Uncertainty Quantification for Conditional Treatment Effect Estimation under Dynamic Treatment Regimes

Leon Deng[1]* LYDENG@MIT.EDU

Hong Xiong[2]* HONGXIONG@HSPH.HARVARD.EDU

Feng Wu[1]* WUFENG@MIT.EDU

Sanyam Kapoor[3] SANYAM@NYU.EDU

Soumya Ghosh[4] GHOSHSO@US.IBM.COM

Zach Shahn[4,5] ZACHARY.SHAHN@SPH.CUNY.EDU

Li-wei H Lehman[1]† LILEHMAN@MIT.EDU

1. Massachusetts Institute of Technology, Cambridge, MA, USA
2. Harvard University, Cambridge, MA, USA
3. New York University, NY, USA
4. MIT-IBM Watson AI Lab, Cambridge, MA, USA
5. City University of New York, NY, USA

Abstract

In medical decision-making, clinicians must choose between different time-varying treatment strategies. Counterfactual prediction via g-computation enables comparison of alternative outcome distributions under such treatment strategies. While deep learning can better model high-dimensional data with complex temporal dependencies, incorporating model uncertainty into predicted conditional counterfactual distributions remains challenging. We propose a principled approach to model uncertainty in deep learning implementations of g-computations using approximate Bayesian posterior predictive distributions of counterfactual outcomes via variational dropout and deep ensembles. We evaluate these methods by comparing their counterfactual predictive calibration and performance in decision-making tasks, using two simulated datasets from mechanistic models and a real-world sepsis dataset. Our findings suggest that the proposed uncertainty quantification approach improves both calibration and decision-making performance, particularly in minimizing risks of worst-case adverse clinical outcomes under alternative dynamic treatment regimes. To our knowledge, this is the first work to propose and compare multiple uncertainty quantification methods in machine learning models of g-computation in estimating conditional treatment effects under dynamic treatment regimes.

1. Introduction

Clinicians often have to choose among multiple treatment options for their patients but do not have the ability to test candidate strategies before making a decision. Thus, counterfactual prediction (i.e. the estimation of potential future trajectories of outcomes of interest under alternative courses of action given observed history) is useful for decision-making. In this work, we focus on treatment strategies that are *time-varying*, in which treatments can vary over time, and *dynamic*, in which treatment decisions vary over time and depend on the patient's preceding history. Recent works have leveraged machine learning (ML) for counterfactual prediction (Lim et al., 2018; Bica et al., 2020a; Melnychuk et al., 2022; Li et al., 2021; Xiong et al., 2024) , but most prior works (Lim et al., 2018; Bica et al., 2020a; Melnychuk et al., 2022) focus on time-varying but non-dynamic treatment settings, and none of these prior works tackle the challenge of uncertainty quantification for conditional treatment effect estimation under dynamic treatment settings.

In choosing among multiple treatment options, it is important to not only estimate expected or most

[*] Co-first authors.
[†] Corresponding author.

likely outcomes for a patient, but also *predictive distributions* to properly navigate risk-reward trade-offs inherent in many clinical treatment decisions. G-computation (Robins, 1986) is uniquely well-suited to provide estimates of conditional counterfactual predictive distributions under *dynamic* treatment strategies that depend on evolving patient history. However, as with all conditional effect estimators, when machine learning is employed for regression modeling, it is challenging to incorporate uncertainty about regression parameters stemming from sampling variability into predictive distributions (Chernozhukov et al., 2017). It is indeed proven impossible to derive confidence intervals with theoretical guarantees if machine learning were used for conditional effect estimation (Chernozhukov et al., 2017). In the absence of methods with theoretical guarantees, one open challenge remains in developing practical but principled approaches to generating counterfactual predictive distributions from deep learning implementations for conditional treatment effect estimation.

In this work, we propose and empirically compare multiple practical but principled approximate-Bayesian approaches to generate counterfactual predictive distributions from deep learning-based g-computation. G-Net (Li et al., 2021) and G-Transformer (Xiong et al., 2024) are two deep learning approaches that support g-computation for counterfactual prediction under dynamic and time-varying treatment regimes, where treatments can depend on past covariate history. However, neither of these prior efforts attempt to incorporate uncertainty from sampling variability into the predictive distributions they generate. While they produce consistent estimates that will approach the truth given unrealistically vast amounts of data, they typically underestimate uncertainty when making predictions from models trained on any health data set available in practice.

Improperly estimating uncertainty can lead to poor decision-making. For example, fluid administration strategies for septic patients must balance the risk of fluid overload from too much fluid against the risk of hypotension from too little fluid. The joint counterfactual distributions of blood pressure and blood volume trajectories (as well as other outcomes that inform the decision maker's utility) under alternative fluid strategies are important to model, not just their expected values. Further, it is important that the tails of these distributions are well approximated.

We investigate multiple approaches of approximating the Bayesian posterior counterfactual predictive distribution, including Deep Ensembles (DE) (Lakshminarayanan et al., 2017), Variational Dropout (VD) (Gal and Ghahramani, 2016b), and a combined VD-DE approach. We use G-Net and G-Transformer without support for model uncertainty as the baseline models. When comparing models with and without uncertainty quantification, we investigate several aspects: the extent to which uncertainty-aware methods enhance calibration in the predictive counterfactual distribution compared to baseline models; the influence of uncertainty quantification on predictive accuracy; and whether improved calibration translates into more effective downstream decision-making.

We evaluate our approach in calibration and predictive accuracy using two simulated datasets from mechanistic models and one real-world dataset. Our contributions are as follows:

- **Incorporate uncertainty from sampling variability into deep-learning based counterfactual prediction models under dynamic treatment regimes.** We apply variational dropout (Gal and Ghahramani, 2016b) and deep ensemble (Lakshminarayanan et al., 2017), two commonly-used methods for uncertainty quantification, to the G-Net and G-Transformer implementations of g-computation.

- **Compare the performance of different uncertainty quantification approaches in counterfactual prediction under dynamic treatment regimes, and demonstrate improved calibration performance over the baseline models.** We evaluate our approach for predicting conditional distributions under counterfactual treatment regimes using two simulated datasets from mechanistic models, and under observational regime using a real-world ICU dataset. Our results indicate that uncertainty quantification methods generally improve the calibration, while maintaining similar levels of predictive accuracy in comparison to the baseline models.

- **Characterize impact of approximate-Bayesian approaches to modeling counterfactual predictive distributions on decision making tasks in mechanistic simulation settings.** We demonstrate how different uncertainty quantification approaches can impact decision making tasks involving minimizing risks for (worst-case) adverse clinical outcomes under alternative dynamic treatment regimes. We design a clinical decision-making task that emulates the decision-making process of a doctor attempting to choose a fluid administration strategy to avoid the adverse outcomes of hypotension and pulmonary edema. We compare the performance of the modified models against that of the base G-Net and G-Transformer models, and demonstrate that uncertainty quantification methods improve decision-making performance, particularly in minimizing risks of worst-case adverse clinical outcomes under alternative dynamic treatment regimes.

2. Related Work

Implementations of g-computation using parametric generalized linear regression models can employ bootstrap to generate valid predictive confidence intervals or perform a standard fully Bayesian analysis (Keil et al., 2017) to draw from the posterior predictive distribution under the strong assumption that models are correctly specified. In complex settings, flexible deep learning implementations of g-computation might be more desirable. Modern machine learning methods are particularly well suited to model high-dimensional data with complex temporal dependencies. However, it is a challenge to properly incorporate model uncertainty into machine learning generated counterfactual predictions, even if interest only centers on the simpler problem of estimating the average population effect (Chernozhukov et al., 2017). Conformal inference has been developed for heterogeneous effects of point exposures (Lei and Candès, 2021), but has not been extended to settings with time-varying treatments. Thus, machine learning based counterfactual prediction under time-varying strategies requires uncertainty quantification without theoretical guarantees.

Numerous prior works have proposed machine learning based approaches for treatment effect estimation (Atan et al., 2018; Alaa et al., 2017; Yoon et al., 2018; Lim et al., 2018; Bica et al., 2020a,b; Melnychuk et al., 2022; Li et al., 2021; Xiong et al., 2024). However, many of the prior works either focus on point-exposure settings (Atan et al., 2018; Alaa et al., 2017; Yoon et al., 2018), or time-varying but non-dynamic treatment settings (Lim et al., 2018; Bica et al., 2020a,b; Melnychuk et al., 2022). While Gaussian Processes (GPs) (Schulam and Saria, 2017), can potentially provide a flexible non-parametric framework for modeling uncertainty, their limited inductive biases are often less effective for complex high-dimensional data. None of these prior ML approaches focus on uncertainty quantification for conditional treatment effect estimation in a dynamic treatment settings.

G-Net (Li et al., 2021) and G-Transformer (Xiong et al., 2024) are neural network based implementations of g-computation, where G-Net uses RNNs and G-Transformer uses transformer encoders. They have been shown to predict outcomes more accurately than other counterfactual prediction methods and g-computation implementations. G-Net and G-Transformer are used to implement g-computation by approximating the conditional distributions of covariates (including outcomes of interest) given history. Although G-Net and G-Transformer models uncertainty around a patient's trajectories, they do not support uncertainty around model parameters.

