Global Deep Forecasting with Patient-Specific Pharmacokinetics

Willa Potosnak WPOTOSNA@ANDREW.CMU.EDU
Auton Lab, School of Computer Science, Carnegie Mellon University
Cristian Challu CRISTIAN@NIXTLA.IO
Nixtla
Kin G. Olivares KIGUTIE@AMAZON.COM
Amazon
Keith A. Dufendach DUFENDACHKA@UPMC.EDU
University of Pittsburgh Medical Center, Division of Cardiac Surgery
Artur Dubrawski AWD@ANDREW.CMU.EDU
Auton Lab, School of Computer Science, Carnegie Mellon University

Abstract

Forecasting healthcare time series data is vital for early detection of adverse outcomes and patient monitoring. However, it can be challenging in practice due to variable medication administration and unique pharmacokinetic (PK) properties of each patient. To address these challenges, we propose a novel hybrid global-local architecture and a PK encoder that informs deep learning models of patient-specific treatment effects. We showcase the efficacy of our approach in achieving significant accuracy gains in a blood glucose forecasting task using both realistically simulated and real-world data. Our PK encoder surpasses baselines by up to 16.4% on simulated data and 4.9% on real-world data for individual patients during critical events of severely high and low glucose levels. Furthermore, our proposed hybrid global-local architecture outperforms patient-specific PK models by 15.8%, on average.

Data and Code Availability We use open-source simulated and real-world data that is available to other researchers. The simulated data includes an open-source Python implementation of the FDA-approved UVa/Padova Simulator (2008 version), which is contained in the 'simglucose' GitHub repository (Xie, 2018; Visentin et al., 2016). The real-world data includes the OhioT1DM 2018 and 2020 datasets (Marling and Bunescu, 2020). Information on accessing this dataset, including a link to the required Data Use Agreement (DUA), is provided in the paper by Marling and Bunescu (2020). Implementations of models used in our work were obtained from the open-source `Neuralforecast` (Olivares et al., 2022b) and `StatsForecast` (Garza et al., 2022) libraries. We implemented our method by adopting the `Neuralforecast` repository to leverage their model training framework. Our research code, along with the `Neuralforecast` models used in our research, is available at `https://github.com/PotosnakW/neuralforecast/tree/pk_paper_code`.

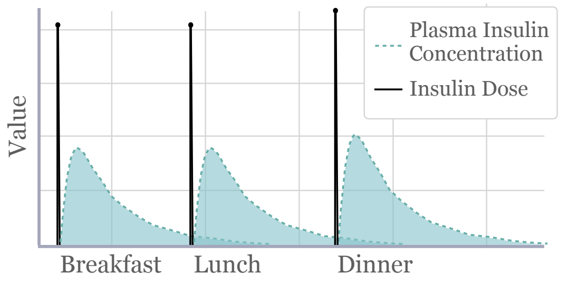

Figure 1: Insulin dose is recorded, but plasma insulin concentration is not directly measurable and typically requires invasive procedures and laboratory tests.

Institutional Review Board (IRB) Our research does not require IRB approval.

1. Introduction

Leveraging the predictive power of exogenous variables presents unique challenges, especially in healthcare, where the frequency of patient signals and exogenous factors, such as medication doses or treatments, often exhibit a mismatch in temporal resolution. Furthermore, while the drug dose can be observed and recorded, the body's interaction with the drug cannot be directly observed and is typically inferred from its plasma concentration, as illustrated in Fig. 1. Consequently, the recorded drug dose alone cannot capture time-dependent pharmacokinetics, including processes such as absorption and elimination.

Consider the task of blood glucose monitoring for patients with Type-1 Diabetes Mellitus (T1DM) or insulin-dependent T2DM patients who must regularly monitor and regulate their blood glucose levels. Careful and prompt administration of insulin and intake of carbohydrates is required to prevent hyper- and hypoglycemic events, which account for numerous hospital admissions annually, and can cause irreversible organ damage or, in severe cases, be fatal. A 2016 report estimated that hypoglycemia causes approximately 100,000 emergency department (ED) visits per year due to insulin-related errors, with nearly one-third resulting in hospitalization (Geller et al., 2014). Another study reports that in 2014, there were 184,255 ED visits and in-patient U.S. hospital admissions for diabetic ketoacidosis (DKA), often associated with hyperglycemia, with 70.6% occurring in T1DM patients (Benoit et al., 2020). Another study reported a trend toward increasing hospitalizations for DKA in T1DM patients from 2008 to 2018 (Kichloo et al., 2021).

For healthcare challenges like this, advanced forecasting technologies can help patients and clinicians monitor, anticipate, and proactively address emerging abnormalities to prevent adverse events. However, accurate forecasting requires models that can leverage exogenous information, such as treatments, while capturing latent treatment kinetics and their effects to generate reliable forecast trajectories. To address this, we propose a hybrid model that integrates cohort-level information with patient-specific pharmacokinetics to improve the prediction of blood glucose responses to treatment.

The main contributions of this paper are:

(i) **Hybrid Global-Local Architecture.** We bolster global architectures (with parameters shared across patients) with a learnable, patient-specific encoder.

(ii) **Pharmacokinetic (PK) Encoder.** We introduce a novel deep learning model-agnostic module that generates plasma concentration profiles to capture time-dependent treatment effects.

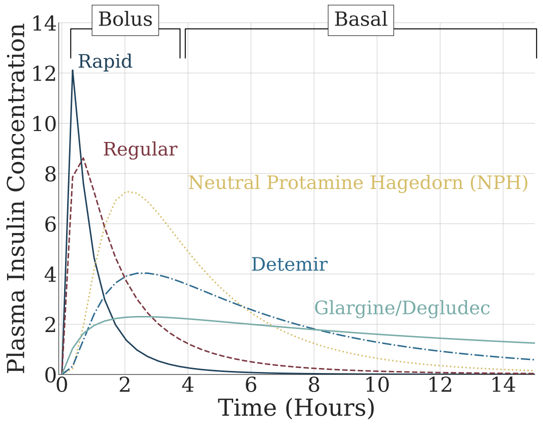

Figure 2: Pharmacokinetic models of plasma insulin concentration over time for various insulin types.

(iii) **Multiple Dose Effects Framework.** We provide a computationally efficient framework for encoding the effects of multiple doses. We verify this approach mathematically under the assumption of linear pharmacokinetics.

(iv) **State-of-the-Art Results.** We obtain significant accuracy improvements on large-scale simulated and real-world patient blood glucose forecasting datasets.

1.1. Pharmacokinetics Background

Pharmacokinetic (PK) modeling is a natural solution to the aforementioned problem, as it can reveal medication effects over time through well-established knowledge of changes in plasma insulin concentration driven by absorption and elimination characteristics (Binder et al., 1984). Empirical and invasive studies show that concentration-time profiles for subcutaneous injections follow right-skewed curves. The extent of skew varies with insulin type and patient-specific factors, as shown in Fig. 2, where the area under the concentration curve measures drug *bioavailability*, or the fraction of the absorbed dose that reaches the intended site of systemic circulation intact. Most drugs used in clinical practice are assumed to follow linear PK (Rosenbaum, 2017). Under this model, the area under the concentration curve is directly proportional to the drug dose. A complete reference for supporting PK information regarding concentration-time profiles, insulin analogs, bioavailability, first-order kinetics, and linear PK can be found in Appendix B.

2. Related Work

Blood Glucose Forecasting. Neural Forecasting has previously been used to predict blood glucose levels, outperforming traditional statistical methods. RNNs and LSTMs have been used both as models and as baselines in numerous studies (Qian et al., 2021; Li et al., 2019a; Fox et al., 2018; Rabby et al., 2021; Rubin-Falcone et al., 2020, 2022; Miller et al., 2020). Similarly, CNNs (Xie and Wang, 2020; Li et al., 2019a,b) and MLPs (McShinsky and Marshall, 2020; Miller et al., 2020; Jahangir et al., 2017) have been used in proposed models and as baselines.

Modeling Blood Glucose Treatment Effects. Prior work has looked at modeling treatment effects on blood glucose levels. Miller et al. introduced one of the first hybrid statistical-physiological models, the Deep T1D Simulator (DTD-Sim), which integrates glucose-insulin dynamics with deep learning through a deep state-space model (Miller et al., 2020). Odnoblyudova et al. (2023) improved the prediction of blood glucose using multi-output Gaussian processes (MOGP) to model the composite effects of meal components, such as carbohydrates and fats. Unlike meals, insulin is a homogeneous protein hormone, making component-based modeling inapplicable to our task. Instead, our objective is to capture the holistic patient-specific effects of the drug and the composite effects of multiple doses.

Despite advances in treatment effect modeling, current approaches face limitations, including (1) reliance on individualized, subject-specific models and (2) inflexible architectures that hinder adaptability to various base models, forecasting tasks, and model inputs. The next paragraph outlines these limitations and how our proposed contributions address them.

Previous work has focused on developing individualized networks, often overlooking the advantages of global models that learn from multiple, diverse time series—a technique known as cross-learning (Smyl, 2019; Semenoglou et al., 2021), and employed by Time Series Foundation Models (TFSMs) (Goswami et al., 2024). Our hybrid global-local architecture enables cohort-level training to develop a global model applicable to multiple patients while also preserving patient-specific information for personalized forecasts. Regarding architectures, models such as the UVA / Padova simulator depend on predefined ODEs for glucose dynamics, restricting adaptability to other targets and exogenous inputs (Visentin et al., 2016; Miller et al., 2020). Similarly, deep learning approaches for sparse exogenous variables often rely on fixed base models like LSTMs (Rubin-Falcone et al., 2022). Our proposed contributions support diverse deep learning models, and are flexible to forecasting targets and input variables, ensuring future adaptability as more advanced tools emerge. Other prior work that proposes the integration of PK knowledge with deep learning focuses on domains such as long-term disease progression in cancer and COVID-19, where such PK models are not directly applicable to blood glucose and insulin kinetics. (Hussain et al., 2021; Qian et al., 2021). Our study provides a formal framework for integrating PK in deep learning under linear PK assumptions that are applicable to many drugs used in clinical practice (Rosenbaum, 2017).

3. Methods

We aim to forecast blood glucose values 30 minutes into the future consistent with prior work (Rubin-Falcone et al., 2022; Xie and Wang, 2020; Li et al., 2019b), based on a 10-hour history of blood glucose, carbohydrate (CHO), insulin basal, and insulin bolus values. We choose a 10-hour duration to balance sufficient historical information with our GPU memory constraints. Moreover, the duration of rapid-acting insulin (Humalog and Novolog) used in insulin pumps by patients in our cohort typically ranges from 2 to 5 hours (Clinic, 2018; Association, n.d.). Thus, a 10-hour time span fully captures multiple dose durations, enabling the model to learn treatment effects across various dosages and durations. We also demonstrate that the significant performance improvement of the proposed methods holds across other various lag times in Appendix C.3.

3.1. PK Encoder

Medication administrations are recorded as instantaneous dosing events, leading to sparse observations in time series data. We propose a novel PK encoder that informs deep learning models of time-dependent plasma drug concentrations to enable more accurate forecasting of treatment outcomes as shown in Fig. 3. The encoder consists of a function C that generates a concentration-time profile for a single specified dose x at time t using a PK parameter k affecting elimination rate:

$$C(t, x, k) = \frac{x}{tk\sqrt{2\pi}} \exp\{-\frac{1}{2k^2}(\log(t) - 1)^2\}. \quad (1)$$

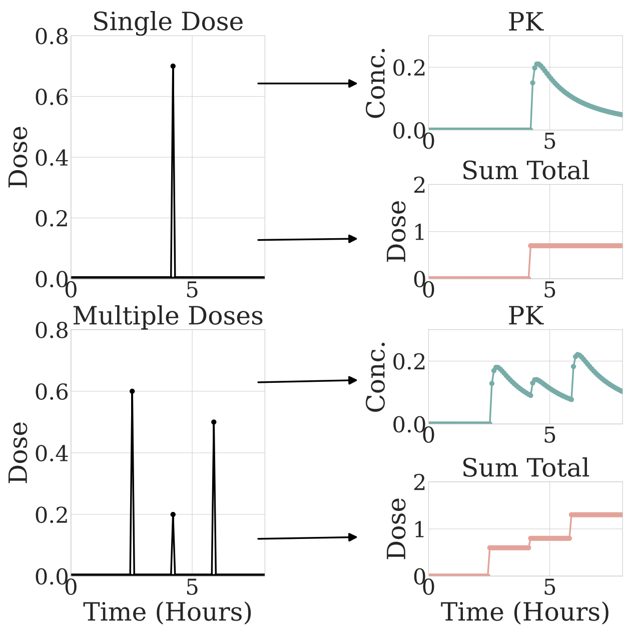

Figure 3: Medication administrations are represented as sparse variables in time series data. We propose a pharmacokinetic (PK) encoder to effectively capture time-dependent plasma drug concentration (Conc.).

Here, C is equivalent to the log-normal probability density function with $\sigma = k$ and $\mu = 1$ and a scaling factor x. The intuition for the log-normal shape of the concentration curves stems from PK literature, as outlined in Section 1.1 and Appendix B.

Under linear PK assumptions, the PK parameters, including bioavailability, are assumed to be constant with respect to the drug. Given this assumption, we show mathematically that the area under the concentration curve is bounded by the dose in Proposition 1 in Appendix A.5. To model bioavailability and incorporate dose information directly into the concentration curves, we use the dose x as a scaling factor in C. This ensures that, under the simplifying assumption of 100% bioavailability, the area under the curve integrates to the dose. Scaling the concentration curve in this way affects its magnitude but does not alter PK parameters that determine its shape.

Let N be the number of subjects in the dataset. The PK parameter vector $\mathbf{k} \in \mathbb{R}^{N\times 1}$ represents patient-specific embedding weights that control the shape of the patient-specific concentration curves, where $\mathbf{k}^{(i)}$ corresponds to the PK parameter k for the i-th patient. The vector $\mathbf{k}$ is initialized with values between 1 and 2 prior to model training, with the initial value treated as a tunable hyperparameter.

The encoder is trained end to end with the model. During training, a sparse treatment feature $\mathbf{x}^{(i)} \in \mathbb{R}^{1\times L}$, representing dose administrations within an input window of length L up to and including the current time T, is passed to the PK encoder along with the corresponding parameter $\mathbf{k}^{(i)}$ to generate a concentration curve feature $\check{\mathbf{x}}^{(i)} \in \mathbb{R}^{1\times L}$, as shown in Fig. 4. In the *single-dose setting*, where a dose is recorded at time step $t_d \in \{T-L+1,\dots,T\}$, the encoder replaces the sparse treatment feature $\mathbf{x}^{(i)}_{T-L+1:T}$ with the corresponding concentration feature $\check{\mathbf{x}}^{(i)}_{T-L+1:T}$, computed as:

$$\check{\mathbf{x}}^{(i)}_t = \begin{cases} 0, & \text{for } t \in [T-L+1, t_d) \\ C\left(\nu(t-t_d), \mathbf{x}^{(i)}_{t_d}, \mathbf{k}^{(i)}\right), & \text{for } t \in [t_d, T] \end{cases} \tag{2}$$

where t indexes time steps within the window, ν is the sampling interval of the data, and $C(\cdot)$ computes the concentration at elapsed time $\nu(t-t_d)$ since the dose was administered. During training, the parameter $\mathbf{k}^{(i)}$ is updated for each patient i via gradient descent.

