatment effects. In other words, we presume that we know in advance that applying, e.g., $2x$ instead of x to the system will increase (or decrease) the value of a score function that depends on the observed dynamic (counterfactual) response. Encoding this information in the loss function helps us achieve a win-win of increased modeling flexibility while biasing the learning towards solutions that match our causal knowledge.

3.1. Limitations of Hybrid Models

To illustrate the potential for hybrid models to lose causal validity, take the MNODE model of Eq. (5) where $c_1 = 0$. Relying solely on the mechanistic dependency graph between states while learning state dynamics can fail to distinguish parameters leading to correct directions of treatment effects. Consider data generated by a mechanistic ODE $ds/dt = -as + bx + c$, where a, b, c are positive. Here, the rate of change of s is monotonically increasing with x and therefore an intervention that increases x should have a positive treatment effect on s. However, after turning these dynamics into MNODE, we instead have $ds/dt = f(s, x; \theta)$; here, we lose the monotonicity with respect to x, unless a monotone neural network is intentionally used—which would severely constrain the flexibility of MNODE, the original motivation for adopting this modeling approach.

One might argue that while not explicitly encoded in the architecture, neural networks should be able to learn relatively simple signals like monotonicity from the data. Unfortunately, this is not the case when learning from noisy, observational datasets. For example, in T1D, the dosing of bolus insulin (negative effect on glucose) frequently occurs in close succession with a planned intake of carbohydrates (positive effect). Due to the confounding effect of insulin, the hybrid model may incorrectly infer that if an individual consumes carbohydrates, glucose drops; see Fig. 3. We provide an exploration of this effect in Sec. 5.4. The resulting dilemma is, while we do not want to make simplifying assumptions like linearity and monotonicity, we want to encourage our hybrid models to learn dynamics and interactions consistent with known signs of treatment effects.

3.2. The Hybrid Objective

Motivated by the preceding discussion, we encode causally relevant domain knowledge in hybrid models through the loss function itself, with the goal of biasing the learning towards causally valid models.

Preliminaries Let $X(\mathcal{T}) = \{x(t); t \in \mathcal{T}\}$ be the set of control inputs at times in $\mathcal{T}$, $\mathcal{T} \subset \mathbb{R}$. In our diabetes example, x can represent interventions such as carbohydrate intake or insulin dosing for a single patient. In our preced-

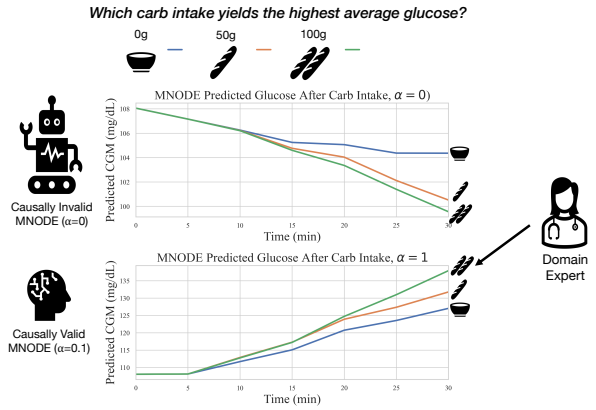


Figure 3. Counterfactual simulations of 3 levels of carbohydrate intake: none (blue), 50g (orange), 100g (green), comparing MNODE trained using predictive loss ($\alpha = 0$) and a hybrid loss ($\alpha = 0.1$). For $\alpha = 0$, MNODE incorrectly selects "none" as the intervention causing the largest resulting average blood glucose, because it incorrectly infers that carb intake decreases glucose. The model using $\alpha = 0.1$ correctly selects "100g", consistent with causally valid domain knowledge ($\mathcal{I}^* = \hat{\mathcal{I}}$).

ing development, we described models for the evolution of the full state $[s(t); z(t)]$. Here we assume we have a partial observation of that state; for example, in our diabetes case study, we observe glucose measurements via a continuous glucose monitor (CGM). We denote this partial state observation by $y(t)$ and assume without loss of generality that $y(t) = H[s(t); z(t)]$ for some projection matrix H (which can be the identity matrix, I, if we have complete observations). We then define $Y(\mathcal{T}) = \{y(t); t \in \mathcal{T}\}$ as the corresponding set of (partial) observations over $\mathcal{T}$.

We assume a context window $\mathcal{T}_{\text{context}}$ of controls and observations, $(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}))$. We are interested in the behavior of $y(t)$ over a prediction window $\mathcal{T}_{\text{pred}}$, represented by $Y(\mathcal{T}_{\text{pred}})$, given future controls $X(\mathcal{T}_{\text{pred}})$. A *model* M takes as input $(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}), X(\mathcal{T}_{\text{pred}}))$ and produces an output trajectory of y over the prediction window. Below we use $M^\dagger$ to represent the unknown ground truth model; this is the model that generated the observations $Y(\mathcal{T}_{\text{pred}})$. We use $\hat{M}$ to denote a fitted model. Throughout we use $\hat{Y}(\mathcal{T}_{\text{pred}}) = \{\hat{y}(t); t \in \mathcal{T}_{\text{pred}}\}$ to denote our predicted output from such a fitted model.

Predictive Loss We evaluate the predictive loss of an estimated model using mean squared error:

$$L_{\text{pred}}(\hat{M}) = \sum_{t \in \mathcal{T}_{\text{pred}}} \left\| y(t) - \hat{y}(t) \right\|_2^2. \tag{6}$$

In practice, we have a collection of observed sequences (e.g., one per patient) and compute the loss summing over all sequences. We then normalize our loss by the total number of observed time points aggregated over the sequences.

Causal Loss To incorporate causal validity into the loss function, we develop a causal loss that examines *counterfactual simulations* rather than *predictions*. For fixed context $X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}})$, we consider a range of hypothetical interventions $X^{(i)}(\mathcal{T}_{\text{pred}})$, $i = 1, \ldots, K$, that differ from the observed $X(\mathcal{T}_{\text{pred}})$. The ground truth model would produce observations $Y^{(i)}(\mathcal{T}_{\text{pred}}) = M^\dagger(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}), X^{(i)}(\mathcal{T}_{\text{pred}}))$ for the i'th intervention. If we could observe these counterfactuals, then we could compute the *causal effect* of each intervention:

$$\tau^{(i)} = u(Y^{(i)}(\mathcal{T}_{\text{pred}})) - u(Y(\mathcal{T}_{\text{pred}})).$$

Here, u is a *score function* that computes a meaningful scalar-valued output from a sequence input. For example, in diabetes, this score might correspond to the average glucose level over $\mathcal{T}_{\text{pred}}$. In other settings, the relevant quantity may be related to extremes, in which case u may compute, e.g., the maximum or minimum value of the sequence.

In many settings with complex dynamics or only very partial observations, such as in modeling glycemic response, the latent sources of variation (e.g., sleep, stress, physiology, etc.) render attempts to precisely estimate treatment effects nearly impossible. Importantly, in our setting the clinically relevant quantity for guiding patient care is the ordering of interventions; the same would be true in a multitude of settings where the model is guiding decision making. This ordinal domain knowledge is often readily available; for example, we know that holding other context constant, increasing insulin dosing should lower glucose levels, and increasingly larger insulin doses lead to greater reductions in glucose levels. Even such ordinal causal grounding is absent in purely predictive models.

To bias our training towards models that capture this ordinal causal domain knowledge, we introduce a causal loss where we evaluate whether the model can indentify the intervention with the maximum score. We define:

$$\mathcal{I}^* = \text{argmax}_i \tau^{(i)} = \text{argmax}_i u(Y^{(i)}(\mathcal{T}_{\text{pred}})).$$

In principle, we can compare this intervention to an estimated maximum score intervention under the trained model $\hat{M}$, i.e.,

$$\hat{\mathcal{I}} = \text{argmax}_i u(\hat{Y}^{(i)}(\mathcal{T}_{\text{pred}})),$$

where for the i'th intervention we have $\hat{Y}^{(i)}(\mathcal{T}_{\text{pred}}) = \hat{M}(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}), X^{(i)}(\mathcal{T}_{\text{pred}}))$.

Of course, to enable model training, we need a loss function that admits a gradient. Accordingly, rather than computing the exact maximum score intervention under the estimated model, we compute the *softmax* vector:

$$\hat{Q} = \sigma\Big(\phi u\big(\hat{Y}^{(1)}(\mathcal{T}_{\text{pred}})\big) \ , \ \ldots \ , \ \phi u\big(\hat{Y}^{(K)}(\mathcal{T}_{\text{pred}})\big)\Big).$$

Here σ is the softmax function and $\phi > 0$ is a scalar "temperature" parameter: the larger ϕ is, the closer $\hat{Q}$ is to approximating a one-hot encoding of the true argmax. In practice we choose ϕ as large as possible while maintaining stable training. We compute the cross entropy (CE) loss between $\hat{Q}$ and a one-hot encoding of the true argmax $\mathcal{I}^*$, denoted $CE(\hat{Q}, \mathcal{I}^*)$. We refer to this as our *causal loss*:

$$L_{\text{causal}}(\hat{M}) = CE(\hat{Q}, \mathcal{I}^*). \tag{7}$$

Again, in practice, we average over all observed sequences, as well as all intervention scenarios under consideration.

If one instead wants to evaluate a loss that considers the entire ranking over interventions, then we can in principle evaluate the CE loss in Eq. (7) for every possible subset of interventions. Of course, this rapidly becomes intractable with increasing K; in future work we plan to directly consider a loss function on the estimated ranking from $\hat{M}$.

Hybrid loss For $\alpha \in [0, 1]$, we define our *hybrid loss* as:

$$L_{\text{hybrid}}(\hat{M}) = (1 - \alpha)L_{\text{pred}}(\hat{M}) + \alpha L_{\text{causal}}(\hat{M}), \tag{8}$$

where we have overloaded notation to interpret L_{pred} and L_{causal} as the average predictive and causal losses, respectively, over all sequences and scenarios. Note that when $\alpha = 0$, model fitting focuses entirely on prediction; when $\alpha = 1$, model fitting focuses entirely on causal validity.

Model Fitting: The H^{2}NCM Approach A critical question in fitting hybrid ODE models is the initial condition $(s_0 = s(t_0), z_0 = z(t_0))$, where t_0 is the beginning of our prediction window. Two primary approaches exist for performing initial state estimation jointly with hybrid ODE parameter learning: (1) statistical state estimation methods (Chen et al., 2022; Brajard et al., 2021; Ribera et al., 2022; Levine & Stuart, 2022) and (2) blackbox encoder models (Chen et al., 2018). We take the latter approach, viewing initial state estimation through the lens of sequence-to-sequence (seq2seq) modeling.

In particular, we produce the predicted sequence $\hat{Y}(\mathcal{T}_{\text{pred}})$ using $X(\mathcal{T}_{\text{pred}})$ and the initial condition (s_0, z_0) encoded from the context data $(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}))$. We consider a general blackbox encoder M_{context} for the initial conditions:

$$(s_0, z_0) = M_{\text{context}}\big(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}); \theta_{\text{context}}\big),$$

where θ_{context} are the parameters of M_{context}.

We take the decoder, M_{pred}, to be a hybrid ODE that takes as input the initial condition (s_0, z_0), controls $X(\mathcal{T}_{\text{pred}})$, and parameters $\theta_{\text{pred}} = \{\beta, \theta\}$, where β are the simulator parameters and θ the neural network parameters. We assume the hybrid model indicator variables $c_1, \ldots, c_4$ and adjacency matrices A_s, A_x are specified. The predicted values are produced as the output of numerical integration of this hybrid ODE decoder over the window $\mathcal{T}_{\text{pred}}$:

$$\hat{y}(t_k) = H \cdot \text{Integrate}(M_{\text{pred}}, \mathcal{T}_{\text{pred}}; X(\mathcal{T}_{\text{pred}}), s_0, z_0, \theta_{\text{pred}}).$$

Here, H is an indicator matrix selecting the observed states. The parameters θ_{context} and θ_{pred} are jointly optimized to obtain a fitted model $\hat{M}$ that minimizes the hybrid loss of Eq. (8). We refer to this approach of training such a model by optimization of hybrid loss as H^{2}NCM.[2]

4. Related Work

Neural Causal Models (NCMs) NCMs use feed-forward neural networks to extend the flexibility of structural equation models (SEM) (Pearl, 1998). Note that both NCM and MNODE face the flexibility-causality dilemma in observational data settings where causal inference is challenging. Similar to the role of SEM in NCM, MNODE replaces system equations–here, dynamical systems defining mechanistic ODEs—with neural networks, while retaining the original state-connectivity graph encoding specific causal relationships. Xia et al. (2021) highlighted NCM's limitations in treatment effect estimation without strong assumptions, while we empirically show MNODE's inability to learn the ordering of treatment effects without causal constraints. For MNODE and related hybrid modeling approaches, we address this challenge through introducing a causal loss; in contrast, Xia et al. (2023) rely on strong distributional assumptions to maintain causality. Similarities between our work may suggest that our hybrid loss can be applied to improve causal alignment of general NCMs.