3. Methods

Background G-computation works by estimating the conditional distribution of relevant covariates given covariate and treatment history at each time point, then producing Monte Carlo (MC) estimates of counterfactual outcomes by simulating forward patient trajectories under treatment strategies of interest (Robins, 1986). We build on deep-learning approaches to g-computation (Robins, 1986), including G-Net (Li et al., 2021) and G-Transformer (Xiong et al., 2024), by adding support for model uncertainty, and use G-Net and G-Transformer as baselines in analyzing results. We detail the implementation of the base G-Net and G-Transformer models in the Appendix.

We enhance the G-Net and G-Transformer models by using two commonly used uncertainty quantification methods: *variational dropout* and *deep ensemble*. We aim to improve calibration, while maintaining predictive accuracy. We also experimented with incorporating a third uncertainty quantification approach, *Stochastic Weight Averaging-Gaussian (SWAG)* (see Appendix for details). We evaluate the models' performance on three datasets: CVSim (Heldt et al., 2010), which simulates the human cardiovascular system; Cancer Growth (Geng et al., 2017), which simulates cancerous tumor growths; and MIMIC-IV (Johnson et al., 2023), a real-world dataset of ICU sepsis patients.

3.1. Uncertainty Quantification

While simulation with G-Net and G-Transformer accounts for the uncertainty in the covariate distribution (by using empirical residuals), it does not account for the uncertainty in the model parameters. To address this limitation, we enhance G-Net and G-Transformer with two different methods to produce uncertainty estimates: *variational dropout* (VD) (Gal and Ghahramani, 2016b) and *deep ensemble* (DE) (Lakshminarayanan et al., 2017), and their combined approach VD-DE. This section describes these uncertainty quantification methods, their implementations, and how we adapt them to work well with the existing G-Net and G-Transformer models. Additionally, we explore using Stochastic Weight Averaging-Gaussian (SWAG) (Maddox et al., 2019; Izmailov et al., 2018) as an alternative uncertainty quantification technique.

Variational Dropout (VD). Dropout is traditionally used in training neural networks as a form of regularization (Srivastava et al., 2014). Activations from network layers are masked during training using a fixed dropout probability, but no masking occurs at test time. However, Gal and Ghahramani (2016b) show that dropout, if used during test time, can be used to produce a variational approximation to the posterior distribution of a Bayesian neural network. Simulating multiple forward passes of a trained model with independent and identically distributed dropout masks gives us samples that take into account the uncertainty over model parameters. To implement variational dropout for the G-Transformer (or G-Net), we start with a trained model and generate sampled trajectories by applying dropout masks during inference. This approach enables us to capture uncertainty in the model parameters in addition to the covariate distribution.

Deep Ensemble (DE). Deep ensemble (Lakshminarayanan et al., 2017) is a flexible method of producing uncertainty estimates for neural network parameters. We train an ensemble of models through *bagging*, with randomization coming from parameter initialization and the order in which data is fed into the model during training. Each model within the ensemble predicts a conditional Gaussian distribution over the continuous covariates, and optimizes the negative log-likelihood. With θ representing a model's parameters, we have:

$$\begin{aligned}\text{Loss}(\theta, \mathbf{x}, y) &\triangleq -\log p_\theta(y|\mathbf{x}) \\ &= \frac{\log \sigma^2_\theta(\mathbf{x})}{2} + \frac{(y-\mu_\theta(\mathbf{x}))^2}{2\sigma^2_\theta(\mathbf{x})} + \text{constant}.\end{aligned} \tag{1}$$

Variational Dropout with Deep Ensembling (VD-DE) We explored combining variational dropout and deep ensemble by independently training $M = 20$ variational dropout models on MSE using random initializations and random shufflings of the training data. Details in Appendix.

SWAG. Stochastic Weight Averaging-Gaussian (SWAG) (Maddox et al., 2019; Izmailov et al., 2018) is an approximate Bayesian inference approach which uses SGD iterates to construct a Gaussian neural network parameter posterior. Details in the Appendix.

3.2. Evaluation Methods

Calibration We assess the calibration of a trained model as follows. Given quantiles α_{low} and α_{high}, calibration is evaluated by calculating the frequency with which the actual ground-truth covariates fall between the α_{low} and α_{high} quantiles of the predicted covariates for each test patient. The frequency is calculated across all patients in test set, time-steps, and covariates. We calculate the proportion of times the actual falls within the target quantile range of the simulated trajectories (e.g. from 100 Monte Carlo simulations), and if it is close to $\alpha_{high} - \alpha_{low}$, the model is well-calibrated. In this work, we use 0.05 and 0.95 as our quantile range, and thus 0.90 indicates perfect calibration.

RMSE We use root mean-squared error (RMSE) to evaluate the accuracy in counterfactual prediction. For each individual test patient, trajectories of 100 Monte Carlo simulations are averaged to represent the expected trajectory and compared to the ground-truth of that patient to compute individual-level RMSE. The variables are normalized before the RMSE computation.

3.3. Data Sets

CVSim Data Generation We use a CVSim (Heldt et al., 2010) 6-compartment circulatory model which takes as input 28 variables that together govern a hemodynamic system. We use the same setting as in (Li et al., 2021) in generating CVSim data. Briefly, we build on CVSim by adding stochastic components and interventions for the purposes of evaluating our counterfactual simulators. We generate an 'observational' dataset (N=10,000) D_o under treatment regime g_o as training and validation data, and two 'counterfactual' datasets D_{c1} and D_{c2} (N=851 each after filtering) under treatment regimes g_{c1} and g_{c2} as our test set. The data generating processes producing D_o and D_{cj} were the same except for the treatment assignment rules, where g_{c1} and g_{c2} represent fluid conservative and liberal regime respectively. For both counterfactual datasets, the first $m-1$ simulation time steps follow the same treatment regime g_o as in the observational dataset before diverging to a different treatment regime for time steps m (m=34) to K (K=66).

Cancer Growth Data Generation We generate simulated 'observational' data from a pharmacokinetic-pharmacodynamic model of tumor growth (Geng et al., 2017) under a stochastic regime (N=10,000 in training/validation data). 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. We use the same experiment settings and data generation procedure as reported in (Li et al., 2021). We generate four test sets (N=1,000 each) in which we follow four counterfactual regimes for the final four time points in the test set: (1) only radiotherapy, (2) only chemotherapy, (3) both chemotherapy and radiotherapy, and (4) no treatment.

MIMIC-IV we evaluate the impact of adding uncertainty quantification to the G-Transformer model using real-world ICU data from the MIMIC-IV database (Johnson et al., 2023; Goldberger et al., 2000). The dataset consists of 8,920 patients meeting sepsis-3 criteria.

Impact on Clinical Decision Making. We investigate how uncertainty predictions can aid clinical decision-making, especially in high-risk scenarios. For the CVSim dataset, we focus on the problem of administering fluid to a patient. For the Cancer Growth dataset, we focus on the problem of minimizing the occurrences of cancerous tumors growth exceeding a target threshold.

Simulation For a given counterfactual treatment strategy, we simulate 100 trajectories per patient. For direct comparison, the ensemble approach produces 100 trajectories per patient across multiple ensemble models, e.g. for ensemble of 20 models, each produces 5 per patient to yield an overall of 100 MC simulations per patient.

4. CVSim Results

4.1. Evaluation of Uncertainty Predictions

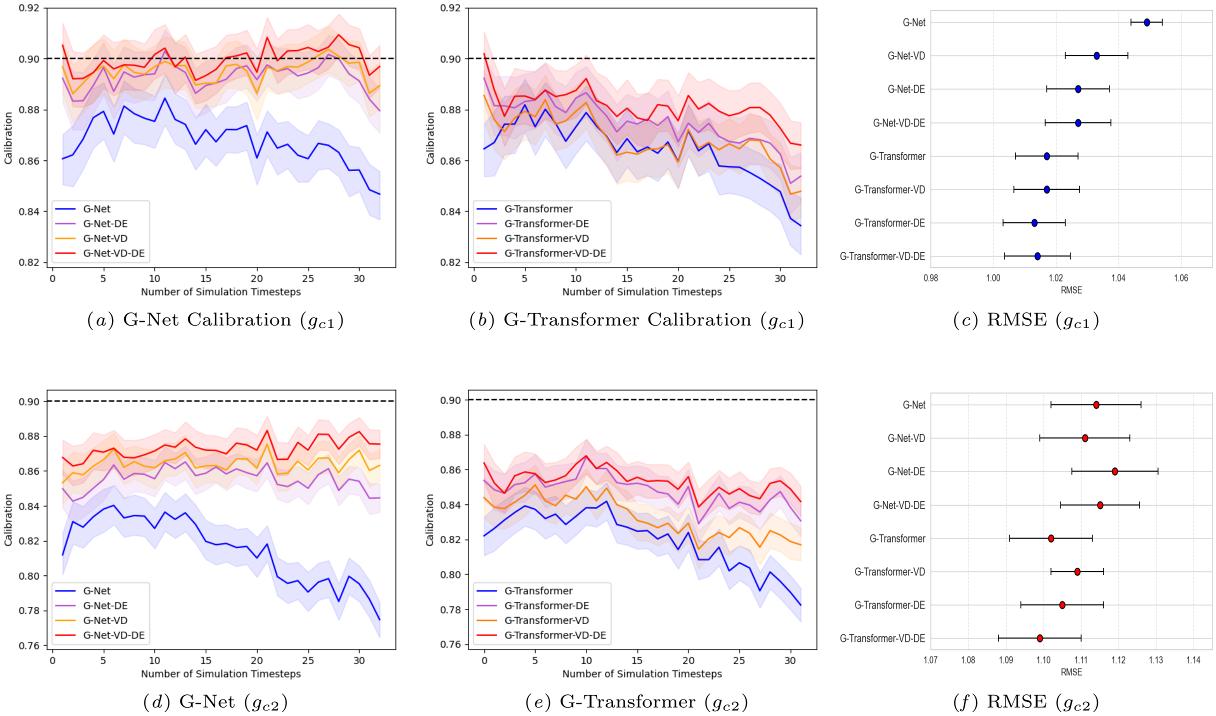
For each model and on each counterfactual regime (g_{c1} and g_{c2}), we calculated the per time-step calibration and the individual-level RMSE. Figure 1 illustrates the calibration performance of uncertainty quantification techniques within the G-Net and G-Transformer frameworks. 95% confidence intervals were calculated based on 1,000 bootstrapped test samples. In general, methods incorporating uncertainty modeling, such as VD, DE, and their combination (VD-DE), consistently improved calibration across all time steps compared to baseline models, with reduced calibration decay over time.