Forecasts $\hat{\mathbf{y}}$ are generated by the model f_θ for each time step within the forecast horizon H as a function of historical glucose values $\mathbf{y}$, treatment concentration curves $\check{\mathbf{x}}$, and static features $\mathbf{s}$, recorded within the input window up to and including time T,

$$\hat{\mathbf{y}}^{(i)}_{T+1:T+H} = f_\theta(\mathbf{y}^{(i)}_{T-L+1:T}, \check{\mathbf{x}}^{(i)}_{T-L+1:T}, \mathbf{s}^{(i)}). \tag{3}$$

In the proposed context, $\mathbf{s}$ includes patient age, weight, one-hot encoded subject identification number, and insulin pump type. $\check{\mathbf{x}}_{T-L+1:T}$ includes concentration curve values for basal insulin $\check{\mathbf{x}}_{\text{basal}}$, bolus insulin $\check{\mathbf{x}}_{\text{bolus}}$, and CHO $\check{\mathbf{x}}_{\text{CHO}}$. We include treatment effects of carbohydrates because, similar to drugs, they undergo absorption and metabolism processes, resulting in glucose and other sugars entering the bloodstream.

3.2. Modeling Multiple Dose Effects

Multiple insulin doses may occur at close intervals in an event often referred to as "insulin stacking" as shown in Fig. 3 and Fig. 15. To model PK effects for multiple doses within an input window, we generate a concentration curve for each dose event and sum the concentration curves within the window. Under the assumption of linear PK (outlined in 1.1

and Appendix B), we show that the overall effect, or concentration, of n doses of the same drug is equal to the sum of the individual effects which is verified mathematically in Proposition 2 in Appendix A.5. To efficiently generate concentration curves for multiple doses within an input window, the PK encoder uses a matrix $\mathbf{w} \in \mathbb{R}^{L\times L}$ to support vectorized computation at each timestamp containing a dose, within the lag time L. This approach enables efficient modeling of multiple effects in $\mathcal{O}(1)$ time. Each row t of $\mathbf{w}$ represents the elapsed time steps until the end of the window, with the time count shifting one step to the right in each subsequent row. A concentration curve is generated for each row of $\mathbf{w}$ using the dose at the corresponding time index. If no dose is administered at a particular time t, the corresponding row of $\mathbf{w}$ is zeroed out. The final concentration curve feature, which captures the combined impact of multiple doses, is obtained by aggregating the individual curves across the rows:

$$\mathbf{w} = \begin{bmatrix} 0 & 1 & 2 & 3 & \cdots & L-1 \\ 0 & 0 & 1 & 2 & \cdots & L-2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{L\times L} \tag{4}$$

$$\breve{\mathbf{x}}^{(i)} = \sum_{t=0}^{L-1} C(\nu \mathbf{w}_{t,:}, \mathbf{x}_t^{(i)}, \mathbf{k}^{(i)}) \tag{5}$$

Here, ν is the sampling interval of the data, $\mathbf{x}^{(i)} \in \mathbb{R}^{1\times L}$ is a sparse exogenous time series treatment feature, and $\breve{\mathbf{x}}^{(i)} \in \mathbb{R}^{1\times L}$ is the new concentration curve feature for patient i.

The PK encoder training algorithm 1 is described in Appendix A.6, which can be integrated into any deep learning architecture capable of handling exogenous variables in addition to modeling the target variable.

3.3. Modeling Dose-Dependent Effects

A majority of drugs are assumed to adhere to linear pharmacokinetics (PK), where a linear relationship is expected between drug dose and its concentration in the body (Rosenbaum, 2017). In practice, however, whether a drug's PK parameters follow linear or nonlinear behavior can vary based on factors such as the drug's physicochemical properties, the route of administration, and the injection site. A key characteristic of the linear PK assumption is that PK parameters remain constant and do not change with variables such as dose or time. Dose- or time-dependent nonlinearity occurs when PK parameters change nonproportionally with dose or time, respectively. In dose-dependent nonlinear PK, the relationship between drug dose and body concentration is no longer linear, resulting in dose-related changes in PK parameters. While insulin is not expected to exhibit classic time-dependent nonlinearity, it may demonstrate dose-dependent nonlinearity. For instance, subcutaneous injections are administered into the subcutaneous tissue, where higher doses can saturate elimination pathways, causing slower absorption rates (Søeborg et al., 2009; Gradel et al., 2018). When PK parameters are dose-dependent, the resulting plasma drug concentration profiles also vary with dose.

Insulin pumps administer the same type of insulin (rapid-acting) for the basal and bolus doses. However, bolus insulin doses are generally much larger than basal doses as highlighted in Figs. 13 and 14 in Appendix A.1. To account for potential dose-dependent effects, we model basal and bolus doses separately by learning distinct PK parameters, $\mathbf{k}_{\text{basal}}$ and $\mathbf{k}_{\text{bolus}}$, allowing us to capture PK variations without resorting to a fully nonlinear model. This approach maintains linear assumptions within each dose type, simplifying the modeling process while still capturing differences between low-dose (basal) and high-dose (bolus) PK. For a single dose of basal insulin $\mathbf{x}_{\text{basal}}$ or bolus insulin $\mathbf{x}_{\text{bolus}}$, administered at time step t_d, we compute the plasma insulin concentration at time $t \geqslant t_d$ for patient i:

$$\breve{\mathbf{x}}_{\text{basal}}^{(i)} = C\left(\nu(t-t_d), \mathbf{x}_{t_d,\text{basal}}^{(i)}, \mathbf{k}_{\text{basal}}^{(i)}\right), \tag{6}$$

$$\breve{\mathbf{x}}_{\text{bolus}}^{(i)} = C\left(\nu(t-t_d), \mathbf{x}_{t_d,\text{bolus}}^{(i)}, \mathbf{k}_{\text{bolus}}^{(i)}\right). \tag{7}$$

Treating each dose category as a distinct entity is an approximation and may not capture all nuances of dose-dependent nonlinear PK. All physiological models are approximations of complex biology. Our goal is to evaluate a novel deep learning-based PK modeling approach based on established linear PK knowledge, while accounting for potential dose-dependent effects, balancing simplicity and precision. This framework also ensures broader applicability across various medications, as most drugs are assumed to follow linear PK (Rosenbaum, 2017).

The equations presented are shown in a patient-specific manner to highlight that the PK encoder generates individualized information. However, these equations can be applied to produce patient-specific

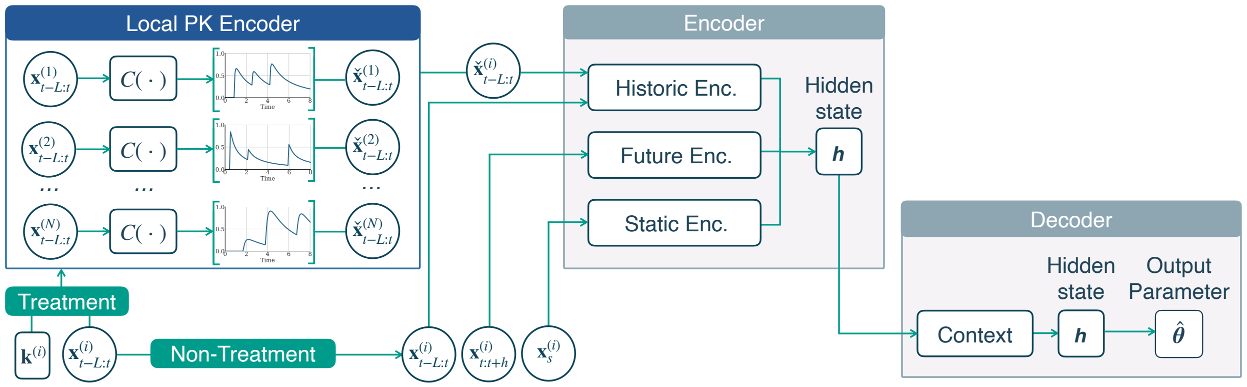
Local PK Encoder
$\mathbf{x}^{(1)}_{t-L:t}$
$C(\cdot)$
$\check{\mathbf{x}}^{(1)}_{t-L:t}$
$\mathbf{x}^{(2)}_{t-L:t}$
$\check{\mathbf{x}}^{(2)}_{t-L:t}$
$\mathbf{x}^{(N)}_{t-L:t}$
$\check{\mathbf{x}}^{(N)}_{t-L:t}$
Time
Treatment
$\mathbf{k}^{(i)}$
$\mathbf{x}^{(i)}_{t-L:t}$
Non-Treatment
$\check{\mathbf{x}}^{(i)}_{t-L:t}$
$\mathbf{x}^{(i)}_{t:t+h}$
$\mathbf{x}^{(i)}_{s}$
Encoder
Historic Enc.
Future Enc.
Static Enc.
Hidden state
h
Decoder
Context
Output Parameter
$\hat{\theta}$

Figure 4: Our hybrid global-local architecture combines global model parameters shared across patients with patient-specific pharmacokinetic (PK) parameters. Sparse treatment time series features, such as medication doses, are input into the PK encoder to generate patient-specific treatment concentration curves $\check{\mathbf{x}}$. PK encoder output $\check{\mathbf{x}}$ and any other exogenous features $\mathbf{x}$ are used as deep learning model inputs. During each training step, the model learns relevant hidden states and output parameters $\hat{\theta}$ across patients, while $\mathbf{k}^{(i)}$ values are updated for each patient i using gradient descent optimization.

outputs for multiple patients simultaneously using vectorized operations.

3.4. Models

To assess forecasting performance across various model architectures, we trained global models with 11 different algorithms as baselines. Additionally, to assess our proposed hybrid global-local architecture, we compare the best performing model with the PK encoder to both purely local and global models.

Statistical and Deep Learning Baselines. We employed Exponential Smoothing (`AutoETS`; Hyndman 2008) as our statistical baseline model. Our deep learning baselines include models from various architectural categories, such as Linear models, Multi-Layer Perceptron (MLP)-based, Recurrent Neural Network (RNN)-based, Convolutional Neural Network (CNN)-based, and Transformer-based models. We employ the following deep learning models: `DLinear` (Zeng et al., 2023), `MLP` (Rosenblatt, 1958), `NHITS` (Challu et al., 2022), `NBEATSx` (Oreshkin et al., 2020; Olivares et al., 2022a), `RNN` (Elman, 1990; Cho et al., 2014), Long-Short-Term Memory (`LSTM`; Sak et al. (2014)), Temporal Convolutional Network with an `MLP` decoder (`TCN`; Bai et al. (2018); van den Oord et al. (2016)), `Informer` (Zhou et al., 2021), patch time series Transformer (`PatchTST`; Nie et al. (2023)), and Temporal Fusion Transformer (`TFT`; Lim et al. (2021)). More information on these models and related prior work is provided in Appendix A.2. Details on model training and hyperparameter selection are provided in Appendix A.3.

Global Model Baselines. Global models are trained on all patient data to create a single model. Global model baselines include: (1) models trained with only the blood glucose signal and static features, (2) models trained with blood glucose signals, static features, and sparse historical exogenous features (CHO and insulin dose values), and (3) models trained with blood glucose signals, static features, and sparse historical exogenous features transformed using the "Sum Total" (ST) approach, which converts features into a cumulative sum of values across an input window (Rubin-Falcone et al., 2022). We include models trained without exogenous variables as baselines to show the impact of incorporating sparse exogenous information for each model. Together with models trained on sparse and Sum Total features, these baselines demonstrate the importance of effective transformations—such as our PK encoder—in reducing forecast error.

With the exception of `TFT`, most Transformer-based models, including `Informer` and `PatchTST`, are not designed to model covariates. For these two models, we can only report global model results for those trained on univariate blood glucose data. Since our proposed PK encoder is intended to supplement model architectures capable of incorporating exogenous variables alongside the endogenous signal, we focus on evaluating the PK encoder in top-performing models that are capable of modeling covariates, such as `TFT`, `NBEATSx`, and `NHITS`. Beyond modeling covariates, residualized MLP-based models like `NBEATSx` and `NHITS` are valuable to demonstrate the utility of our PK encoder as `NHITS` has achieved SOTA performance, outperforming RNNs and Transformers in long-range forecasting tasks while avoiding the quadratic complexity of self-attention in Transformer-based architectures (Challu et al., 2022; Ansari et al., 2024).

Local Model Baselines. Patient-specific models using sparse exogenous inputs, as well as those incorporating the PK encoder, were compared against our hybrid global-local PK encoder model. These baselines highlight the effectiveness of hybrid global-local architectures that leverage cross-learning from multiple time series.

3.5. Model Training

Models were trained using `Adam` optimizer (Kingma and Ba, 2017). Optimal hyperparameters were selected via cross-validation on a validation set by minimizing Huber loss across all prediction time points. Given the use of forward-filling to account for missing values in the OhioT1DM dataset, timestamps with forward-filled values were omitted from the validation set. Huber estimator was selected as the loss function, $\mathcal{L}$, given its ability to reduce the impact of outliers and offer more stable results compared to other functions:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}, \delta) = \begin{cases} \frac{1}{2}(\mathbf{y} - \hat{\mathbf{y}})^2, & \text{if } |\mathbf{y} - \hat{\mathbf{y}}| \leqslant \delta, \\ \delta(|\mathbf{y} - \hat{\mathbf{y}}| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases} \tag{8}$$

More information on model training and hyperparameters in provided in Appendix A.3.

3.6. Model Evaluation

Trained models were used to generate rolling window forecasts at one-step intervals. Similar to prior work (Rubin-Falcone et al., 2022; Xie and Wang, 2020), we evaluate forecasts for each test set in the two datasets using two metrics: mean absolute error (MAE) and root mean squared error (RMSE). We compute results across all points within the forecast horizon, and present the average results over 8 trials of individually trained models. We also evaluate the performance of the models exclusively at time

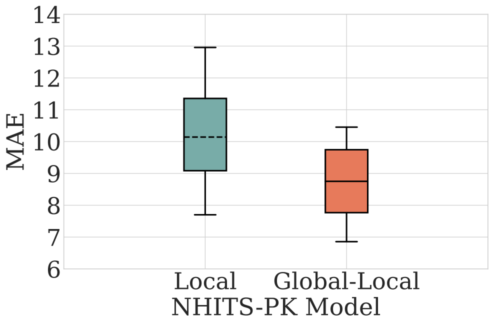

Figure 5: The hybrid global-local `NHITS`-PK model trained on all OhioT1DM individuals (orange, solid median) has a lower MAE than `NHITS`-PK local models (blue, dashed median).

points where blood glucose levels reach or fall below critical thresholds relevant to hypoglycemic and hyperglycemic events (i.e., $\leqslant$70 mg/dL and $\geqslant$180 mg/dL). For the OhioT1DM dataset, timestamps with forward-filled values were omitted from evaluation. To evaluate whether hybrid global-local PK models significantly outperform baselines, we use paired t-tests computed on the patients' mean MAE results. Metric formulations and statistical tests are detailed in Appendix A.4.

4. Cohort

We use simulated and real-world datasets containing patient insulin, CHO time series, and blood glucose measurements. These open-source datasets ensure reproducibility and are among the largest publicly available cohorts for blood glucose forecasting, making them widely used in current research within this domain (Xie and Wang, 2020; Rubin-Falcone et al., 2020, 2022; Rabby et al., 2021; Li et al., 2019b).

Simulated data. An open-source python implementation of the FDA-approved UVa/Padova Simulator (2008 version) (Xie, 2018; Visentin et al., 2016) provides blood glucose, CHO, and insulin values for 30 simulated patients with Type-I diabetes. We generate 54 days of data for each patient to match the median training set of the OhioT1DM, and approximately 9 days for the test set. Appendix A.1 contains examples of blood glucose time series for 6 individuals, shown in Fig. 11, as well as basal and bolus dose distributions for 12 individuals, shown in Fig. 13.