Physics-informed Neural Networks (PINNs) PINNs, introduced by Raissi et al. (2019), use neural networks for solving partial differential equations (PDEs) by approximating solutions that conform to known differential equations. In contrast to the hybrid models of this work, PINNs: 1) typically focus on learning *solutions* to PDEs (rather than a governing vector field), and 2) do so via regularization in which a physics-based loss encourages the learned solution to comply with known physical laws. This method is advantageous when the system's underlying physics are well-understood, but computationally intensive or costly to simulate directly. Related is the systems-biology-informed deep learning approach by Yazdani et al. (2020), which constrains the learned models to exhibit properties similar to pre-specified mechanistic ODEs.

Graph Network Simulator (GNS) GNS applies the message passing architecture of graph convolution networks (GCN) to physical system simulation (Sanchez-Gonzalez

[2]Our implementation of H^{2}NCM is available at https://github.com/bobjz/H2NCM.

et al., 2018; 2020; Pfaff et al., 2021; Wu et al., 2022; Allen et al., 2023). Poli et al. (2019) connect GNS with blackbox neural ODEs (GNODE) (see also Jin et al., 2022; Bishnoi et al., 2023; Wu et al., 2022; Allen et al., 2023; Li et al., 2022). MNODE focuses on causal relationships between features *across* time, rather than between spatial interactions; as such, we use directed rather than undirected graphs.

5. Experiments

5.1. Motivation: Safely Managing Exercise in T1D

Type 1 diabetes (T1D) is an autoimmune condition characterized by the destruction of insulin producing beta-cells. To manage glucose concentrations and prevent diabetes-related complications, intensive insulin therapy through externally delivered sources (e.g., insulin pumps) is required. Regular physical activity and exercise lead to numerous health benefits such as increased insulin sensitivity, weight management, and improved psychosocial well-being. However, for individuals with T1D, due to the inability to rapidly decrease circulating insulin concentrations, exercise can also increase the risk of hypoglycemia. Despite technological advancements such as continuous glucose monitoring (CGM) and automated insulin delivery systems, individuals with T1D still face significant challenges with managing glucose concentrations around exercise. Many adults with T1D are not meeting current exercise recommendations of at least 30-minutes of moderate-to-vigorous physical activity per day (Riddell et al., 2017), with fear of exercise-related hypoglycemia a leading barrier (Brazeau et al., 2008). To address these challenges, there is a pressing demand for precise, reliable models that can predict individualized glycemic responses during and after exercise and encourage safe physical activity for all individuals with T1D.

Past efforts on modeling glycemic response to physical activity have focused on developing more intricate mechanistic models (Dalla Man et al., 2009; Liu et al., 2018; Deichmann et al., 2023), rather than devising a performant predictive model. Although glucose prediction using ML and, more recently, deep learning methods has received significant attention, a dearth of papers consider exercise periods (Oviedo et al., 2017) and the few that do do not perform well (Hobbs et al., 2019; Xie & Wang, 2020; Tyler et al., 2022). We aim to leverage hybrid modeling to predict glucose concentrations of an individual with T1D in the first 30 minutes following physical activity, given historical context and expected future covariates. Exercise leads to increased insulin sensitivity post-exercise and this may in turn contribute to a heightened risk of hypoglycemia after activity. The post-exercise period represents both a period of glycemic risk and one in which interventions can be readily applied; our goal is to enable better guidance during this period.

5.2. Data Preparation

Our data come from the Type 1 Diabetes Exercise Initiative (T1DEXI) (Riddell et al., 2023), which can be requested via `https://doi.org/10.25934/PR00008428`. This is a real-world study of exercise effects on 497 adults with T1D. Participants in the study were randomly assigned to aerobic, resistance, or interval exercise videos for a total of six sessions over four weeks. Participants were asked to self-report their food intake and exercise habits, while their insulin dosage and relevant physiological responses were recorded by corresponding wearable devices such as insulin pumps, CGM devices, and smart watches.

We select participants on open-loop pumps, which enables real-time recording of insulin dosing with levels not proactively adjusted (and thus correlated with proximal CGM readings). We also limit our cohort to participants under 40 years old and with BMI below 30 because these participants tend to exercise more regularly and thus represent the general physically-active T1D population better.

For selected participants, we filter out exercises shorter than 30 minutes. Then, for each exercise instance, we extract the participant's metabolic history (basal/bolus insulin delivery, carbohydrate intake, heart rate, step count, and CGM readings) 4 hours before to 30 minutes after the end of exercise to form a 5-dimensional time series. The 4-hour context window was chosen because the effect of most bolus insulin (fast-acting insulin) lasts for around 4 hours. It is common within the T1D community to consider this time range when historical context is relevant in decision making. Finally, we filter out exercise instances with missing heart rate values. We end up with 143 exercise instances from 78 participants. See the Appendix for further data preprocessing details.

Intervention Sets Our causal loss relies on the introduction of intervention sets. Here, we consider sets of size $K = 3$. For our training procedure, for each training post-exercise instance, we create 3 replicates of $(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}))$ and append each replicate with an input sequence $X^{(i)}(\mathcal{T}_{\text{pred}})$ from a set selected uniformly at random from the following categories: (1) adding 0/50/100 grams (g) of carbohydrates at the end of exercise; (2) adding 0/2.5/5.0 units of insulin uniformly throughout the first 30 minutes post-exercise; (3) no-change/50g carbs/10.0 units of insulin at the end of exercise; (4) replacing the real post-exercise heart rate trajectory with a prototypical trajectory seen in aerobic/resistance/interval training. All of these modifications are relative to the observed inputs $X(\mathcal{T}_{\text{pred}})$. These intervention sets capture the most integral pieces of well-understood and well-researched domain knowledge, allowing us to compute a reliable true ranking. In particular, increasing levels of carbohydrates (resp., decreasing levels of insulin) progressively increase mean glucose; and increasing exercise intensity (as measured by

heart rate) leads to progressively increasing mean glucose during the short exercise period being considered (Aronson et al., 2019; Riddell et al., 2017). For our evaluation, we consider counterfactual simulations defined similarly. For each test exercise instance, we create a set of three instances $(X(\mathcal{T}_{\text{context}}), Y(\mathcal{T}_{\text{context}}), X^{(i)}(\mathcal{T}_{\text{pred}}))$ used to generate a counterfactual simulation $\hat{Y}^{(i)}(\mathcal{T}_{\text{pred}})$, $i = 1, 2, 3$. The intervention sets defining $X^{(i)}(\mathcal{T}_{\text{pred}})$ are again drawn uniformly at random. In all cases (train and test), we take score to be average glucose in the 30-minute prediction window.

5.3. Key Implementation Details

Discretization There are currently two mainstream methods to obtain approximate solutions to hybrid ODE variants: differentiate-then-discretize and discretize-then-differentiate (Ayed et al., 2019). We choose the latter for its simplicity and adaptability. This method approximates the hybrid ODE system with stacks of residual networks (He et al., 2016) via a forward-Euler discretization scheme

$$\begin{aligned} s_{t+1} &= s_t + \Delta t\big[c_1 m(s_t, x_t;\ A_s, A_x, \beta_t) \\ &\qquad + c_2 f_1(s_t, x_t, c_3 z_t;\ A_s, A_x, \theta)\big] \\ z_{t+1} &= z_t + \Delta t f_2(s_t, x_t;\ \theta) \\ \beta_{t+1} &= \beta + c_4 f_3(x_{t+1}, z_{t+1};\ \theta), \end{aligned}$$

and then computes the gradient via backpropagation. Here, we use subscript instead of parentheses to indicate the transition from the continuous to discrete time. In our implementation, we set Δt to be 5 minutes, which is the sampling rate of CGM readings in the T1DEXI data.

Model Reduction Instead of the highly parameterized UVA/Padova S2013 model (Man et al., 2014), our hybrid models build upon a reduced model with the aim of (1) reducing variance and improving generalization performance, and (2) reducing training time. We applied a data-driven reduction method to the full UVA/Padova mechanistic model and its causal graph; see the Appendix. An empirical comparison to hybridizations of the full, non-reduced UVA/Padova model are in the Appendix. The results illustrate that the flexibility of the neural networks in the hybrid models is a satisfactory replacement for many of the full mechanistic model compartments.

Model Variants and Evaluation Procedure We consider 3 hybrid models in the form of Eq. (5) with increasing amounts of flexibility: (1) latent parameter dynamics (LP) defined via $c_1 = c_4 = 1, c_2 = c_3 = 0$; (2) latent parameter dynamics plus state closure (LPSC) defined via $c_1 = c_2 = c_4 = 1, c_3 = 0$; and (3) mechanistic neural ODE (MNODE) defined via $c_2 = 1, c_1 = c_3 = c_4 = 0$. In all of these cases, the adjacency matrices A_s and A_x are specified based on our reduced UVA/Padova mechanistic model. We implement MNODE by defining a neural network for each

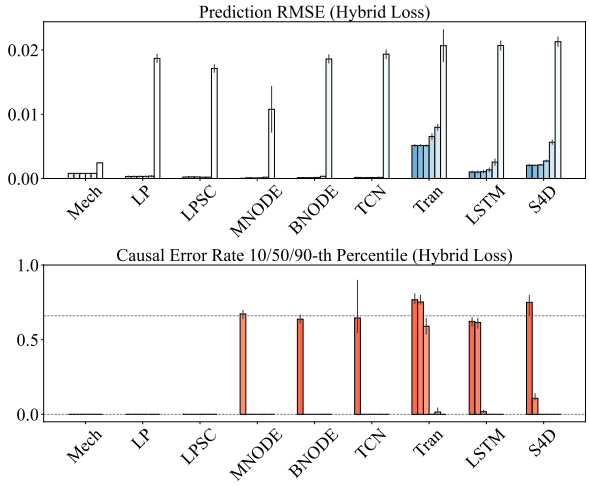


Figure 4. For the synthetic data, predictive loss (RMSE) and standard error (*top*) and 10/50/90-th(upper/bar/lower) percentile causal classification error rate (*bottom*). Within each bar group for a single model, the parameter α (hybrid loss parameter) increases from left to right, taking the values $\{0, 1e{-}4, 1e{-}3, 1e{-}2, 1e{-}1, 1\}$. The dashed line in the bottom figure corresponds to the causal classification error of random guessing $(2/3)$.

output dimension and masking appropriate inputs, though the methods of Chen et al. (2024) could be adapted to our setting to provide a more efficient alternative. We also consider the blackbox neural ODE model (BNODE) of Eq. (3) and tune the latent state dimension. Finally, we provide a set of non-ODE-based blackbox sequence models: the LSTM, Transformer (Trans), temporal convolutional network (TCN), and diagonal approximation to S4 (S4D) (Gu et al., 2022a). For our mechanistic model baseline, we consider the full (not reduced) UVA/Padova S2013 simulator (UVA). We limit all models to have fewer than 25,000 parameters for fair comparison. For all of our ODE-based models, we use a multi-stack LSTM for our initial-condition encoder; the non-ODE baseline models also take the historical context as input, with model-specific encoding choices. See the Appendix for further model implementation details.

To tune our models and compute test error, due to the small dataset size, we use repeated nested cross validation (CV) with 3 repeats, 6 outer folds and 4 inner folds. The inner folds are used to tune hyperparameters and outer folds to estimate generalization error; results are averaged over three runs for which the sequences are randomly shuffled. For more details of the evaluation procedure, see the Appendix.

5.4. Results

Synthetic Data We start by exploring a synthetic data example generated by a simple, single-state ODE with two correlated inputs, mimicking the carb/insulin correlation in the T1DEXI data. In particular, the ODE is specified as:

$$dy/dt = -y(t) + x_1(t) - x_2(t), \quad y(0) = 0$$

$$x_1(t) = a\exp(-bt),\ a \sim \text{Unif}[1,2],\ b \sim \text{Unif}[5,15]$$
$$x_2(t) = 1.5x_1(t) + \epsilon, \quad \epsilon \sim N(0, 1\text{e}{-4}).$$

We simulate 600 training, 200 validation, and 200 test sequences and corresponding intervention sets, each selected uniformly at random from: (1) raising x_1 by 0/+1/+2; (2) raising x_2 by 0/+1/+2; or (3) no-change/adding +1 to x_1/adding +1 to x_2. The sequences are of length 100.