As depicted in Figure 1, adding uncertainty quantification significantly enhances calibration relative to the baseline. Moreover, Figures 1(c) and 1(f) confirm that incorporating uncertainty quantification does not compromise RMSE performance. Approaches with uncertainty modeling achieve RMSE values comparable to their respective base G-Net or G-Transformer models, maintaining accuracy while improving calibration.

4.2. Impact on Clinical Decision Making

CVSim Decision Rules We design our decision rule to emulate a doctor's decision making with an aim to avoid worst-case scenarios of patient developing hypotension, i.e. in which a patient's mean arterial pressure (MAP) drops too low, and pulmonary edema, i.e. represented as patient's pulmonary venous pressure (PVP) rising too high, in CVSim.

Our decision-making task involves selecting the "optimal" dynamic treatment strategy for individual patient from two alternatives, g_{c1} and g_{c2}, to minimize adverse clinical outcomes over the patient's remaining trajectory. After observing a patient for K time steps ($K = 34$ in CVSim), we use counterfactual simulations to predict the patient's remaining trajectory under each strategy. Based on these simulations, we choose the strategy that minimizes the proportion of time the patient experiences adverse outcomes, defined as either low mean arterial pressure (MAP) or high pulmonary venous pressure (PVP). Specifically, for each patient, we simulate 100 counterfactual trajectories under g_{c1} and g_{c2}, compute the proportion of time steps with adverse outcomes for each set of simulations, and select the strategy with the lower value of this objective function. This decision-making pro-



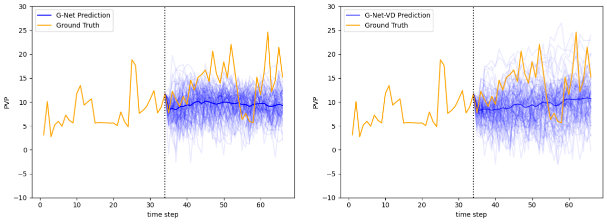
(a) G-Net Calibration (g_{c1}) (b) G-Transformer Calibration (g_{c1}) (c) RMSE (g_{c1})

(d) G-Net (g_{c2}) (e) G-Transformer (g_{c2}) (f) RMSE (g_{c2})

Figure 1: CVSim: Per time-step calibration for G-Net and G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Dashed horizontal line represents the perfect calibration, 0.90, for the given setting. The shaded area represents the 95% confidence intervals (CI). Subplots (c) and (f) show comparison of models in average RMSE with 95% CI for g_{c1} and g_{c2} respectively. Methods that properly model uncertainty generally outperform their base models (G-Net or G-Transformer) in calibration while maintaining comparable RMSE.

cess is conducted on an individualized basis for every patient.

To evaluate model performance, we compute the objective function, defined by target MAP and PVP thresholds, using the ground truth trajectories. The performance metric is the percentage of times adverse events occur across the testset (with 851 test trajectories over 32 time steps each), totaling 27,232 time steps. We assess this performance across all G-Net and G-Transformer-based models, using multiple pairs of MAP and PVP thresholds. Additionally, we compare these results to baseline strategies where g_{c1} or g_{c2} is applied universally for the entire patient cohort without individualized decision-making.

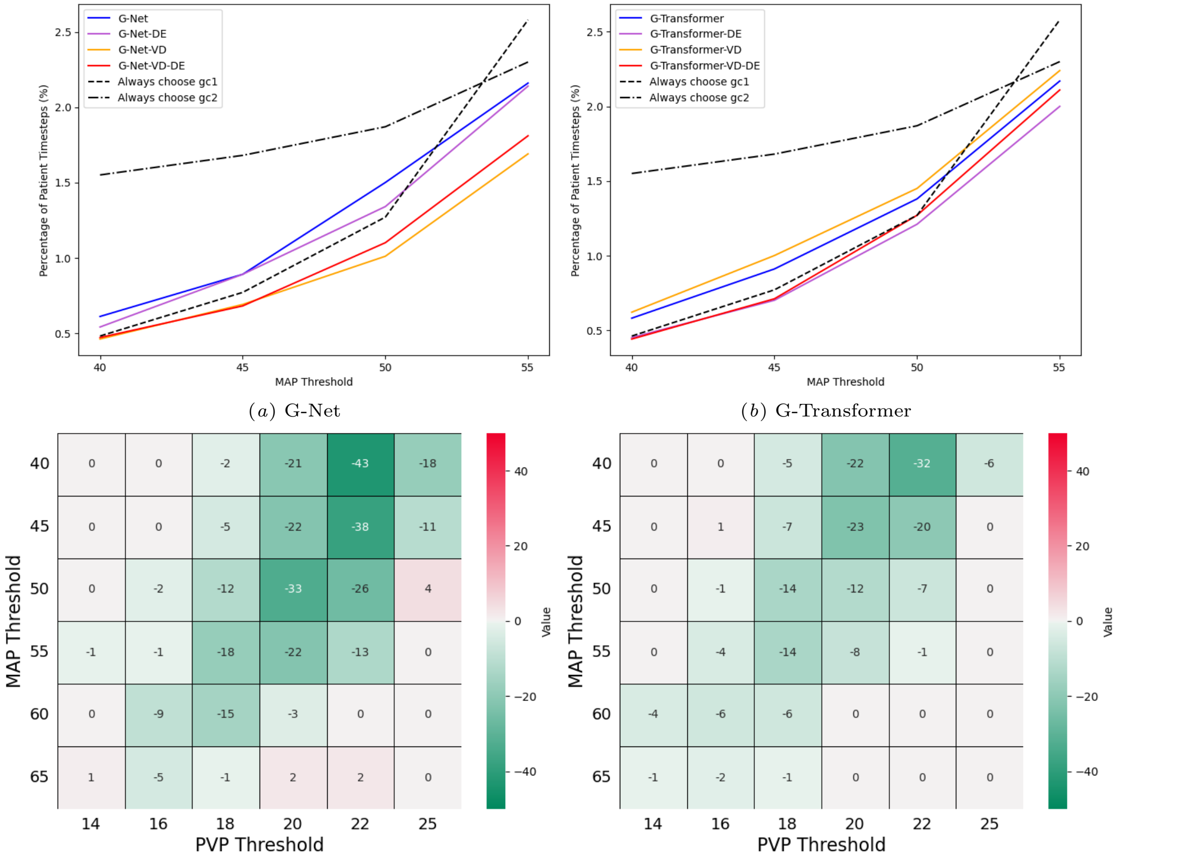


(a) G-Net (b) G-Net-VD

Figure 2: CVSim: Illustrative plots of simulated PVP trajectories for one CVSim subject using G-Net and G-Net-VD under counterfactual treatment strategy g_{c1}.

Case Study. To demonstrate the potential added value of uncertainty modeling in predicting adverse events such as pulmonary edema, Figure 2 presents a case study using the CVSim dataset. The patient's ground truth trajectory is shown in orange, alongside 100 simulated trajectories from the model in light blue and the average of these predicted trajectories in dark blue. Notably, while the base G-Net model failed to capture the PVP spike (i.e., PVP exceeding 20), simulations from G-Net with Dropout (G-Net-VD) highlighted the likelihood of this adverse event, providing a more informative prediction.