Table 1: Mean absolute error (MAE) computed for model predictions across all values in the forecast horizon and across only critical values in the forecast horizon (blood glucose $\leqslant 70 \mid \geqslant 180$). Models are separated with horizontal lines into statistical, linear, RNN-based, CNN-based, Transformer-based, and MLP-based, and PK model groups, respectively. Result for models with (w/) and without (w/o) exogenous (exog.) variables are shown where applicable. Best results are highlighted in **bold**, second best results are underlined. PK models with improved forecasts over their respective exogenous baseline model are highlighted in blue.

	Simulated Dataset				**OhioT1DM Dataset**			
	All Values		**Critical Values**		**All Values**		**Critical Values**	
Model	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.
AutoETS	9.450	–	13.457	–	9.022	–	10.028	–
DLinear	7.802	7.989	10.575	10.790	11.514	13.011	12.512	14.580
RNN	12.728	12.930	20.308	20.945	11.535	10.907	12.641	12.989
LSTM	9.250	9.152	12.034	12.000	10.217	10.615	12.102	12.983
TCN	14.340	11.380	23.370	14.817	10.769	11.516	12.812	13.203
Informer	8.599	–	12.074	–	13.714	–	14.867	–
PatchTST	7.284	–	9.842	–	9.922	–	10.999	–
TFT	6.587	5.826	8.518	7.132	9.402	9.197	10.577	10.643
MLP	10.424	8.407	13.932	10.297	10.300	13.254	11.550	14.997
NBEATSx	9.187	6.886	12.757	8.145	9.868	10.280	11.037	11.359
NHITS	8.268	7.304	11.114	9.094	9.060	8.873	10.337	10.137
TFT-PK	–	**5.743**	–	**7.042**	–	9.529	–	10.461
NBEATSx-PK	–	6.806	–	7.998	–	9.983	–	10.927
NHITS-PK	–	7.037	–	8.492	–	**8.758**	–	**9.965**

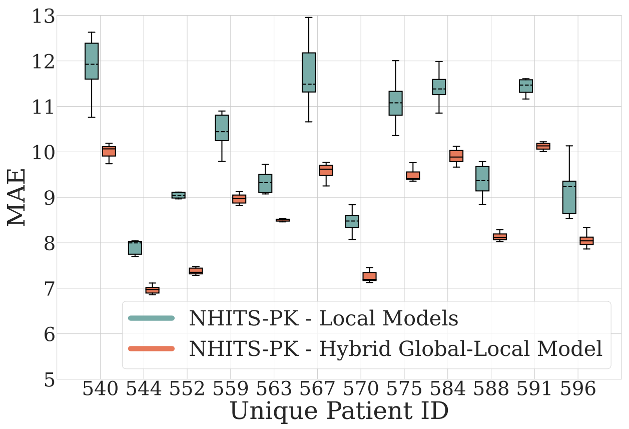

Figure 6: The NHITS-PK hybrid global-local model (orange, solid median) consistently outperforms the local NHITS-PK model (blue, dashed median).

Real-world data. Twelve de-identified individuals with Type 1 diabetes are included in the OhioT1DM 2018 and 2020 datasets (Marling and Bunescu, 2020). Each patient has approximately 8 weeks of data that includes blood glucose (mg/dL) measurements recorded with a continuous glucose monitor (CGM) at a frequency of 5-minutes, basal rate information, bolus insulin (units), and CHO values. The maximum number of test samples consistent across all patients (2691 timestamps) was used to obtain equal test sets of approximately 9 days. Missing values were forward filled to prevent data leakage. Sample counts and missing data percentages for each subject are included in Table 4 in Appendix A.1. Appendix A.1 also contains examples of blood glucose time series plots for 6 subjects, shown in Fig. 12, as well as basal and bolus dose distributions for the 12 patients, shown in Fig. 14.

5. Results

PK-encoder models outperform their sparse exogenous model counterparts. For simulated data, the TFT-PK model achieves the lowest MAE, on average, among all models for both evaluations across all models and critical values. The PK models, including TFT-PK, NBEATSx-PK, and NHITS-PK outperform their sparse exogenous model counterparts by up to 5.4%, 3.7%, 9.2%, respectively, on evaluations across all values and 7.7%, 6.6%, 16.4% on evaluations across critical values for individual patients. For the OhioT1DM dataset, the `NHITS`-PK model achieves the lowest MAE, on average, among all models for both evaluations across all values and critical values. The PK models, including, `TFT`-PK, `NBEATSx`-PK, and `NHITS`-PK, outperform their sparse exogenous model counterparts by up to 5.2%, 8.5%, 4.9% on evaluations across critical values for individual patients. Mean MAE and RMSE results and their standard deviations computed across 8 trials are shown in Appendix C.7 Tables 11 and 12.

Hybrid global-local models significantly improve forecasts over global models. For the simulated dataset, the `TFT`-PK, `NBEATSx`-PK, and `NHITS`-PK models achieve significantly lower forecast errors for patients for evaluations on all values (p-values: 4.6e-4 [`TFT`-PK]; 7.9e-5 [`NBEATSx`-PK]; 2.0e-7 [`NHITS`-PK]) and critical values (p-values: 5.7e-4 [`NBEATSx`-PK]; 3.0e-5 [`NHITS`-PK]) compared to their sparse exogenous global model counterparts. For the OhioT1DM dataset, the `NBEATSx`-PK and `NHITS`-PK models also achieve significantly lower forecast errors for patients for both evaluations on all values (p-values: 3.4e-3 [`NBEATSx`-PK]; 1.3e-3 [`NHITS`-PK]) and critical values (p-values: 4.4e-4 [`NBEATSx`-PK]; 7.0e-3 [`NHITS`-PK]) compared to their sparse exogenous global model counterparts. The `TFT`-PK model does not outperform its sparse exogenous counterpart for $L = 120$. However, for $L = 180$, it achieves significantly lower error (p-value: 4.8e-2) than its sparse exogenous variant and records the lowest error among all `TFT` model variants, as shown in Table 8. Additionally, Table 2 shows that `TFT`-PK and `NHITS`-PK models achieve significantly lower errors, on average—2.4% (p-value: 3.6e-12) and 2.3% (p-value: 4.1e-5), respectively—for subjects in evaluations across all values, compared to their "Sum Total" model counterparts.

Hybrid global-local models significantly improve forecasts over local models. For the OhioT1DM dataset, the `NHITS`-PK hybrid global-local model outperforms local `NHITS`-PK models developed for individual OhioT1DM patients by 15.8%, on average (p-value: 1.7e-5) as shown in Fig. 5. Furthermore, this improvement in forecast accuracy is consistent across all patients as shown in Fig. 6 and Table 3 (standard deviations are provided in Table 6 in Appendix C.1). Our hybrid global-local `NHITS`-PK model also significantly outperforms local sparse exogenous baseline models by 14.8%, on average, for individual subjects (p-value: 1.7e-6).

Table 2: PK models outperform Sum Total (ST) models for both Simulated and OhioT1DM datasets. The best results are in **bold**.

	Simulated Dataset		OhioT1DM Dataset	
	`TFT`-PK	**`TFT`-ST**	**`NHITS`-PK**	**`NHITS`-ST**
All Values	**5.743** (0.159)	5.885 (0.187)	**8.758** (0.080)	8.958 (0.087)
Critical Values	**7.042** (0.270)	7.233 (0.282)	**9.965** (0.095)	10.131 (0.100)

Table 3: MAE computed across forecast horizon values for local `NHITS`, global `NHITS`, local `NHITS`-PK, and hybrid global-local `NHITS`-PK on the 12 subjects from the OhioT1DM dataset. The best results are in **bold**.

ID	w/ Exog. (Sparse)		w/ Exog. (PK)	
	Local	Global	Local	Hybrid
540	12.106	10.119	11.880	**10.015**
544	7.952	7.188	8.067	**6.967**
552	8.559	7.400	9.132	**7.374**
559	10.845	9.013	13.051	**8.967**
563	9.527	8.731	9.427	**8.506**
567	12.599	9.640	12.132	**9.562**
570	8.361	7.523	8.466	**7.258**
575	10.891	9.712	11.095	**9.496**
584	11.111	**9.846**	11.518	9.905
588	10.172	8.172	9.376	**8.137**
591	11.987	10.246	11.428	**10.151**
596	8.990	8.133	9.147	**8.059**
All	10.324	8.873	10.425	**8.758**

The `NHITS`-PK model effectively learns the inhibitory treatment effects of insulin on blood glucose. We examine whether the `NHITS`-PK model can accurately capture treatment effects of insulin on blood glucose by generating forecasts for three scenarios: the original bolus insulin doses, bolus insulin doses removed, and bolus insulin doses increased by a factor of 10. When insulin doses are removed, blood glucose predictions increase, and when insulin doses are augmented, blood glucose predictions decrease as shown in Fig. 8. This result underscores the importance of models that can effectively capture treatment effects, enabling them to adapt to new treatment doses for more reliable and accurate forecasts.

6. Discussion

The proposed PK encoder can substantially reduce forecasting errors in deep learning models for both

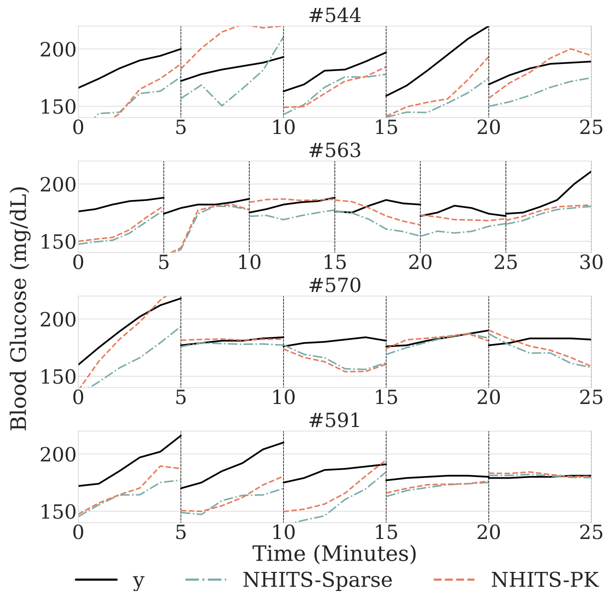

Figure 7: Blood glucose forecasts of the final horizon point during 5 randomly selected hyperglycemic periods (blood glucose $\geqslant$ 180) for 4 randomly selected OhioT1dm patients. Forecasts for the NHITS-PK model (orange, dashed) are more aligned with ground truth (black, solid) compared to the NHITS sparse exogenous model blue, dash-dotted).

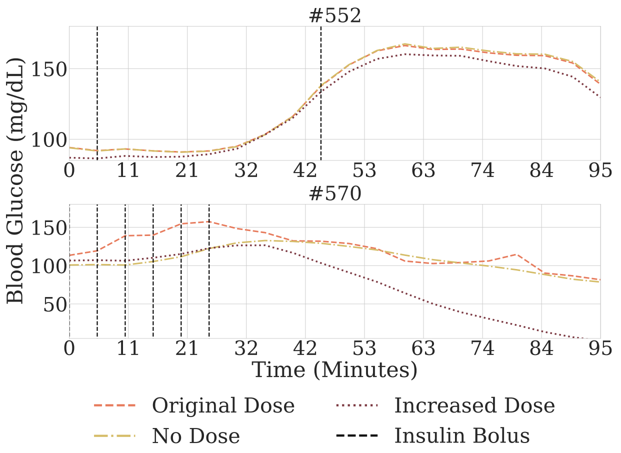

Figure 8: The NHITS-PK model captures insulin treatment effects: forecasts with removed bolus doses (yellow, dash-dotted) and augmented doses (red, dotted), result in higher and lower glucose predictions, respectively, compared to the original doses (orange, dashed). Each point represents the final forecasted time step from a rolling window forecast.

simulated and real-world data. In addition to providing more accurate forecasts during hyperglycemic events, our PK encoder predicts these events earlier, up to 3.6 minutes compared to the NHITS sparse exogenous baseline model, for individual OhioT1DM patients, as shown in Fig. 10. The TFT-PK model achieves a true positive rate of 0.96 on the simulated dataset, while the NHITS-PK model reaches 0.89 on the OhioT1DM dataset, showcasing their effectiveness in predicting critical events while leaving room for further improvement. A complete analysis of hyper- and hypoglycemic event prediction is provided in Appendix C.4.

In addition to improving performance over global model baselines, our hybrid global-local model architecture produces forecasts with significantly smaller errors than patient-specific models. This result is important in demonstrating that global models can integrate cohort-level information with patient-specific parameters to produce more accurate personalized forecasts. Evaluations on the OhioT1DM dataset also demonstrate that our hybrid global-local model remains effective even with 20% of missing timestamps in the training set for individual patients, as shown in Table 3, with missing data detailed in Table 4. Moreover, our proposed hybrid global-local model offers practical benefits beyond improved forecasting performance as it requires less computational resources and time to train than individual patient models. Training patient-specific NHITS-PK models took approximately seven times longer than training a single NHITS-PK hybrid model, despite using a common hyperparameter space. Additionally, the PK models are as computationally efficient as or more efficient than their sparse exogenous counterparts in terms of floating-point operations per second (FLOPs) and the number of trainable parameters, as shown in Table 9 in Appendix C.5.

Our study demonstrates that leveraging PK domain knowledge to integrate exogenous variables in models offers performance improvements over using sparse features or the “Sum Total” feature engineering approach. Furthermore, dose-dependent PK encoding strategies, such as those proposed in Section 3.3, can lead to more realistic models, as highlighted by our ablation study in Appendix C.2 where we demonstrate that using our PK encoder to infer the same **k** value for both bolus and basal insulin, rather than separate values, results in worse performance on the OhioT1DM dataset.

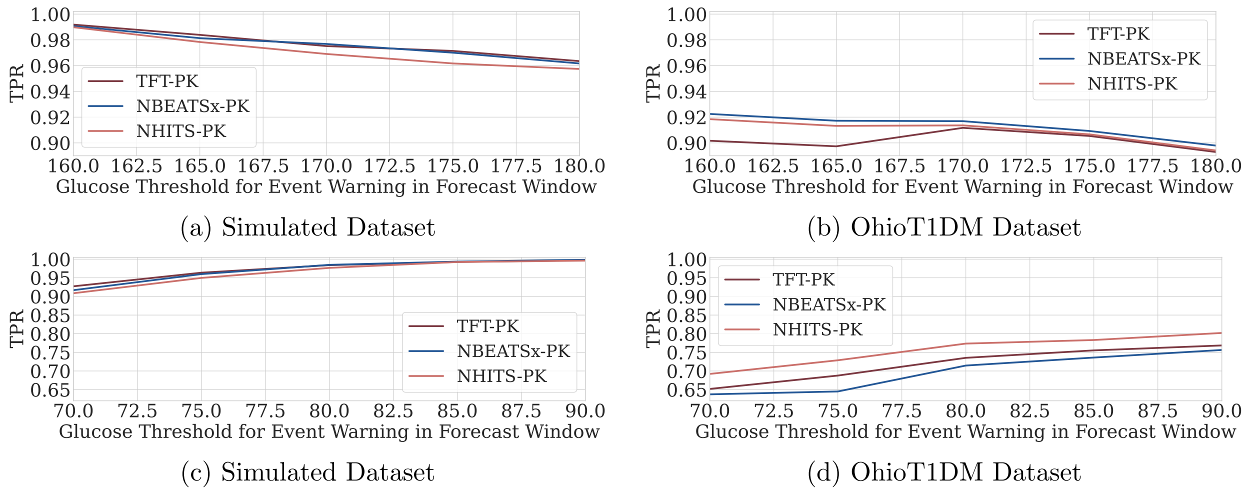

Figure 9: True positive rate (TPR) for the Simulated and OhioT1DM datasets at various glucose thresholds for **(a, b) hyperglycemia (blood glucose** $\geqslant 180$**)** and **(c, d) hypoglyemia (blood glucose** $\leqslant 70$**)** event warnings. TPR represents the proportion of forecast horizon windows in which the model correctly predicts an event when one actually occurs. The best performing `TFT`-PK model for the Simulated dataset predicts hyperglycemic and hypoglycemic events in forecast windows at an approximate TPR of 0.96 and 0.93, respectively. The best performing `NHITS`-PK model for the OhioT1DM dataset predicts hyperglycemic and hypoglycemic events in forecast windows at an approximate TPR of 0.89 and 0.69, respectively.

