G-Net: a Recurrent Network Approach to G-Computation for Counterfactual Prediction Under a Dynamic Treatment Regime

Rui Li[1] * RUILI@ALUM.MIT.EDU
Stephanie Hu[1]* SHU98@MIT.EDU
Mingyu Lu[1] MINGYULU@MIT.EDU
Yuria Utsumi[1] YUTSUMI@MIT.EDU
Prithwish Chakraborty[2,3] PRITHWISH.CHAKRABORTY@IBM.COM
Daby M. Sow[2,3] SOWDABY@US.IBM.COM
Piyush Madan[2,3] PIYUSH.MADAN1@IBM.COM
Mohamed Ghalwash[2,3] MOHAMED.GHALWASH@IBM.COM
Zach Shahn[2,3] ZACH.SHAHN@IBM.COM
Li-wei H. Lehman[1] LILEHMAN@MIT.EDU
[1] *Massachusetts Institute of Technology, Cambridge, MA, USA*
[2] *IBM Research, Yorktown Heights, NY, USA*
[3] *MIT-IBM Watson AI Lab, Cambridge, MA, USA*

Abstract

Counterfactual prediction is a fundamental task in decision-making. This paper introduces G-Net, a sequential deep learning framework for counterfactual prediction under dynamic time-varying treatment strategies in complex longitudinal settings. G-Net is based upon g-computation, a causal inference method for estimating effects of general dynamic treatment strategies. Past g-computation implementations have mostly been built using classical regression models. G-Net instead adopts a recurrent neural network framework to capture complex temporal and nonlinear dependencies in the data. To our knowledge, G-Net is the first g-computation based deep sequential modeling framework that provides estimates of treatment effects under *dynamic and* time-varying treatment strategies. We evaluate G-Net using simulated longitudinal data from two sources: CVSim, a mechanistic model of the cardiovascular system, and a pharmacokinetic simulation of tumor growth. G-Net outperforms both classical and state-of-the-art counterfactual prediction models in these settings.

Keywords: Counterfactual prediction; deep sequential models; representation learning; causal inference; dynamic treatment regimes; g-computation.

Introduction

Counterfactual prediction is a fundamental task in decision-making that entails the estimation of expected future trajectories of variables of interest under alternative courses of action given observed history. This is particularly important in clinical settings, where physicians may have to choose between multiple treatment strategies for their patients but are unable to test all of them before making a decision. Treatment strategies of interest are usually *time-varying* (meaning they comprise decisions at multiple time points) and *dynamic* (meaning the treatment decision at each time point is a function of the history up to that time point). To aid in the choice between competing dynamic treatment strategies, it would be desirable to obtain counterfactual predictions of a patient's probability of adverse outcomes were they to follow each alternative strategy going forward given their observed covariate history up to the current time.

As an example, consider the problem of fluid administration to septic patients in intensive care units (ICUs) Shahn et al. (2020). It is frequently necessary for physicians to adopt strategies that administer large volumes of fluids to increase blood pressure and promote blood perfusion through organs in these patients; however, such strategies can lead to fluid overload, which can have serious adverse downstream effects such as pulmonary edema. Fluid administration strategies are time varying and dynamic because at each time point, physicians decide the vol-

* These authors contributed equally

ume of fluid to administer based on observed patient history (e.g. blood pressure and volume of fluid already administered) up that time point. To aid in the choice between competing dynamic fluid administration strategies, it would be desirable to obtain counterfactual predictions of a patient's probability of fluid overload (and other outcomes of interest) were they to follow each alternative strategy going forward given their observed covariate history up to the current time.

Counterfactual prediction is an inherently *causal* task in that it must account for the causal effects of following different treatment strategies. When treatment strategies of interest are time-varying and there is treatment-confounder feedback, so-called "g-methods" (Hernan and Robins, 2020; Robins and Hernan, 2009) are required to estimate their effects. G-methods include g-computation (Robins, 1986, 1987), structural nested models (Robins, 1994; Vansteelandt and Joffe, 2014), and marginal structural models (Robins et al., 2000; Orellana et al., 2008). Of these methods, g-computation is best suited for estimating the effects of general dynamic treatment strategies conditioned on high dimensional patient histories (Daniel et al., 2013).

G-computation works by estimating the conditional distribution of relevant covariates given covariate and treatment history at each time point, then producing Monte Carlo estimates of counterfactual outcomes by simulating forward patient trajectories under treatment strategies of interest. Critical to this method is the use of regression models for estimating the covariates and outcomes at each time point conditioned on observed history. While any regression models could theoretically be input to the g-computation algorithm, most g-computation implementations have employed simple regression models with limited capacity to capture complex temporal and nonlinear dependence structures. To capture long-term dependencies using such models, features summarizing patient history must be hand chosen by the analyst. In recent years, sequential deep learning methods such as recurrent neural networks (RNNs) have achieved state of the art performance in predictive modeling of complex time series data while imposing minimal modeling assumptions and without requiring custom feature construction.

In this paper, we propose G-Net, a sequential deep learning framework tailored for g-computation.G-Net admits the use of recurrent networks such as LSTMs to model time-varying covariates and treatment in a manner suitable for g-computation. The G-Net framework supports a flexible representation learning architecture that allows for various configurations that can be tailored to the task at hand. To our knowledge, this is the first work to propose an RNN-based approach to g-computation.

To evaluate G-Net, we used simulated data in which counterfactual ground truth can be known. Specifically, we used CVSim (Heldt et al., 2010), a well established mechanistic model of the cardiovascular system, to simulate counterfactual patient trajectories under various dynamic fluid and vasopressor administration strategies. Additionally, using a simulated tumor growth data set, we compared G-Net with recently introduced Counterfactual Recurrent Neural Networks (CRN) (Bica et al., 2020a), and a Recurrent Marginal Structural Network (R-MSN) (Lim et al., 2018), a recurrent neural network implementation of a history adjusted marginal structural model for estimating static time-varying treatment effects.

Overall, our contributions are three-fold: (**1**) We introduce a novel network architecture tailored to g-computation (Figure 1) that enables counterfactual predictions under dynamic treatment strategies using RNNs to provide estimates of individual or population-level time-varying treatment effects. (**2**) We provide a flexible sequential representation learning framework for g-computation under minimal modeling assumptions and evaluate alternative representation learning approaches to summarize patient history in settings with complex temporal dependencies. (**3**) Our simulation experiments demonstrating the superior performance of G-Net compared to other state-of-the-art approaches provide a template for causal model evaluation using complex and physiologically realistic simulated longitudinal data.

Related Work

Several recent works have proposed a deep learning framework for counterfactual prediction from observational data, including Atan et al. (2018); Alaa et al. (2017); Yoon et al. (2018). However, these studies have mostly focused on learning point exposure as opposed to time-varying treatment effects, which are the focus of this paper.

G-computation for estimating time-varying treatment effects was first proposed by Robins (1986). Illustrative applications of the general approach are provided in Taubman et al. (2009); Young et al.

(2011), and summaries of g-computation (and other "g-methods" for estimating time-varying treatment effects) can be found in Hernan and Robins (2020). The g-computation algorithm takes arbitrary regression models as inputs. While most applications (e.g. Taubman et al. (2009); Young et al. (2011)) have thus far employed classical generalized linear models, there is no conceptual barrier to using more complex regression models. RNNs, and in particular LSTMs, have achieved state of the art performance on a wide variety of time series regression tasks, including healthcare related tasks (Tomašev et al., 2019; Xiao et al., 2018; Choi et al., 2016). However, despite the success of RNNs for time series regression, we have not seen any "deep" implementations of g-computation.

Recent works by Lim et al. (2018); Bica et al. (2020a,b) presented deep learning approaches to estimate time-varying treatment effects. Bica et al. (2020a) applied ideas from domain adaptation to estimate treatment effects over time while Lim et al. (2018) used RNN regression models with history adjusted marginal structural models (MSMs) (Van der Laan et al., 2005) to make counterfactual predictions. However, none of these approaches are applicable to *dynamic* treatment strategies.

G-computation relies on different modeling assumptions than alternative approaches to estimating time-varying treatment effects, such as MSMs or structural nested models (Robins, 1994; Vansteelandt and Joffe, 2014), and can better handle dynamic treatment strategies, especially conditional on high dimensional health history. MSMs can only make counterfactual predictions under *static* time-varying treatment strategies that do not depend on recent covariate history. For example, a history adjusted MSM could estimate the probability of fluid overload given patient history under the (static) treatment strategy "*give 1 liter fluid each hour for the next 3 hours*", but it could not estimate the probability of fluid overload given patient history under the (dynamic) treatment strategy "*at each hour for the next 3 hours, if blood pressure is less than 65 then give 1 liter of fluids, otherwise give 0 liters*". History adjusted MSMs cannot estimate effects of time-varying treatment strategies that respond to changes in the patient's health history, but g-computation can. Further, g-computation is able to straightforwardly estimate the *distribution* of a counterfactual outcome under a time-varying treatment strategy. This is not straightforward to do with (history adjusted) MSMs.