In Fig. 4, we show prediction root mean squared error (RMSE) and classification error, calculated using the nested CV procedure described above, for each of the considered models as a function of α. Here, the classification task is to correctly predict the intervention that yields the highest score (taken to be average of $\hat{Y}^{(i)}(\mathcal{T}_{\text{pred}})$) among the set of 3 choices. We see that even when the true mechanistic model is used in the hybrid modeling—and even when that system is very simple—our causal loss is critical in disambiguating the sign of the treatment effects for x_1 and x_2. For example, when $\alpha = 0$ (fully predictive loss), MNODE (and the blackbox models) have large causal loss because of lack of identifiability between the sign of the treatment effects of x_1 and x_2. However, even a small $\alpha > 0$ rapidly enables disambiguation between x_1 and x_2. These synthetic results illustrate the importance of moving beyond predictive loss, especially when there is significant correlation between inputs; the importance is heightened when working with partial observations, as in our real-data setting below.

T1DEXI Data We now turn to our results on the real-world T1DEXI data. In Fig. 5, we see that as α—the amount we emphasize the causal loss component of our hybrid loss—increases, all hybrid and blackbox models have significant *decreases* in classification error; however, the hybrid causal error rate decreases more rapidly. Critically, across a broad range of α values, the hybrid models have stable RMSE; the blackbox models appear to be more sensitive to increasing α. (The RMSE of all models is impacted for sufficiently large α, notably $\alpha = 1$ when a purely causal loss is considered.) Note that the UVA/Padova mechanistic model does not achieve zero classification error because the mechanistic model does not include all mechanistic components relevant to exercise. Our hybrid loss provides a win-win for hybrid ODEs: predictive performance exceeding pure mechanistic or blackbox approaches while also having extremely low classification error, again outperforming both the mechanistic and blackbox baselines.

Although our causal loss focuses on intervention *ranking*, this information alone may bias learning towards causally-grounded models that better capture the direction of treatment effects even outside the domain knowledge encoded in the causal loss. For example, in the Appendix, we define our causal loss in terms of rankings on carbohydrate and insulin intervention sets, but evaluate on insulin-to-carbohydrate ratios, which represents the most common intervention considered by clinicians for T1D. Our experiments demonstrate that even when we train with limited domain knowledge (i.e., the impact of carbohydrates or insulin individually), all models appear to exhibit improved insulin-to-carbohydrate rankings relative to models that do not leverage causal loss.

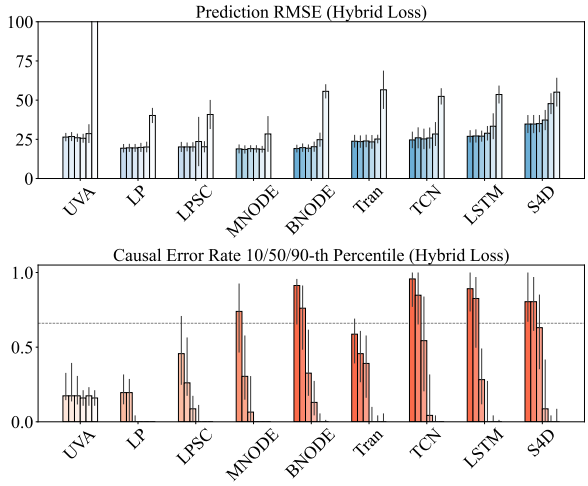


Figure 5. As in Fig. 4, but for T1DEXI data and using a full UVA/Padova mechanistic model baseline and hybridizations of the reduced UVA/Padova simulator for LP, LPSC, and MNODE.

In the Appendix, we also consider the impact of increasing levels of corrupted domain knowledge where our causal loss is given the incorrect label for the top-ranked intervention, $\mathcal{I}^*$, for some fraction of intervention sets. The results indicate robustness to moderate amounts of corruption.

6. Discussion

We presented H^{2}NCM, a method for integrating domain knowledge not only through a hybrid *model*, but also through a hybrid *loss* that encourages causal validity. We consider this in the challenging real-world setting of predicting post-exercise glycemic response in T1D. Our experiments illustrate a win-win where—across a wide range of settings of α—our state-of-the-art predictive performance does not drop while the causal validity dramatically improves. In theory, models that can do both tasks well should align better with the true underlying system $M^\dagger$ and thus generalize better to unseen data, especially in the presence of distribution shifts, noise, and incompleteness, as our promising initial results in the Appendices suggest.

We assume a known mechanistic model from which to build our hybrid models. One could instead consider softer forms of prior mechanistic knowledge. For example, Wang et al. (2024) specify priors over graphical structures in neural differential equations. Indeed, our hybrid loss may be formulated in a Bayesian context as well, and could aid other modeling frameworks (cf., Takeishi & Kalousis, 2021).

Acknowledgements

This work was supported in part by AFOSR Grant FA9550-21-1-0397, ONR Grant N00014-22-1-2110, NSF Grant 2205084, the Stanford Institute for Human-Centered Artificial Intelligence (HAI), the Helmsley Charitable Trust, and the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard. EBF is a Chan Zuckerberg Biohub – San Francisco Investigator. This publication is based on research using data from Jaeb Center for Health Research Foundation that has been made available through Vivli, Inc. Vivli has not contributed to or approved, and is not in any way responsible for, the contents of this publication. We thank Ke Alexander Wang for helpful discussions on the implementation and training of various models.

Impact Statement

All clinical decision support systems come with some risks, even with a trained human in the loop. If our intervention-ranking system gains the trust of the clinician, they may rely on its recommendations blindly. There are of course cases where it will make errors. In the type 1 diabetes setting, an error of recommending too much insulin may lead to potentially life-threatening hypoglycemia. A benefit of our system, however, is that instead of searching over all possible interventions, we are ranking an expert-provided list of possible interventions, all of which are assumed to be safe alternatives that would not deviate from clinically established standards of care.

References

Allen, K. R., Guevara, T. L., Rubanova, Y., Stachenfeld, K., Sanchez-Gonzalez, A., Battaglia, P., and Pfaff, T. Graph network simulators can learn discontinuous, rigid contact dynamics. In *Conference on Robot Learning*, pp. 1157–1167. PMLR, 2023.

Aronson, R., Brown, R. E., Li, A., and Riddell, M. C. Optimal insulin correction factor in post–high-intensity exercise hyperglycemia in adults with type 1 diabetes: the fit study. *Diabetes care*, 42(1):10–16, 2019.

Asikis, T., Böttcher, L., and Antulov-Fantulin, N. Neural ordinary differential equation control of dynamics on graphs. *Physical Review Research*, 4(1):013221, 2022.

Ayed, I., de Bézenac, E., Pajot, A., Brajard, J., and Gallinari, P. Learning dynamical systems from partial observations. *Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada*, 2019.

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

Bisgaard Bengtsen, M. and Møller, N. Mini-review: Glucagon responses in Type 1 diabetes–a matter of complexity. *Physiological Reports*, 9(16):e15009, 2021.

Bishnoi, S., Bhattoo, R., Ranu, S., and Krishnan, N. Enhancing the inductive biases of graph neural ODE for modeling dynamical systems. *International Conference on Learning Representations (ICLR) 2023 poster*, 2023.

Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L. Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A*, 379(2194):20200086, 2021.

Brazeau, A.-S., Rabasa-Lhoret, R., Strychar, I., and Mircescu, H. Barriers to physical activity among patients with Type 1 diabetes. *Diabetes care*, 31(11):2108–2109, 2008.

Canini, L. and Perelson, A. S. Viral kinetic modeling: State of the art. *Journal of pharmacokinetics and pharmacodynamics*, 41:431–443, 2014.

Chen, A., Shi, R. I., Gao, X., Baptista, R., and Krishnan, R. G. Structured neural networks for density estimation and causal inference. *Advances in Neural Information Processing Systems*, 36, 2024.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Chen, Y., Sanz-Alonso, D., and Willett, R. Autodifferentiable ensemble Kalman filters. *SIAM Journal on Mathematics of Data Science*, 4(2):801–833, 2022.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Neural Information Processing Systems Workshop on Deep Learning*, 2014.

Dalla Man, C., Breton, M. D., and Cobelli, C. Physical activity into the meal glucose—insulin model of Type 1 diabetes: In silico studies, 2009.

Deichmann, J., Bachmann, S., Burckhardt, M.-A., Pfister, M., Szinnai, G., and Kaltenbach, H.-M. New model of glucose-insulin regulation characterizes effects of physical activity and facilitates personalized treatment evaluation in children and adults with Type 1 diabetes. *PLOS Computational Biology*, 19(2):e1010289, 2023.

Didona, D., Quaglia, F., Romano, P., and Torre, E. Enhancing performance prediction robustness by combining analytical modeling and machine learning. In *Proceedings of the 6th ACM/SPEC international conference on performance engineering*, pp. 145–156, 2015.

Du, M., Liu, N., and Hu, X. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1): 68–77, 2019.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35: 35971–35983, 2022a.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Representations (ICLR)*, 2022b.

Gupta, A. and Lermusiaux, P. F. Neural closure models for dynamical systems. *Proceedings of the Royal Society A*, 477(2252):20201004, 2021.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse problems*, 34(1):014004, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, S., Peng, Y., and Sun, K. SEIR modeling of the COVID-19 and its dynamics. *Nonlinear dynamics*, 101:1667–1680, 2020.

Hilgemann, D. and Noble, D. Excitation-contraction coupling and extracellular calcium transients in rabbit atrium: Reconstruction of basic cellular mechanisms. *Proceedings of the Royal society of London. Series B. Biological sciences*, 230(1259):163–205, 1987.

Hobbs, N., Hajizadeh, I., Rashid, M., Turksoy, K., Breton, M., and Cinar, A. Improving glucose prediction accuracy in physically active adolescents with Type 1 diabetes. *Journal of diabetes science and technology*, 13(4):718–727, 2019.

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

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500, 1952.

Holz, M. and Fahr, A. Compartment modeling. *Advanced Drug Delivery Reviews*, 48(2-3):249–264, 2001.

Hussain, Z. M., Krishnan, R. G., and Sontag, D. Neural pharmacodynamic state space modeling. In *International Conference on Machine Learning*, pp. 4500–4510. PMLR, 2021.

Jin, M., Zheng, Y., Li, Y.-F., Chen, S., Yang, B., and Pan, S. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kidger, P. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Ladenbauer, J., McKenzie, S., English, D. F., Hagens, O., and Ostojic, S. Inferring and validating mechanistic models of neural microcircuits based on spike-train data. *Nature communications*, 10(1):4933, 2019.

Lea, C., Vidal, R., Reiter, A., and Hager, G. D. Temporal convolutional networks: A unified approach to action segmentation. In Hua, G. and Jégou, H. (eds.), *Computer Vision – ECCV 2016 Workshops*, pp. 47–54, Cham, 2016. Springer International Publishing. ISBN 978-3-319-49409-8.

Levine, M. and Stuart, A. A framework for machine learning of model error in dynamical systems. *Communications of the American Mathematical Society*, 2(07):283–344, 2022.

Li, Z., Meidani, K., Yadav, P., and Barati Farimani, A. Graph neural networks accelerated molecular dynamics. *The Journal of Chemical Physics*, 156(14), 2022.

Liu, C., Vehi, J., Oliver, N., Georgiou, P., and Herrero, P. Enhancing blood glucose prediction with meal absorption and physical exercise information. *arXiv preprint arXiv:1901.07467*, 2018.

Lu, J., Deng, K., Zhang, X., Liu, G., and Guan, Y. Neural-ODE for pharmacokinetics modeling and its advantage to alternative machine learning models in predicting new dosing regimens. *Iscience*, 24(7), 2021.

Man, C. D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., and Cobelli, C. The UVA/PADOVA Type 1 diabetes simulator: New features. *Journal of diabetes science and technology*, 8(1):26–34, 2014.

Marsh, D. J., Sosnovtseva, O. V., Chon, K. H., and Holstein-Rathlou, N.-H. Nonlinear interactions in renal blood flow regulation. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 288(5):R1143–R1159, 2005.

McArdle, W. D., Katch, F. I., and Katch, V. L. *Essentials of exercise physiology*. Lippincott Williams & Wilkins, 2006.

Miller, A. C., Foti, N. J., and Fox, E. Learning insulin-glucose dynamics in the wild. In *Machine Learning for Healthcare Conference*, pp. 172–197. PMLR, 2020.