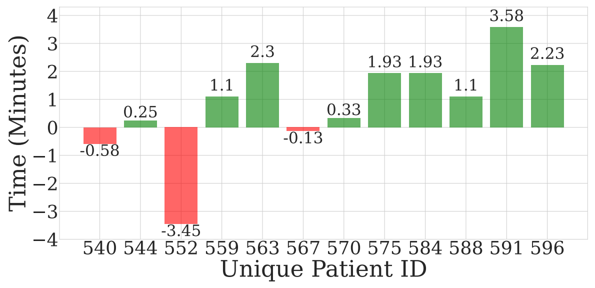

Figure 10: The PK encoder enables improved `NHITS` model prediction lead times for hyperglycemic events (CGM $\geqslant 180$) for 9 of 12 OhioT1DM patients.

Our PK encoder can have multiple beneficial applications in clinical practice, such as issuing early warnings of unexpected treatment responses and characterizing patient-specific effects for personalized treatment dosing. We discuss the feasibility of integrating such deep learning models into clinical decision-support systems in section C.6. While we demonstrate performance improvements across the top three models, our approach can be easily integrated into other deep learning architectures as new tools emerge. It can also be adapted for other types of medication administration, such as oral or intravenous injections.

Limitations and Future Work. This study has two limitations related to the PK encoder: we assume 100% bioavailability and a consistent peak concentration across patients. These assumptions focus model learning on the parameter k, rather than the area or peak of the concentration curve, to help mitigate potential issues with parameter identification. Future work will explore expanding concentration curve parameters as well as using nonlinear PK frameworks to potentially enable more flexible concentration curves. The proposed method is demonstrated on T1DM patient data but has potential to be adapted to other forecasting tasks and populations, such as insulin-dependent T2DM patients, with appropriate cohort data, PK model adjustments, and model training. Future work will apply our approach to additional cohorts, including T2DM, as more data becomes available to further validate our findings. Future work will also explore methods for estimating patient-specific parameters in a global model for new patients with limited or no historical data.

Acknowledgments

This work has been partially supported by the NIH awards R01NR013912 and R01DK131586, and NSF awards 2406231 and 2427948. The authors wish to thank Dr. Frank Guyette, Dr. Leonard Weiss and Dr. Gilles Clermont for their guidance and informative conversations.

References

Mohn Angelika, M. Loredana Marcovecchio, and Francesco Chiarelli. Insulin analogues. *The New England journal of medicine*, 352:1822–4; author reply 1822, 05 2005. doi: 10.1056/NEJM200504283521721.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024.

James Arthur Harris and Francis Gano Benedict. *A biometric study of basal metabolism in man.* Carnegie institution of Washington, 21919.

American Diabetes Association. Insulin basics, n.d. URL https://diabetes.org/health-wellness/medication/insulin-basics.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

Stephen R. Benoit, Israel Hora, Francisco J. Pasquel, Edward W. Gregg, Ann L. Albright, and Giuseppina Imperatore. Trends in emergency department visits and inpatient admissions for hyperglycemic crises in adults with diabetes in the u.s., 2006–2015. *Diabetes Care*, 43(5):1057—-1064, 2020.

Christian Binder, Torsten Lauritzen, Ole Faber, and Stig Pramming. Insulin pharmacokinetics. *Diabetes care*, 7(2):188–199, 1984.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-HiTS: Neural hierarchical interpolation for time series forecasting. In *AAAI-23*, 2022.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Cleveland Clinic. Injectable insulin medications, 2018. URL https://my.clevelandclinic.org/health/drugs/13902-injectable-insulin-medications.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Ian Fox, Lynn Ang, Mamta Jaiswal, Rodica Pop-Busui, and Jenna Wiens. Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1387–1395, 2018.

Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biol. Cybernetics*, 20: 121—136, 1975.

Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022. URL https://github.com/Nixtla/statsforecast.

Andrew I. Geller, Nadine Shehab, Maribeth C. Lovegrove, , Scott R. Kegler, Kelly N. Weidenbach, Gina J. Ryan, and et al. National estimates of insulin-related hypoglycemia and errors leading to emergency department visits and hospitalizations. *JAMA Internal Medicine*, 174(5):678—686, 2014.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *41st International Conference on Machine Learning*, 2024.

A. K. J. Gradel, T. Porsgaard, J. Lykkesfeldt, T. Seested, S. Gram-Nielsen, N. R. Kristensen, and H. H. F. Refsgaard. Factors affecting the absorption of subcutaneously administered insulin: Effect on variability. *J Diabetes Res*, 2018:1205121, 2018. doi: 10.1155/2018/1205121. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6079517/.

Zeshan M Hussain, Rahul G Krishnan, and David Sontag. Neural pharmacodynamic state space modeling. In *International Conference on Machine Learning*, pages 4500–4510. PMLR, 2021.

Rob Hyndman. *Forecasting with exponential smoothing: the state space approach*. Springer Berlin, Heidelberg, 2008.

Maham Jahangir, Hammad Afzal, Mehreen Ahmed, Khawar Khurshid, and Raheel Nawaz. An expert system for diabetes prediction using auto tuned multi-layer perceptron. In *2017 Intelligent systems conference (IntelliSys)*, pages 722–728. IEEE, 2017.

Asim Kichloo, Zain El-amir, Farah Wani, and Hafeez Shaka. Hospitalizations for ketoacidosis in type 1 diabetes mellitus, 2008 to 2018. *Proceedings (Baylor University. Medical Center)*, 35(1):1–5, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, 2017.

Kezhi Li, John Daniels, Chengyuan Liu, Pau Herrero, and Pantelis Georgiou. Convolutional recurrent neural networks for glucose prediction. *IEEE journal of biomedical and health informatics*, 24(2): 603–613, 2019a.

Kezhi Li, Chengyuan Liu, Taiyu Zhu, Pau Herrero, and Pantelis Georgiou. GluNet: A deep learning framework for accurate glucose forecasting. *IEEE journal of biomedical and health informatics*, 24(2): 414–423, 2019b.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, and Joseph E Gonzalez. Ray tune: A scalable hyperparameter tuning library. GitHub, 2018. URL https://github.com/ray-project/ray.

Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

Cindy Marling and Razvan Bunescu. The ohiot1dm dataset for blood glucose level prediction: Update 2020. In *CEUR Workshop Proc*, 2020.

Richard McShinsky and Brandon Marshall. Comparison of forecasting algorithms for type 1 diabetic glucose prediction on 30 and 60-minute prediction horizons. In *KDH@ ECAI*, pages 12–18, 2020.

Andrew C. Miller, Nicholas J. Foti, and Emily Fox. Learning insulin-glucose dynamics in the wild. In *Proceedings of Machine Learning Research*, pages 1–25, 2020.

Vinod Nair and Jeoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML-23*, 2010.

Yuqi Nie, Nam H. Nguyen, and Phanwadee Sinthong an Jayant Kalagnanam2. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Arina Odnoblyudova, Çağlar Hizli, ST John, Andrea Cognolato, Anne Juuti, Simo Särkkä, Kirsi Pietiläinen, and Pekka Marttinen. Nonparametric modeling of the composite effect of multiple nutrients on blood glucose dynamics. *Proceedings of Machine Learning Research*, 225:1–16, 2023.

Kin G. Olivares, Cristian Challu, Grzegorz Marcjasz, Rafal Weron, and Artur Dubrawski. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with nbeatsx. *International Journal of Forecasting*, 39(2):884–900, 2022a.

Kin G. Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022, 2022b. URL https://github.com/Nixtla/neuralforecast.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Zhaozhi Qian, William R. Zame, Lucas M. Fleuren, Paul Elbers, and Mihaela van der Schaar. Integrating expert odes into neural odes: Pharmacology and disease progression. In *35th Conference on Neural Information Processing Systems*, 2021.

Md Fazle Rabby, Yazhou Tu, Md Imran Hossen, Insup Lee, Anthony S Maida, and Xiali Hei. Stacked lstm based deep recurrent neural network with kalman smoothing for blood glucose prediction.

BMC Medical Informatics and Decision Making, 21:1–15, 2021.

Sara E. Rosenbaum. *Basic Pharmacokinetics and Pharmacodynamics: an Integrated Textbook and Computer Simulations.* Hoboken, New Jersey: John Wiley & Sons, Incorporated, 2017.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386—-408, 1958.

Harry Rubin-Falcone, Ian Fox, and Jenna Wiens. Deep residual time-series forecasting: Application to blood glucose prediction. In *KDH@ ECAI*, pages 105–109, 2020.

Harry Rubin-Falcone, Joyce M Lee, and Jenna Wiens. Forecasting with sparse but informative variables: A case study in predicting blood glucose. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2022. URL https://kdd-milets.github.io/milets2022/papers/MILETS_2022_paper_0765.pdf.

Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

Artemios-Anargyros Semenoglou, Evangelos Spiliotis, Spyros Makridakis, and Vassilios Assimakopoulos. Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, 37 (3):1072–1084, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2020.11.009. URL https://www.sciencedirect.com/science/article/pii/S0169207020301850.

Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 07 2019. doi: 10.1016/j.ijforecast.2019.03.017.

Tue Søeborg, Christian Hove Rasmussen, Erik Mosekilde, and Morten Colding-Jørgensen. Absorption kinetics of insulin after subcutaneous administration. *European Journal of Pharmaceutical Sciences*, 36(1):78–90, January 2009. doi: 10.1016/j.ejps.2008.08.007.

Tue Søeborg, Christian Hove Rasmussen, Erik Mosekilde, and Morten Colding-Jørgensen. Bioavailability and variability of biphasic insulin mixtures. *European journal of pharmaceutical sciences*, 46(4):198–208, 2012.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio, 2016.

Roberto Visentin, Chiara Dalla Man, and Claudio Cobelli. One-day bayesian cloning of type 1 diabetes subjects: Toward a single-day uva/padova type 1 diabetes simulator. *IEEE Transactions on Biomedical Engineering*, 63(11), 2016.

Jinyu Xie. Simglucose v0.2.1 (2018), 2018. URL https://github.com/jxx123/simglucose.

Jinyu Xie and Qian Wang. Benchmarking machine learning algorithms on blood glucose prediction for type i diabetes in comparison with classical time-series models. *IEEE Transactions on Biomedical Engineering*, 67(11):3101–3124, 2020. doi: 10.1109/TBME.2020.2975959.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.

Appendix A. Supplemental Methods

A.1. Data

Simulated data An open-source python implementation of the FDA-approved UVa/Padova Simulator (2008 version) (Xie, 2018; Visentin et al., 2016) provides clinical and biological parameters for 30 patients (10 adults, 10 adolescents, and 10 children) with Type-I diabetes. Clinical parameters include information on age and weight, and biological parameters include information on insulin-glucose kinetics, such as absorption constants. We generate 54 days of data for each patient to match the median training set of the OhioT1DM, and approximately 9 days for the test set. In a similar approach to (Rubin-Falcone et al., 2022), the meal schedule used to generate simulated data was based on the Harrison-Benedict equation (Arthur Harris and Gano Benedict, 21919) with approximately 3 meals per day and no additional snacks. Height data was not provided in the patient parameters, and so was estimated at 140cm, 170cm, and 175cm for adults, adolescents, and children, respectively.

The 'Dexcom' continuous glucose monitor (CGM) option was used to obtain glucose readings at 5-minute intervals. The default basal-bolus controller provided in the simulation was used to administer insulin at the time of each meal, where the bolus amount is computed based on the current glucose level, the target glucose level, the patient's correction factor, and the patient's carbohydrate ratio. However, we adapt the controller to deliver basal doses once per hour, consistent with the function of real insulin pumps used in the OhioT1DM dataset. The resulting simulated data consists of CGM (in mg/dL) values, insulin (units), and carbohydrates (in grams), referred to as CHO. We extract the bolus insulin rate from the overall insulin rate as the difference in the continuous basal insulin rate, which is consistent hourly doses. Blood glucose time series for 6 simulated patients are shown in Fig. 11 in Appendix A.1. Basal and bolus dose distributions for 12 simuluted patients are shown in Fig. 13 in Appendix A.1.

OhioT1DM 12 de-identified individuals with Type 1 diabetes are included in the OhioT1DM 2018 and 2020 datasets (Marling and Bunescu, 2020). Each patient has approximately 8 weeks of data that includes blood glucose (mg/dL) measurements recorded with a CGM at a frequency of 5-minutes. The dataset consists of both female (n=5) and male (n=8) patients with unspecified ages within the ranges of 20-40, 40-60, and 60-80. Information on insulin pump model and type of insulin is also provided.

Sparse exogenous features in the data include basal rate (in units per hour), temporary basal rate (units per hour), and bolus insulin (units), and CHO values. Bolus insulin consists of both 'normal' insulin, which is recorded as an instantaneous administration, and 'square dual', where the insulin dose is stretched over a defined period. 'Square dual' insulin doses were divided evenly across the specified administration window with the assumption that insulin units are administered consistently across specified timestamps. If multiple bolus administrations occurred within the same minute, the insulin units were summed. For Medtronic insulin pumps 530G and 630G versions, basal insulin is administered hourly based on the basal rate. For timestamps with overlapping basal rates, the last basal rate value was considered. When specified, the temporary basal rate superseded the basal rate.

We use the specified train and test partitions provided in the dataset. The maximum number of test samples consistent across all patients (2691 samples) was used to obtain equal test sets of approximately 9 days. The sample count and percent missing data for each subject are included in Table 4 in Appendix A.1. Missing values were forward filled to prevent data leakage. Blood glucose time series plots for 6 subjects are shown in Fig. 12 in Appendix A.1. Basal and bolus dose distributions for the 12 patients are shown in Fig. 14 in Appendix A.1.

Table 4: Sample count (% missing samples) for the original OhioT1DM datasets and preprocessed test set

ID	Original		Preprocessed
	Train	Test	Test
540	13109 (8.87%)	3066 (6.43%)	2691 (6.91%)
544	12671 (16.16%)	3137 (13.42%)	2691 (14.08%)
552	11097 (18.80%)	3950 (40.15%)	2691 (53.36%)
559	12081 (10.64%)	2876 (12.59%)	2691 (13.45%)
563	13098 (7.44%)	2691 (4.50%)	2691 (4.50%)
567	13536 (19.78%)	2871 (16.79%)	2691 (16.91%)
570	11611 (5.42%)	2880 (4.69%)	2691 (4.91%)
575	13103 (9.44%)	2719 (4.74%)	2691 (4.79%)
584	13248 (8.29%)	2995 (11.02%)	2691 (11.93%)
588	13105 (3.55%)	2881 (3.12%)	2691 (3.34%)
591	12755 (14.96%)	2847 (3.06%)	2691 (3.23%)
596	13630 (20.20%)	3003 (8.66%)	2691 (9.66%)

A.2. Models

Exponential Smoothing (`AutoETS`) - The `AutoETS` model applies weighted average of past observations and exponentially decreases weights for observations further into the past (Hyndman, 2008).