Schulam and Saria (2017) propose Counterfactual Gaussian Processes (GPs), an implementation of continuous time g-computation. They only consider static time-varying treatment strategies, and while it appears that their method might straightforwardly be extended to handle dynamic strategies as well, it is known that GPs are intractable for large datasets (Titsias, 2009). Sparse GPs, which introduce M inducing points, have at least $O(M^2N)$ time complexity (Titsias, 2009), where N is the number of observations. Borgne et al. Borgne et al. (2021) investigated various machine learning implementations of g-computation; however, they only considered point exposures, i.e. treatments administered at a single time point. In contrast, our work is concerned with time-varying and dynamic sequential treatment strategies.

Recently, deep reinforcement learning (DRL) has been proposed for treatment decision support in a healthcare setting Lu et al. (2020); Peng et al. (2018); Yu et al. (2019). In contrast to our approach, these DRL approaches typically make simplifying Markov assumptions to make the backward iterative procedure of identifying an optimal regime tractable.

Background and Problem Definition

G-computation for Counterfactual Prediction

Our goal is to predict patient outcomes under various future treatment strategies given observed patient histories. Let:

- $t \in \{0, \dots, K\}$ denote time, assumed discrete, with K being the end of followup;
- A_t denote the observed treatment action at time t;
- Y_t denote the observed outcome at time t
- L_t denote a vector of covariates at time t that may influence treatment decisions or be associated with the outcome;
- $\bar{X}_t$ denote the history $X_0, \dots, X_t$ and $\underline{X}_t$ denote the future $X_t, \dots, X_K$ for arbitrary time varying variable X.

At each time point, we assume the causal ordering (L_t, A_t, Y_t). Let $H_t \equiv (\bar{L}_t, \bar{A}_{t-1})$ denote patient history preceding treatment at time t. A dynamic treatment strategy g is a collection of functions $\{g_0, \dots, g_K\}$, one per time point, such that g_t maps H_t onto a treatment action at time t. A simple dynamic strategy for fluid volume might be $g_t(H_t) = .5 \times \mathbf{1}\{bp_t < 65\}$, i.e. give .5 liters of fluid if mean arterial blood pressure is less than 65 at time t.

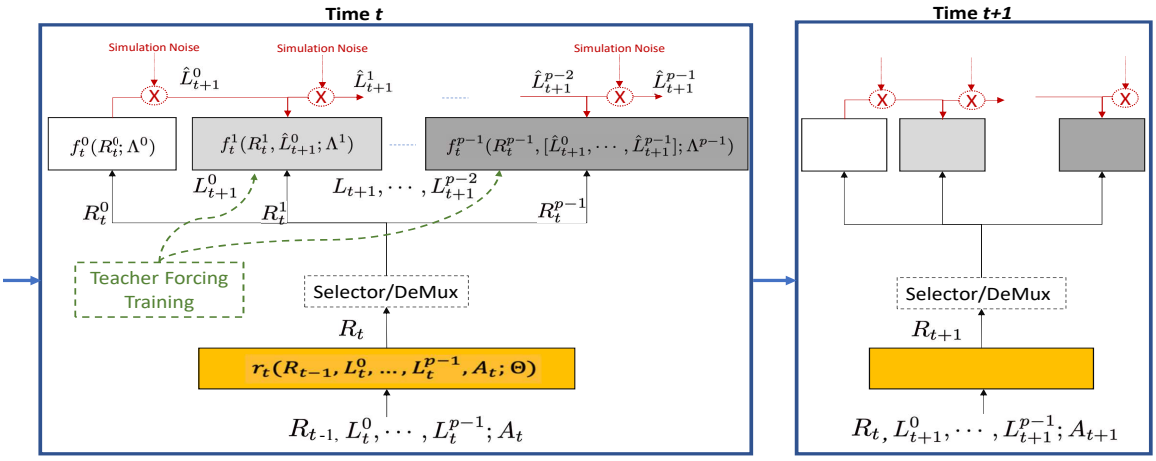

Figure 1: The G-Net: A flexible sequential deep learning framework for g-computation.

Let $Y_t(g)$ denote the counterfactual outcome that would be observed at time t had, possibly contrary to fact, treatment strategy g been followed from baseline (Robins, 1986). Further, let $Y_t(\bar{A}_{m-1}, \underline{g}_m)$ with $t \geq m$ denote the counterfactual outcome that would be observed had the patient received their observed treatments $\bar{A}_{m-1}$ through time $m-1$ then followed strategy g from time m onward, where g can be specified by the domain experts, e.g. clinicians.

In counterfactual point prediction, our goal is to estimate expected counterfactual patient outcome trajectories

$$\{E[Y_t(\bar{A}_{m-1}, \underline{g}_m)|H_m], t \geq m\} \tag{1}$$

given observed patient history through time m for any m and any specified treatment strategy g, where g is specified by a domain expert, e.g. a clinician. We might also be interested in estimating the counterfactual outcome distributions at future time points

$$\{p(Y_t(\bar{A}_{m-1}, \underline{g}_m)|H_m), t \geq m\}. \tag{2}$$

If we do not condition on anything in H_m, then (1) is an expectation (and (2) a distribution) over the full population. If we condition on a small subset of variables contained in patient history, then (1) is an expectation (and (2) a distribution) over a sub-population. If we condition on all elements of a patient history, then (1) is still technically only an expectation (and (2) a distribution) over a hypothetical sub-population with the exact patient history conditioned on, but in this case (1) and (2) practically amount to what is usually meant by personalized prediction.

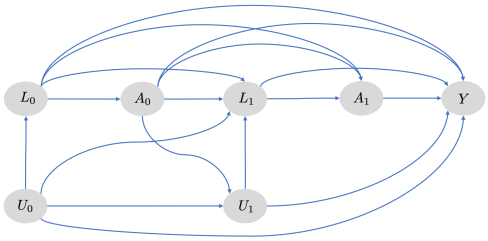

Figure 2: A causal DAG representing a data generating process in which Assumption 2 (sequential exchangeability) holds. Note that all variables influencing treatment (i.e. with arrows directly into treatment) and associated with future outcomes are measured.

Under the below standard assumptions, we can estimate (1) and (2) through g-computation (Robins, 1986).

1. **Consistency:** $\bar{Y}_K(\bar{A}_K) = \bar{Y}_K$
2. **Sequential Exchangeability:** $\underline{Y}_t(g) \perp\!\!\!\perp A_t|H_t \ \forall t$
3. **Positivity:** $P(A_t = g_t(H_t)) > 0 \ \forall\{H_t : P(H_t) > 0\}$

Consistency states that the observed outcome is equal to the counterfactual outcome corresponding to the observed treatment. Exchangeability states that there is no unobserved confounding. This would hold, e.g., if all drivers of treatment decisions that were prognostic for the outcome were observed as in Figure 2. Positivity states that the counterfactual treatment strategy of interest has some non-zero probability of actually being followed. Under the assumption that we specify certain extrapolative predictive models correctly, positivity is not strictly necessary.

Under assumptions 1-3, for $t = m$ we have simply that

$$p(Y_m(\bar{A}_{m-1}, g_m)|H_m) = p(Y_m|H_m, A_m = g_m(H_m)), \quad (3)$$

i.e. the conditional distribution of the counterfactual is simply the conditional distribution of the observed outcome given patient history and given that treatment follows the strategy of interest. For $t > m$, things are slightly more complex because we need to adjust for time-varying confounding. With $X_{i:j} = X_i, \ldots, X_j$ for any random variable X, under assumptions 1-3 the g-formula yields

$$\begin{aligned} &p(Y_t(\bar{A}_{m-1}, \underline{g}_m) = y|H_m) \\ &= \int_{l_{m+1:t}} p(Y_t = y|H_m, L_{m+1:t} = l_{m+1:t}, A_{m:t} = g(H_{m:t})) \\ &\qquad \times \prod_{j=m+1}^{t} p(L_j = l_j|H_m, L_{m+1:j-1} = l_{m+1:j-1}, \\ &\qquad\qquad A_{m,j-1} = g(H_m, l_{m+1:j-1})). \end{aligned} \quad (4)$$

It is not generally possible to compute this integral in closed form, but it could be approximated through Monte-Carlo simulation. We repeat Algorithm 1 M times. (There the outcome Y_t is without loss of generality deemed to be a variable in the vector L_{t+1}.) At the end of this process, we have M simulated draws of the counterfactual outcome for each time $t = \{m, \ldots, K\}$. For each t, the empirical distribution of these draws constitutes a Monte-Carlo approximation of the counterfactual outcome distribution (2). The sample averages of the draws at each time t are an estimate of the conditional expectations (1) and can serve as point predictions for $Y_t(\bar{A}_{m-1}, \underline{g}_m)$ in a patient with history H_m.

Key to the g-computation algorithm is the ability to simulate from joint conditional distributions $p(L_t|\bar{L}_{t-1}, \bar{A}_{t-1})$ of the covariates given patient history at time t. Of course, in practice we do not have knowledge of these conditional distributions and need to estimate them from data. Most implementations use generalized linear regression models to estimate the conditional distributions of the covariates. Often, these models do not capture temporal dependencies present in the patient data. We propose the G-Net for this task.