Results For Figure 3(a) we fixed the PVP threshold at 20 and analyzed results with MAP thresh-



(a) G-Net

(b) G-Transformer

(c) G-Net-VD/G-Net

(d) G-Transformer-DE/G-Transformer

Figure 3: CVSim: (a-b) Percent time steps with adverse events. Lower is better. (c-d) Percent difference in percentage of time-steps with adverse events between (c) G-Net-VD vs the base G-Net model, and (d) G-Transformer-DE vs G-Transformer. Greener is better.

olds of $40, 45, 50, 55$ mmHg, comparing performance against baselines that always choose g_{c1} or g_{c2} for each patient. For G-Net models, uncertainty quantification methods improved decision-making performance over the base G-Net model. The G-Net-DE performed comparably to the base G-Net, while G-Net-VD significantly outperformed it. Notably, G-Net failed to outperform always choosing g_{c1} for low MAP threshold, but G-Net-VD and G-Net-VD-DE surpassed g_{c1} and g_{c2} across all thresholds, demonstrating that uncertainty quantification enhances G-Net's clinical decision-making and viability for individual-level treatment. For G-Transformer models (Figure $3(b)$), the deep ensemble method and the combined VD-DE approach improved performance over the base G-Transformer model, and consistently outperformed both g_{c1} and g_{c2}, further indicating that uncertainty quantification enhances G-Transformer's effectiveness for individual-level predictions. Here, G-Transformer-DE is trained on MSE loss and simulated with empirical noise.

Figure $3(c)$ illustrates the percent difference between G-Net-VD and the baseline G-Net model in the proportion of time steps with adverse events defined by varying MAP and PVP thresholds. Let p represents the percentage of time steps with adverse events from G-Net-VD, and p_b represents the percentage from the base model G-Net. The percent difference is calculated as the relative change: $(p - p_b)/p_b$. Figure $3(d)$ shows a similar comparison for G-Transformer-DE versus the baseline G-Transformer model, calculated using the same formula.

Both G-Net and G-Transformer models demonstrated improved prediction of rare adverse events when incorporating uncertainty quantification. For example, with G-Net, rare events such as MAP falling below 40 or PVP exceeding 20 occurred in 0.61% of time steps (∼166 adverse events across

94 patients). Using G-Net-VD, these events decreased to 0.48% of time steps (∼131 adverse events across 77 patients), reflecting a 21.3% reduction. Similarly, G-Transformer-DE reduced percentages of adverse events by 22% compared to the base G-Transformer model. These results highlight the importance of proper uncertainty modeling in clinical decision-making for reducing critical adverse events.

5. Cancer Growth Results

For the Cancer Growth experiment, we simulate under the four counterfactual treatment strategies for four time-steps, and report the average RMSE and calibration in the Appendix. Our results show that the models with uncertainty quantification had similar or better RMSEs (Table 2 in Appendix) than the base model under every treatment strategy, and had improved calibration performance (see Table 3 in Appendix) in all strategies except in the case of no-treatment. **Clinical Decision-Making:** For the decision making task, we analyzed results with varying cancer volume thresholds from 20 to 80. In Figure 4, across every threshold and model type, the models with uncertainty quantification outperform the base models in the decision making task and had less percentage of times in having the adverse outcomes of tumor size exceeding a target threshold.

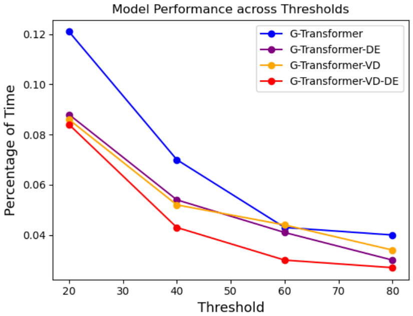


Figure 4: Cancer Growth with G-Transformer: percentage of times the tumor volume threshold is exceeded at the final time-step. Lower is better.

6. MIMIC Results

We evaluate the impact of adding (model) uncertainty quantification to the G-Transformer model using real-world ICU data from the MIMIC-IV database. The dataset consists of 8,920 patients meeting sepsis-3 criteria, and we use a 80-10-10 train/validation/test splits. G-Transformer-DE and G-Transformer-VD-DE both use 20 models.

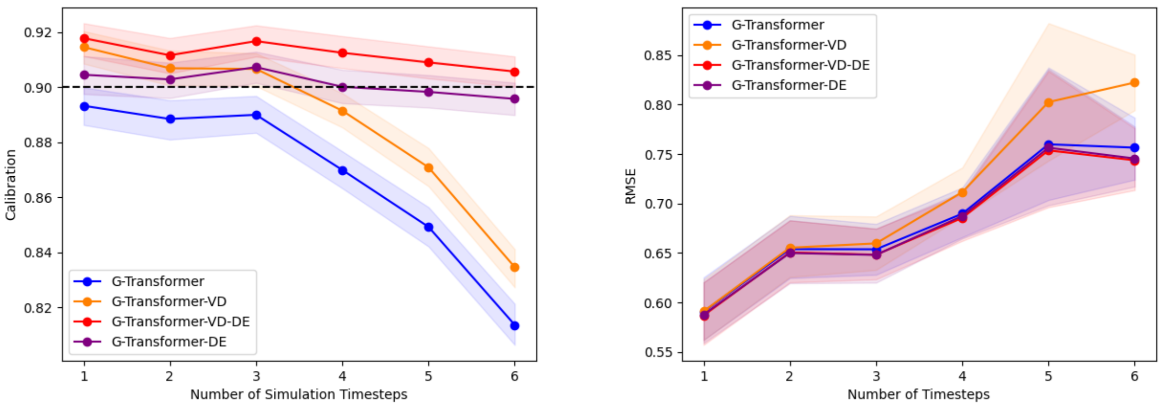
We quantitatively evaluate G-Transformer-based model's performance in predicting patient trajectories under observational treatment regime learned from the ICU data. For each test set patient, we simulate forward patients' trajectories for the next 6 hours conditioned on the observations from the first hour in the ICU. We evaluate by comparing the predicted time-varying outcomes (averaged across M Monte-Carlo simulations) with the actual observed trajectories from individual patients. See Appendix for more details, including a list of variables included for calculating calibration.

Calibration. Figure $5(a)$ compares calibration of various uncertainty quantification techniques in G-Transformer framework. Based on 1000 bootstrapped samples from test dataset, the 95% confidence interval was based on 2.5 percentile and 97.5 percentile of 1000 calibrations for all continuous variables over time for each time step. Both methods of uncertainty quantification improved the calibration performance uniformly across all time-steps over the baseline G-Transformer. Notably, the calibration result from G-Transformer-DE approximately meets the ideal of 0.90 across time-steps without decay.

RMSE. Figure $5(b)$ compares RMSEs over time of various uncertainty quantification techniques in G-Transformer framework. Based on 1000 bootstrapped samples from test dataset, the 95% confidence interval was based on 2.5 percentile and 97.5 percentile of 1000 RMSEs over time for each time step. We observe that the RMSEs of the models are at similar levels. Adding uncertainty quantification from deep ensemble slightly improves RMSE from baseline G-Transformer. See the Appendix for additional results.

7. Discussion and Conclusion

In this work, we propose approximate Bayesian methods as a practical and principled approach to explicitly account for uncertainty due to variability in model parameters in generating counterfactual predictive distributions. Previous deep learning approaches to g-computation, including G-Net and G-Transformer, estimate uncertainty in counterfactual predictions (based on distributions from Monte Carlo simulations) conditioned on model parameters. Our results demonstrate that incorporating uncertainty around model parameters through the proposed approach significantly enhances the calibration perfor-



(a) Calibration

(b) RMSE

Figure 5: MIMIC with G-Transformer: Per time-step calibration and RMSE on MIMIC data, conditioned on the first time step using 100 MC simulations per patient. The shaded area represents the 95% confidence interval. G-Transformer-DE (in purple) outperforms the base model G-Transformer (in blue) and other uncertainty quantification approaches in calibration while maintaining RMSE performance comparable to the base model.

mance of the base models while preserving predictive accuracy.

Deep ensembles (DE) are generally more effective than variational dropout (VD) in capturing uncertainty, particularly when the true posterior is complex and multimodal. By training multiple independent models, DEs provide a better approximation to the Bayesian posterior (Wilson and Izmailov, 2020; Lakshminarayanan et al., 2017). While VD is computationally efficient, it is limited in capturing the full complexity of the posterior. Empirically, our results demonstrate that DE and the combined VD-DE generally outperformed VD alone when applied to G-Transformer, especially for more complex settings such as CVSim and MIMIC. In CVSim, VD improved calibration of G-Net, but was less effective with G-Transformer overall. In MIMIC experiments, G-Transformer-DE achieved the best calibration performance, significantly outperforming the base G-Transformer and G-Transformer-VD models, while maintaining RMSE on par with the base model.

We should note that simply incorporating uncertainty quantification methods may not improve calibration if the underlying model exhibits high bias (i.e., high RMSE). For example, a model with poor predictive accuracy may produce biased predictions, and adding uncertainty quantification techniques cannot necessarily correct this underlying issue. If the model is fundamentally unable to capture the data's underlying structure, the uncertainty estimates may not be meaningful, and the model could still fail to provide well-calibrated probabilities.

Our study highlights the importance of modeling counterfactual predictive distributions to navigate the inherent risk-reward trade-offs in clinical decision-making. By incorporating practical and principled approximate-Bayesian methods into deep learning-based g-computation, we addressed the challenges in estimating conditional counterfactual predictive distributions under dynamic treatment strategies. Our results confirm that incorporating uncertainty to the model parameters significantly improves their ability to represent the distribution of potential outcomes, including critical tail events, compared to prior methods such as G-Net and G-Transformer, which do not account for sampling variability from model uncertainty.