Oviedo, S., Vehí, J., Calm, R., and Armengol, J. A review of personalized blood glucose prediction strategies for T1DM patients. *International journal for numerical methods in biomedical engineering*, 33(6):e2833, 2017.

Owoyele, O. and Pal, P. ChemNODE: a neural ordinary differential equations framework for efficient chemical kinetic solvers. *Energy and AI*, 7:100118, 2022.

Passini, E., Britton, O. J., Lu, H. R., Rohrbacher, J., Hermans, A. N., Gallacher, D. J., Greig, R. J., Bueno-Orovio, A., and Rodriguez, B. Human in silico drug trials demonstrate higher accuracy than animal models in predicting clinical pro-arrhythmic cardiotoxicity. *Frontiers in physiology*, 8:668, 2017.

Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B. R., Girvan, M., and Ott, E. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

Pearl, J. Graphs, causality, and structural equation models. *Sociological Methods & Research*, 27(2):226–284, 1998.

Perelson, A. S., Neumann, A. U., Markowitz, M., Leonard, J. M., and Ho, D. D. HIV-1 dynamics in vivo: Virion clearance rate, infected cell life-span, and viral generation time. *Science*, 271(5255):1582–1586, 1996.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Qian, Z., Zame, W., Fleuren, L., Elbers, P., and van der Schaar, M. Integrating expert ODEs into neural ODEs: Pharmacology and disease progression. *Advances in Neural Information Processing Systems*, 34:11364–11383, 2021.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ribera, H., Shirman, S., Nguyen, A. V., and Mangan, N. M. Model selection of chaotic systems from data with hidden variables using sparse data assimilation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(6), June 2022. ISSN 1089-7682. doi: 10.1063/5.0066066. URL http://dx.doi.org/10.1063/5.0066066.

Rico-Martinez, R., Krischer, K., Kevrekidis, I., Kube, M., and Hudson, J. Discrete-vs. continuous-time nonlinear signal processing of Cu electrodissolution data. *Chemical Engineering Communications*, 118(1):25–48, 1992.

Rico-Martinez, R., Anderson, J., and Kevrekidis, I. Continuous-time nonlinear signal processing: A neural network based approach for gray box identification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pp. 596–605. IEEE, 1994.

Riddell, M. C., Gallen, I. W., Smart, C. E., Taplin, C. E., Adolfsson, P., Lumb, A. N., Kowalski, A., Rabasa-Lhoret, R., McCrimmon, R. J., Hume, C., et al. Exercise management in Type 1 diabetes: A consensus statement. *The lancet Diabetes & endocrinology*, 5(5):377–390, 2017.

Riddell, M. C., Li, Z., Gal, R. L., Calhoun, P., Jacobs, P. G., Clements, M. A., Martin, C. K., Doyle III, F. J., Patton, S. R., Castle, J. R., et al. Examining the acute glycemic effects of different types of structured exercise sessions in Type 1 diabetes in a real-world setting: The Type 1 diabetes and exercise initiative (T1DEXI). *Diabetes care*, 46(4):704–713, 2023.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Shi, J., Wang, K. A., and Fox, E. Sequence modeling with multiresolution convolutional memory. In *International Conference on Machine Learning*, pp. 31312–31327. PMLR, 2023.

Sottile, P. D., Albers, D., DeWitt, P. E., Russell, S., Stroh, J., Kao, D. P., Adrian, B., Levine, M. E., Mooney, R.,

Larchick, L., et al. Real-time electronic health record mortality prediction during the COVID-19 pandemic: A prospective cohort study. *Journal of the American Medical Informatics Association*, 28(11):2354–2365, 2021.

Takeishi, N. and Kalousis, A. Physics-integrated variational autoencoders for robust and interpretable generative modeling. *Advances in Neural Information Processing Systems*, 34:14809–14821, 2021.

Tauschmann, M., Forlenza, G., Hood, K., Cardona-Hernandez, R., Giani, E., Hendrieckx, C., DeSalvo, D. J., Laffel, L. M., Saboo, B., Wheeler, B. J., et al. ISPAD clinical practice consensus guidelines 2022: Diabetes technologies: Glucose monitoring. *Pediatric Diabetes*, 23(8):1390–1405, 2022.

ten Tusscher, K. H., Noble, D., Noble, P.-J., and Panfilov, A. V. A model for human ventricular tissue. *American Journal of Physiology-Heart and Circulatory Physiology*, 286(4):H1573–H1589, 2004.

Tyler, N. S., Mosquera-Lopez, C., Young, G. M., El Youssef, J., Castle, J. R., and Jacobs, P. G. Quantifying the impact of physical activity on future glucose trends using machine learning. *Iscience*, 25(3):103888, 2022.

Wang, B., Jennings, J., and Gong, W. Neural structure learning with stochastic differential equations. *International Conference on Learning Representations (ICLR)*, 2024.

Wang, L., Adiga, A., Chen, J., Sadilek, A., Venkatramanan, S., and Marathe, M. Causalgnn: Causal-based graph neural networks for spatio-temporal epidemic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 12191–12199, 2022.

Weinan, E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11, 2017.

Wellen, K. E., Hotamisligil, G. S., et al. Inflammation, stress, and diabetes. *The Journal of clinical investigation*, 115(5):1111–1119, 2005.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, 55(4):1–37, 2022.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Wu, T., Wang, Q., Zhang, Y., Ying, R., Cao, K., Sosic, R., Jalali, R., Hamam, H., Maucec, M., and Leskovec, J. Learning large-scale subsurface simulations with a hybrid graph network simulator. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4184–4194, 2022.

Xia, K., Lee, K.-Z., Bengio, Y., and Bareinboim, E. The causal-neural connection: Expressiveness, learnability, and inference. *Advances in Neural Information Processing Systems*, 34:10823–10836, 2021.

Xia, K., Pan, Y., and Bareinboim, E. Neural causal models for counterfactual identification and estimation. *International Conference on Learning Representations (ICLR)*, 2023.

Xie, J. and Wang, Q. Benchmarking machine learning algorithms on blood glucose prediction for Type 1 diabetes in comparison with classical time-series models. *IEEE Transactions on Biomedical Engineering*, 67(11):3101–3124, 2020.

Yazdani, A., Lu, L., Raissi, M., and Karniadakis, G. E. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS computational biology*, 16(11):e1007575, 2020.

Yin, Y., Le Guen, V., Dona, J., de Bézenac, E., Ayed, I., Thome, N., and Gallinari, P. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

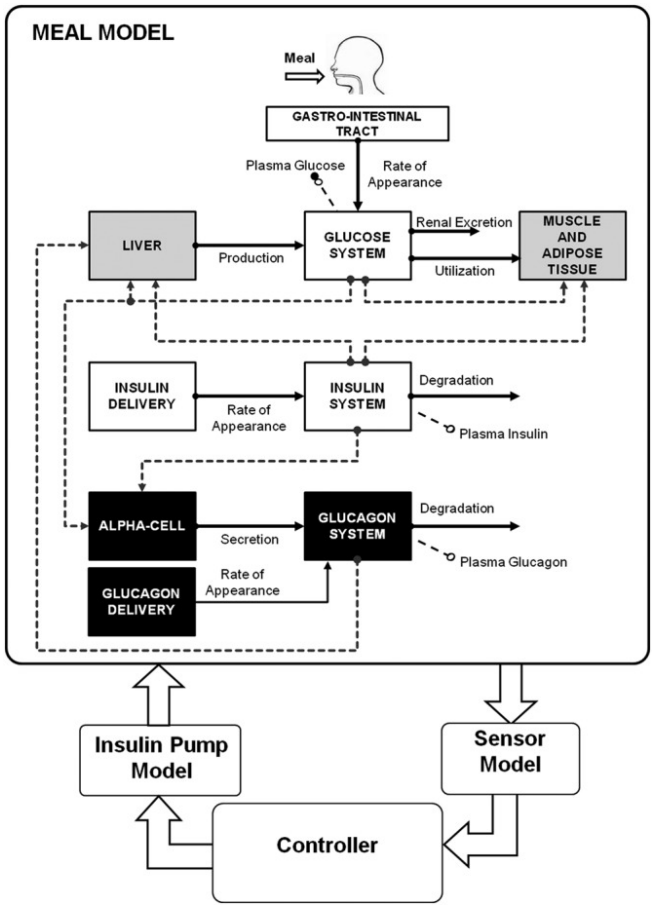


Figure 6. UVA/Padova Simulator S2013, taken from Figure 1 of Man et al. (2014)

A. UVA-Padova Simulator S2013

Here we provide the exact full UVA-Padova S2013 model equations. Variables that are not given meaningful interpretations are model parameters.

A.1. Summary Diagram

At a high level, UVA-Padova can be summarized by the diagram in Figure 6, which is taken from Figure 1 in Man et al. (2014). It divides the complex physiological system into 10 subsystems, which are linked by key causal states such as Rate of Appearance, Endogenous Glucose Production and Utilization. Next, we will introduce each subsystem one by one and also explain the physiological meanings behind state variables.

A.2. Glucose Subsystem

$$\dot{G}_p = EGP + Ra - U_{ii} - E - k_1 G_p + k_2 G_t \tag{9}$$

$$\dot{G}_t = -U_{id} + k_1 G_p - k_2 G_t \tag{10}$$

$$G = G_p / V_G \tag{11}$$

G_p: Plasma Glucose, G_t Tissue Glucose, EGP: Endogenous Glucose Production Rate, Ra Rate of Glucose Appearance, U_{ii}: Insulin-independent Utilization Rate, U_{id}: Insulin-dependent Utilization Rate, E Excretion Rate, V_G Volume Parameter, G Plasma Glucose Concentration

A.3. Insulin Subsystem

$$\dot{I}_p = -(m_2 + m_4)I_p + m_1 I_l + Rai \tag{12}$$
$$\dot{I}_t = -(m_1 + m_3)I_t + m_2 I_p \tag{13}$$
$$I = I_p / V_I \tag{14}$$

I_p Plasma Insulin, I_l Liver Insulin, Rai Rate of Insulin Appearance, V_l Volume Parameter, I Plasma Insulin Concentration

A.4. Glucose Rate of Appearance

$$Q_{sto} = Q_{sto1} + Q_{sto2} \tag{15}$$
$$\dot{Q}_{sto1} = -k_{gri} Q_{sto1} + D \cdot \delta \tag{16}$$
$$\dot{Q}_{sto2} = -k_{empt}(Q_{sto}) \cdot Q_{sto2} + k_{gri} Q_{sto1} \tag{17}$$
$$\dot{Q}_{gut} = -k_{abs} Q_{gut} + k_{empt}(Q_{sto}) \cdot Q_{sto2} \tag{18}$$
$$Ra = f k_{abs} Q_{gut} / (BW) \tag{19}$$
$$k_{empt}(Q_{sto}) = k_{min} + (k_{max} - k_{min})(\tanh(\alpha Q_{sto} - \alpha b D) - \tanh(\beta Q_{sto} - \beta c D) + 2)/2 \tag{20}$$

Q_{sto1}: First Stomach Compartment, Q_{sto2}: Second Stomach Compartment, Q_{gut}: Gut Compartment, δ Carbohydrate Ingestion Rate

A.5. Endogenous Glucose Production

$$EGP = k_{p1} - k_{p2} G_p - k_{p3} X^L + \xi X^H \tag{21}$$
$$\dot{X}^L = -k_i (X^L - I_r] \tag{22}$$
$$\dot{I}_r = -k_i (I_r - I) \tag{23}$$
$$\dot{X}^H = -k_H X^H + k_H \max(H - H_b) \tag{24}$$

X^L: Remote Insulin Action on EGP, X^H: Glucagon Action on EGP, I_r Remote Insulin Concentration, H Plasma Glucagon Concentration, H_b: Basal Glucagon Concentration Parameter

A.6. Glucose Utilization

$$U_{ii} = F_{cns} \tag{25}$$
$$U_{id} = \frac{(V_{m0} + V_{mx} X(1 + r_1 \cdot risk)) G_t}{K_{m0} + G_t} \tag{26}$$
$$\dot{X} = -p_{2U} X + p_{2U}(I - I_b) \tag{27}$$
$$risk = \begin{cases} 0 & G_b \leq G \\ 10(\log(G) - \log(G_b))^{2r_2} & G_{th} \leq G < G_b \\ 10(\log(G_{th}) - \log(G_b))^{2r_2} & G < G_{th} \end{cases} \tag{28}$$

F_{cns}: Glucose Independent Utilization Constant, X: Insulin Action on Glucose Utilization, I_b Basal Insulin Concentration Constant, $risk$ Hypoglycemia Risk Factor, G_b Basal Glucose Concentration Parameter, G_{th} Hypoglycemia Glucose Concentration Threshold.