The G-Net Framework

The G-Net framework depicted in Figure 1 enables the use of sequential deep learning models to estimate conditional distributions $p(\bar{L}_t|\bar{L}_{t-1}, \bar{A}_{t-1})$ of covariates given history at each time and perform the g-computation algorithm described in Algorithm 1 to simulate variables under various treatment strategies. Without loss of generality, we set Y_t as one of the covariates in L for notational simplicity.

Algorithm 1 G-Computation (One simulation)

Set $a^*_m = g_m(H_m)$
Simulate l^*_{m+1} from $p(L_{m+1}|H_m, A_m = a^*_m)$
Set $a^*_{m+1} = g_m(H_m, l^*_{m+1}, a^*_m)$
Simulate l^*_{m+2} from $p(L_{m+2}|H_m, L_{m+1} = l^*_{m+1}, A_m = a^*_m, A_{m+1} = a^*_{m+1})$
Continue simulations through time K

Let $L^0_t, \ldots, L^{p-1}_t$ denote p components of the vector L_t. Here, a 'component' L^j_t is just a potentially multivariate subset of the covariate vector that is jointly modeled in the same 'box' in our architecture. We impose an arbitrary ordering $L^0_1, L^1_1, L^2_1, \ldots, L^{p-1}_1, A_1, \ldots, L^0_K, L^1_K, L^2_K, \ldots, L^{p-1}_K, A_k$ and estimate the conditional distributions of each L^j_t given all variables preceding it in this ordering. At simulation time, we exploit the basic probability identity

$$\begin{aligned} p(L_t|\bar{L}_{t-1}, \bar{A}_{t-1}) &= p(L^0_t|\bar{L}_{t-1}, \bar{A}_{t-1}) \times p(L^1_t|L^0_t, \bar{L}_{t-1}, \bar{A}_{t-1}) \\ &\quad \times \cdots \times p(L^{p-1}_t|L^0_t, \ldots, L^{p-2}_t, \bar{L}_{t-1}, \bar{A}_{t-1}) \end{aligned} \quad (5)$$

to simulate from $p(L_t|\bar{L}_{t-1}, \bar{A}_{t-1})$ by sequentially simulating each L^j_t from $p(L^j_t|L^0_t, \ldots, L^{j-1}_t, \bar{L}_{t-1}, \bar{A}_{t-1})$. There are at least two reasons to allow for subdivision of the covariates. First, if covariates are of different types (e.g. continuous, categorical, count, etc.), it is difficult to simultaneously simulate from their joint distribution. Second, customizing models for each covariate component can potentially lead to better performance. Figure 1 illustrates this decomposition where at each time point the components are depicted via ordered (grey shaded) boxes that are responsible for the estimation of the various terms needed to compute the conditional distributions. One could set $p = 1$ and model all covariates simultaneously or at the other extreme set p to be the total number of variables.

The sequential model used in G-Net provides us with estimates of the conditional expectations $E[L^j_t|\bar{L}_{t-1}, L^0_t, \ldots, L^{j-1}_t, \bar{A}_{t-1}]$ for all t and j. To simulate from $p(L^j_t|\bar{L}_{t-1}, L^0_t, \ldots, L^{j-1}_t, \bar{A}_{t-1})$, we proceed as follows. If L^j_t is multinomial, its conditional expectation defines its conditional density. If L^j_t has a continuous density, there are various approaches we

might take to simulate from its conditional distribution. Without making parametric assumptions, we could simulate from $L_t^j|L_t^0,\ldots,L_t^{j-1},\bar{L}_{t-1},\bar{A}_{t-1} \sim \hat{E}[L_t^j|L_t^0,\ldots,L_t^{j-1},\bar{L}_{t-1},\bar{A}_{t-1}] + \epsilon_t^j$, where ϵ_t^j is a draw from the empirical distribution of the residuals $L_t^j - \hat{L}_t^j$ in a holdout set not used to fit the model parameters used to generate $\hat{L}_t^j$ as an estimate of $E[L_t^j|L_t^0,\ldots,L_t^{j-1},\bar{L}_{t-1},\bar{A}_{t-1}]$. This method makes the simplifying assumption that the covariate error distribution does not depend on patient history. This is the approach we take in the experiments in this paper, and is depicted in the simulation noise nodes at the top of Figure 1. Alternatively, we might specify a parametric distribution for $L_t^j - E[L_t^j|L_t^0,\ldots,L_t^{j-1},\bar{L}_{t-1},\bar{A}_{t-1}]$, e.g. a Gaussian, and directly estimate its parameters by maximum likelihood.

As shown in the yellow boxes in Figure 1, at each time t, a representation R_t of patient history can be computed as $R_t = r_t(\bar{L}_t,\bar{A}_t;\Theta)$, where Θ represents model parameters learned during training. In its simplest form, r_t may just be an identity function passing the covariates without transformation. In other configurations, r_t can provide abstractions of histories using sequential learning architectures such as RNNs. Using the selector in Figure 1, different components of R_t can be passed to models for different covariates.This formulation of r_t allows for much flexibility in how information is shared across variables and time.

Estimates from each of the p covariate groups can then be obtained, as shown in Figure 1, by successive estimation of conditional expectations of covariates. Specifically, the conditional expectation of each $L_{t+1}^j, 0 \le j < p$ given the representation of patient history R_t and the other variables from time $t+1$ that precede it in the arbitrary predefined ordering is estimated by the functions f_t^j:

$$\begin{aligned} L_{t+1}^0 &= f_t^0(R_t;\Lambda_0) \\ L_{t+1}^1 &= f_t^1(R_t, L_{t+1}^0;\Lambda_1) \\ &\cdots \\ L_{t+1}^j &= f_t^j(R_t, L_{t+1}^0,\ldots,L_{t+1}^{j-1};\Lambda_j) \\ &\cdots \end{aligned} \tag{6}$$

where each f_t^j represents the specialized estimation function for group j and Λ_j are learnable parameters. The f_t^j might be sequential models (e.g. RNN) or models focused only on the representation at t (e.g. linear models).

The parameters (Θ,Λ) are learned by optimizing a loss function forcing G-Net to accurately estimate covariates L_t at each time point t using standard gradient descent techniques. We use teacher-forcing (i.e. using observed values of $L_{t+1}^{0:j-1}$ as inputs to f_t^j in equation 6) as shown in Figure 1 Williams and Zipser (1989). Given G-Net parameters, the distribution of the Monte Carlo simulations produced by Algorithm 1 constitute an estimate of uncertainty about a counterfactual prediction.

Evaluation and Experiment Settings

Simulations Using CVSim

To evaluate counterfactual predictions, it is necessary to use simulated data in which counterfactual ground truth for outcomes under alternative treatment strategies is known. To this end, we performed experiments on data generated by CVSim, a program that simulates the dynamics of the human cardiovascular system (Heldt et al., 2010). We used a CVSim 6-compartment circulatory model which takes as input 28 variables that together govern a hemodynamic system. We built on CVSim by adding stochastic components and interventions for the purposes of evaluating our counterfactual simulators. Full details are in the Appendix.

Data Generation: We generated an 'observational' dataset D_o under treatment regime g_o and two 'counterfactual' datasets D_{c1} and D_{c2} under treatment regimes g_{c1} and g_{c2}. The data generating processes producing D_o and D_{cj} were the same except for the treatment assignment rules. For each j, g_{cj} was identical to g_o for the first $m-1$ simulation time steps before switching to a different treatment rule for time steps m to K as illustrated in Figure 3.

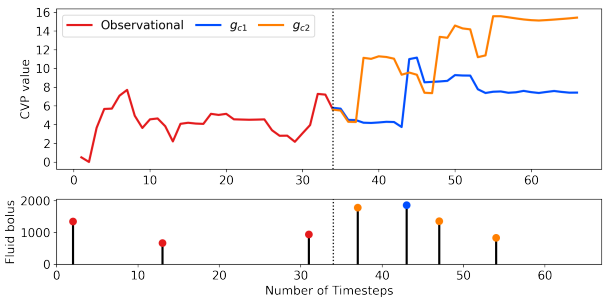

Figure 3: Covariate trajectories for the same patient under two different treatment strategies: g_{c1} (blue) and g_{c2} (orange) starting at $t = 34$ (black dashed line).

Under the (stochastic) observational treatment strategy g_o, the probability of receiving a non-zero dose of vasopressor or fluid at a given time increases as mean arterial pressure (MAP) and central venous pressure (CVP) decrease according to a logistic regression function. Given that a dose is non-zero, its amount is drawn from a normal distribution with mean inversely proportional to MAP and CVP. Since all drivers of treatment under g_o are observed in our data, the sequential exchangeability assumption holds and g-computation may be validly applied. Both g_{c1} and g_{c2} are similar to g_o, except they are deterministic treatment strategies with different coefficients linking treatment and dose volume to covariates. Again, details are provided in the Appendix.