In scenarios like fluid administration for septic patients, where treatment decision making aims at reducing probabilities of adverse outcomes (e.g., fluid overload or hypotension), our approach demonstrated improved performance in modeling the tails of joint outcome distributions. This improvement underscores the necessity of uncertainty-aware models for clinical contexts where the consequences of underestimating uncertainty can lead to suboptimal or even harmful decisions. Our findings thus highlight the utility of combining deep learning with principled uncertainty quantification for informed sequential treatment decision making in dynamic treatment settings.

8. Acknowledgments

The authors are grateful for Professor Roger Mark for his insightful comments. This research was in part funded by the MIT-IBM Watson AI Lab. Lehman was in part funded by NIH grant R01EB030362.

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 (ICLR)*, 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.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, Jun 2017. doi: 10.3386/w23564.

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

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, 2016b.

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*, 2017.

AL Goldberger, LAN Amaral, L Glass, JM Hausdorff, PCh Ivanov, RG Mark, JE Mietus, GB Moody, C-K Peng, and HE Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, June 2000.

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.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

AEW Johnson, L Shen, A Gayles, A Shammout, S Horng, TJ Pollard, B Moody, B Gow, LH Lehman, L Celi, and RG Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1), 2023.

Alexander P Keil, Eric J Daza, Stephanie M Engel, Jessie P Buckley, and Jessie K Edwards. A Bayesian approach to the g-formula. *Statistical Methods in Medical Research*, 27(10):3183–3204, March 2017. ISSN 1477-0334. doi: 10.1177/0962280217694665.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Lihua Lei and Emmanuel J Candès. Conformal inference of counterfactuals and individual treatment effects. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(5):911–938, 2021.

Rui Li, Stephanie Hu, Mingyu Lu, Yuria Utusumi, Prithwish Chakraborty, Daby Sow, Piyush Madan, Mohamed Ghalwash, Zach Shahn, and Li-wei H Lehman. G-Net: a Recurrent Network Approach to G-Computation for Counterfactual Prediction Under a Dynamic Treatment Regime. *Proceedings of Machine Learning for Health*, 2021.

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.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal Transformer for Estimating Counterfactual Outcomes. In *Proceedings of the 39 th International Conference on Machine Learning*, 2022.

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.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

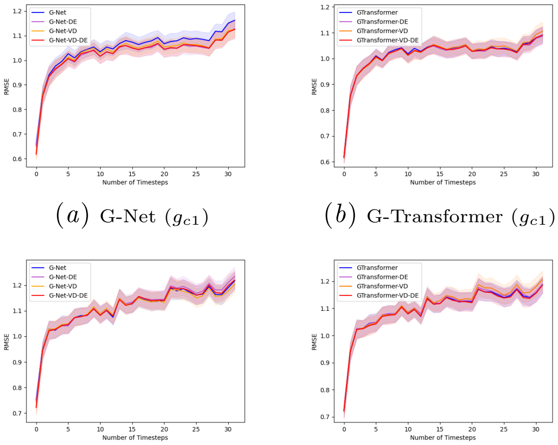
Hong Xiong, Feng Wu, Leon Deng, Megan Su, and Li-wei H Lehman. G-Transformer: Counterfactual Outcome Prediction under Dynamic and Time-Varying Treatment Regimes. In *Proceedings of Machine Learning for Healthcare*, 2024.

Jinsung Yoon, James Jordan, and Mihaela Van der Schaar. GANITE: Estimation of Individualized Treatment Effects Using Generative Adversarial Nets. In *ICLR.*, 2018.

Appendix A. CVSim Experiment Settings and Results

A.1. CVSim RMSE Over Time

Figure 6 shows the per-timestep RMSE over time for CVSim experiments. Approaches with uncertainty modeling achieve RMSE values comparable to their respective base G-Net or G-Transformer models under both counterfactual regimes g_{c1} and g_{c2}, indicating that incorporating uncertainty quantification using the proposed methods does not negatively impact RMSE performance. Based on 1000 bootstrapped samples from test dataset, the 95% confidence interval was based on 2.5 percentile and 97.5 percentile of 1000 RMSEs over time for each time step.



(a) G-Net (g_{c1}) (b) G-Transformer (g_{c1})

(c) G-Net (g_{c2}) (d) G-Transformer (g_{c2})

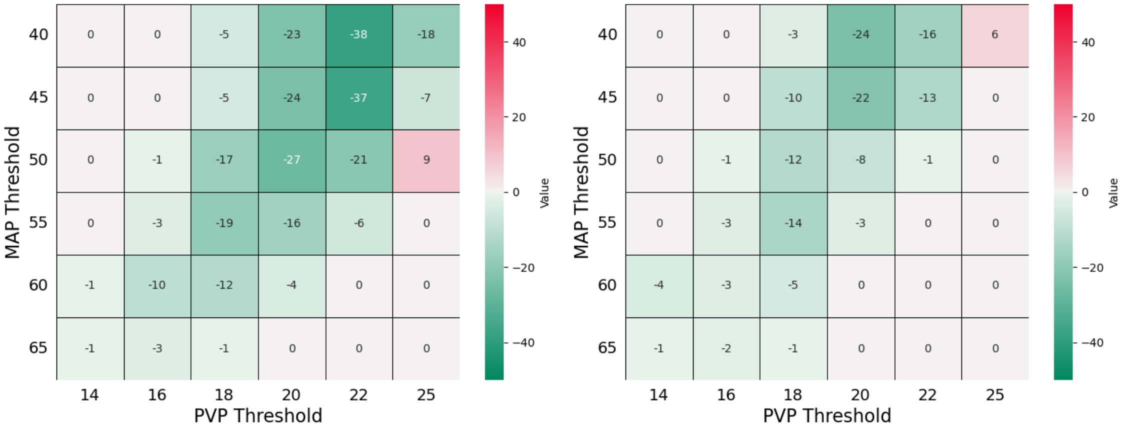
Figure 6: CVSim: Per time-step RMSE for G-Net and G-Transformer under counterfactual regimes g_{c1} and g_{c2}. The shaded area represents the 95% confidence interval.

A.2. CVSim Decision Making Additional Results

Figure 7 shows the percent difference in proportion of time-steps performance between VD-DE and the base G-Net and G-Transformer model.

A.3. Counterfactual Regimes g_{c1} vs g_{c2}

Under g_{c1}, treatment (fluids or vasopressors) is administered if and only if mean arterial blood pressure (MAP) $<$ 65 mmHg; fluids are administered as treatment if and only if patient does not have pulmonary edema. Counterfactual regime g_{c2} follows the same rules as g_{c1}, except that treatments are administered when MAP $<$ 75 mmHg. For more details, including the dosage administered, please see Li et al. (2021).



(a) G-Net-VD-DE/G-Net

(b) G-Transformer-VD-DE/G-Transformer

Figure 7: CVSim: (a) Percent difference in proportion of time-steps performance between G-Net-VD-DE and the base G-Net model. Lower (greener) is better. (b)Percent difference in proportion of time-steps performance between G-Transformer-VD-DE and the base G-Transformer model. Lower (greener) is better. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

A.4. CVSim List of Covariates

Table 1 lists output covariates from CVSim used in our experiments. Calibration calculation included all (continuous-valued) selected output covariates, marked by *, except Pulmonary Edema which is a binary indicator variable.

Appendix B. Effect of Ensemble Size on Calibration

We also compared calibration results for smaller ensemble sizes to determine how much a larger ensemble may benefit model performance. For the G-Net Ensemble and G-Transformer Ensemble models, we plotted per time-step calibration on g_{c1} and g_{c2} while varying ensemble sizes. We chose $\{1, 2, 5, 10, 20\}$ as the sizes to examine.

We found that increasing ensemble size improved calibration across all models and counterfactual regimes. For most models and counterfactual regime pairings, calibration improved only marginally between the 10-model and 20-model ensembles. For G-Transformer Ensemble under g_{c2}, however, the 20-model ensemble displays significant improvement over the 10-model ensemble during the later time-steps.

Appendix C. Cancer Growth Experiments

As in Li et al. (2021), we divide by the maximum tumor volume ($1150cm^3$) for percentage RMSEs.

Our results in Table 2 show that the models with uncertainty quantification had better RMSEs than the base model under every treatment strategy, with G-Transformer-DE having the best RMSE under no treatment and G-Transformer-VD having the best under radiotherapy, chemotherapy, and chem-rad. For calibration (see Table 3), G-Transformer-DE and G-Transformer-VD are better calibrated than the baseline G-Transformer under chemotherapy but are worse under no treatment. The models with uncertainty quantification have noisy estimates for the no treatment regime, with calibrations close to 1. In the simpler Cancer-Growth experiment, where only one time-varying covariate is present, the advantage of DE over VD is less pronounced as in the case of CVSim and MIMIC experiments.