DLinear (`DLinear`) - A linear model that leverages linear layers to model the trend and seasonal components. A moving average filter is used to separate the time series into its trend and seasonal components (Zeng et al., 2023).

Multi Layer Perceptrons (`MLP`) - A neural network architecture composed of stacked fully connected neural networks trained with backpropagation (Nair and Hinton, 2010; Fukushima, 1975; Rosenblatt, 1958).

Neural Hierarchical Interpolation for Time Series (`NHITS`) - A deep learning model that applies multi-rate input pooling, hierarchical interpolation, and backcast residual connections together to generate additive predictions with different signal bands. The hierarchical interpolation technique promotes efficient approximations of arbitrarily long horizons and reduced computation (Challu et al., 2022).

Neural Basis Expansion Analysis with Exogenous (`NBEATSx`) - An MLP-based deep learning model that decomposes the input signal into trend and seasonality components using polynomial and harmonic basis projections. The `NBEATSx` extension enables modeling of exogenous variables in addition to the univariate signal (Oreshkin et al., 2020; Olivares et al., 2022a).

Recurrent Neural Network (`RNN`) - A recurrent neural network (`RNN`) architecture that applies recurrent transformations to obtain hidden states. The hidden states are transformed into contexts which are used as inputs to `MLPs` to generate forecasts (Elman, 1990; Cho et al., 2014).

Long Short-Term Memory Recurrent Neural Network (`LSTM`) - A recurrent neural network (`RNN`) architecture that transforms hidden states from a multilayer `LSTM` encoder into contexts which are used as inputs to `MLPs` to generate forecasts (Sak et al., 2014).

Temporal Convolution Network (`TCN`) - A 1D causal-convolutional network architecture that transforms hidden states into contexts which are used as inputs to `MLP` decoders to generate forecasts. To generate contexts, the prediction at time t is convolved only with elements from time t and earlier in the previous layer in what is referred to as causal convolutions (Bai et al., 2018; van den Oord et al., 2016).

Informer (`Informer`) - A Transformer-based deep learning model with a multi-head attention mechanism that uses autoregressive features from a convolution network. A ProbSparse self-attention mechanism and a self-attention distilling process are leveraged to reduce computational complexity (Zhou et al., 2021).

Patch time series Transformer (`PatchTST`) - A Transformer-based deep learning model with a multi-head attention mechanism. The input time series is segmented into fixed-length sub-series-level patches, which serve as input tokens (Nie et al., 2023).

Temporal Fusion Transformer (`TFT`) - An attention-based deep learning architecture that learns temporal relationships at different scales using `LSTM`s for local processing and self-attention layers to model long-term dependencies. `TFT` also leverages variable selection networks as well as a series of gating layers to suppress unnecessary parts of the architecture (Lim et al., 2021).

A.3. Model Training and Hyperparameters

The deep learning models were trained using an NVIDIA A100 Tensor Core GPU. For all models, we use the hyperparameter values shown in Table 5 and tune the learning rate and random seed, in addition to architecture-specific hyperparameters discussed below. Additionally, for all models with the PK encoder, we tune the initial $\mathbf{k}$ values as shown in the second section of table 5. We tuned the hyperparameters of our model using 'HyperOptSearch' with the Tree-structured Parzen Estimator (TPE) algorithm and 20 samples.

Hyperparameter	**Considered Values**
Learning rate	Loguniform(1e-4, 1e-1)
Batch size	4
Windows batch size'	256
Dropout	0.0
Training steps	2000
Validation check steps	100
Early stop patience steps	5
Random seed	DiscreteRange(1, 10)
Initial $\mathbf{k}_{\text{basal}}$	{1.0, 1.1, 1.2}
Initial $\mathbf{k}_{\text{bolus}}$	{1.7, 1.8, 1.9}
Initial $\mathbf{k}_{\text{CHO}}$	{1.7, 1.8, 1.9}

Table 5: Common hyperparameter search space

In addition to tuning the learning rate and random seed for each model, we also tune architecture-specific hyperparameters. For the `DLinear` model, we tune the moving average window size ([5, 15, 25, 35]). For the `MLP` model, we tune the number of hidden units per layer ([128, 256, 512]). For the `LSTM` and `RNN` models, we tune the hidden size ([128, 256, 512]). For the `TCN` model, we tune the hidden size ([128, 256, 512]) and number of dilations ([2, 4, 8]). For the `PatchTST` model, we tune the hidden size ([128, 256, 512]), number of attention heads ([4, 8, 16]), number of encoder layers ([3, 4]), and patch length ([8, 16, 32, 64, 96, 128]). For the `Informer` model, we tune the hidden size ([128, 256, 512]), number of attention heads ([4, 8, 16]), number of encoder layers ([3, 4]), and number of decoder layers ([1, 2]). For the `TFT` model, we tune the hidden size ([128, 256, 512]) and number of attention heads ([4, 8, 16]). For the `NBEATSx` and `NHITS` models, we did not tune additional hyperparameters. The code to train models and tune hyperparameters for each architecture is included in the paper's code repository.

A.4. Model Evaluation

Metrics. We evaluate forecasts for each test set in the two datasets using two metrics: mean absolute error (MAE) and root mean squared error (RMSE),

$$\text{MAE}(\mathbf{y}_\tau, \hat{\mathbf{y}}_\tau) = \frac{1}{H}\sum_{\tau=t+1}^{t+H} |\mathbf{y}_\tau - \hat{\mathbf{y}}_\tau| \quad \text{and} \tag{9}$$

$$\text{RMSE}(\mathbf{y}_\tau, \hat{\mathbf{y}}_\tau) = \sqrt{\frac{1}{H}\sum_{\tau=t+1}^{t+H} (\mathbf{y}_\tau - \hat{\mathbf{y}}_\tau)^2}. \tag{10}$$

Here, $\mathbf{y}_\tau$ are the blood glucose values, and $\hat{\mathbf{y}}_\tau$ are the model predictions for the forecast horizon H, with $\tau = \{T+1, \dots, T+H\}$. We compute the metrics across all points within the forecast horizon and present the average results over 8 trials of independently trained models.

Statistical Significance Tests. To evaluate the performance differences between our hybrid global-local PK model and the baseline models, we conducted statistical significance testing on the forecasting MAE (Mean Absolute Error) results. For each statistical comparison, patient results were averaged over eight trials of individually trained models, and the paired t-test was computed by pairing patient results between the hybrid global-local PK model and the baseline model. The null hypothesis for each test was that there is no difference in performance between our hybrid global-local PK model and the compared baseline across patients. We considered results statistically significant at the conventional threshold of p-value < 0.05, indicating strong evidence against the null hypothesis.

A.5. Propositions

Proposition 1 *Under linear pharmacokinetic (PK) assumptions, the area under the concentration curve is directly proportional to the dose of the drug. As such, PK parameters, such as bioavailability, are assumed to be constant for a given drug. The bioavailability, F, can be assessed from the dose-normalized area under the concentration curve (AUC). When bioavailability is constant, we show that the AUC is bounded by the dose, x, given that $0 \leqslant F \leqslant 1$,*

$$F \propto \frac{AUC}{x} \tag{11}$$

$$n \cdot F = n \cdot \frac{AUC}{x} \tag{12}$$

$$AUC = F \cdot x \tag{13}$$

$$\leqslant x \tag{14}$$

where n where is a constant of proportionality.

Proposition 2 *Given some mathematical model, $C(t, \cdot)_x$, of concentration for a drug dose, x, that is a function of time, t, and pharmacokinetic parameters, $\cdot$, we show that the overall effect (concentration) of n drug doses of the same drug is equal to the sum of the individual effects under linear pharmacokinetic assumptions:*

$$x \propto \int_0^T C(t, \cdot)_x dt \tag{15}$$

$$n \cdot x = n \int_0^T C(t, \cdot)_x dt \tag{16}$$

$$= \int_0^T nC(t, \cdot)_x dt \tag{17}$$

$$= \int_0^T \underbrace{C(t, \cdot)_x + C(t, \cdot)_x + \cdots + C(t, \cdot)_x}_{n\ times} dt \tag{18}$$

$$= \int_0^T \sum_{i=1}^{n} C(t, \cdot)_x dt \tag{19}$$

Here, T represents the total duration of drug exposure or the time interval over which drug concentrations are being measured and t represents a specific point in time within that interval.

A.6. Algorithm

We present the training procedure for the hybrid global-local PK model in Algorithm 1. This framework is compatible with any deep learning architecture capable of modeling exogenous variables alongside the endogenous target forecast signal.

The algorithm initializes $\mathbf{k}$ as a vector of network embedding weights with values between 1 and 2 before model training. The initial value of $\mathbf{k}$ is treated as a hyperparameter and tuned using the `Hyperopt` package from the `RayTune` library (Liaw et al., 2018). At each training step, the algorithm iterates over batches of data. The model f_θ generates forecasts $\hat{\mathbf{y}}_\tau$ based on the input data. Network parameters, including $\mathbf{k}$, are updated using stochastic gradient descent (SGD) based on the loss of the model $\ell(f_\theta)$, computed between the predicted values $\hat{\mathbf{y}}$ and the true values $\mathbf{y}$ using the loss function $\mathcal{L}$. After a prespecified number of training steps, the algorithm performs a validation step by calculating the total validation loss $\ell(f_\theta)$. The final hyperparameters are selected based on the configuration that minimizes validation loss across all runs and are used to train the final model. The algorithm returns the final trained model f_θ.

Algorithm 1 `Hybrid Global-Local PK Model Training`.
E is the number of training steps, η is the learning rate, N is the number of subjects, ν is the sampling interval of the data, L is the lag time, and H is the forecast horizon. The model f_θ hyperparameters include the learning rate η, the hidden size $\mathbf{h}$, and the initial (init) $\mathbf{k}$ values for bolus, basal, and CHO concentration curves, among other hyperparameters. For brevity, we define $\mathbf{k}_{\text{init}} = (\mathbf{k}_{\text{init, bolus}}, \mathbf{k}_{\text{init, basal}}, \mathbf{k}_{\text{init, CHO}})$ to represent the initial concentration curve parameters. $\mathcal{L}$ is the loss function (Huber Loss). V is the specified validation step, or the number of training steps between each validation loss check. Forecasts $\hat{\mathbf{y}}$ are generated by f_θ as a function of historical glucose values $\mathbf{y}$, treatment concentration curves $\breve{\mathbf{x}}$, and static features $\mathbf{s}$.

```
Input: $N > 0$, $\nu > 0$, $L > 0$, $H > 0$, $\eta > 0$, $\mathbf{h} \in \{128, 256, 512, 1024\}$, $\mathbf{k}_{\text{init}} \in [1, 2]^{N \times 1}$
Output: $f_\theta$
 $\mathcal{B}_{\text{train}} \leftarrow$ (split training data into batches)
 $\mathcal{B}_{\text{val}} \leftarrow$ (split validation data into batches)

for each run $o$ in hyperparameter tuning samples (runs) $O$ do
    $f_\theta \leftarrow$ (initialize the model)
    $\mathbf{k} \leftarrow \mathbf{k}_{\text{init}}$                    ▷ Each entry in $\mathbf{k}$ is initialized to the same scalar value
    $\ell(f_\theta)^{(o)}_{\text{val}} \leftarrow 0$                    ▷ Initialize validation loss for run $o$

    for each step $i$ from 1 to $E$ do
        for batch $b \in \mathcal{B}_{train}$ do
            $\breve{\mathbf{x}} \leftarrow \sum_{t=0}^{L} C(\nu \mathbf{w}_{t,:}, \mathbf{x}_t, \mathbf{k})$
            $\hat{\mathbf{y}}_\tau \leftarrow f_\theta(\mathbf{y}, \breve{\mathbf{x}}, \mathbf{s})$
            $\ell_{\text{batch}}(f_\theta) \leftarrow \mathcal{L}(\mathbf{y}_\tau, \hat{\mathbf{y}}_\tau)$                    ▷ $\tau$ represents the forecast horizon
            $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \ell_{\text{batch}}(f_\theta)$                    ▷ update network parameters using SGD
        end
        if $i \mod V == 0$ then
            $\ell(f_\theta)_{\text{total}} \leftarrow 0$
            for batch $b \in \mathcal{B}_{val}$ do
                $\breve{\mathbf{x}} \leftarrow \sum_{t=0}^{L} C(\nu \mathbf{w}_{t,:}, \mathbf{x}_t, \mathbf{k})$
                $\hat{\mathbf{y}}_\tau \leftarrow f_\theta(\mathbf{y}, \breve{\mathbf{x}}, \mathbf{s})$
                $\ell_{\text{batch}}(f_\theta) \leftarrow \mathcal{L}(\mathbf{y}_\tau, \hat{\mathbf{y}}_\tau)$
                $\ell(f_\theta)_{\text{total}} \leftarrow \ell(f_\theta)_{\text{total}} + \ell_{\text{batch}}(f_\theta)$
            end
            $\ell(f_\theta)^{(o)}_{\text{val}} \leftarrow \frac{\ell(f_\theta)_{\text{total}}}{|\mathcal{B}_{\text{val}}|}$                    ▷ Average loss over entire validation set and update loss
        end
    end
end
 $\{\eta, \mathbf{h}, \mathbf{k}_{\text{init}}, \ldots\} \leftarrow \arg\min_{o \in O} \ell(f_\theta)_{\text{val}}$                    ▷ select best hyperparameters; retrain final model
 return $f_\theta$
```

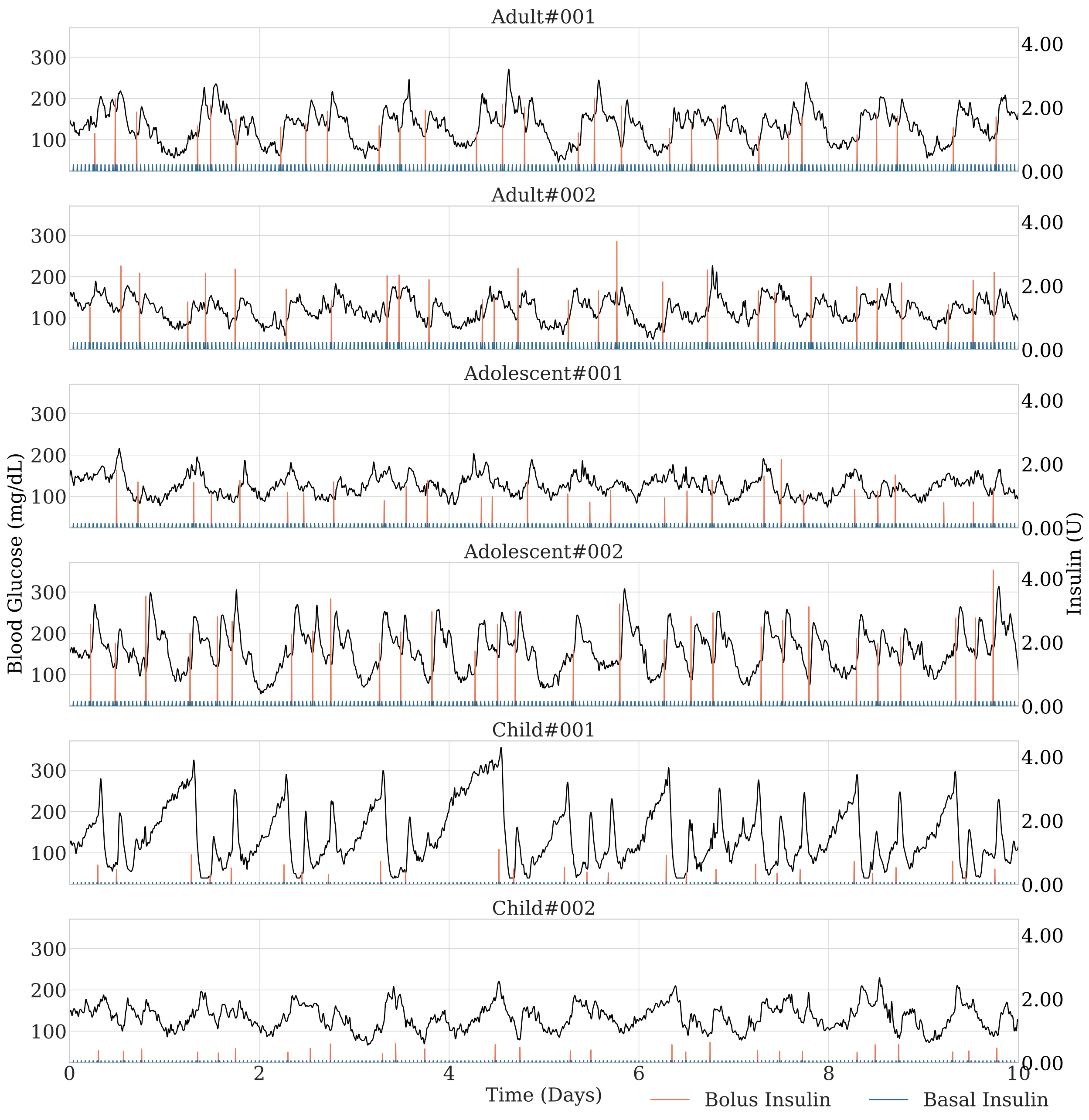

Figure 11: Time series for the first 10 days of 6 patients from the simulated dataset.