Experimental Setup: We set ourselves the task of training G-Net on the observational regime dataset D_o and using it to predict the trajectories of patients in the counterfactual regime dataset D_{cj} for time steps m to K for each j. This setup is designed to evaluate the performance of G-Net in a situation in which we observe data from past patients (D_o) who received usual care (g_o) for K time steps and would like to predict how a new patient who has been observed for m time steps would fare were they to follow a different treatment strategy of interest (g_{cj}) for time steps m to K. This is a standard use case for counterfactual prediction. D_{cj} provides ground truth data for a collection of patients whose trajectories follow the path we are interested in predicting. By aggregating predictive performance metrics across simulated patients in D_{cj}, we generate measures of population level performance of G-Net at the counterfactual prediction task for which it was intended.

Table 1: G-Net Experimental Model Setup

	With pass thru r_t	**With sequential r_t**
f_i:LR	(*Linear*) • r_t: Identity • $p = 2$ • (f_0, f_1): linear	(*LSTM1*) • r_t: LSTM • $p = 2$ • (f_0, f_1): linear
f_i:RNN	(*LSTM2*) • r_t: Identity • $p = 2$ • (f_0, f_1): LSTMs.	(*LSTM3*) • r_t: LSTM • $p = 2$ • (f_0, f_1): LSTMs

As shown in Table 1, we explore two specific criteria: (a) using sequential vs. identity functions for r_t, and (b) using sequential models on the entire patient history vs. linear models focused only on the current time point for f_t. This provides us four implementations of G-Net: (*Linear*), (*LSTM1*), (*LSTM2*), and (*LSTM3*). Additionally, we implemented G-Net using multi-layer perceptron (MLP) as a nonlinear estimator.

We employed a version of G-Net with 2 boxes (one for all continuous variables and one for categorical variables). We chose $p = 2$ because we have both categorical and continuous variables in our experiments, and it is nontrivial to simulate from joint continuous and discrete error distributions. Parameter settings are in the Appendix.

The LSTM-based implementations of G-Net were compared to a baseline implementation of g-computation using generalized linear models (GLMs) and MLPs with observations from the previous time point as predictors. This comparison is meant to illustrate the importance of both flexible nonlinear modeling and automatic construction of relevant summaries of history that G-Net provides.

Evaluation: We evaluate the accuracy (Root Mean Squared Error, RMSE) of the counterfactual simulations generated by a G-Net as follows. Say D_{cj} comprises N_c trajectories of random variable $(\bar{L}_K^{cj}, \bar{A}_K^{cj})$. Given observed history $H_{mi}^{cj} = (\bar{L}_{mi}^{cj}, \bar{A}_{m-1i}^{cj})$ for patient i from D_{cj}, a G-Net G fit to D_o produces M (in our experiments, 100) simulations of the counterfactual covariate trajectory $\{\tilde{L}_{ti}^{cj}(H_{mi}^{cj}, G, k) : t \in m : K; k \in 1 : M\}$. These simulated trajectories are the light blue lines in Figure 4.

The G-Net's point prediction of L_{ti} is its estimate of $E[L_t(g_{cj})|H_m = H_{mi}]$, i.e. the average of the M simulations $\hat{L}_{ti}(G) \equiv \frac{1}{M}\sum_{k=1}^{M} \tilde{L}_t^c(H_{mi}^c, G, k)$. This is the dark blue line in Figure 4. If L_t has dimension d, we compute the MSE of counterfactual predictions by a G-Net G in the dataset D_c as $\frac{1}{N_c(K-m)d}\sum_{i=1}^{N_c}\sum_{t=m}^{K}\sum_{h=1}^{d}(L_{ti}^{h,CF} - \hat{L}_{ti}^{h,CF}(G))^2$. The RMSE is the square root of this value.

Experiments/Results: We used a total of 12,000 simulated trajectories generated from CVSim in D_o ($N_o = 12,000$), of which 83% (10,000) were used for training, and the remaining 17% (2,000) for validation. For testing, we generated 1000 simulated trajectories in the D_{cj} datasets ($N_c = 851$ after filtering as described in the Appendix). We included a total of 20 CVSim output variables (i.e. two treatment variables and 18 covariates, including all variables influencing treatment assignment under g_o) to construct D_o and D_{cj}; each trajectory is of length 66 time steps (d=20,

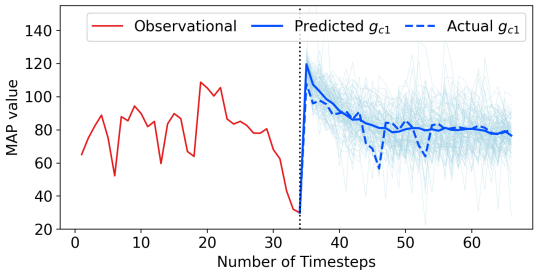

Figure 4: G-Net simulated MAP trajectories (100 Monte Carlo simulations in light blue, average in solid dark blue) and ground truth (dashed dark blue) for one patient under g_{c1} (starting $t = 34$).

K=66). In each D_{cj}, the switching time point m from g_o to g_c is fixed at 34 for all trajectories ($m = 34$).

We fit G-Net to the training portion of D_o (83%), and used the remaining portion as validation. Next, given observed covariate history through 34 time steps and treatment history through 33 time steps of each trajectory in each D_{cj}, we computed the RMSE of the G-Nets' counterfactual predictions for time steps 35 to 66, inclusive (total 32 predicted time steps).

Figure 5 compares the performance of LSTM-based implementations of G-Net to a GLM baseline in terms of RMSE under g_{c1} and g_{c2}. The advantage of LSTM-based G-Net over the GLM baseline increases over time for both counterfactual regimes. Among the three LSTM-based models, the models with LSTM representation layers, i.e. $LSTM1$ and $LSTM3$, performed slightly better than $LSTM2$, which did not have any representation layers. MLP results are not shown in these plots as the RMSE over time (1.03 to 1.80) was much higher than both the LSTM and GLM implementations of G-Net. The MLP regression models had better validation loss (0.48) than the GLM (0.51), meaning that the MLP's flexibility enabled it to perform better on the task of one-step-ahead prediction than the GLM (as expected), but flexibility without suitable incorporation of patient history led to unstable counterfactual prediction.

G-Net can be used to estimate population average counterfactual outcomes, quantities sometimes more relevant to policy decisions. The estimated and actual population-level average trajectories under g_{c1} and g_{c2} from the CVSim data set for one of the selected variable, AP (arterial pressure), are plotted in Figure 6. The plot shows that LSTM-based models outperform GLMs and more accurately predict counterfactual population AP trajectories. Predicted trajectories for additional covariates are shown in the Appendix.

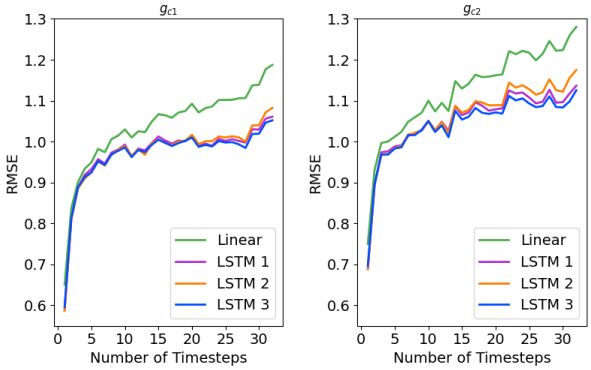

Figure 5: G-Net overall normalized RMSE (averaged across all output covariates) over time for g_{c1} and g_{c2}.

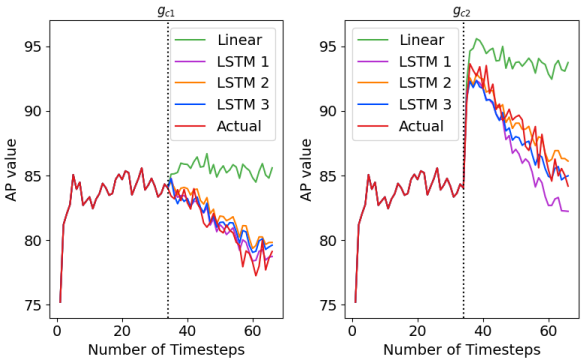

Figure 6: Estimated and actual population average trajectories under g_{c1} vs g_{c2} for arterial pressure (AP).

Comparison with R-MSN & CRN

We compared G-Net to a recurrent neural network implementation of a history adjusted marginal structural model (R-MSN) introduced in Lim et al. (2018) and a domain adaptive Counterfactual Recurrent Network (CRN) introduced in Bica et al. (2020a), using the simulation setup from Lim et al. (2018). As in Lim et al. (2018), we generate simulated 'observational' data from a pharmacokinetic-pharmacodynamic model of tumor growth under a stochastic regime (Geng et al., 2017a). In this simulation, chemotherapy and radiation therapy comprise a two dimensional time-varying treatment impacting tumor growth. Under the observational regime, probability of receiving each treatment at each time depends on volume history, so there is time-varying confounding. Details of data generation are in the Appendix.