Appendix D. MIMIC Results and Calibration Settings

Table 4 shows individual-level RMSEs (averaged across 6 time steps) with 95% confidence interval for G-Transformer-based models on MIMIC data, conditioned on the first time step and using 100 Monte Carlo trajectories per patient under the observational regime. Based on 1000 bootstrapped samples from

Table 1: CVSim output covariates. 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

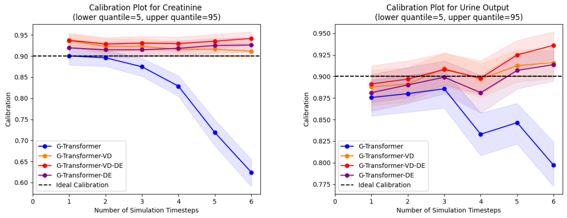
	GT	GT-DE	GT-VD	GT-VD-DE
No Treat	0.82	**0.70**	0.74	0.88
Radio	3.61	3.56	**3.36**	3.66
Chemo	1.87	1.90	1.56	**1.49**
RadioChemo	2.24	2.21	**2.17**	2.27

Table 2: Cancer Growth with G-Transformer: Percent RMSEs (averaged over 4 simulation time steps). Best performing models in bold. GT = G-Transformer.

test dataset, the 95% confidence interval was based on 2.5 percentile and 97.5 percentile of 1000 timestep-averaged RMSEs.

	GT	GT-DE	GT-VD	GT-VD-DE
No Treat	**0.87**	1.00	0.98	1.00
Radio	0.60	0.76	0.74	**0.83**
Chemo	0.72	**0.88**	**0.92**	0.96
RadioChemo	0.69	0.83	0.78	**0.86**

Table 3: Cancer Growth with G-Transformer: Calibration Results. Best performance (in absolute difference from 0.90) in bold. GT = G-Transformer.



(a) Creatinine

(b) Urine Output

Figure 8: MIMIC with G-Transformer: Per time-step calibration of selected variables for G-Transformer-based models on MIMIC data, conditioned on the first time step and using 100 Monte Carlo trajectories per patient under the observational regime. The shaded area represents the 95% confidence interval.

Model	RMSE (95%CI)
G-Transformer	0.687 (0.663, 0.712)
G-Transformer-VD	0.712 (0.689, 0.736)
G-Transformer-VD-DE	**0.681 (0.657, 0.705)**
G-Transformer-DE	0.682 (0.659, 0.706)

Table 4: MIMIC with G-Transformer: Individual-level RMSEs (averaged across 6 time steps) with 95% confidence interval (CI) for G-Transformer-based models on MIMIC data, conditioned on the first time step and using 100 Monte Carlo trajectories per patient under the observational regime.

In Table 5, we show all the continuous variables that went into calculating calibration for the MIMIC-IV experiments.

Figure 8 shows per time-step calibration of selected variables with 95% confidence interval for G-Transformer-based models on MIMIC data, conditioned on the first time step and using 100 Monte Carlo trajectories per patient under the observational regime. Based on 1000 bootstrapped samples from test dataset, the 95% confidence interval was based on 2.5 percentile and 97.5 percentile of 1000 calibrations for the selected variables over time for each time step. We see that G-Transformer-DE in general per-

Table 5: MIMIC variables used in the calibration calculation.

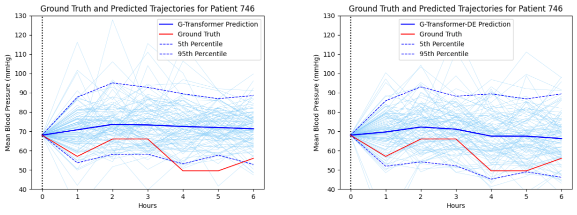
Variable Name	Units
Heart Rate	beats/min
Diastolic Blood Pressure	mmHg
Systolic Blood Pressure	mmHg
Mean Blood Pressure	mmHg
Minimum Mean Blood Pressure	mmHg
Temperature	degree C
Platelet	counts/10^9L
Hemoglobin	g/dL
Calcium	mg/dL
BUN	mmol/L
Creatinine	mg/dL
Bicarbonate	mmol/L
Lactate	mmol/L
pO2	mmHg
sO2	%
spO2	%
pCO2	mmHg
Total CO2	mEq/L
pH	Numerical[1,14]
Base excess	mmol/L
Weight	kgs
Respiratory Rate	breaths/min
Fluid Volume	mL
Urine Output	mL
Vasopressor Amount	N/A
Bolus Volume	mL

form the best in term of calibration across selected variables.

D.1. Example MIMIC Case Studies

Case Study. Figure 9 demonstrates the calibration check for selected variables on an example MIMIC test set patients. We observe that the ground truth falls within the target percentile range from the G-Transformer-DE and G-Transformer-VD-DE more frequently compared to the baseline G-Transformer. This demonstrates the improvement in calibration achieved through adding uncertainty quantification and highlights the benefit from the perspective of an individual patient.

Appendix E. SWAG Methods and Results in CVSim



(a) G-Transformer (b) G-Transformer-DE

Figure 9: MIMIC with G-Transformer: Illustration of calibration check on an example MIMIC test set patient. Subplots (a) and (b) display simulations for Mean Blood Pressure. The values are simulated from the baseline G-Transformer and G-Transformer-DE models, conditioned on the first time step and using 100 Monte Carlo trajectories under the observational regime.

Stochastic Weight Averaging-Gaussian (SWAG) (Maddox et al., 2019; Izmailov et al., 2018) is an approximate Bayesian inference approach which uses SGD iterates to construct a Gaussian neural network parameter posterior. SWAG has been shown to provide improved generalization in deep learning. Starting from a pretrained checkpoint, we denote the weights of the network obtained after iteration i of SWA traning θ_i, the SWA solution after T epochs is given by $\theta_{\text{SWA}} = \frac{1}{T}\sum_{i=1}^{T}\theta_i$. In the training process, we maintain a running average of the second uncentered moment for each weight. After training, we compute the covariance by following identity: $\bar{\theta}^2 = \frac{1}{T}\sum_{i=1}^{T}\theta_i^2, \quad \Sigma_{\text{diag}} = \text{diag}(\bar{\theta}^2 - \bar{\theta}^2_{\text{SWA}})$, here θ_i^2 and θ^2_{SWA} are the results of element-wise squaring of θ_i and θ_{SWA} respectively. The approximate posterior distribution is $\mathcal{N}(\theta_{\text{SWA}}, \Sigma_{\text{Diag}})$. During the simulation process, we sample model parameters from this posterior distribution and use the Bayesian model average of the results as the model output.

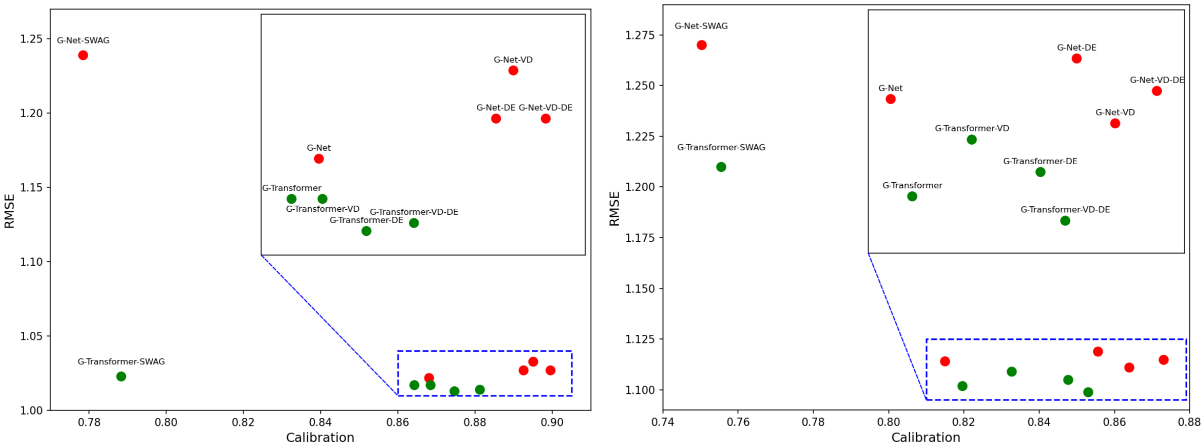
Figure 10 shows the calibration and RMSE performance from SWAG relative to the base G-Net and G-Transformer models other uncertainty quantificaiton methods. We observe that, in general, SWAG underperformed both in terms of calibration and RMSE relative to other uncertainty quantification techniques.

Appendix F. Background

F.1. G-Net and G-Transformer

G-Net (Li et al., 2021) and G-Transformer (Xiong et al., 2024) are neural network based implementations of g-computation, where G-Net uses RNNs and G-Transformer uses transformer encoders. They have been shown to predict patient outcomes more accurately than other counterfactual prediction methods and g-computation implementations. They are also the models that we build upon in this work, and we will use them as baselines in analyzing results.

This section describes the theory and implementation behind the base G-Net and G-Transformer mod-



(a) RMSE and calibration under g_{c1}

(b) RMSE and calibration under g_{c2}

Figure 10: CVSim with SWAG results: comparison of models in RMSE vs Calibration.

els that do not incorporate model uncertainty into their predictive distributions.