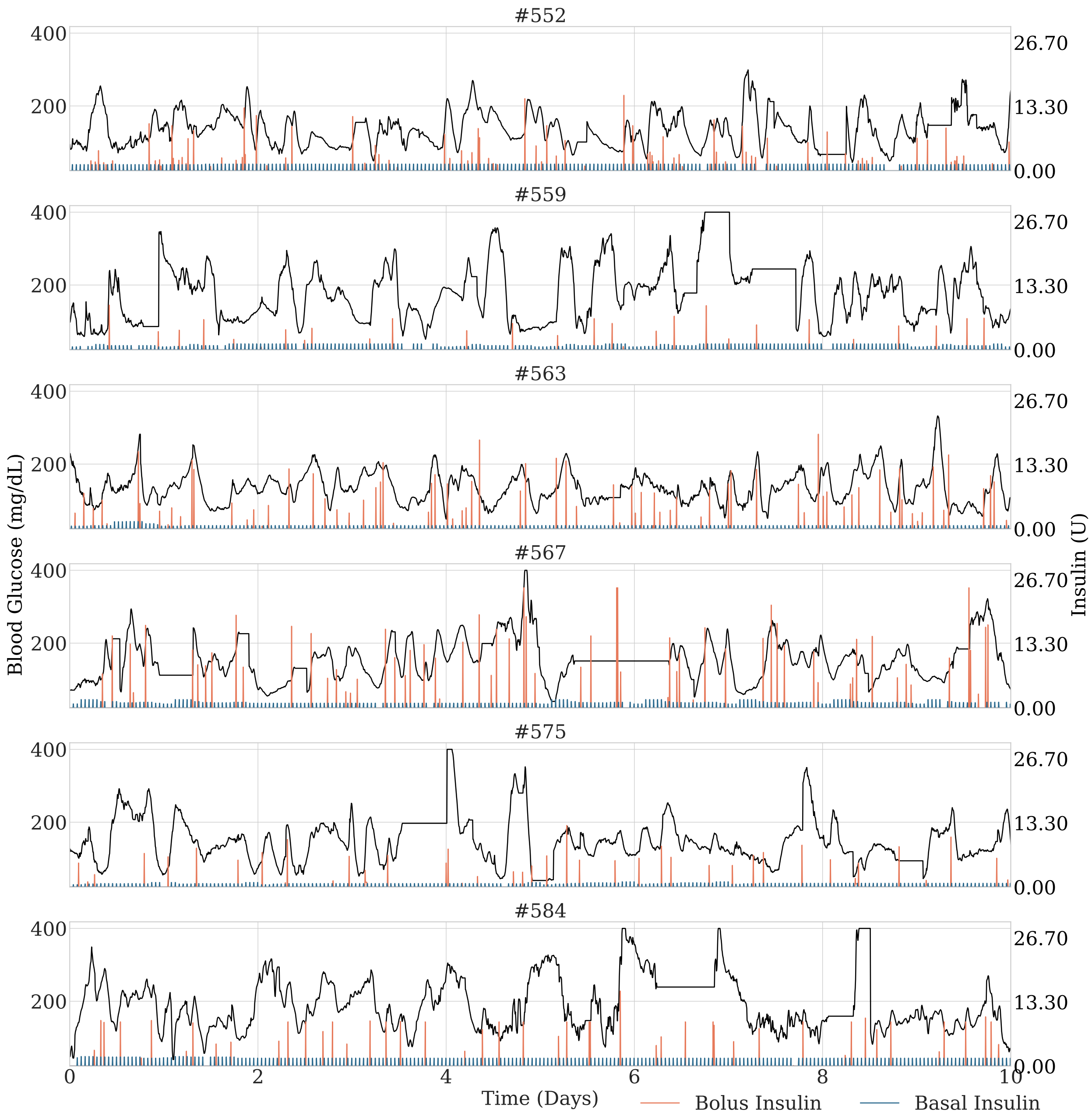

Figure 12: Time series for the first 10 days of 6 patients from the OhioT1DM dataset.

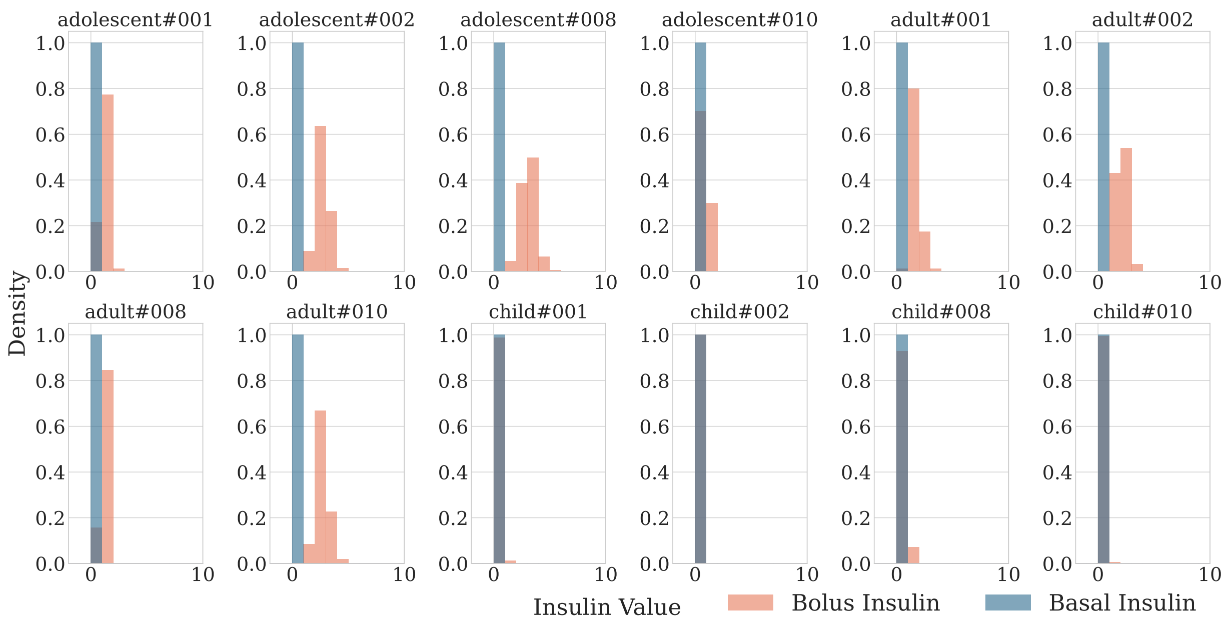

Figure 13: Density plots of bolus and basal insulin for 12 simulated data patients. Bolus doses are general larger than basal doses.

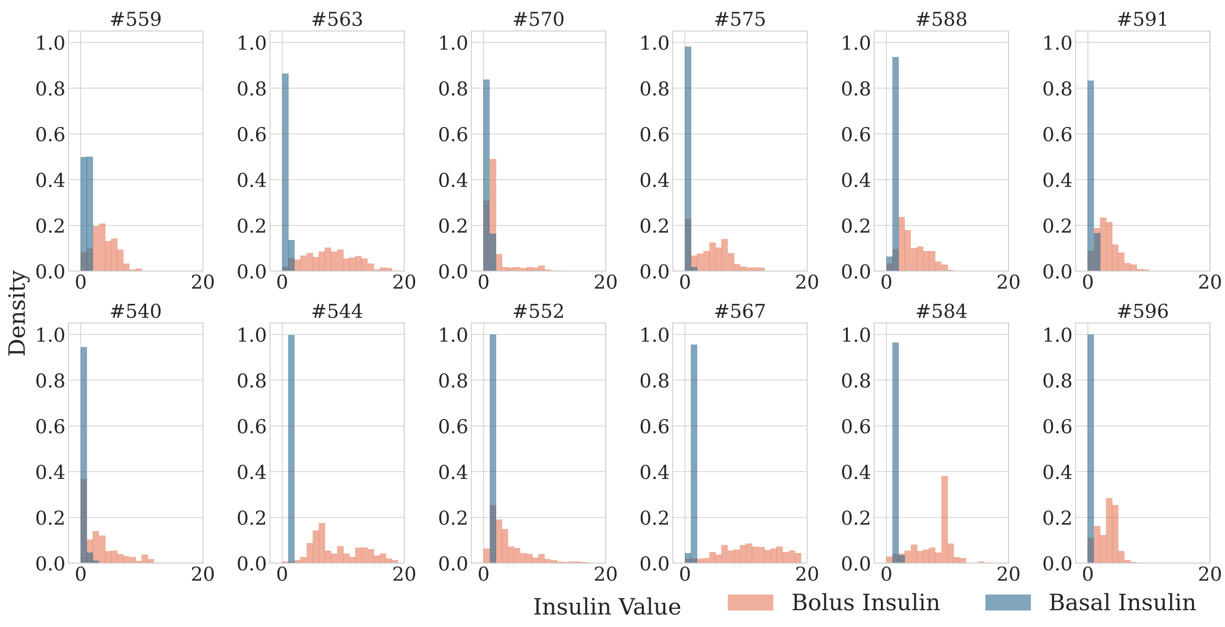

Figure 14: Density plots of bolus and basal insulin for OhioT1DM patients. Bolus doses are general larger than basal doses.

Appendix B. Pharmacokinetic Background

Concentration-time Profiles From PK literature, exponential decay functions are commonly used to model drug concentration over time for intravenous injections, where the drug is directly injected into the bloodstream, leading to instantaneous effects (Rosenbaum, 2017; Angelika et al., 2005). Alternatively, subcutaneous injections involve absorption and elimination processes where the peak onset time may not be instantaneous depending on the drug profile. Regarding knowledge on concentration-time profiles established through empirical and invasive studies, we know that concentration-time profiles for subcutaneous injections adhere to right-skewed curves. The degree of right-skew in the concentration curve will differ depending on the type of insulin, as illustrated in Fig. 2, along with various patient-specific factors.

Insulin Analogs Insulin analogs are modified forms of insulin designed to exhibit certain PK properties, such as changes in drug onset and duration of action. Fig. 2 is adapted from PK literature and depicts PK modeling of plasma insulin concentration over time for different insulin analogs (Angelika et al., 2005). Generally, bolus doses are quick-acting and, thus, often taken before meals. Contrastingly, basal insulin is longer-acting and serves to regulate blood glucose levels throughout the day. However, continuous glucose monitors (CGMs) and insulin pumps generally only hold and administer one type of insulin, such as rapid-acting insulin. For these devices, 'bolus' and 'basal' are terms used to refer to different timings of injections for different purposes. Bolus doses are administered discretely, generally around meals, while basal doses are administered continuously at a defined frequency, such as hourly, to regulate blood glucose levels throughout the day.

Bioavailability In Fig. 2, the area under the concentration curve measures drug *bioavailability*, the fraction of the absorbed dose that reaches the intended site of systemic circulation intact. Bioavailability varies based on many factors, including type of medication, dosage, and patient-specific characteristics. For instance, the bioavailability of an intravenous dose of any drug is close to 100 percent (Søeborg et al., 2012). However, for subcutaneous injections, bioavailability is typically lower due to unintended drug absorption or elimination.

First-order Kinetics First-order kinetics, a common assumption in pharmacokinetics, describes the rate at which a drug is eliminated from the body as being proportional to its concentration. This means that a constant fraction of the drug is eliminated per unit of time, regardless of its concentration levels. Exponential functions are often used to represent concentration-time profiles under the assumption of first-order kinetics, simplifying the modeling of plasma drug concentration in the body.

Linear Pharmacokinetics Linear pharmacokinetics refers to a situation where the relationship between the dose of a drug and its concentration in the body is linear (Rosenbaum, 2017). In linear pharmacokinetics, the area under the concentration curve is directly proportional to the dose of the drug. If you double the dose of a drug, the area under the curve will also double. Linearity results from the assumption that the drug processes exhibit first-order kinetics and that the PK parameters remain constant with dose, indicating no significant changes in the body's ability to eliminate the drug with higher doses. The majority of drugs used in clinical practice are assumed to follow linear pharmacokinetics (Rosenbaum, 2017).

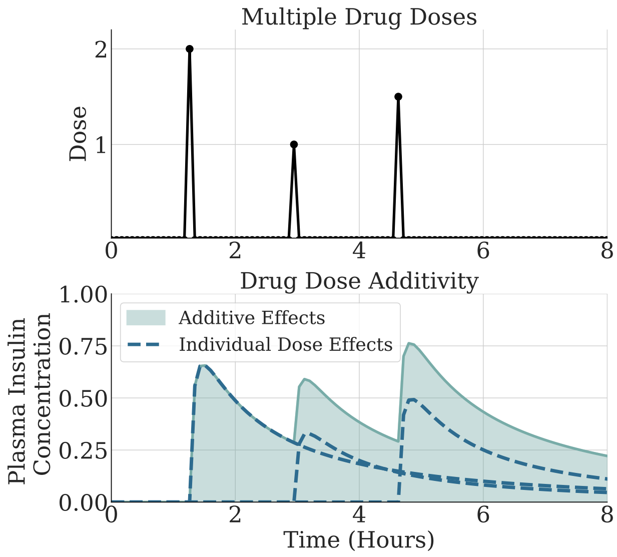

Figure 15: When doses of insulin are administered at close intervals, the subsequent dose may start to act before the previous dose has fully finished its effect. This can lead to a stacking of insulin effects. The figure depicts additive effects under linear pharmacokinetic assumptions.

Appendix C. Additional Results

C.1. Patient-specific (Local) Models

We showcase the effectiveness of global models, with parameters shared across subjects, in generating more accuracy forecasts compared to local models trained for individual subjects. Global PK models consistently outperform local PK models, on average across 8 trials, for each OhioT1DM patient as shown in Table 6 and Fig. 6. Furthermore, the global `NHITS`-PK model outperforms the global `NHITS`sparse exogenous model for 11 out of 12 OhioT1DM patients, on average across 8 trials, as shown in Table 6.