To evaluate model performance, we generated four test sets in which various counterfactual regimes (static, as R-MSNs can only estimate counterfactuals under static regimes) were followed for the final four time points in the test set: give only radiotherapy, only chemotherapy, both chemotherapy and radiotherapy, and no treatment. Table 2 shows the percent RMSE of predictions from both models in the final four time points (when counterfactual strategies were in effect) conditioned on previous time points. (RMSEs were divided by the maximum possible tumor volume, 1150 cm^3, as in Lim et al. (2018).) LSTM-based implementations of G-Net outperformed CRN and R-MSN at this task for all treatment strategies, which is not surprising given that g-computation is known to be more statistically efficient than MSMs when all models are correctly specified Daniel et al. (2013). We also observe that the G-Net model $LSTM1$, with an LSTM representation layer as input to a linear regressor, achieved the best overall RMSE performance in three of the four cancer growth datasets. In contrast, using the higher-dimensional CVSim dataset, $LSTM3$, a G-Net model using an LSTM-representation-layer as input to an LSTM regressor, performed slightly better than the other variants of LSTM-based G-Net models.

Table 2: Cancer growth data: Percent RMSE for various prediction horizons. A-Overall. Best performing models in bold.

	t	rMSN	CRN	GNet Linear	GNet MLP	GNet LSTM1	GNet LSTM2	GNet LSTM3
No Treat	1	1.13	1.00	0.63	0.45	0.25	**0.25**	0.28
	2	1.24	1.20	1.21	0.87	**0.47**	0.47	0.54
	3	1.85	1.49	1.78	1.27	0.72	**0.70**	0.83
	4	2.60	1.78	2.35	1.67	1.01	**0.94**	1.20
	A	1.68	1.40	1.62	1.16	0.67	**0.64**	0.79
Radio	1	5.27	4.91	7.14	4.29	3.29	3.84	**3.10**
	2	5.38	4.92	7.43	4.03	3.14	3.60	**2.89**
	3	5.13	4.94	7.05	3.81	**3.02**	3.59	3.09
	4	4.81	4.92	6.50	3.68	**3.00**	3.76	3.40
	A	5.15	4.92	7.04	3.96	**3.11**	3.70	3.13
Chemo	1	1.42	1.04	1.58	0.56	**0.34**	0.47	0.39
	2	1.27	1.09	3.14	0.97	**0.63**	0.84	0.70
	3	1.46	1.03	4.52	1.64	**0.84**	1.08	1.04
	4	1.69	1.02	5.47	2.15	**0.89**	1.20	1.33
	A	1.47	1.05	3.96	1.46	**0.71**	0.94	0.93
Radio Chemo	1	4.76	4.66	7.76	4.21	3.10	3.72	**2.89**
	2	3.59	4.36	7.32	3.25	**2.28**	2.89	2.31
	3	2.76	3.65	6.13	2.43	**1.50**	2.34	2.05
	4	2.30	2.95	4.88	1.65	**1.18**	2.12	1.71
	A	3.48	3.96	6.62	3.04	**2.15**	2.84	2.28

Discussion and Conclusion

We introduced a novel and flexible framework, G-Net, for counterfactual prediction under dynamic treatment strategies through g-computation using sequential deep learning models and evaluated it in two realistically complex simulation settings where we had access to ground truth counterfactual outcomes. Our extensive experiments show strong empirical performance of G-Net under both static and dynamic time-varying treatment strategies. Using a cancer-growth dataset, we have demonstrated the superior performance of G-Net over two other state-of-the-art RNN based techniques, rMSN and CRN.

In the dynamic treatment setting, which we explored using data from CVSim, we sought to tease apart how much improvement in counterfactual predictive performance is coming from capturing nonlinear or long term temporal dependencies by comparing LSTM based implementations with both GLM and MLP-based implementations. The finding that MLPs (which capture nonlinearity but not long term temporal dependency) did not outperform GLMs suggests that much of the gain in performance from LSTM-based implementations comes from incorporation of patient history. Something to be cautious about when using flexible regression models is not to extrapolate to counterfactual treatment strategies too far afield from what appears in the data. We would also not expect our method to outperform alternatives when there are not nonlinear or long term dependencies in the data.

An area of future work is quantification of model uncertainty. Given G-Net parameters, the distribution of the Monte Carlo simulations produced by g-computation Algorithm 1 constitute an estimate of uncertainty about a counterfactual prediction. But this estimate ignores uncertainty about the G-Net parameter estimates themselves. One way to incorporate such model uncertainty would be to fit a Bayesian model and, before each Monte Carlo trajectory simulation in g-computation, draw new network parameters from their posterior distribution. These Monte Carlo draws would be from the posterior predictive distribution of the counterfactual outcome. Bayesian deep learning can be prohibitively computationally intensive, but can be approximated through dropout Gal and Ghahramani (2016).

In on-going work, we are investigating alternative representation learning architectures for summarizing patient history and the use of G-Net on real-world data to inform sequential treatment decision making in a critical care setting.

Acknowledgements

This work was funded by the MIT-IBM Watson AI Lab. We thank Dr. Jun Li and Professor Roger Mark at MIT for helpful discussions.

References

M Ahmed Alaa, Michael Weisz, and Mihaela van der Schaar. Deep counterfactual networks with propensity-dropout. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Onur Atan, James Jordan, and Mihaela Van der Schaar. Deep-treat: Learning optimal personalized treatments from observational data using neural networks. In *Proceedings of AAAI*, 2018.

Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. *International Conference on Learning Representations*, 2020a.

Ioana Bica, Ahmed M Alaa, and Mihaela van der Schaar. Time series deconfounder: Estimating treatment effects over time in the presence of hidden confounders. *International Conference on Machine Learning*, 2020b.

Florent Le Borgne, Arthur Chatton, Maxime Léger, Rémi Lenain, and Yohann Foucher. G-computation and machine learning for estimating the causal effects of binary exposure statuses on binary outcomes. *Scientific Reports*, 2021.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512. Curran Associates, Inc., 2016.

R.M. Daniel, S.N. Cousens, B.L. De Stavola, M.G. Kenward, and J.A.C. Sterne. Methods for dealing with time-dependent confounding. *Statistics in Medicine*, 32:1584–1618, 2013.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2016.

C. Geng, H. Paganetti, and C. Grassberger. Prediction of treatment response for combined chemo- and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model. *Scientific Reports*, 2017a.

Changran Geng, Harald Paganetti, and Clemens Grassberger. Prediction of treatment response for combined chemo- and radiation therapy for non-small cell lung cancer patients using a biomathematical model. *Scientific Reports*, 2017b.

T Heldt, R Mukkamala, GB Moody, and RG Mark. CVSim: An open-source cardiovascular simulator for teaching and research. *Open Pacing, Electrophysiol & Ther J*, 3:45–54, 2010.

Miguel Hernan and James Robins. *Causal Inference*. Boca Raton: Chapman & Hall/CRC, 2020.

Bryan Lim, Ahmed Alaa, and Mihaela Van der Schaar. Forecasting treatment responses over time using recurrent marginal structural networks. In *Neural Information Processing Systems (NIPS).*, 2018.

MingYu Lu, Zachary Shahn, Daby Sow, Finale Doshi-Velez, and Li-wei H. Lehman. Is deep reinforcement learning ready for practical applications in healthcare? a sensitivity analysis of Duel-DDQN for hemodynamic management in sepsis patients. In *Proceedings of the AMIA Annual Symposium*, 2020.

Liliana Orellana, James Robins, and Andrea Rotnitzky. Dynamic regime marginal structural mean models for estimation of optimal dynamic treatment regimes. *The international journal of biostatistics*, 2008.

Xuefeng Peng, Yi Ding, David Wihl, Omer Gottesman, Matthieu Komorowski, Li-wei H. Lehman, Andrew Ross, Aldo Faisal, and Finale Doshi-Velez. Improving sepsis treatment strategies by combining deep and kernel-based reinforcement learning. In *Proceedings of the AMIA Annual Symposium*, 2018.

Andrew Rhodes, Waleed Evans, Laura E.and Alhazzani, Mitchell M. Levy, Massimo Antonelli, Ricard Ferrer, Anand Kumar, Jonathan E. Sevransky, Charles L. Sprung, Mark E. Nunnally, et al. Surviving sepsis campaign: International guidelines for management of sepsis and septic shock:

2016. *Critical Care Medicine*, 2017. URL https://www.ncbi.nlm.nih.gov/pubmed/28101605.

James Robins. A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect. *Mathematical Modelling*, 1986.

James Robins. A graphical approach to the identification and estimation of causal parameters in mortality studies with sustained exposure periods. *Journal of Chronic Diseases*, 1987.

James Robins. Correcting for non-compliance in randomized trials using structural nested mean models. *Communications in Statistics-Theory and Methods*, 1994.

James Robins and Miguel Hernan. Estimation of the causal effects of time varying exposures. In Garrett Fitzmaurice, Marie Davidian, Geert Verbeke, and Geert Molenberghs, editors, *Longitudinal Data Analysis*, pages 553–599. Chapman and Hall, 2009.

James Robins, Miguel Hernan, and Babette Brumback. Marginal structural models and causal inference in epidemiology. *Epidemiology*, 2000.