F.1.1. G-Net and G-Transformer Framework

G-Net and G-Transformer are used to implement g-computation by approximating the conditional distributions of covariates (including outcomes of interest) given history, i.e. $p(L_t|L_{1:t-1}, A_{1:t-1})$. However, L_t in many cases is high-dimensional and complex, so we partition the covariates represented by L_t into G groups or "boxes", $\{L_t^0, L_t^1, \ldots, L_t^{G-1}\}$, and train multiple regressors or classifiers to learn their distributions separately. This technique is represented mathematically by the chain rule of probability, where $p(L_t|L_{1:t-1}, A_{1:t-1})$ is rewritten as

$$p(L_t^0|L_{1:t-1}, A_{1:t-1}) \times p(L_t^1|L_t^0, L_{1:t-1}, A_{1:t-1}) \times \ldots \\ \times p(L_t^{G-1}|L_t^0, L_t^1, \ldots, L_t^{G-2}, L_{1:t-1}, A_{1:t-1}). \tag{2}$$

In this work, we use $G = 2$ and set the groups to be the categorical and continuous covariates, respectively. We call this a *two-box* architecture, with one classifier for the categorical covariates and one regressor for the continuous covariates.

To be concrete, we describe in detail one time-step's worth of prediction when $G = 2$. We set L_t^0 and L_t^1 to be the categorical and the continuous covariates at time t, respectively. Let R^0 and R^1 be a classifier and a regressor that approximate $p(L_t^0|L_{1:t-1}, A_{1:t-1})$ and $p(L_t^1|L_t^0, L_{1:t-1}, A_{1:t-1})$. To simulate for time-step $t + 1$, R^0 takes $(L_{1:t}^0, L_{1:t}^1, A_{1:t})$ as input and outputs a predicted $\hat{L}_{t+1}^0$. This predicted $\hat{L}_{t+1}^0$ is passed to R^1, which takes $(L_{1:t}^0, \hat{L}_{t+1}^0, L_{1:t}^1, A_{1:t})$ and outputs a predicted $\hat{L}_{t+1}^1$. The predicted covariates ($\hat{L}_{t+1}^0$ and $\hat{L}_{t+1}^1$) are concatenated to get $\hat{L}_{t+1}$. Lastly, $\hat{A}_{t+1}$ is decided by the counterfactual strategy as a function of $(L_{1:t}^0, \hat{L}_{t+1}^0, L_{1:t}^1, \hat{L}_{t+1}^1, A_{1:t})$. This process repeats for further time-steps, with R^0 taking $(L_{1:t}^0, \hat{L}_{t+1}^0, L_{1:t}^1, \hat{L}_{t+1}^1, A_{1:t}, \hat{A}_{t+1})$ as input to predict $\hat{L}_{t+2}^0$.

This framework is flexible without making assumptions on the model architectures of R^0 and R^1. Any model that conforms to the desired input and output format can be employed. G-computation is generally performed with R^0 and R^1 as generalized linear models. G-Net uses recurrent neural networks, and G-Transformer uses transformer encoders.

F.1.2. Training Procedure

For the two-box model, we train with the outputs from the first box of categorical covariates fed as input to the second box of continuous covariates. The categorical box is trained with cross-entropy loss. For the base G-Net and G-Transformer models, the continuous box is trained with mean squared error. When these models are implemented with deep ensemble, we may use a different loss function (see Section 3.1).

For model training, we also use *teacher forcing*. Instead of using the predicted $\hat{L}_{t+1}$ to make predictions on time-step t+2, we use the observed L_{t+1}. This is equivalent to training the model on one time-step ahead prediction. We also use teacher forcing for the individual "boxes". Instead of passing a predicted $\hat{L}^0_{t+1}$ to R_1, we pass the observed L^0_{t+1}. Since we evaluate performance on multiple time-step ahead prediction, teacher-forcing notably optimizes a different loss function than our true objective. However, we find that teacher-forcing performs better than our alternative, *student forcing*.

F.1.3. Simulation Procedure

After we've trained our G-Net and G-Transformer models, we simulate. Our trained models, on their own, only output conditional expectations — that is, our models output estimates of $\mathbb{E}[L_t | L_{1:t-1}, A_{1:t-1}]$. However, for continuous covariates, we want the distribution of covariates $p(L_t | L_{1:t-1}, A_{1:t-1})$. We obtain an estimate of these distributions by sampling from every box as follows:

$$\begin{aligned} L_t^g | L_t^0, \ldots, L_t^{g-1}, L_{1:t-1}, A_{1:t-1} \sim \\ \hat{\mathbb{E}}[L_t^0, \ldots, L_t^{g-1}, L_t | L_{1:t-1}, A_{1:t-1}] \quad + \epsilon_t^g \end{aligned} \tag{3}$$

where g denotes the gth box and ϵ_t^g is drawn randomly from a set of residuals $L_t^g - \hat{L}_t^g$ calculated empirically from a validation dataset. This way, we can non-parametrically simulate from an approximate conditional distribution of covariates at each time-step.

Categorical covariates are treated more simply. Again, our models output estimates of $\mathbb{E}[L_t | L_{1:t-1}, A_{1:t-1}]$, which we enforce to be in the range $[0, 1]$ with a sigmoid activation. To simulate, we draw from a Bernoulli distribution with parameter $\hat{\mathbb{E}}[L_t | L_{1:t-1}, A_{1:t-1}]$.

We repeat this simulation process for a number of time-steps dependent on the specifications of the dataset. For example, the CVSim dataset involves 32 time-steps in which the patient undergoes a counterfactual treatment strategy.

Appendix G. Methods Details

G.1. Computing Residuals with Variational Dropout

Implementing variational dropout for G-Net and G-Transformer isn't as straightforward as turning on dropout masks for simulations. We must also make modifications to how residuals are computed.

Recall that for G-Net and G-Transformer, the residuals ϵ_t^g were computed empirically from the $L_t^g - \hat{L}_t^g$ values on an validation dataset, where $\hat{L}_t^g$ is treated as an estimate of the conditional expectation $\mathbb{E}[L_t^g | L_t^0, \ldots, L_t^{g-1}, L_{1:t-1}, A_{1:t-1}]$. For G-Net with Dropout and G-Transformer with Dropout, however, forward passes during validation time no longer can be treated as conditional expectations but rather as samples from an approximate distribution. To correct this discrepancy, we conduct 100 forward passes of our model to obtain 100 $L_t^g - \hat{L}_t^g$ values per patient in our validation dataset. These 100 forward passes represent uncertainty about the model parameters. The number of forward passes only dictates the granularity of our estimate of the residual distribution, and was chosen arbitrarily. During simulation time, we simulate $L_t^g \sim \hat{L}_t^g + \epsilon_t^g$ as per usual. These residuals represent uncertainty about the model output. This way, we capture both the uncertainty in model parameters and the uncertainty in patient covariates.

G.2. G-Net with Variational Dropout (G-Net-VD)

G-Transformer with Dropout can be implemented simply as the base G-Transformer model along with the modifications discussed earlier. However, G-Net with Dropout is implemented with a slightly different model architecture as well. The traditional LSTM architecture involves using independently sampled dropout masks on the inputs and outputs of every LSTM layer. G-Net with Dropout, on the other hand, uses an LSTM architecture recommended by Gal and Ghahramani (2016a) for improved regularization. In their model, dropout masks are placed not only between layers but also between adjacent time-steps. Connections between the same pair of layers share dropout masks, and connections within the same layer also share dropout masks.

G.3. Deep Ensemble Negative Log-Likelihood Loss

For deep ensemble models, we use negative log-likelihood as the loss function for training. Since we use multiple different modeling assumptions for the deep ensemble model, the implementation differs slightly for each.

For the Gaussian with identity covariance matrix, we note that optimizing the NLL of such a Gaussian is equivalent to optimizing the MSE of the mean estimates. Since the G-Net and G-Transformer models already use MSE to optimize the continuous covariates, we use their implementations. Nothing changes except that we train 20 models for an ensemble instead of just one.

For the Gaussian with diagonal covariance matrix, we need to model variances. To do so, we add a second linear layer alongside the first. We treat the first linear layer as the predictor for the means and we treat the second as the predictor for the variances. The second layer comes with a softplus activation function that ensures that positive variances are predicted. From there, the predicted means and variances are fed into the negative log-likelihood calculation and are summed across all covariates for the loss on our continuous covariates.

For the Gaussian with full covariance matrix, we need to model variances and covariances. Instead of modeling the entire matrix Σ, we use the Cholesky decomposition $\Sigma = LL^T$ and model L, a lower triangular matrix with positive diagonal entries. As in the diagonal Gaussian case, we add a second linear layer alongside the first, which outputs all entries of L. We use the softplus activation on the diagonal entries of L to ensure positivity.

Deep Ensemble Alternative Parametric Assumptions Note that the deep ensemble method as presented above makes the modeling assumption that the conditional distribution of covariates at each time-step is a diagonal Gaussian; that is, the distribution's covariance matrix is diagonal. By training an ensemble of these models, we're approximating the data distribution as a mixture of diagonal Gaussians. By modifying our modeling assumptions, we thereby modify the way we fit the data.

In addition to modeling the conditional distributions as diagonal Gaussians, we also modeled them as Gaussians with identity covariance matrices and as Gaussians with all covariance terms. The loss function changes accordingly. For example, modeling Gaussians with identity covariances with negative log-likelihood loss is equivalent to training with mean squared error as the loss function. Details about their implementations are in G.3.