A.7. Renal Excretion

$$\dot{E} = k_{e1} \max(G_p - k_{e2}, 0) \tag{29}$$

A.8. Subcutaneous Insulin Kinetics

$$Rai = k_{a1} I_{sc1} + k_{a2} I_{sc2} \tag{30}$$
$$\dot{I}_{sc1} = -(k_d + k_{a1}) I_{sc1} + IIR \tag{31}$$
$$\dot{I}_{sc2} = k_d I_{sc1} - k_{a2} I_{sc2} \tag{32}$$

I_{sc1}: First Subcutaneous Insulin Compartment, I_{sc2}: Second Subcutaneous Insulin Compartment, IIR Exogenous Insulin Delivery Rate

A.9. Subcutaneous Glucose Kinetics

$$\dot{G}_s = -T_s G_s + T_s G \tag{33}$$

G_s: Subcutaneous Glucose Concentration

A.10. Glucagon Secretion and Kinetics

$$\dot{H} = -nH + SR_H + Ra_H \tag{34}$$
$$SR_H = SR_H^s + SR_H^d \tag{35}$$
$$\dot{SR}_H^s = \begin{cases} -\rho \left[SR_H^s - \max\left(\sigma_2(G_{th} - G) + SR_H^b, 0\right) \right] & G \geq G_b \\ -\rho \left[SR_H^s - \max\left(\frac{\sigma(G_{th} - G)}{I+1} + SR_H^b, 0 \right) \right] & G < G_b \end{cases} \tag{36}$$
$$\dot{SR}_H^d = \eta \max(-\dot{G}, 0) \tag{37}$$

SR_H^s: First Glucagon Secretion Compartment, SR_H^d: Second Glucagon Secretion Compartment, SR_H^b: Basal Glucagon Secretion Parameter, Ra_H: Rate of Glucagon Appearance

A.11. Subcutaneous Glucagon Kinetics

$$\dot{H}_{sc1} = -(k_{h1} + k_{h2}) H_{sc1} + H_{inf} \tag{38}$$
$$\dot{H}_{sc2} = k_{h1} H_{sc1} - k_{h3} H_{sc2} \tag{39}$$
$$Ra_H = k_{h3} H_{sc2} \tag{40}$$

H_{sc1}: First Subcutaneous Glucagon Compartment, H_{sc2}: Second Subcutaneous Glucagon Compartment, H_{inf} Subcutaneous Glucagon Infusion Rate.

B. Reduced UVA-Padova for DTDSim2

We used the a reduced version of UVA-Padova S2013 in the implementation of Latent Parameter Model and Latent Parameter with State Closure Model. It comprises the following equations:

$$\dot{G}_p = EGP + Ra - U_{ii} - k_1 G_p + k_2 G_t \tag{41}$$
$$\dot{G}_t = -U_{id} + k_1 G_p - k_2 G_t \tag{42}$$
$$\dot{I}_p = -(m_2 + m_4) I_p + m_1 I_l + IIR \tag{43}$$
$$\dot{I}_l = -(m_1 + m_3) I_l + m_2 I_p \tag{44}$$
$$Q_{sto} = Q_{sto1} + Q_{sto2} \tag{45}$$
$$\dot{Q}_{sto1} = -k_{gri} Q_{sto1} + D \cdot \delta \tag{46}$$
$$\dot{Q}_{sto2} = -k_{empt}(Q_{sto}) \cdot Q_{sto2} + k_{gri} Q_{sto1} \tag{47}$$
$$\dot{Q}_{gut} = -k_{abs} Q_{gut} + k_{empt}(Q_{sto}) \cdot Q_{sto2} \tag{48}$$
$$Ra = f k_{abs} Q_{gut} / (BW) \tag{49}$$
$$k_{empt}(Q_{sto}) = k_{min} + (k_{max} - k_{min})(\tanh(\alpha Q_{sto} - \alpha b D) - \tanh(\beta Q_{sto} - \beta c D) + 2)/2 \tag{50}$$
$$EGP = k_{p1} - k_{p2} G_p - k_{p3} X^L \tag{51}$$
$$\dot{X}^L = -k_i (X^L - I_p) \tag{52}$$
$$U_{ii} = F_{cns} \tag{53}$$
$$U_{id} = \frac{(V_{m0} + V_{mx} X) G_t}{K_{m0} + G_t} \tag{54}$$
$$\dot{X} = -p_{2U} X + p_{2U} I_p \tag{55}$$
$$\tag{56}$$

Comparing to the full model, we performed the following:

1. We replaced states G, I with G_p, I_p as they only differ by a constant.
2. We removed the whole glucagon system to reduce model variance because most T1D patients do not have exogenous glucagon delivery, and their body's own glucagon regulation system is often impaired, a common symptom of T1D (Bisgaard Bengtsen & Møller, 2021).
3. We removed the renal excretion system, as renal excretion of glucose only takes place during episodes of severe hyperglycemia, which does not happen very often to patients on insulin pumps.
4. We removed the subcutaneous glucose/insulin kinetics systems/the remote insulin state they are solely meant to introduce delays, which already exist in the data (CGM readings are only taken every 5 minutes).
5. We removed the hypoglycemia risk factor $risk$ as it is used to model a relatively uncommon phenomenon.

We verified these reduction changes on a validation set that was taken out of the training set to make sure they do not break our models.

C. MNODE Graph Reduction Heuristic

We start with the full UVA/Padova causal graph as shown in Figure 7, this graph is obtained by adding the physical activity model graph from Dalla Man et al. (2009) to the UVA/Padova S2013 (Man et al., 2014) causal graph. Note that in the illustration, the node HR actually refers to both heart rate and step count, as we consider these two features both crucial indicators of physical activity intensity.

1. Step 1: For each Strongly Connected Component in the graph (indicated in Figure 8a), try collapsing it into a single node and evaluate MNODE's performance with the resulting graph on the validation set. Adopt the change that (1) has

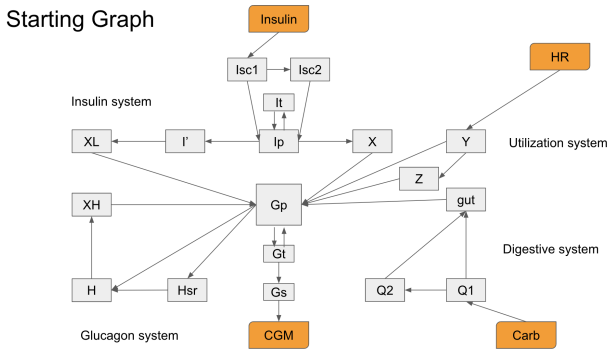
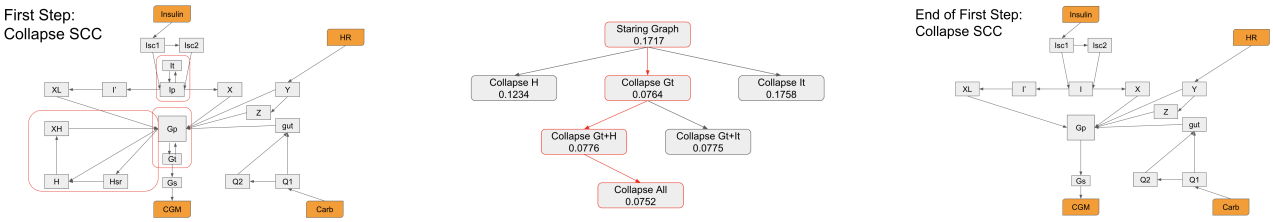


Figure 7. The Starting Graph for MNODE Graph Reduction Heuristic

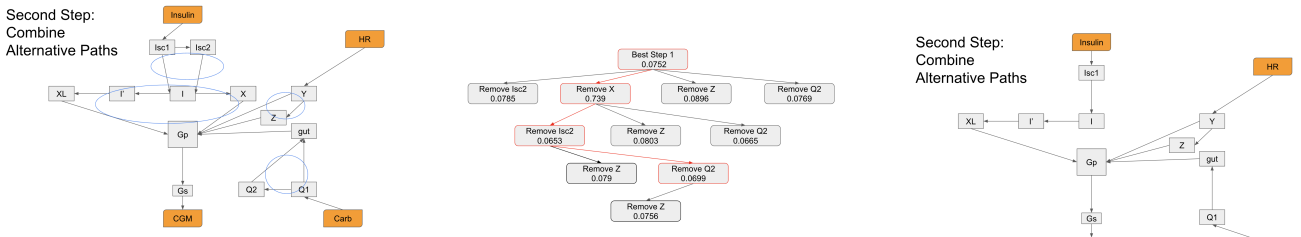


(a) SCCs in the Starting Graph (b) Reduction Tree of the SCCs (c) End Graph of the SCC Step

Figure 8. Illustration of the Collapsing SCC Step of Our Heuristic

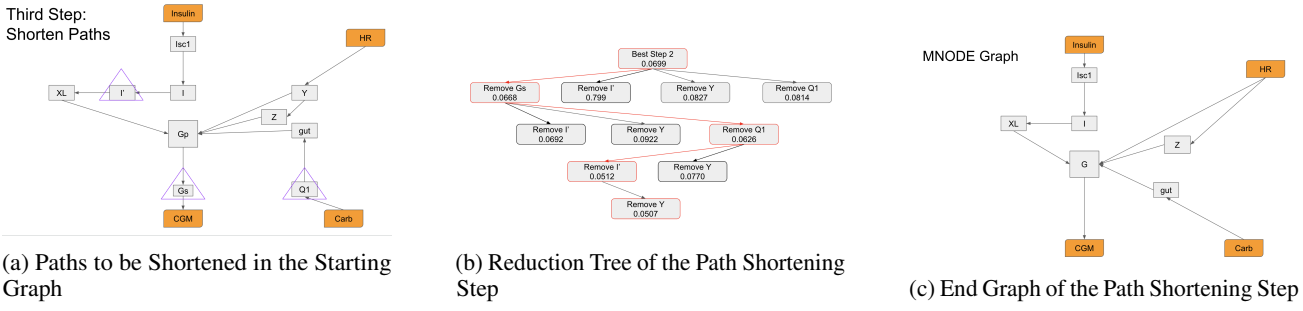
the least loss among all trials and (2) has loss that is within 10 percent increase of the best loss ever achieved so far. The second condition is to make sure we are not picking among a set of bad choices and at the same time to encourage exploration (i.e. can proceed as long as loss does not increase too much). The metric we use is the hybrid loss with $\alpha = 0.6$ (we picked an α that is not used in the actual experiments to avoid bias).

2. Step 2: Repeat step 1 until no change satisfy both criteria. Figure 8 shows an illustrative summary of step 1 and step 2.
3. Step 3: For each group of non-overlapping paths with same source and destination nodes, try merge them by keeping only the path of greatest length and evaluate MNODE's performance with the resulting graph on the validation set. Adopt the change with the same criteria as in step 1.
4. Step 4: Repeat step 3 until no change satisfy both criteria. Figure Figure 9 shows an illustrative summary of step 3 and step 4.
5. Step 5: For each path between an input node and the output node (at this point they should all be disjoint), try reducing its length by 1 via removing one intermediate node and evaluate MNODE's performance with the resulting graph on the validation set. Adopt the change with the same criteria as in step 1.



(a) Non-overlapping Path Groups in the Starting Graph (b) Reduction Tree of the Non-overlapping Paths (c) End Graph of the Path Combine Step

Figure 9. Illustration of the Combining Non-overlapping Paths Step of Our Heuristic



(a) Paths to be Shortened in the Starting Graph

(b) Reduction Tree of the Path Shortening Step

(c) End Graph of the Path Shortening Step

Figure 10. Illustration of the Combining Non-overlapping Paths Step of Our Heuristic

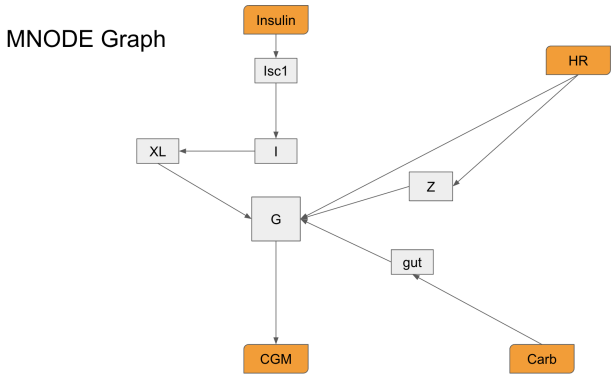


Figure 11. Final Graph used by MNODE

6. Step 6: Repeat step 5 until no change satisfy both criteria. Figure Figure 10 shows an illustrative summary of step 3 and step 4. Figure is the final graph we used in MNODE. The final graph we use for MNODE is therefore Figure 11, we showed it again with larger size for better readability.