Peter Schulam and Suchi Saria. Reliable decision support using counterfactual models. In *Neural Information Processing Systems (NIPS)*., 2017.

Zach Shahn, Nathan I. Shapiro, Patrick D. Tyler, Daniel Talmor, and Li-wei H. Lehman. Fluid-limiting treatment strategies among sepsis patients in the icu: a retrospective causal analysis. *Journal of Critical Care*, 24(62), 2020.

Sarah Taubman, James Robins, Murray Mittleman, and Miguel Hernan. Intervening on risk factors for coronary heart disease: An application of the parametric g-formula. *International journal of epidemiology*, 2009.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Aritifical Intelligence and Statistics*, 2009.

Nenad Tomašev, Xavier Glorot, Jack W Rae, Michal Zielinski, Harry Askham, Andre Saraiva, Anne Mottram, Clemens Meyer, Suman Ravuri, Ivan Protsyuk, et al. A clinically applicable approach to continuous prediction of future acute kidney injury. *Nature*, 572(7767):116, 2019.

Mark Van der Laan, Maya Petersen, and Marshall Joffe. History-adjusted marginal structural models and statically-optimal dynamic treatment regimens. *The international journal of biostatistics*, 2005.

Stijn Vansteelandt and Marshall Joffe. Structural nested models and g-estimation: The partially realized promise. *Statistical Science*, 2014.

R.J. Williams and D Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428, 06 2018.

Jinsung Yoon, James Jordan, and Mihaela Van der Schaar. Ganite: Estimation of individualized treatment effects using generative adversarial nets. In *ICLR.*, 2018.

Jessica Young, Lauren Cain, James Robins, Eilis O'Reilly, and Miguel Hernan. Comparative effectiveness of dynamic treatment regimes: An application of the parametric g-formula. *Statistics in biosciences*, 2011.

Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. In *https://arxiv.org/abs/1908.08796*, 2019.

Appendix A: Example Estimated Individual Trajectories in CVSim

Figure 7 illustrates how G-Net could be used for decision making in individual patients. Both patients (a) and (b) have relatively similar MAP trajectories in the first half of the simulated trajectories; however, while Patient (a)'s MAP is projected to increase significantly under a more aggressive fluid administration strategy (g_{c2}) compared to a less aggressive strategy (g_{c1}), Patient (b)'s MAP is not, possibly because their blood volume is already high. If these were real patients, a physician looking at these results might choose to administer fluids to patient (a) but not to patient (b) because the small predicted gain in MAP in patient (b) would not be worth the risk of fluid overload.

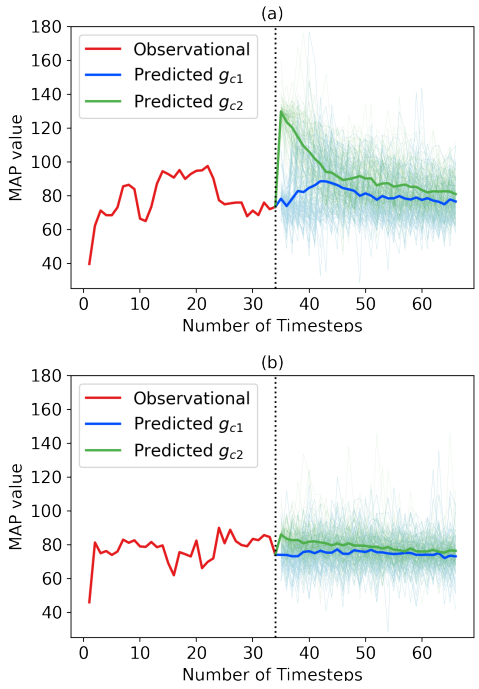

Figure 7: Estimated effects of two treatment strategies g_{c1} (blue) vs g_{c2} (green) in two patients with similar MAP. 100 Monte Carlo simulated trajectories in light blue and green respectively. Both patients would receive similar fluid amount under g_{c2}, but patient (a)'s predicted treatment effect is larger compared to no treatment than patient (b) due to differences in their underlying physiology. Predicted treatment effect under g_{c1} is similar for both patients.

Appendix B: Example Population-Level Trajectories in CVSim

The estimated and actual population-level average trajectories under g_{c1} and g_{c2} from the CVSim data set are plotted in Figure 8 for selected covariates. For every covariate shown, LSTM-based G-Net outperforms GLMs in predicting counterfactual trajectories.

Appendix C: CVSim Data and Settings

CVSim is an open-source cardiovascular simulator available at PhysioNet[1]. In this work, we focus on CVSim-6C, which consists of 6 components functioning as pulmonary and systemic veins, arteries, and micro-circulations. CVSim-6C is regulated by an arterial baroreflex system to simulate Sah's lumped hemodynamic model Heldt et al. (2010). The aggregate model is capable of simulating pulsatile waveforms, cardiac output and venous return curves, and spontaneous beat-to-beat hemodynamic variability. In this work, we modified and built on CVSim by adding stochastic components and interventions for the purposes of evaluating our counterfactual simulators. We call our stochastic simulation engine S-CVSim.

A CVSim 6-compartment circulatory model takes as inputs 28 variables that together govern a hemodynamic system. It then deterministically simulates forward in time a set of 25 output variables according to a collection of differential equations (parameterized by the input variables) modeling hemodynamics. By modeling only a subset of these variables in our dataset, we ensure that there are long range temporal dependencies present in our observed data that are mediated by the excluded variables. Important variables in CVSim include arterial pressure (AP), central venous pressure (CVP), total blood volume (TBV), and total peripheral resistance (TPR). In real patients, physicians observe AP and CVP and seek to keep them at safe levels by intervening on TBV (through fluid administration) and TPR (through vasopressors).

We defined simulated treatment interventions that were designed to mimic the impact of fluids and vasopressors. These simulated interventions alter the natural course of the simulation by increasing either

1. https://www.physionet.org/content/cvsim/1.0.0/

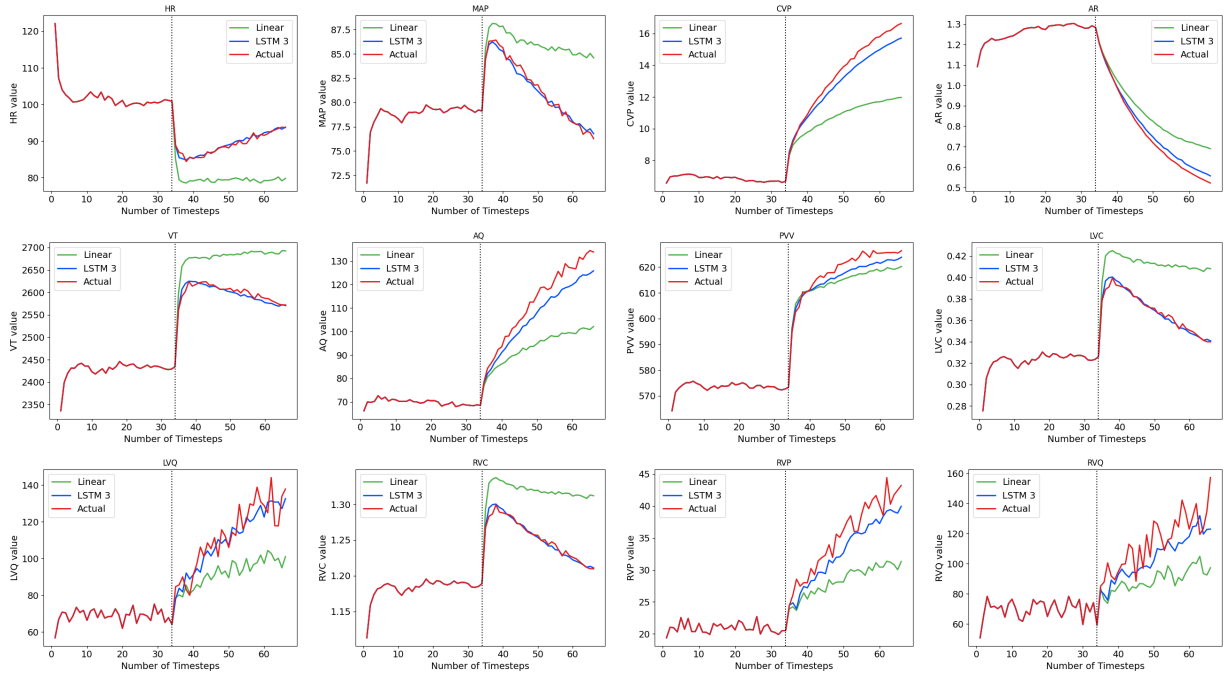

Figure 8: Estimated and actual population average trajectories under g_{c2} for selected covariates. The best-performing G-Net implementation (*LSTM3*) is shown in comparison to GLM baselines.

TBV (in the case of the simulated fluids intervention) or TPR (in the case of the simulated vasopressor intervention). We generated patients by randomly initiating baseline inputs (which we hid from our G-Nets to make this a stochastic modeling problem) within plausible physiologic ranges, then using CVSim to simulate covariates forward and intervening according to the relevant treatment strategy at each time step.