Table 6: MAE computed across forecast horizon values using local and global models for the OhioT1DM dataset. The best results are in **bold**.

ID	w/ Exog. (Sparse)		w/ Exog. (PK)	
	Local	Global	Local	Global
540	12.106 (0.779)	10.119 (0.139)	11.880 (0.596)	**10.015** (0.147)
544	7.952 (0.679)	7.188 (0.062)	8.067 (0.490)	**6.967** (0.083)
552	8.559 (0.243)	7.400 (0.118)	9.132 (0.403)	**7.374** (0.070)
559	10.845 (1.266)	9.013 (0.176)	13.051 (7.027)	**8.967** (0.102)
563	9.527 (0.379)	8.731 (0.065)	9.427 (0.402)	**8.506** (0.084)
567	12.599 (1.649)	9.640 (0.138)	12.132 (1.667)	**9.562** (0.189)
570	8.361 (0.496)	7.523 (0.141)	8.466 (0.223)	**7.258** (0.121)
575	10.891 (0.419)	9.712 (0.099)	11.095 (0.470)	**9.496** (0.151)
584	11.111 (0.547)	**9.846** (0.091)	11.518 (0.538)	9.905 (0.156)
588	10.172 (1.737)	8.172 (0.081)	9.376 (0.321)	**8.137** (0.087)
591	11.987 (1.274)	10.246 (0.083)	11.428 (0.172)	**10.151** (0.134)
596	8.990 (0.246)	8.133 (0.103)	9.147 (0.502)	**8.059** (0.141)
All	10.324 (0.401)	8.873 (0.064)	10.425 (0.659)	**8.758** (0.080)

Table 7: Mean absolute error (MAE) computed for model predictions across all values in the forecast horizon and across only critical values in the forecast horizon (blood glucose $\leqslant 70| \geqslant 180$). Result for models with (w/) and without (w/o) exogenous variables are shown. The best results are in **bold**.

Model	All Values		Critical Values	
	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.
Simulated Dataset				
`NHITS`	8.268 (0.039)	7.304 (0.055)	11.114 (0.237)	9.094 (0.236)
`NHITS`-ST	–	7.055 (0.038)	–	8.648 (0.068)
`NHITS`-PK$_{\text{linear}}$	–	7.103 (0.069)	–	**8.449** (0.139)
`NHITS`-PK$_{\text{nonlinear}}$	–	**7.037** (0.093)	–	8.492 (0.220)
OhioT1DM Dataset				
`NHITS`	9.060 (0.142)	8.873 (0.064)	10.337 (0.137)	10.137 (0.108)
`NHITS`-ST	–	8.958 (0.087)	–	10.131 (0.100)
`NHITS`-PK$_{\text{linear}}$	–	8.830 (0.060)	–	10.026 (0.064)
`NHITS`-PK$_{\text{nonlinear}}$	–	**8.758** (0.080)	–	**9.965** (0.089)

C.2. Forecasting with Dose-Independent Parameters

We conducted an ablation study to assess whether modeling patient-specific effects of bolus and basal insulin under the assumption of dose-independent effects results in improved forecasting performance. We use our PK encoder to infer the same $\mathbf{k}$ for both bolus and basal insulin instead of separate values as shown in Table 7 for `NHITS`-PK$_{\text{linear}}$. We compare these results to `NHITS`-PK$_{\text{nonlinear}}$, where the PK encoder infers separate $\mathbf{k}$ values for bolus and basal insulin. A separate $\mathbf{k}$ value is inferred for the CHO variable, consistent with the method in the main paper.

We found that learning a single $\mathbf{k}$ value for both bolus and basal insulin improved performance on the simulated dataset only when forecasts were evaluated at critical values. However, this approach resulted in worse performance when evaluated across all values in the simulated dataset, as well as on the OhioT1DM dataset. The improved performance on the simulated dataset may be due to the simulated data generator relying on linear PK by treating bolus and basal doses in aggregate and using static parameters for insulin-glucose kinetics (Xie, 2018; Visentin et al., 2016). In contrast, insulin may exhibit nonlinear dose-dependent PK in real-world scenarios, as mentioned in section 3.3.

C.3. Lag Time Ablation

We performed an ablation study on model performance for various lag times L, including 2.5 hours (30 time steps), 5 hours (60 time steps), and 10 hours (120 time steps; results presented in the main paper), as well as 15 hours (180 time steps), for the top three performing deep learning models: `TFT`, `NBEATSx`, and `NHITS`. The study was conducted at a fixed forecast horizon H of 30 minutes (6 time steps). For each lag time, the `NBEATSx`-PK and `NHITS`-PK models had a lower MAE than their sparse exogenous model counterparts, except for $L = 180$ in the OhioT1DM dataset, as shown in Table 8. The `TFT`-PK model had a lower MAE than its sparse exogenous model counterpart for $L \in \{60, 120, 180\}$ in the simulated dataset and for $L = 180$ in the OhioT1DM dataset.

For each lag time, we evaluate whether the PK models have a significantly lower forecasting error, in terms of MAE, than their sparse exogenous model counterparts using paired t-tests, as outlined in Section A.4. For the simulated dataset, the `NBEATSx`-PK and `NHITS`-PK models showed a statistically significant performance improvement over their sparse exogenous model counterparts (p-value < 0.05) for all lag times, while the `TFT`-PK model showed a statistically significant performance improvement for all lag times except $L = 2.5$. For the OhioT1DM dataset, the `NBEATSx`-PK and `NHITS`-PK models demonstrated a statistically significant performance improvement over their sparse exogenous model counterparts (p-value < 0.05) for lag times of 2.5, 5, and 10 hours, while the `TFT`-PK model showed a statistically significant performance improvement for $L = 180$. Notably, for $L = 180$, the `TFT`-PK model was also the best-performing `TFT` model variant across all lag times for the OhioT1DM dataset.

C.4. Critical Event Prediction Rate Analysis

We evaluate the ability of models to predict critical events using two metrics: the true positive rate (TPR), or the proportion of forecast horizon windows in which the model correctly predicts an event when one actually occurs, and the false positive rate (FPR), or the proportion of forecast horizon windows in which the model predicts an event when none occurs. From the TPR, one can also obtain the false negative rate ($\text{FNR} = 1 - \text{TPR}$), or the proportion of forecast horizon windows in which the model does not predict an event when one actually occurs. We measure each metric on multiple blood glucose thresholds up to critical event thresholds (blood glucose $\leqslant 70 | \geqslant 180$).

The TPR and FPR for the simulated and OhioT1DM datasets at various glucose thresholds for hyperglycemia event warnings are shown in Fig. 16. The best performing `TFT`-PK model for the simulated dataset predicts hyperglycemic events (blood glucose $\geqslant 180$) in forecast windows at an approximate TPR of 0.96 and an FPR of 0.09. The best performing `NHITS`-PK model for the OhioT1DM dataset predicts hyperglycemic events in forecast windows at an approximate TPR of 0.89 and an FPR of 0.02. The TPR increases as blood glucose thresholds decrease toward 160 mg/dL, with lower thresholds potentially serving as preliminary warnings, prompting patients to monitor their blood glucose and stay alert for further warnings at higher glucose levels.

The TPR and FPR for the simulated and OhioT1DM datasets at various glucose thresholds for hypoglycemia event warnings are shown in Fig. 17. The best performing `TFT`-PK model for the simulated dataset predicts hypoglycemic events (blood glucose $\leqslant 70$) in forecast windows at an approximate TPR of 0.93 and an FPR of 0.11. The best performing `NHITS`-PK model for the OhioT1DM dataset predicts hypoglycemic events in forecast windows at an approximate TPR of 0.69 and an FPR of 0.01. The TPR increases as blood glucose thresholds decrease toward 90 mg/dL, with higher thresholds potentially serving as preliminary warnings to prompt patients to monitor their blood glucose and stay alert for further warnings at lower glucose levels.

These results highlight the effectiveness of the models in predicting critical glucose events, demonstrating high TPRs while maintaining relatively low FPRs. The ability to provide early warnings at varying glucose thresholds can be valuable for proactive patient monitoring and timely intervention. However, there is still room for improvement, as achieving higher TPRs with even lower FPRs could further enhance reliability and reduce false alarms.

Table 8: Comparing lag time L effect on model performance in terms of mean absolute error (MAE) computed for model predictions across all values in the forecast horizon and only across critical values in the forecast horizon (blood glucose $\leqslant 70| \geqslant 180$) for simulated dataset and OhioT1DM dataset. Average results are computed across 8 trials. Best results are highlighted in **bold**.

Lag Time (L)	**Model**	**Simulated Dataset**				**OhioT1DM Dataset**			
		All Values		**Critical Values**		**All Values**		**Critical Values**	
		w/ Exog.	**w/ Exog. (PK)**	**w/ Exog.**	**w/ Exog. (PK)**	**w/ Exog.**	**w/ Exog. (PK)**	**w/ Exog.**	**w/ Exog. (PK)**
$L = 30$ timesteps (2.5 hours)	TFT	**5.836** (0.248)	6.064 (0.466)	**7.318** (0.364)	7.756 (0.847)	**9.136** (0.363)	9.145 (0.480)	**10.271** (0.260)	10.464 (0.790)
	NBEATSx	7.619 (0.092)	**7.506** (0.180)	9.230 (0.126)	**8.915** (0.241)	9.263 (0.089)	**8.789** (0.146)	10.583 (0.148)	**9.841** (0.167)
	NHITS	6.807 (0.050)	**6.566** (0.076)	8.454 (0.212)	**7.784** (0.250)	8.356 (0.020)	**8.231** (0.031)	9.585 (0.080)	**9.393** (0.057)
$L = 60$ timesteps (5 hours)	TFT	6.087 (0.466)	**5.686** (0.073)	7.344 (0.461)	**6.748** (0.177)	**9.012** (0.195)	9.236 (0.544)	**10.090** (0.151)	10.207 (0.485)
	NBEATSx	7.241 (0.052)	**7.200** (0.103)	8.511 (0.064)	**8.395** (0.117)	9.810 (0.122)	**9.572** (0.710)	10.995 (0.145)	**10.621** (0.689)
	NHITS	6.934 (0.050)	**6.662** (0.092)	8.543 (0.151)	**7.931** (0.216)	8.528 (0.046)	**8.462** (0.059)	9.731 (0.089)	**9.691** (0.079)
$L = 120$ timesteps (10 hours)	TFT	5.826 (0.192)	**5.743** (0.159)	7.132 (0.243)	**7.042** (0.270)	**9.197** (0.455)	9.529 (0.597)	10.643 (0.792)	**10.461** (0.498)
	NBEATSx	6.886 (0.031)	**6.806** (0.060)	8.145 (0.072)	**7.998** (0.076)	10.280 (0.214)	**9.983** (0.264)	11.359 (0.188)	**10.927** (0.271)
	NHITS	7.304 (0.055)	**7.037** (0.093)	9.094 (0.236)	**8.492** (0.220)	8.873 (0.064)	**8.758** (0.080)	10.137 (0.108)	**9.965** (0.095)
$L = 180$ timesteps (15 hours)	TFT	5.967 (0.749)	**5.829** (0.122)	7.243 (0.884)	**7.048** (0.257)	8.926 (0.229)	**8.852** (0.195)	10.314 (0.593)	**10.165** (0.400)
	NBEATSx	6.803 (0.035)	**6.731** (0.045)	7.900 (0.061)	**7.854** (0.073)	**10.356** (0.307)	10.543 (0.363)	**11.493** (0.366)	11.630 (0.488)
	NHITS	7.446 (0.046)	**7.289** (0.040)	9.283 (0.238)	**8.867** (0.122)	**8.981** (0.058)	9.087 (0.131)	**10.130** (0.043)	10.249 (0.128)

Model	**FLOPS**	**# Trainable Parameters**	**Inference Time (ms)**
TFT w/o Exog.	3.66e8	2.53 million	15.9
TFT w/ Exog.	6.11e8 (↑ 66.94%)	2.78 million (↑ 9.88%)	17.7
TFT-PK w/ Exog.	6.07e8 (↑ 65.85%)	2.78 million (↑ 9.88%)	21.3
NBEATSx w/o Exog.	8.01e7	10.02 million	5.2
NBEATSx w/ Exog.	8.89e7 (↑ 10.99%)	11.13 million (↑ 11.08%)	5.8
NBEATSx-PK w/ Exog.	8.30e7 (↑ 3.62%)	10.39 million (↑ 3.69%)	5.7
NHITS w/o Exog.	8.19e7	10.25 million	5.5
NHITS w/ Exog.	9.08e7 (↑ 10.87%)	11.36 million (↑ 10.83%)	5.8
NHITS-PK w/ Exog.	8.49e7 (↑ 3.66%)	10.62 million (↑ 3.61%)	8.4

Table 9: Computational efficiency characteristics including FLOPS (floating-point operations per second), number of trainable parameters, and inference time measured for the OhioT1DM dataset models. The percentage increase in FLOPs and the number of trainable parameters over the baseline model without (w/o) exogenous (exog.) variables is shown in parentheses (e.g., ↑ x.xx%). As expected, models with (w/) exogenous variables exhibit larger FLOPs, more trainable parameters, and longer inference times due to the processing of higher-dimensional input. However, the PK models have the same or fewer FLOPs and trainable parameters compared to their sparse exogenous model counterparts, as the exogenous features are leveraged more effectively in the PK model architectures.