D. Detailed Data Preparation Procedure for T1DEXI

D.1. T1DEXI Exercise Instances

Here we provide a step-by-step procedure of how we pre-processed the T1DEXI dataset described in Section 5.2.

We select patients on open-loop pumps with age under 40 and body mass index (BMI) less than 30. For each selected exercise instance in our dataset, we focus on the time window from 4 hours prior to the end of exercise, to 30 minutes after the end of exercise. We anchor time zero as the end of exercise, so this time window (in minutes) is $[-240, 30]$. Since CGM measurements are taken in 5 minute increments, we divide this time window into 5 minute increments, and divide the *end time* of each interval by 5, so that we obtain 54 discrete time steps: $t = -47, -46, -45, \ldots, 4, 5, 6$, with $t = 0$ denoting the end of exercise.

For each exercise instance, we use the following (54-dimensional) features derived from the T1DEXI data:

1. *CGM readings*. CGM readings are taken every 5 minutes for all patients in T1DEXI.

2. *Insulin injection*. If the patient injected insulin at a given time in the T1DEXI dataset, we add that amount of insulin to the corresponding 5 minute interval to create a bolus insulin injection time series. We add basal insulin to the bolus insulin time series to produce the (total) insulin time series used in our experiments.

3. *Carbohydrate intake*. Suppose the patient consumed carbohydrates at a given time in the T1DEXI dataset. We assume a constant meal consumption rate of 45 grams per minute, and then compute the average consumption rate over each 5 minute interval to create a carbohydrate consumption time series.

4. *Heart rate*. We average the heart rate over each 5 minute interval to obtain a heart rate time series.
5. *Step count*. Similarly, we average step counts over each 5 minute interval to obtain a step count time series.

We only consider exercise instances for which:

1. all 54 CGM readings were available;
2. the patient logged carbohydrates during the instance;
3. the exercise duration was at least 30 minutes;
4. it was either the first or second exercise instance for that patient; and
5. HR was consistently recorded every 10 seconds without any missingness throughout the 54 5-min intervals.

After this process we end up with 143 exercise instances (13 patients with 1 exercise instance, and 65 patients with 2 exercise instances).

Figure 12 depicts the raw and processed data for a single exercise time-window for a single patient. The first row shows a complete timeseries of CGM, while the second row shows the recorded changes in basal insulin rates. While insulin pump data only includes the change points (red dots in the second plot), the devices function by executing the new constant rate (blue line) until a new basal rate is set (a subsequent red dot). The third plot shows bolus insulin recordings (red dots), which are sometimes thought of as being delivered instantaneously. However, insulin pumps in fact administer these doses by delivering insulin at a constant rate over a suitable short time window (typically < 5 min), such that the sum total of disbursed bolus (integral of the blue curve in the third plot) corresponds to the recorded dose (red dot). In the case of insulin pumps, both the basal and bolus insulins are fundamentally the same drug and are only distinguishable by their administration pattern; for this reason, we convert all insulin administration to U/min and sum the basal and bolus, producing the black curve in the fourth figure.

The fifth plot shows the amount of carbohydrates (g) that were recorded by the participant (either as a meal or as part of the clinical study as "rescue" carbohydrates explicitly intended to mitigate hypoglycemia). We assume that meals are consumed at a constant $45g/min$ and plot the corresponding carbohydrate consumption rate in the sixth plot (blue); we then align this impulse function with the 5 minute grid defined by CGM data (black). Finally, the last two figures show heart rate and step counts, respectively. In both cases, we average the raw data (red) over each 5 minute window, and perform piecewise-constant interpolation of these values (blue). Observe that step count and heart rate begin to rise at $t = -60$; the exercise lasts for approximately 1 hour, then the step counts and heart rates drop for $t > 0$, once exercise is complete.

D.2. Intervention Sets

For each exercise time series, we first create an intervention set containing three copies of the original time series. And then we uniformly randomly (simulated by the numpy default random number generator with seed 2024) apply one of the four following intervention sets to it:

1. Add 0/50/100 grams of carbohydrate to the three copies at the end of exercise (time step $t = 0$);
2. Add 0/2.5/5 units of insulin to the three copies from the end of exercise onwards (time steps $t = 0, 1, \ldots, 6$);
3. Add nothing/50 grams of carbohydrate/10 units of insulin to the three copies at the end of exercise (time step $t = 0$);
4. Change the recorded heart rate in the last 7 time steps $t = 0, 1, \ldots, 6$ to typical medium intensity aerobic training (80,90,100,110,120,130,120)/ typical interval training (80,170,80,170,80,170,80)/typical high intensity resistance training (160,170,180,170,160,180,160).

For each exercise instance this leads to four time series (each consisting of 5 features and 54 time steps): the original observed time series, and three copies corresponding to the intervention set.

In addition, we compute the class label for each intervention set as the index of the intervention that leads to the highest mean glucose: 100 grams of carbohydrates for category 1; zero insulin for category 2; 50 grams of carbohydrate for category 3; and the highest heart rate level for category 4.

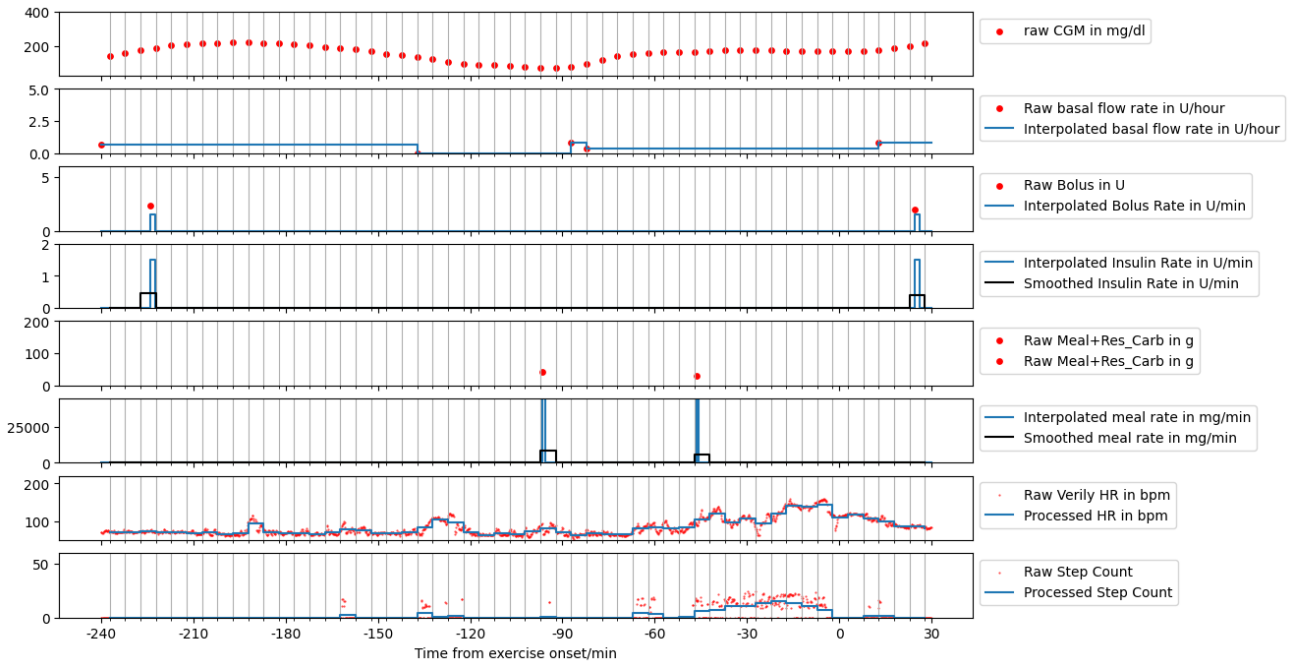


Figure 12. We show an example of the raw and processed data from a single patient window. Here, time 0 represents the end of exercise, and we plot all features for the 4 hours prior to exercise termination, as well as the subsequent 30 minutes. In all plots, red dots indicate raw measurements, blue lines represent our clinically-informed representations and interpolations of the raw data, and the black curves (when relevant) represent subsequent mapping of the interpolated data to 5 minute intervals.

E. Model Implementations and Hyperparameters

Set-up For each model mentioned in the experiment section, here we offer a detailed description of the corresponding computational method and the hyperparameters used. Throughout this section we are given an exercise time series of 54 time steps (corresponding to 54 5 minute intervals that made up the time window starting from 4 hours prior to exercise termination, and ending at 30 minutes after exercise termination) and 5 features (corresponding to CGM reading, insulin, carbohydrate, heart rate, and step count, in that order). We denote the first feature (CGM reading) as y, and the other 4 features as x. We use a subscript to indicate discrete time steps and superscript to indicate feature indices. For example, x_1^2 is the 2nd feature of x (carbohydrate) at discrete time step $t = 1$.

Our goal is to predict the CGM trace during the first 30 minutes following exercise completion corresponding to the output $y_{1:6} \in \mathbb{R}^6$ and therefore we set the number of prediction steps q to be 6 for all models. We further split the given time series into historical context $p = (y, x)_{-47:-1} \in \mathbb{R}^{47\times 5}$, starting glucose $y_0 \in \mathbb{R}$, inputs during exercise $x_{0:5} \in \mathbb{R}^{6\times 4}$ (six inputs that are recorded 1 time step ahead of the expected outputs). We use $\hat{y} \in \mathbb{R}^6$ to indicate the CGM trace predicted by models, s for modeled states and z for latent states. h, c for the final hidden state and cell state of the LSTM initial condition learner. For ease of computation and without loss of generality, we set the Δt term in forward-Euler style discretization to be 1 for all relevant models, and thus we omit it in the equations.

Learning Rate, Initialization and Optimizer For all experiments, we use the Adam optimizer (Kingma & Ba, 2015) to perform gradient descent.

Learning rates and weight initialization are set with model stability as a consideration. In particular, in all experiments using T1DEXI data, we set the default learning rate to be 2×10^{-3} for all models except UVA, for which we use a larger learning rate of 10^{-1}. For all models except UVA, we initialize model weights with the PyTorch default setting if they are part of a pre-defined PyTorch model class or standard normal if they are custom weights. For UVA we initialize model weights from a normal distribution with mean zero and variance $1/400$. UVA is treated differently because ODE-based mechanistic models are notoriously highly sensitive to parameter initialization, and it is both important to initialize the parameters to be small in magnitude, and to make sure they do not get too close to zero during training.

For synthetic experiments, we set the learning rate of the purely mechanistic model to 5×10^{-1}; to 1×10^{-2} for LP and LPSC; and to 2×10^{-3} for the remainder of the models.

Training Epochs Unless otherwise specified in the model description (LPSC), in T1DEXI experiments, we train for 100 epochs and pick the epoch with best validation loss. In synthetic experiments, we train for 50 epochs and pick the epoch with best validation loss.

Dropout For the hybrid models, we use 0 dropout rate. This is because we are already doing regularization via early stopping, and our trial runs indicate that there is no need for additional dropout. This is not true for black-box models, for which we still tune the dropout rate.

Hyperparameter Search We use grid search to tune hyperparameters. When choosing the grid, we restrict the search space to areas where the models have less than 20000 parameters and we also try to limit the number of grid points to around 10. This is to make sure the computational cost of the experiments is capped at a reasonable level for small data sets and individual users. The grid used for each model in each experiment will be provided below together with model descriptions.

E.1. UVA

UVA is our baseline mechanisitc model, whose description is given in Appendix A. The computation equation for the UVA model is given in Algorithm 1.

Algorithm 1 UVA Padova Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:5}$, the original UVA mechanistic ODEs m_{UVA}, $\Delta t = 1$
$h, c = \text{LSTM}(p)$
$h^1 = y_0$
$s_0 = h$
for $i = 0 : q - 1$ **do**
  $s_{i+1} = s_i + \Delta t \cdot m_{\text{UVA}}(s_i, x_i; \beta)$
  $\hat{y}_{i+1} = s^1_{t+1}$
end for
Output: $\hat{y}_{1:q}$

T1DEXI Experiments The LSTM network has 2 layers and 21 hidden dimensions (same as the UVA/Padova model), and we set the first state of the estimated initial condition (represented by hidden state h) to be the true initial value of CGM y_0 to keep consistency with the assumption that s^1 represents glucose. The UVA/Padova Simulator has 53 trainable parameters $\beta \in \mathbb{R}^{53}$. We do not tune hyperparameters for the UVA model.