Inputs of S-CVSim

Varying hemodynamic parameters of CVSim causes it to simulate the cardiovascular system under various conditions. We randomly generate values of a subset of model parameters when initiating simulation at time 0 ($t = 0$) to obtain a population distribution of trajectories. These model parameters are listed in Table 3. **Note that initial value ranges do not bound values of input covariates for times $t > 0$.**

Table 3: Names and corresponding ranges of input parameters of S-CVSim.

Input Covariates	Range
Total Blood Volume	3,500 - 6,500
Nominal Heart Rate	40 - 160
Total Peripheral Resistance	0.1 - 1.3
Arterial Compliance	0.4 - 1.1
Total Zero-pressure filling Volume	500 - 3,500
Pulmonary Arterial Compliance	2.0 - 3.4
Pulmonary Microcirculation Resistance	0.4 - 1.00

Outputs of S-CVSim

At each time t, CVSim-6C generates hemodynamic data including vascular resistance and variables characterizing flow, pressure, and volume. In addition to the original 25 hemodynamic outputs, we introduce 4 new outputs, including systolic blood pressure, diastolic blood pressure, mean arterial pressure, and a pulmonary edema binary indicator, bringing the total number of output variables to 29. For this work, we only predict a subset of output covariates highlighted in **Table 4**. The complete list of outputs is also in **Table 4**. Below are definitions of the new variables we added.

- **Systolic Blood Pressure (SBP)** is defined as the highest measured arterial blood pressure during a cardiac cycle.
- **Diastolic Blood Pressure (DBP)** is defined as the lowest measured arterial blood pressure during a cardiac cycle.
- **Mean Arterial Pressure (MAP)** is defined as the average pressure during one cardiac cycle, computed as follows, $MAP = \frac{2*DBP+1*SBP}{3}$.
- **Pulmonary Edema (PE)** is defined as the indicator of pulmonary venous pressure above a threshold, computed as follows, $PE = [PVP > 25]$.

Table 4: Outputs of S-CVSim. Covariates highlighted with * are the selected outputs. (Note that treatment variables are not included in this table.)

Output Covariates	
Left Ventricle Pressure*	**LVP**
Left Ventricle Flow*	**LVQ**
Left Ventricle Volume	LVV
Left Ventricle Contractility*	**LVC**
Right Ventricle Pressure*	**RVP**
Right Ventricle Flow*	**RVQ**
Right Ventricle Volume	RVV
Right Ventricle Contractility*	**RVC**
Central Venous Pressure*	**CVP**
Central Venous Flow	CVQ
Central Venous Volume	CVV
Arterial Pressure*	**AP**
Arterial Flow*	**AQ**
Arterial Volume*	**AV**
Pulmonary Arterial Pressure	PAP
Pulmonary Arterial Flow	PAQ
Pulmonary Arterial Volume	PAV
Pulmonary Edema*	**PE**
Pulmonary Venous Pressure	PVP
Pulmonary Venous Flow	PVQ
Pulmonary Venous Volume*	**PVV**
Heart Rate*	**HR**
Arteriolar Resistance*	**AR**
Venous Tone*	**VT**
Total Blood Volume*	**TBV**
Intra-thoracic Pressure	PTH
Mean Arterial Pressure*	**MAP**
Systolic Blood Pressure*	**SBP**
Diastolic Blood Pressure	DBP

Disease Simulation, S_t

We introduce the concept S_t to simulate hemodynamic instability of the cardiovascular system, such as sepsis and bleeding at each time step. S_t consists of two events, **sepsis** and **blood loss**. In the module S_t, we set $P(S_t|L_t) = 0.05$, and $P(Sepsis|S_t) = P(BloodLoss|S_t) = 0.5$, with blood loss and sepsis being mutually exclusive events at any time t in the simulation process.

- When sepsis occurs, $L^{tpr}_{t+1} = \alpha_{sep} * L^{tpr}_t$, where $\alpha_{sep} = 0.7$. This means that L^{tpr}_t, total peripheral resistance at time t, decreases in α_{sep} at time $t+1$
- When blood loss happens, $L^{tbv}_{t+1} = \alpha_{loss} * L^{tbv}_t$, where $\alpha_{loss} = 0.85$ meaning that L^{tbv}_t, total blood volume at time t, would decrease in α_{loss} at time $t+1$.

Treatment Simulation, A_t

Under treatment rule g, treatment at time t is given by $A_t = g(\bar{L}_t)$. $A_t = (A^1_t, A^2_t)$ was two dimensional, with A^1_t indicating the amount by which total blood volume should increase (emulating the effect of a fluid bolus) and A^2_t indicating the amount by which total peripheral resistance should increase (emulating a vasopressor). The probability of choosing fluids or vasopressor is dependent on whether the patient has pulmonary edema; the treatments will not be administered at the same time. The dosage of A_t depends on a subset of L_t, which indicates hemodynamic balance. More specifically, since adequate blood pressure is an important clinical goal (Rhodes et al., 2017), we denote mean arterial pressure, MAP, of 70 mmHg and central venous pressure, CVP, of 10 mmHg as target goals. Therefore, we define $\Delta_{map,t} \equiv 70 - L^{map}_t$ and $\Delta_{cvp,t} \equiv 10 - L^{cvp}_t$ as proxies of how much dosage should be delivered. The following section will discuss the differences between between our counterfactual strategies g_{c1} and g_{c2} and our observational regime g_o.

Observational Regime, g_o

Under g_{obs}, the probability and dosage of treatment are denoted as the following

- Probability of treatment, $P(A_t|L_t) = \frac{1}{1+e^{-x}}$,where $x = C_1 * \Delta_{map} + C_2 * \Delta_{cvp} + C_0$, where probability of fluids $P(A^1_t|L_t) = B_0 - B_1 * L^{PE}_t$.

- If we administer fluids, we generate the dose (in mL) $A_t^1 \sim max(0, \beta_1^1 * \Delta_{map,t} + \beta_2^1 * \Delta_{cvp,t} + \mathcal{N}(0, 500))$.
- If we administer vasopressors, we generate the dose $A_t^2 \sim max(0, \beta_1^2 * \Delta_{map} + \beta_2^2 * \Delta_{cvp} + \mathcal{N}(0, 1))$

Administering a fluid dose of A_t^1 increases TBV as $L_{t+1}^{tbv} = L_t^{tbv} + A_t^1$, while administering a vasopressor dose of A_t^2 increases as $L_{t+1}^{tpr} = L_t^{tpr} + 1 - \frac{1}{A_t^2+1}$.

Counterfactual Regime, g_{c1}

Under g_{c1}, probability and dosage of treatment are denoted as following

- Probability of treatment, $P(A_t|L_t) = 1$ if and only if $L_t^{map} < 65$, where probability of fluids, $P(A_t^1|L_t) = 1$ if and only if $L_t^{pe} = 0$.
- If we administer fluids, we generate the dose (in mL) $A_t^1 \sim max(0, \beta_1^1 * \Delta_{map,t} + \beta_2^1 * \Delta_{cvp,t} + \beta_0^1)$.
- If we administer vasopressors, we generate the dose $A_t^2 \sim max(0, \beta_1^2 * \Delta_{map} + \beta_2^2 * \Delta_{cvp})$.

We experimented with multiple parameters and opted to use $C_0 = 0.65, C_1 = 0.3, C_2 = 0.24, B_0 = 0.5, B_1 = 0.2, \beta_0^1 = 1000, \beta_1^1 = 20, \beta_2^1 = 120, \beta_1^2 = 0.2, \beta_2^2 = 0.3$.

Counterfactual Regime, g_{c2}

Counterfactual regime g_{c2} follows the same rules as g_{c1}, except that $P(A_t|L_t) = 1$ if and only if $L_t^{map} < 75$.

Data Generation

To simulate the trajectory for a single patient under treatment strategy g,

- Initialize input variables $V_1, ..., V_N$ by drawing from independent uniform distributions using predefined plausible physiological ranges for each variable (Table 3).
- For t in $0 : K$,
  - generate L_t as $F_{sim,t}(V, \bar{L}_{t-1}^{'}, A_{t-1}, S_{t-1})$, where A_{-1} is taken to be 0, $F_{sim,t}(\cdot)$ denotes the CVSim simulation function, and $\bar{L}_{t-1}^{'}$ is the 'natural' value of the covariates but with total blood volume and arterial pressure at $t-1$ altered post hoc (i.e. after L_{t-1} values were recorded) according to A_{t-1};
  - generate A_t as $g(\bar{L}_t)$.

Using S-CVSim, we generated 15,000 trajectories under the observational regime and 1,000 trajectories under each counterfactual regime. All trajectories spanned 66 time steps in length. To filter physiologically improbable patient trajectories, we removed any trajectories whose non-treatment covariates at the first visit were greater than the 99^{th} percentile of the dataset or beyond 250 for MAP and 250 for HR. After filtering, we were left with 12,774 trajectories under the observational regime and 851 trajectories under each counterfactual regime. We removed an additional 774 trajectories under the observational regime so that we were left with exactly 12,000 samples, of which 10,000 (83%) were used for training and 2,000 (17%) were used for validation. There were no overlapping patients across the training, validation, and held-out test datasets. All continuous variables were z-score standardized using mean and standard deviations from the training set.