Simulation Methods After an ensemble is trained, we can also simulate from it in different ways. Most simply, we can ignore the variance estimates and simulate by adding empirical noise terms to the mean estimates as in the base G-Net and G-Transformer models in 3. This is our non-parametric approach to estimating the conditional distribution of covariates. We can also take the parametric route and, at each time-step, sample from the Gaussians parametrized by the mean and (co)variance estimates without using any empirical noise terms.

Both methods are explored and evaluated in our experiments. For G-Net, we found that training for a diagonal Gaussian and simulating with empirical noise worked best. For G-Transformer, we found that training on MSE and simulating with empirical noise worked best. Results from other alternative methods are in the appendix.

G.3.1. Variational Dropout with Ensembling

We explored combining variational dropout and deep ensemble. This was accomplished by independently training $M = 20$ variational dropout models on MSE using random initializations and random shufflings of the training data. Each model independently simulates using empirical noise, and the simulations are aggregated for a full set of simulations from the ensemble. When $M = 20$, each individual variational dropout model produces 5 simulations for 100 simulations in total.

Appendix H. Experiment Settings

H.1. Evaluation of Uncertainty Predictions

In this section, we describe our methods for evaluating how well our models quantify uncertainty. We evaluate the base, variational dropout, and deep ensemble versions of G-Net and G-Transformer on the CVSim and Cancer Growth datasets. For the deep ensemble models, we examine using the different modeling assumptions as described in Section 3.1.

We explain in detail about the implementations of the G-Net and G-Transformer based models. Training and simulation of G-Net and G-Transformer models revolve around the training and simulation of the (two) individual boxes as described in Appendix F.1.

G-Net Based Models For both the base G-Net and the G-Net Ensemble models, the individual boxes are implemented with multilayer LSTM models that output one embedding per time-step with dimensionality controlled by a hyperparameter. We also have a linear layer that brings transforms the embedding to the desired output dimension. G-Net with Dropout is implemented nearly identically, but with the dropout masks implemented as described in G.2.

At each time-step, the base G-Net model takes in the patient history $(L_{1:t}, A_{1:t})$ and conditions on it to produce an estimate of the patient's covariates for the next time-step. However, with G-Net based models, the patient histories are actually not directly input to the model. Instead, to predict for time-step $t+1$, the LSTM (in the first box) predicts using (h_{t-1}, L_t, A_t), where h_t is the LSTM's hidden state from time-step $t-1$ and acts as a learned representation for $(L_{1:t-1}, A_{1:t-1})$.

G-Transformer Based Models For all the G-Transformer based models, the individual boxes are implemented with transformer encoders. To predict for time-step $t+1$, the first box takes the patient history $(L_{1:t}, A_{1:t})$ and feeds it through a positional embedding layer and a transformer encoder to produce a sequence of embeddings. The last embedding in this sequence is fed through a linear layer to produce our prediction. Note that the G-Transformer differs from G-Net in that it conditions on the patient history in its raw form rather than in a learned representation.

Training We use binary cross-entropy loss for the categorical covariates and either mean squared-error or negative log-likelihood loss for the continuous covariates, depending on the model. We use Adam as our optimizer and CosineAnnealingWarmRestarts as our learning rate scheduler. To train the ensembles, we train 20 individual base models with different random seeds that control parameter initializations and the order in which training data is fed into the model. See Appendix for details in hyperparameter settings in grid search.

Simulation For the CVSim dataset, we simulate under two counterfactual strategies g_{c1} and g_{c2}; for the Cancer Growth dataset, we simulate under no treatment, radiotherapy, chemotherapy, and both. Simulation is as described in F.1. For a given counterfactual treatment strategy, we simulate 100 trajectories per patient. For direct comparison, the deep ensemble approach produces an overall of 100 trajectories per patient across multiple ensemble models, e.g. for deep ensemble with 20 models, each produces 5 trajectories per patient to yield an overall of 100 MC simulations per patient.

Evaluation We perform model evaluation with two metrics: calibration and individual-level RMSE. For each of the models, we use the 100 simulated trajectories per patient to compute per-time-step calibrations and individual-level RMSEs as described in 3.2.

RMSE We use root mean-squared error (RMSE) to evaluate the accuracy in counterfactual prediction. For each individual patient, trajectories of 100 Monte Carlo simulations are averaged to represent the expected trajectory and compared to the ground-truth of that patient to compute individual-level RMSE. With N_c as the number of patients, $\{m, m+1, \ldots, t\}$ as the predicted time-steps, d as the number of covariates, f as our model, and $\hat{L}^{CF}(f)$ as the average of the predicted trajectories of a patient as predicted by f, we can express the individual-level RMSE as

$$\sqrt{\frac{1}{N_c(K-m)d}\sum_{i=1}^{N_c}\sum_{t=m}^{K}\sum_{h=1}^{d}(L_{ti}^{h,CF} - \hat{L}_{ti}^{h,CF}(f))^2}. \quad (4)$$

Ensemble Size Comparison We also compared calibration results for smaller ensemble sizes to determine how much a larger ensemble may benefit model performance. For the G-Net Ensemble and G-Transformer Ensemble models, we plotted per time-step calibration on g_{c1} and g_{c2} while varying ensemble sizes. We chose $\{1, 2, 5, 10, 20\}$ as the sizes to examine.

Impact on Clinical Decision Making While well-calibrated models are preferable to badly-calibrated ones, calibration doesn't exactly reflect the usefulness of a model in clinical settings. Doctors care greatly about preventing worst-case outcomes in patients, and a model's ability to predict such outcomes aren't perfectly reflected in the calibration metric. This mismatch in objectives motivates this section's experiment, which emulates how a doctor might use uncertainty predictions in a clinical setting. For the CVSim dataset, we focus on the problem of administering fluid to a patient. If a patient's blood pressure (mean arterial pressure, or MAP) is low, a doctor might want to administer fluids to increase blood pressure and blood perfusion to the organs. However, too much fluid leads to a higher risk of developing pulmonary edema, which is indicated by the pulmonary venous pressure (PVP) covariate. This balancing act in administering just the right amount of fluid may be difficult for a doctor and can be made easier with uncertainty prediction. For the Cancer Growth, we focus on the problem of minimizing the chance of cancerous tumors growing too large. There is no trade-off between treatment effects similar to the one in the CVSim dataset. Rather, we seek to reduce occurrences in which patient tumors grow to dangerous sizes. This goal of predicting for and preventing uncommon but significant outcomes is also made easier with uncertainty prediction.

Appendix I. Hyperparameter Settings

In this section, we detail the hyperparameter settings for CVSim (Table 6), cancer growth (Table 7) and the MIMIC experiments (Table 8).

The baseline G-Net and G-Transformer model are established using the hyperparameter configuration that achieves the best validation performance after a grid search. For the Ensemble approach, no additional tuning is performed; it directly adopts the hyperparameters optimized for the baseline models.

Table 6: Hyperparameter search space in CVSim experiments. HD = Hidden Dimension.

	Hyperparameters	**Range**
G-Net	Number of Layers	**2**, 3
	HD (Categorical)	64, **128**
	HD (Continuous)	64, **128**
	Batch Size	**16**
	Learning Rate	**0.0001**
G-Net-DE (NLL, diagonal)	Batch Size	16, **32**
	Number of Layers	**1**, 2, 4
	HD (Continuous)	32, 64, **128**
	Learning Rate	**0.01**, 0.001
G-Net-VD with Dropout	Batch Size	16, **32**
	Number of Layers	1, **2**, 4
	HD (Continuous)	32, **64**, 128
	Dropout Rate	0.05, **0.1**, 0.2
	Learning Rate	**0.01**, 0.001
G-Transformer	Number of Layers	**3**
	Hidden Dimension	32, 64, **128**
	Batch Size	**16**
	Learning Rate	**0.0001**
G-Transformer with Dropout	Number of Layers	**3**
	Hidden Dimension	32, 64, **128**
	Batch Size	**16**
	Learning Rate	**0.0001**

Table 7: Hyperparameter search space in Cancer Growth experiments.

	Hyperparameters	**Range**
G-Transformer	Number of Layers	1, **2**, 3
	HD (Continuous)	**64**, 128
	Batch Size	**16**, 32
	Learning Rate	0.0001, **0.001**
G-Transformer-VD	Number of Layers	**1**, 2, 3
	HD (Continuous)	**64**, 128
	Batch Size	**16**, 32
	Learning Rate	0.0001, **0.001**

Table 8: Hyperparameter settings in MIMIC experiments with 7-hour data. We used the best hyperparameter settings based on 24-hour data from previous experiments and perform no tuning for G-Transformer Ensemble because we use the same hyperparameters from the baseline G-Transformer.

	Hyperparameters	**Range**
G-Transformer	Number of Layers	3
	HD (Continuous)	128
	Batch Size	16
	Learning Rate	0.0001
G-Transformer-VD	Number of Layers	3
	HD (Continuous)	128
	Batch Size	16
	Learning Rate	0.0001
G-Transformer-VD-DE	Number of Layers	3
	HD (Continuous)	128
	Batch Size	16
	Learning Rate	0.0001
G-Transformer-DE	Number of Layers	3
	HD (Continuous)	128
	Batch Size	16
	Learning Rate	0.0001