Synthetic Experiments In the synthetic setting, we assume to know the ground truth model. And therefore we remove the LSTM initial condition encoder and instead set $s_0 = y_0$ since in the synthetic setting, there is only one state. Same as T1DEXI, we do not tune hyperparameters.

E.2. Reduced UVA Latent Parameter Learning

The Reduced UVA Latent Parameter Learning model uses a lower-fidelity, reduced version of UVA/Padova defined in Appendix B, and applies the idea of latent parameter learning to it. The computation equations are given in Algorithm 2

T1DEXI Experiments In our implementation, the latent dynamics of z only depends on itself and inputs that are not used by the UVA S2013 model (the 3rd and 4th feature, which correspond to heart rate and step count). We made this choice to keep consistency with the implementation in Miller et al. (2020). The LSTM has 2 layers and d hidden dimension (since this time the cell states are used to initialize latent state z, whose dimension needs to be tuned) and MLP2 serves to map d dimensional h_0 to a 8-dimensional initial condition vector. All MLPs have n hidden layers and m hidden units with dropout

Algorithm 2 Reduced UVA Latent Parameter Learning Model

Input: number of prediction steps q, historical context p, starting glucose y_0, exogenous inputs x, the reduced UVA mechanistic ODEs m_{RUVA}, $\Delta t = 1$
$h, c = \text{LSTM}(p)$
$s_0 = \text{concatenate}(y0, \text{MLP1}(h))$
$z_0 = c$
for $i = 0 : q - 1$ **do**
  $s_{i+1} = s_i + \Delta t \cdot m_{\text{RUVA}}(s_i, x_i; \beta_i = \text{MLP2}(z_i))$
  $z_{i+1} = Az_i + Bx_i^{3\sim4}$
  $\hat{y}_{i+1} = s_{t+1}^1$
end for
Output: $\hat{y}_{1:q}$

0 and activation ReLu, we tune these hyperpameters with grid search on the following grid:

$$n = \{2, 3, 4\} \times m = \{16, 32, 48\} \times d = \{8, 12, 16\}$$

Synthetic Experiments Similar to the purely mechanistic model, in the synthetic setting we directly set $s_0 = y_0$ and therefore do not use h or MLP1. We still need the 2-layer LSTM to generate z_0. The search grid for synthetic experiments is:

$$n = \{2, 3\} \times m = \{16, 32\} \times d = \{2, 4\}$$

E.3. Reduced UVA Latent Parameter and State Closure Learning

The Reduced UVA Latent Parameter and State Closure Learning model uses the same reduced UVA/Padova defined in Appendix B, and applies both the idea of latent parameter learning and state closure learning to it. The computation equations are given in Algorithm 3

Algorithm 3 Reduced UVA Latent Parameter and State Closure Learning Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:5}$, the reduced UVA mechanistic ODEs m_{RUVA} the adjacency matrices (as defined in Section 3) of reduced UVA: A_s, A_x, closure switch constant w, $\Delta t = 1$
$h, c = \text{LSTM}(p)$
$s_0 = \text{concatenate}(y_0, \text{MLP1}(h))$
$z_0 = c$
for $i = 0 : q - 1$ **do**
  $s_{i+1} = s_i + \Delta t \cdot m_{\text{RUVA}}(s_i, x_i; \beta_i = \text{MLP2}(z_i)) + w \cdot \text{MLPs}(s_i, x_i; A_s, A_x)$
  $z_{i+1} = Az_i + Bx_i^{3\sim4}$
  $\hat{y}_{i+1} = s_{t+1}^1$
end for
Output: $\hat{y}_{1:q}$

T1DEXI Experiments The LSTM has 2 layers and d hidden dimension (since this time the cell states are used to initialize latent state z, whose dimension needs to be tuned). MLP1 and MLP2 have n hidden layers and m hidden units with dropout 0 and activation ReLu, while MLPs that form the closure learning network have 2 hidden layers and m hidden units with dropout 0 and activation ReLu. We tune these hyperpameters with grid search on the following grid:

$$n = \{2, 3\} \times m = \{16, 32\} \times d = \{8, 16\}$$

Note that our search space is smaller as the introduction of extra MLPs significantly increases the total number of parameters.

Synthetic Experiments We again set $s_0 = y_0$ and do not use h or MLP1. The hyperparameter grid is:

$$n = \{2, 3\} \times m = \{16, 32\} \times d = \{8, 16\}$$

Special Training Routine The training routine of models adopting state closure is more complicated. We first set $w = 0$ and just train the reduced UVA latent parameter model and the initial condition learner LSTM for 100(T1DEXI)/50(Synthetic) epochs (the closure part is masked out by w). Then we freeze the weights of the reduced UVA latent parameter model and the LSTM, set $w = 1$ and train the closure neural network on the residuals for 50 more epochs. This trick makes sure the closure network is truly learning the residuals as intended and not subsuming the mechanistic model.

E.4. Mechanistic Neural ODE

The MNODE model no longer relies on the functional forms of UVA models, and therefore does not require any ODE equations m. Instead, it uses the adjacency matrices of a reduced version of the UVA/Padova graph. We obtain this reduced graph with the reduction heuristic described in Appendix C. We provide its computation equations in Algorithm 4.

Algorithm 4 Mechanistic Neural ODE Model

Input: number of prediction steps $q = 6$, historical context p,starting glucose y_0, exogenous inputs $x_{0:5}$, the adjacency matrices (as defined in Section 3) of the reduced UVA graph A_s, A_x, $\Delta t = 1$
$h, c = \text{LSTM}(p)$
$h^1 = y_0$
$s_0 = h$
for $i = 0 : q - 1$ **do**
  $s_{i+1} = s_i + \Delta t \cdot \text{MLPs}(s_i, x_i; A_s, A_x)$
  $\hat{y}_{i+1} = s^1_{t+1}$
end for
Output: $\hat{y}_{1:q}$

T1DEXI Experiments The LSTM has 2 layers and 5 hidden dimension (this corresponds to the number of states in the reduced graph) and for the same reason we set the 1st features of the initial condition state vector to be y_0. All MLPs have n hidden layers and m hidden units with dropout 0 and activation ReLu, we tune these hyperpameters with grid search on the following grid:

$$n = \{2, 3\} \times m = \{16, 24, 32\}.$$

Synthetic Experiments We set $s_0 = y_0$ and do not use h or the LSTM (note that our code implementation still has the LSTM component for the sake of consistency but its output is actually never used by MNODE.) The search grid is:

$$n = \{2, 3\} \times m = \{16, 32\}.$$

E.5. BNODE

For BNODE and the subsequent black-box models, we point the reader to implementations referenced in the associated citations in the main paper. Here we describe our implementation.

Algorithm 5 Black-box Neural ODE Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:5}$, $\Delta t = 1$
$h, c = \text{LSTM}(p)$
$h^1 = y_0$
$s_0 = h$
for $i = 0 : q - 1$ **do**
  $s_{i+1} = s_i + \Delta t \cdot \text{MLPs}(s_i, x_i)$
  $\hat{y}_{i+1} = s^1_{t+1}$
end for
Output: $\hat{y}_{1:q}$

T1DEXI Experiments The LSTM has 2 layers and d hidden dimension. Note that here the hidden dimension of LSTM also determines the state dimension of the neural ODE, which is a tunable hyperparameter. All MLPs have n hidden layers and m hidden units with dropout a and activation ReLU, we tune these hyperpameters with grid search on the following grid:

$$d = \{4, 5, 6\} \times n = \{2, 3\} \times m = \{32, 48, 60\} \times a = \{0, 0.1, 0.2\}.$$

Synthetic Experiments We keep all the settings the same (note we can no longer assume there is only 1 state as in the black-box regime we have zero domain knowledge) as the T1DEXI experiments except that the search grid for hyperparameters is now:

$$d = \{2, 3\} \times n = \{2, 3\} \times m = \{64\} \times a = \{0, 0.2\}.$$

E.6. TCN

Algorithm 6 Temporal Convolutional Network Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:5}$
$\tilde{x} = \mathbf{0} \in \mathbb{R}^q$
$\tilde{x}_0 = y_0$
$x' = \text{concatenate}(\tilde{x}, x, \text{dim} = -1)$
$seq_{in} = \text{concatenate}(p, x', \text{dim} = 0)$
$seq_{out} = \text{TCN}(seq_in)$
$\hat{y} = \text{Linear}(seq_{out})$
Output: $\hat{y}$

T1DEXI Experiments The TCN model is taken directly from the code repository posted on https://github.com/locuslab/TCN/blob/master/TCN/tcn.py, with input size set to 5, number of channels set to a list of n copies of m, kernel size set to l and dropout set to a. We tune these hyperpameters with grid search on the following grid:

$$n = \{2, 3\} \times m = \{16, 24, 32\} \times l = \{2, 3, 4\} \times a = \{0, 0.1, 0.2\}.$$

Synthetic Experiments We keep all the settings the same as the T1DEXI experiments except that the input size is set to 3 and the search grid for hyperparameters is now:

$$n = \{2, 3\} \times m = \{16, 32\} \times l = \{2, 3, 4\} \times a = \{0, 0.2\}.$$

E.7. LSTM

Algorithm 7 Long Short Term Memory Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:5}$
$h, c = \text{Encoder LSTM}(p)$
Set initial hidden state and cell state of Decoder LSTM to h, c respectively
$seq_{out}, h_q, c_q = \text{Decoder LSTM}(x_{0:5})$
$\hat{y} = \text{Linear}(seq_{out})$
Output: $\hat{y}$

T1DEXI Experiments Both Encoder and Decoder LSTM have n layers and d hidden states with dropout set to a. We tune these hyperpameters with grid search on the following grid:

$$n = \{2, 3.4\} \times m = \{8, 12, 16\} \times a = \{0, 0.1, 0.2\}.$$

Synthetic Experiments

$$n = \{2, 3\} \times m = \{8, 16\} \times a = \{0, 0.2\}.$$

Algorithm 8 Transformer Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:q-1}$, true output $y_{1:q-1}$ (needed during training)
$\tilde{x} = \mathbf{0} \in \mathbb{R}^q$
$\tilde{x}_0 = y_0$
$x' = \text{concatenate}(\tilde{x}, x, \text{dim} = -1)$
encoder_in $= \text{concatenate}(p, x', \text{dim} = 0)$ (concatenating all inputs to form a masked context)
if Model in Training Mode **then**
  decoder_in $= \text{concatenate}(y_0, y_{1:q-1})$ (expected output shifted to the right)
  decode_out $= \text{Transformer}(\text{encoder\_in}, \text{decoder\_in}, \text{decoder\_causal\_mask})$
end if
if Model in Evaluation Mode **then**
  decoder_in $= \text{concatenate}(y_0, \mathbf{0} \in \mathbb{R}^{q-1})$
  for $i = 1 : q - 1$ **do**
    decode_out $= \text{Transformer}(\text{encoder\_in}, \text{decoder\_in})$
    $\text{decoder\_in}_{i+1} = \text{decode\_out}_i$
  end for
  decode_out $= \text{Transformer}(\text{encoder\_in}, \text{decoder\_in})$
end if
$y = \text{Linear}(\text{decode\_out})$
Output: y

E.8. Transformer

T1DEXI Experiments We use the transformer model provided by the pytorch nn class, and its hyperparameters are set as follows: d_model set to d, number of encoder layers set to l_1, number of decoder layers set to l_2, the dim_feedforward is set to m and dropout is set to a. We tune the hyperparameters with the following grid:

$$d = \{4, 8\} \times l_1 = \{2, 3\} \times l_2 = \{2, 3\} \times m = \{32, 64\} \times a = \{0, 0.1\}$$

Synthetic Experiments We keep all experiment settings the same except setting $q = 10$ and the hyperparameter search grid to be:

$$d = \{4, 8\} \times l_1 = \{2\} \times l_2 = \{2\} \times m = \{32, 64\} \times a = \{0, 0.2\}$$

E.9. S4D

Algorithm 9 S4 Diagonal Model

Input: number of prediction steps $q = 6$, historical context p, starting glucose y_0, exogenous inputs $x_{0:q-1}$
$\tilde{x} = \mathbf{0} \in \mathbb{R}^q$
$\tilde{x}_0 = y_0$
$x' = \text{concatenate}(\tilde{x}, x_{0:5}, \text{dim} = -1)$
$seq_{in} = \text{concatenate}(p, x', \text{dim} = 0)$
$seq_{in} = \text{Linear}(Seq_{in})$
$seq_{in} = \text{Transpose}(Seq_{in}, 1, 2)$
$seq_{out} = \text{S4D}(seq_i n)$
$seq_{out} = \text{Transpose}(Seq_{out}, 1, 2)$
$\hat{y} = \text{Linear}(seq_{out})_{-q:}$
Output: $\hat{y}$

T1DEXI Experiments We take the S4D model directly from the following github repository https://github.com/thjashin/multires-conv/blob/main/layers/s4d.py, and its hyperparameters are set as: d_model set to d,

d_state set to m, dropout set to a.We tune the hyperparameters with the following grid:

$$d = \{4, 6, 8\} \times \{m = \{32, 64\} \times a = \{0, 0.1, 0.2\}$$

Synthetic Experiments We set $q = 10$ and the hyperparameter search grid to be:

$$d = \{3, 4, 5\} \times \{m = \{32, 64\} \times a = \{0, 0.2\}$$

F. Repeated Nested Cross Validation

We use the following algorithm to evaluate both predictive and causal generalization errors of our models.