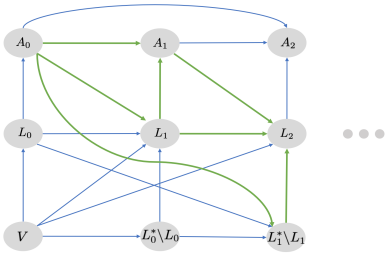

Figure 9: Causal DAG for D_o generated under observational regime g_o. Green arrows denote directed paths from treatments to outcomes. Treatment only depends on current covariates. L_t denotes 'observed' covariates and L_t^* denotes all variables generated. All covariates with arrows into treatment (i.e. MAP, CVP, and pulmonary edema) are observed, satisfying the sequential exchangeability assumption as in Figure 1.

The DAG in Figure 9 depicts the causal structure of D_o. Note that only observed covariates have arrows pointing into treatment (treatment is actually only a function of CVP, MAP, and pulmonary edema), so that there is no unobserved confounding and g-computation may be applied. Also, omitting certain variables from the analysis induces long term dependencies mediated by the missing variables. The unobserved V acting as a common cause of all non-treatment variables across time also induces long term dependencies.

CVSim Experimental Settings

Hyperparameter optimization was performed by searching over the hyperparameter space shown in **Table 5**. G-Net was trained using the Adam opti-

mizer with early stopping for a maximum of 50 epochs and a batch size of 32.

The experiments were performed on NVIDIA Tesla V100 SXM2 GPUs. After searching over the hyperparameter space, the optimal parameters for the network were found to have representation number of layers of 1, categorical and continuous number of layers of 1, representation hidden dimension of 32, categorical RNN hidden dimension of 8, continuous RNN hidden dimension of 64, and learning rate of 0.01.

Appendix D: Tumor Growth Simulation

Following PK-PD model in Geng et al. (2017b), tumor volume $V(t)$ was generated according to the formula:

$$V(t) = (1 + \underbrace{\rho log \frac{K}{V(t-1)}}_{\text{Tumor Growth}} - \underbrace{\beta_c C(t)}_{\text{Chemotherapy}} - \underbrace{(\alpha d(t) + \beta d(t)^2)}_{\text{Radiation}} + \underbrace{\epsilon_t}_{\text{Noise}})V(t-1) \quad (7)$$

where ρ, K, β, α are model parameters sampled for each patient according to prior distributions. $d(t)$ is the dose of radiation applied at t, and drug concentration $C(t)$ is generated according to an exponential decay with a half life 1 day.

Treatment Policy Assignment

Following Lim et al. (2018) to introduce time-dependent confounding, probabilities $p_c(t)$ and $p_d(t)$ of delivering nonzero doses of chemotherapy or radiotherapy, respectively, depend on tumor diameter.

$$p_c(t) = \rho(\frac{\gamma_c}{D_{max}}(\bar{D}(t) - \theta_c)); p_d(t) = \rho(\frac{\gamma_d}{D_{max}}(\bar{D}(t) - \theta_d))$$

where $\bar{D}(t)$ is the average tumor diameter over the last 15 days, ρ is the inverse logit function, and θ_* and γ_* are constant parameters. θ_* is fixed such that $\theta_c = \theta_d = \frac{D_{max}}{2}$, giving the model a 0.5 probability of treatment application when the tumor is half its maximum size. In this paper we set both γ_c and γ_d to 10.

10,000 trajectories were simulated and used for training, and 1,000 simulated trajectories for hyperparameter optimization (validation data), with another 1,000 for testing.

Settings for Propensity Networks, Encoder & Decoder for R-MSN

Hyperparameter optimization was performed using 50 iterations of random search and networks were trained using the Adam optimizer. For each set of samples, simulation trajectories were grouped into B minibatches and networks were trained for a maximum of 100 epochs. LSTM state sizes were also defined in relation to the number of inputs for the network C. The experiments were performed on NVIDIA Tesla K80 GPU with 2 vCPUs + 13 GB memory. The hyperparameter search space is shown in **Table 6**. The optimal hyperparameters for propensity score are dropout rate of 0.1, state size of 4c, minibatch size of 64, learning rate of 0.01, Max Gradient Norm of 0.5. The optimal parameter for encoder are dropout rate of 0.2, state size of 2c, minibatch size of 128, learning rate of 0.01, Max Gradient Norm of 0.5. The optimal hyperparameter of decoder are dropout rate of 0.1, state size of 4C, minibatch size of 128, learning rate of 0.01, Max Gradient Norm of 2.0.

Settings for Encoder & Decoder for CRN

Hyperparameter optimization was performed using 50 and 30 iterations of random search for the encoder and decoder, respectively, and networks were trained using the Adam optimizer. CRNs were implemented as described in Bica et al. (2020a)[2]. Parameters are also defined in relation to the size of the input, C, and the size of the balancing representation, R. The experiments were performed on NVIDIA Tesla K80 GPU with 12 vCPUs + 110 GB memory. The hyperparameter search space for the encoder and decoder is shown in **Table 7** and **Table 8**, respectively, with optimal parameters starred.

Settings for G-Net

Hyperparameter optimization was performed by searching over the hyperparameter space shown in **Table 9** with optimal parameters starred. G-Net was trained using the Adam optimizer with early stopping with patience of 10 epochs for a maximum of 50 epochs. The experiments were performed on NVIDIA Tesla K80 GPU with 12 vCPUs + 110 GB memory.

2. We used the publicly available implementation from https://github.com/ioanabica/Counterfactual-Recurrent-Network.

Table 5: Hyperparameter search space for G-Net in CVSim experiments. Note that when training *LSTM3*, the number of layers was set to 1 for the representational layer, as well as the categorical and continuous boxes (e.g. no tuning over number of layers).

	Hyperparameters	**Search Range**
Linear	Learning Rate	0.001*, 0.01
MLP	Number of Layers	2*, 4
	Hidden Dimension	16, 32*
	Learning Rate	0.001*, 0.01
LSTM 1	Number of Layers (Representation)	1, 2, 4*
	Hidden Dimension (Representation)	16, 32*
	Learning Rate	0.001, 0.01*
LSTM 2	Number of Layers (Categ & Contin)	1, 2, 4*
	Hidden Dimension (Categorical)	8*, 16
	Hidden Dimension (Continuous)	32, 64*
	Learning Rate	0.001*, 0.01
LSTM 3	Hidden Dimension (Representation)	16, 32*
	Hidden Dimension (Categorical)	8*, 16
	Hidden Dimension (Continuous)	32, 64*
	Learning Rate	0.001, 0.01*

Table 6: Hyperparameter Search Range for Propensity Networks, Encoder, and Decoder

Hyperparameters	**Search Range**
Search Iteration	50
Dropout Rate	0.1 , 0.2 , 0.3, 0.4, 0.5
State Size	1C, 2C, 4C, 8C, 16C
Minibatch Size	64, 128, 256
Learning Rate	0.01, 0.005, 0.001
Max Gradient Norm	0.5, 1.0, 2.0

Table 7: Hyperparameter Search Range for Encoder.

Hyperparameters	**Search Range**
Search Iteration	50
Learning Rate	0.01, 0.001*, 0.0001
Minibatch Size	64*, 128, 256
RNN Hidden Units	0.5C, 1C, 2C, 3C, 4C*
Balancing Representation Size	0.5C, 1C, 2C*, 3C, 4C
FC Hidden Units	0.5R, 1R, 2R*, 3R, 4R
RNN Dropout Probability	0.1*, 0.2, 0.3, 0.4, 0.5

Table 8: Hyperparameter Search Range for Decoder.

Hyperparameters	**Search Range**
Search Iteration	30
Learning Rate	0.01, 0.001*, 0.0001
Minibatch Size	256*, 512, 1024
RNN Hidden Units	Balancing Representation Size of Encoder
Balancing Representation Size	0.5C, 1C, 2C, 3C*, 4C
FC Hidden Units	0.5R, 1R, 2R, 3R*, 4R
RNN Dropout Probability	0.1*, 0.2, 0.3, 0.4, 0.5

Table 9: Hyperparameter search space for G-Net in cancer experiments.

	Hyperparameters	**Search Range**
Linear	Learning Rate	0.001*, 0.01
MLP	Number of Layers	1, 2, 4*
	Hidden Dimension	2, 4*, 8
	Learning Rate	0.001*, 0.01
LSTM 1	Number of Layers (Representation)	1*, 2
	Hidden Dimension (Representation)	8, 16, 32, 64*
	Learning Rate	0.001, 0.01*
LSTM 2	Number of Layers (Continuous)	1*, 2, 4
	Hidden Dimension (Continuous)	4, 16, 64*
	Learning Rate	0.001, 0.01*
LSTM 3	Number of Layers (Representation)	1*, 2
	Number of Layers (Continuous)	1*, 2
	Hidden Dimension (Representation)	16, 32, 64*
	Hidden Dimension (Continuous)	16, 32*, 64
	Learning Rate	0.001, 0.01*