C.5. Computational Efficiency Analysis

To assess the computational efficiency characteristics of the deep learning models, we calculate three key metrics: FLOPs (floating-point operations per second), the number of trainable parameters, and the inference time for the OhioT1DM dataset models. These metrics provide insights into the computational complexity, memory usage, and real-time efficiency of

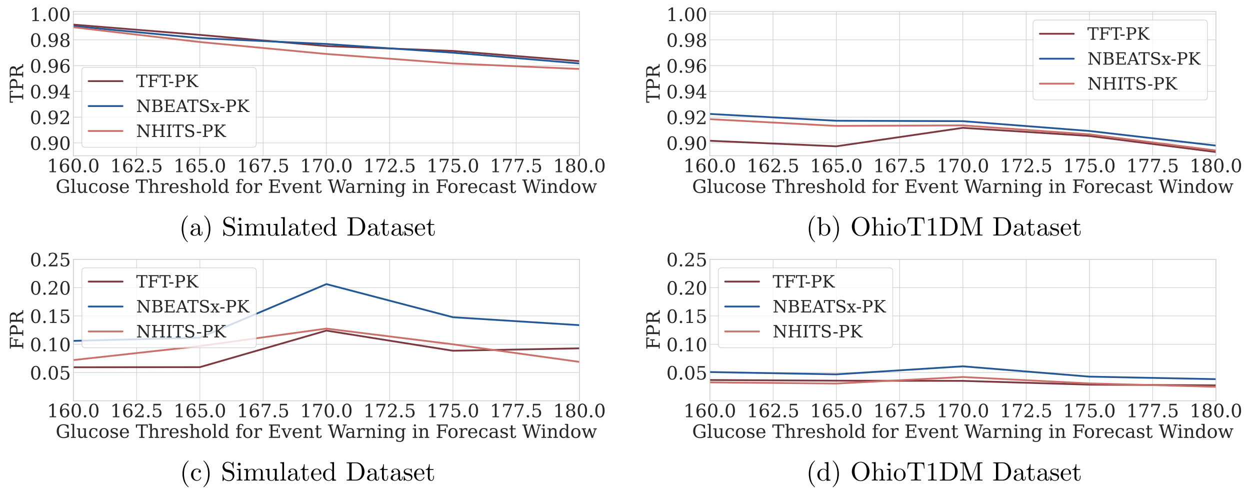

Figure 16: **(a, b)** True positive rate (TPR) and **(c, d)** false positive rate (FPR) for the simulated and OhioT1DM datasets at various glucose thresholds for **hyperglyemia** event warnings (blood glucose $\geqslant 180$). TPR represents the proportion of forecast horizon windows in which the model correctly predicts an event when one actually occurs, while FPR represents the proportion of forecast horizon windows in which the model predicts an event when none occurs. The best performing `TFT`-PK model for the simulated dataset predicts key glucose thresholds in forecast windows at an approximate TPR above of 0.96 and an FPR of 0.09. The best performing `NHITS`-PK model for the OhioT1DM dataset predicts key glucose thresholds in forecast windows at an approximate TPR of 0.89 and an FPR of 0.02. The TPR increases as blood glucose thresholds decrease toward 160, with lower thresholds potentially serving as preliminary warning triggers to prompt patients to monitor their blood glucose and stay alert for further triggers at higher glucose levels.

the model. We calculate FLOPs using the FlopCounterMode class from Pytorch, which tracks operations during model execution. The resulting FLOPs represent the computational load of the model for each forward pass. The number of trainable parameters in a model reflects its size. We calculate this by iterating through all the model's parameters and summing the number of elements for those that require gradients. Inference time refers to the time it takes for the model to generate predictions from input data. To measure this, we execute the model multiple times (100 runs) and record the time taken for each forward pass. The inference time is then calculated as the average time taken across these runs, providing an estimate of the model's efficiency during real-time deployment. This metric is essential for applications like clinical decision support or real-time monitoring devices, where timely predictions are crucial. The metrics for the three best performing deep learning models, `TFT`-PK, `NBEATSx`-PK, and `NHITS`-PK are presented in Table 9. In addition to providing metrics for the PK models, the table also includes metrics for models with (w/) and without (w/o) exogenous (exog.) features for reference. As expected, models with exogenous variables exhibit larger FLOPs, more trainable parameters, and longer inference times due to the processing of higher-dimensional input. However, the PK models have the same or fewer FLOPs and trainable parameters compared to their sparse exogenous model counterparts, as the exogenous features are leveraged more effectively in the PK model architectures. For `NBEATSx` and `NHITS` models, this efficiency stems from our proposed approach, where exogenous variables are processed in only one of the deep stacks of fully connected layers rather than across all stacks. This design improves processing efficiency and reduces model size. Without the PK encoder, the approach of isolating sparse exogenous variables to a single stack generally results in worse performance, as shown in Table 10.

The `TFT`-PK, `NBEATSx`-PK, and `NHITS`-PK models with sparse exogenous features show increases in FLOP of 65.30%, 3.62%, and 3.66%, respectively, compared to the baseline models without exogenous variables. In contrast, the models with sparse exogenous features exhibit higher FLOP increases of

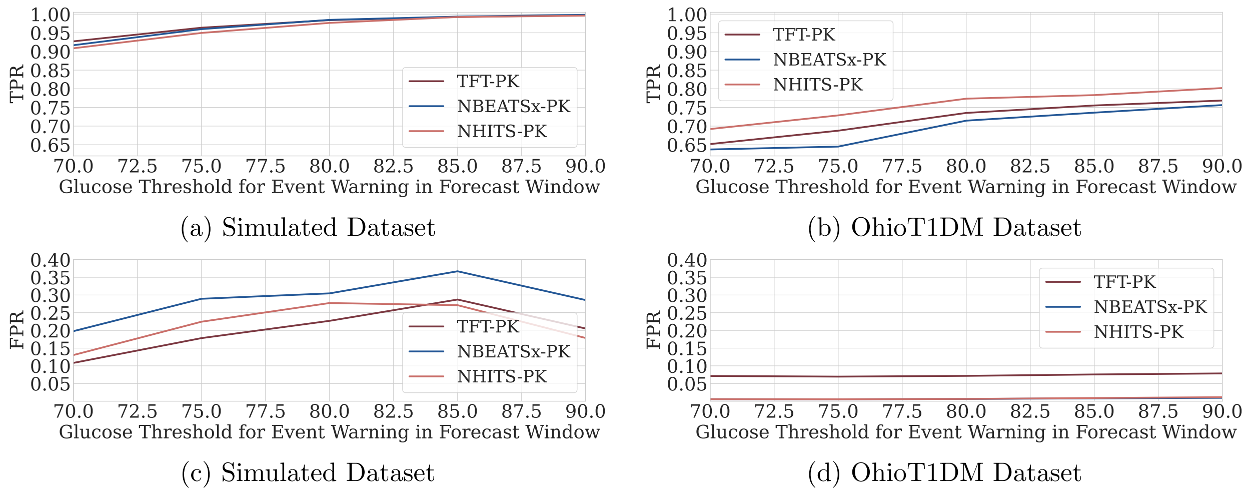

Figure 17: **(a, b)** True positive rate (TPR) and **(c, d)** false positive rate (FPR) for the simulated and OhioT1DM datasets at various glucose thresholds for **hypoglyemia** event warnings (blood glucose $\leqslant 70$). TPR represents the proportion of forecast horizon windows in which the model correctly predicts an event when one actually occurs, while FPR represents the proportion of forecast horizon windows in which the model predicts an event when none occurs. The best performing `TFT`-PK model for the simulated dataset predicts key glucose thresholds in forecast windows at an approximate TPR of 0.93 and an FPR of 0.11. The best performing `NHITS`-PK model for the OhioT1DM dataset predicts key glucose thresholds in forecast windows at an approximate TPR of 0.69 and an FPR of 0.01. The TPR increases as blood glucose thresholds increase toward 90, with higher thresholds potentially serving as preliminary warning triggers to prompt patients to monitor their blood glucose and stay alert for further triggers at lower glucose levels.

Table 10: Processing covariates in 1 versus 3 of the deep stacks of fully connected layers for `NBEATSx` and `NHITS`. Without the PK encoder, the approach of isolating sparse exogenous variables to a single stack generally results in worse performance.

	Simulated Dataset		**OhioT1DM Dataset**	
	`NBEATSx`	`NHITS`	`NBEATSx`	`NHITS`
1 Stack	6.908 (0.101)	7.429 (0.042)	**10.020** (0.569)	8.897 (0.246)
3 Stacks	**6.886** (0.031)	**7.304** (0.055)	10.280 (0.214)	**8.873** (0.064)

66.94%, 10.99%, and 10.87%, respectively, compared to the baseline models without exogenous variables. This result highlights the utility of the PK models, as they not only provide lower forecasting errors but also offer greater computational efficiency. Additionally, the models have fast inference speeds, on the order of magnitudes smaller than 1 second.

C.6. Feasibility of Model Integration into Clinical Systems

Integrating deep learning forecasting models into clinical decision-support systems or real-time monitoring devices can provide timely, data-driven alerts for critical events, aiding decision-making for proactive interventions. To ensure smooth integration into a clinical decision-support system or real-time monitoring device, deep learning models must be both data-efficient and capable of delivering reasonable inference times to facilitate updated forecasts at appropriate time intervals. The OhioT1DM patients use Medtronic 530G or 630G insulin pumps and Medtronic Enlite Continuous Glucose Monitor (CGM) sensors. The MiniMed 630G Insulin Pump is capable of storing 90 days of pump history and glucose sensor data. Given that such devices are equipped to store data on the scale of many days, the feasibility of integrating the proposed hybrid global-local PK models into such systems is high, as our PK models only require a maximum of 10 hours of historical data to generate forecasts. The PK models are also memory-efficient. For example, the `NHITS`-

PK model requires less than 20MB of memory during inference on a CPU in a cold-start scenario. A cold start refers to the initial model inference, where no preloaded state or cached data is used, providing an upper bound on memory usage. Furthermore, the `NHITS`-PK model demonstrates an inference time of approximately 8 milliseconds when executed on a CPU, ensuring rapid predictions that support real-time operation. This combination of low storage requirements and fast inference times makes the integration of such models into clinical systems feasible to help monitor and provide early warnings for critical events. While a graphics processing unit (GPU) is recommended for efficient model training, a central processing unit (CPU) is sufficient for inference with the trained model, offering space, cost, and energy efficiency benefits over dedicated graphics processing devices.

In addition to CGMs, deep learning models could be integrated into other mobile devices, such as a phone app. The model could provide continuous forecast windows that update every 5 minutes (the frequency of the data) and trigger a beep or vibration if the patient is at risk of a critical event occurring within the forecast horizon. The device could alert the patient with customizable blood glucose thresholds, accounting for warnings of varying severity. For example, preliminary warnings could be set at less severe blood glucose levels to prompt patients to monitor their blood glucose and remain alert for further triggers at more severe thresholds. Furthermore, integrating the model into a phone app could facilitate sharing alert information, notifying patients and allowing the inclusion of other monitoring users, such as parents, school nurses, or caregivers.

C.7. Model MAE and RMSE Results

This appendix presents the complete results for all models for both simulated data and OhioT1DM. Standard deviation across 8 trials are provided in parenthesis. Tables 11 and 12 provide complete MAE and RMSE results, respectively.

Table 11: Mean absolute error (MAE) computed for model predictions across all values in the forecast horizon and only across critical values in the forecast horizon (blood glucose $\leqslant 70 | \geqslant 180$) for simulated dataset and OhioT1DM dataset. Average results are computed across 8 trials. Models are separated with horizontal lines into statistical, linear, RNN-based, CNN-based, Transformer-based, and MLP-based, and PK model groups, respectively. Result for models with (w/) and without (w/o) exogenous (exog.) variables are shown where applicable. Best results are highlighted in **bold**, second best results are underlined. PK models with improved forecasts over their respective exogenous baseline model are highlighted in blue.

Model	Simulated Dataset				OhioT1DM Dataset			
	All Values		Critical Values		All Values		Critical Values	
	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.	w/o Exog.	w/ Exog.
AutoETS	9.450 (0.00)	–	13.457 (0.00)	–	9.022 (0.00)	–	10.028 (0.00)	–
DLinear	7.802 (0.054)	7.989 (0.141)	10.575 (0.126)	10.790 (0.169)	11.514 (1.595)	13.011 (3.590)	12.512 (1.228)	14.580 (4.080)
RNN	12.728 (8.829)	12.930 (8.819)	20.308 (19.533)	20.945 (19.925)	11.535 (3.056)	10.907 (0.809)	12.641 (2.342)	12.989 (1.822)
LSTM	9.250 (2.041)	9.152 (2.097)	12.034 (1.830)	12.000 (3.983)	10.217 (0.930)	10.615 (0.807)	12.102 (1.218)	12.983 (1.464)
TCN	14.340 (8.271)	11.380 (2.049)	23.370 (20.857)	14.817 (3.258)	10.769 (1.093)	11.516 (1.846)	12.812 (0.901)	13.203 (1.882)
Informer	8.599 (1.060)	–	12.074 (2.083)	–	13.714 (3.753)	–	14.867 (3.620)	–
PatchTST	7.284 (0.554)	–	9.842 (0.939)	–	9.922 (0.949)	–	10.999 (1.054)	–
TFT	6.587 (0.115)	5.826 (0.192)	8.518 (0.178)	7.132 (0.243)	9.402 (0.811)	9.197 (0.455)	10.577 (1.038)	10.643 (0.792)
MLP	10.424 (1.478)	8.407 (0.387)	13.932 (0.861)	10.297 (0.329)	10.300 (0.329)	13.254 (0.245)	11.550 (0.359)	14.997 (0.353)
NBEATSx	9.187 (0.236)	6.886 (0.031)	12.757 (0.547)	8.145 (0.072)	9.868 (0.528)	10.280 (0.214)	11.037 (0.548)	11.359 (0.188)
NHITS	8.268 (0.039)	7.304 (0.055)	11.114 (0.237)	9.094 (0.236)	9.060 (0.142)	8.873 (0.064)	10.337 (0.137)	10.137 (0.108)
TFT-PK	–	**5.743** (0.159)	–	**7.042** (0.270)	–	9.529 (0.597)	–	10.461 (0.498)
NBEATSx-PK	–	6.806 (0.060)	–	7.998 (0.076)	–	9.983 (0.264)	–	10.927 (0.271)
NHITS-PK	–	7.037 (0.093)	–	8.492 (0.220)	–	**8.758** (0.080)	–	**9.965** (0.095)

Table 12: Root mean square error (RMSE) computed for model predictions across all values in the forecast horizon and only across critical values in the forecast horizon (blood glucose $\leqslant 70| \geqslant 180$) for simulated dataset and OhioT1DM dataset. Average results are computed across 8 trials. Models are separated with horizontal lines into statistical, linear, RNN-based, CNN-based, Transformer-based, and MLP-based groups, and PK model groups, respectively. Result for models with (w/) and without (w/o) exogenous (exog.) variables are shown where applicable. Best results are highlighted in **bold**, second best results are underlined. PK models with improved forecasts over their respective exogenous baseline model are highlighted in blue.

Model	**Simulated Dataset**				**OhioT1DM Dataset**			
	All Values		**Critical Values**		**All Values**		**Critical Values**	
	w/o Exog.	**w/ Exog.**	**w/o Exog.**	**w/ Exog.**	**w/o Exog.**	**w/ Exog.**	**w/o Exog.**	**w/ Exog.**
AutoETS	14.751 (0.00)	–	21.454 (0.00)	–	14.569 (0.00)	–	**16.225** (0.00)	–
DLinear	12.545 (0.085)	12.673 (0.119)	17.606 (0.340)	17.707 (0.216)	17.138 (1.462)	18.609 (3.511)	18.679 (0.970)	20.743 (3.869)
RNN	17.679 (10.978)	18.017 (10.935)	26.312 (19.591)	26.849 (19.935)	16.433 (2.734)	15.762 (0.657)	18.274 (2.199)	18.375 (1.654)
LSTM	12.993 (1.896)	12.526 (2.389)	17.027 (1.599)	16.222 (4.528)	15.337 (0.750)	15.571 (0.684)	17.759 (0.949)	18.399 (1.295)
TCN	20.040 (10.281)	15.181 (2.571)	30.812 (20.017)	19.573 (4.054)	16.073 (0.855)	16.483 (1.544)	18.642 (0.601)	18.677 (1.451)
Informer	12.348 (0.918)	–	17.400 (1.865)	–	18.536 (3.459)	–	19.904 (3.220)	–
PatchTST	11.467 (0.603)	–	15.755 (1.000)	–	15.375 (0.838)	–	17.065 (0.928)	–
TFT	10.691 (0.139)	8.898 (0.314)	14.188 (0.329)	10.979 (0.446)	14.832 (0.791)	14.571 (0.345)	16.523 (0.959)	16.577 (0.684)
MLP	15.196 (1.322)	11.267 (0.405)	21.246 (0.692)	13.832 (0.336)	15.908 (0.276)	18.986 (0.312)	17.762 (0.421)	21.340 (0.440)
NBEATSx	14.105 (0.348)	9.570 (0.053)	20.526 (0.756)	11.397 (0.104)	15.934 (0.548)	16.224 (0.675)	17.540 (0.599)	17.742 (0.303)
NHITS	12.918 (0.064)	10.546 (0.106)	18.436 (0.369)	13.371 (0.387)	14.844 (0.192)	14.505 (0.096)	16.789 (0.189)	16.453 (0.132)
TFT-PK	–	**8.620** (0.188)	–	**10.517** (0.367)	–	14.725 (0.414)	–	16.265 (0.437)
NBEATSx-PK	–	9.480 (0.084)	–	11.244 (0.117)	–	16.945 (1.467)	–	17.659 (0.658)
NHITS-PK	–	10.039 (0.188)	–	12.339 (0.435)	–	**14.385** (0.092)	–	16.251 (0.089)