Algorithm 10 Repeated Nested Cross Validation

Input: Data D, Model M, Hybrid Loss Function l_h, Alpha for Hybrid Loss α, List of Hyperparameter Settings Λ, Number of Repeats $R = 3$, Number of Outer Fold $N = 6$, Number of Inner Fold $M = 4$, Random Seed $s = 2024$
Generate R different permutations of $[1, \ldots, \text{length}(D)]$: $P_1, \ldots, P_R$ with numpy default random number generator and seed s
Initialize Error Lists $e_{\text{pred}}, e_{\text{causal}}$
for $r = 1 : R$ **do**
  Permute D with permutation P_r, save the resulting data as D_r
  Split D_r into N folds $D_r^1, \ldots D_r^N$
  for $i = 1 : N$ **do**
    Form the outer training set as $D_{\text{tout}} = D_r \setminus D_r^i$, and set the test set as $D_{\text{test}} = D_r^i$
    Split D_{tout} into M folds $D_{\text{tout}}^1, \ldots, D_{\text{tout}}^M$
    for $j = 1 : M$ **do**
      Form the training set as $D_{\text{train}} = D_{\text{tout}} \setminus D_{\text{tout}}^j$
      Form the validation set as $D_{\text{val}} = D_{\text{tout}}^j$
      if $j < M$ **then**
        for each $\lambda \in \Lambda$ **do**
          Set pytorch seed to $s + r - 2$
          Standardize $D_{\text{train}}, D_{\text{val}}, D_{\text{test}}$ with sample mean and standard deviation of D_{train}
          Train M on D_{train} with hyperparameter setting λ, add the best validation hybrid loss $l_h(\cdot, D_{\text{val}}; \alpha)$ achieved during training to the score of λ
        end for
      end if
      if $j = M$ **then**
        Select λ^* from Λ with the lowest score
        Set pytorch seed to $s + r - 2$
        Standardize $D_{\text{train}}, D_{\text{val}}, D_{\text{test}}$ with sample mean and standard deviation of D_{train}
        Train M on D_{train} with hyperparameter setting λ^*, select the epoch M^* with best validation hybrid loss $l_h(\cdot, D_{\text{val}}; \alpha)$.
        Add $l_h(M^*, D_{\text{test}}; \alpha = 0)$ to e_{pred}
        Add $l_h(M^*, D_{\text{test}}; \alpha = 1)$ to e_{causal}
      end if
    end for
  end for
end for
Output: $e_{\text{pred}}, e_{\text{causal}}$

G. Additional Experiments

In this section we provide details of additional experiments that may be of interest to readers. We carried out three sets of additional experiments: (1) we apply novel interventions to the test set that are unseen in the training set; (2) we corrupt a

small portion of training ranking information; and (3) we test hybrid models that are based on full (un-reduced) mechanistic models.

G.1. Different Intervention Sets in Training and Test Sets

In this experiment, the training intervention sets consist of only category (1) and (2) described in Section 5.2 (Intervention Sets). Crucially, note that our training intervention sets *do not* include interventions where both insulin and carbohydrates can vary, as in category (3) in Section 5.2; this ensures that our training is *only* teaching the models about dependence on insulin alone and carbs alone. On the other hand, the test intervention set contains only the following category: add 45 grams of carbohydrate at the end of exercise, and at the same time add 2.25/3.00/4.50 units of insulin. These interventions correspond to a insulin-carb ratio of 1:20/1:15/1:10 respectively. We keep other experimental settings the same as the main T1DEXI experiments.

Results The results of the experiments are summarized in Figure 13. The results illustrate that for moderate values of α that balance the prediction loss with the causal loss, the causal error rate of most models improves even in this setting where the test intervention set contains interventions not seen in the training data. Further, we see that in general, the hybrid models offer more rapid improvement in causal loss as α increases, with virtually no increase in predictive loss. BNODE appears to offer comparable performance to MNODE, but with generally higher variance in its causal loss.

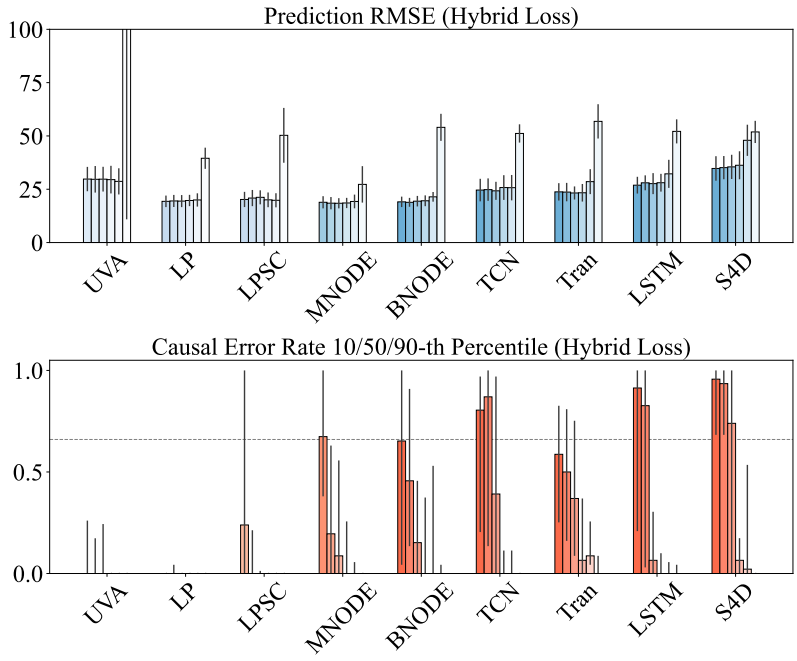


Figure 13. As in Fig. 4, but for T1DEXI data and using a full UVA/Padova mechanistic model baseline and hybridizations of the reduced UVA/Padova simulator for LP, LPSC, and MNODE. In these experiments, the training data includes only interventions on either insulin or carb intake, but not both together; however, the test data includes interventions on the insulin-carb ratio.

G.2. Corrupted Ranking Knowledge in Training Set

In this experiment, the ground truth ranking of each training intervention set in $\mathcal{I}^*$ has a probability of $\rho \in \{0.05, 0.10, 0.20\}$ of being corrupted by circularly shifting the ranking to the right by 1 position. We carry out these simulations only for MNODE as a representative example. We keep the testing ground truth intact and other experimental settings the same as the main T1DEXI experiments.

Results The results of the experiments are summarized in Figure 14. We see that at moderate values of α, the performance on causal tasks degrades with increasing levels of corruption. Notably, we also see that when the corruption rate is high, *high*

α (high weight on the causal loss) can actually *deteriorate* causal performance, suggesting that the model is erroneously overemphasizing rankings from the corrupted data (introducing bias). Regardless, even when the corruption rate is high, choosing any $\alpha > 0$ (i.e., using a hybrid loss) lowers the causal error rate relative to a pure predictive loss ($\alpha = 0$).

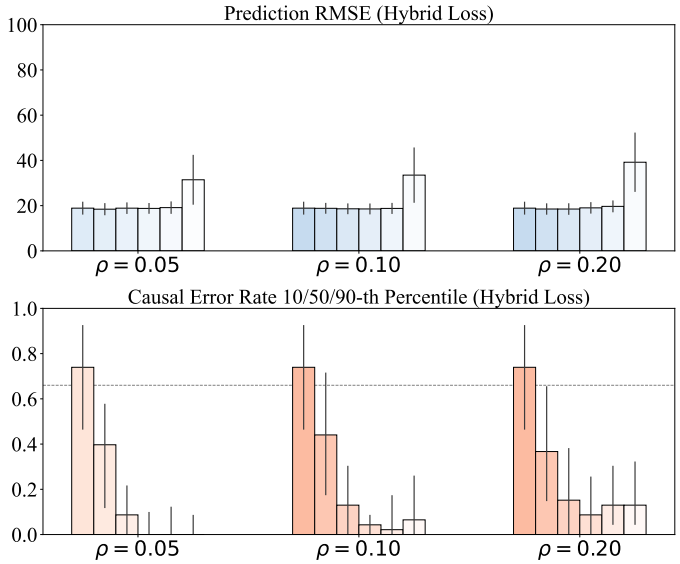


Figure 14. As in Fig. 4, but for T1DEXI data with MNODE, using a hybridization of the reduced UVA/Padova simulator. In these experiments, a fraction ρ of the training data is corrupted.

G.3. Hybrid Models Based on Full UVA/Padova

In the main paper, we work with a reduced UVA/Padova model. For completeness, we also implemented the latent parameter (LP) model and mechanistic neural ODE (MNODE) model using the full UVA/Padova model described in Appendix A. Their implementations are exactly the same as described in Appendix E except that they use model equations/adjacency matrices of the full UVA/Padova model, and their hyperparameters are tuned on slightly different grids to count for the drastic increase in the number of mechanistic states and parameters. The search grid for hyperparameters of each full hybrid model is given below:

1. LP:
$$n = \{2, 3\} \times m = \{16, 24, 32\} \times d = \{20, 24, 28\}$$
2. MNODE:
$$n = \{2, 3\} \times m = \{16, 32\}.$$

Note that we did not include latent parameter learning with state closure (LPSC) in this set of simulations, as its full version requires an unreasonably large number of both states and model parameters when applied to the full UVA/Padova model.

Results The results of the experiments are summarized in Figure 15. We see that in general, the hybrid models based on the reduced UVA/Padova simulator offer generalization performance on par with those of hybrid models based on the full simulator. One of the primary benefits of building off of a reduced mechanistic model is that the resulting hybrid model has significantly reduced training time. These results suggest model reduction is a valuable (but not necessary) implementation step in applying H^{2}NCM in practice.

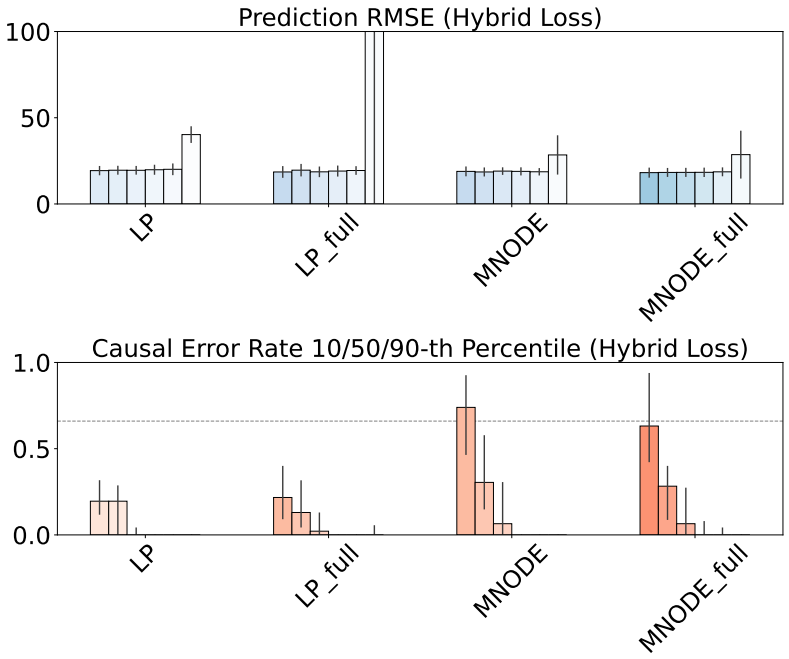


Figure 15. As in Fig. 4, but for T1DEXI data and comparing LP and MNODE hybrid models based on both the full and reduced UVA/Padova simulator as the mechanistic model.