Uncertainty Quantification in Deep Learning Models of G-Computation for Outcome Prediction under Dynamic Treatment Regimes

by

Leon Deng

S.B. Computer Science and Engineering and Mathematics, MIT, 2024

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by: Leon Deng
Department of Electrical Engineering and Computer Science
August 16, 2024

Certified by: Roger G. Mark
Distinguished Professor, Thesis Supervisor

Certified by: Li-wei H. Lehman
Research Scientist, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Uncertainty Quantification in Deep Learning Models of G-Computation for Outcome Prediction under Dynamic Treatment Regimes

by

Leon Deng

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

G-Net is a neural network framework that implements g-computation, a causal inference method for making counterfactual predictions and estimating treatment effects under dynamic and time-varying treatment regimes. Two G-Net models have been successfully implemented: one that uses recurrent neural networks (RNNs) as its predictors, and one that uses transformer encoders (G-Transformer). However, one limitation of G-Net is that its counterfactual predictive density estimates do not take into account uncertainty about model parameter estimates. These uncertainty estimates are necessary for establishing confidence intervals around the effect estimation, enabling a robust assessment of whether the effects of two treatment options exhibit statistically significant differences.

An important area of work is adding support for quantification of model uncertainty for conditional effect estimation. This thesis aims to add uncertainty quantification to both the RNN-based G-Net and the G-Transformer. To achieve this, we use two well-known techniques in uncertainty modeling, namely variational dropout and deep ensembling. We evaluate our methods using two simulated datasets based on mechanistic models. We demonstrate that G-Net and G-Transformer models with uncertainty quantification are better-calibrated and perform better for individual-level clinical decision making than their baseline counterparts.

Thesis supervisor: Roger G. Mark
Title: Distinguished Professor

Thesis supervisor: Li-wei H. Lehman
Title: Research Scientist

Acknowledgments

Firstly, I'd like to thank Li-Wei Lehman for all the guidance she has given me in the research process. Thanks to her guidance, I have become a more inquisitive, thorough, and rigorous researcher. She was also a great help in defining the direction and scope of my thesis. Without her many helpful suggestions, the research process would be nearly impossible for me.

I thank Professor Roger Mark for accepting me into the Laboratory for Computational Physiology. I am very grateful for the research opportunity he has provided for me.

I thank Professor Zach Shahn, Dr. Soumya Ghosh, and Sanyam Kapoor for their amazing technical guidance. With their expertise, I was able to more deeply understand the theory and motivations behind my research, in turn making the research process much smoother.

I thank Hong Xiong and Feng Wu, who were integral in my onboarding process to this project. I am grateful for their help in answering my many questions early on in the research process. They were a pleasure to work with, and I wish them success in their research.

Finally, I thank my family and friends for their constant support throughout the last year. Shout out to my mom!

Contents

Title page 1

Abstract 3

Acknowledgments 5

List of Figures 9

List of Tables 11

1 Introduction 13

1.1 G-Computation . . . 13
1.2 G-Net and G-Transformer . . . 14
1.3 Thesis Overview and Contributions . . . 14
1.4 Thesis Outline . . . 15

2 Background and Related Works 17

2.1 Uncertainty Quantification Methods . . . 17
2.2 Counterfactual Prediction . . . 18
2.3 Counterfactual Prediction with Uncertainty Quantification . . . 18

3 Problem Statement and G-Computation 19

3.1 Problem Statement . . . 19
3.2 G-computation . . . 20

4 Methods 21

4.1 G-Net and G-Transformer . . . 21
4.1.1 G-Net and G-Transformer Framework . . . 21
4.1.2 Training Procedure . . . 22
4.1.3 Simulation Procedure . . . 22
4.2 Uncertainty Quantification . . . 23
4.2.1 Variational Dropout . . . 23
4.2.2 Deep Ensemble . . . 24
4.2.3 Variational Dropout with Ensembling . . . 26
4.3 Evaluation Methods . . . 26

4.3.1 Calibration 26
4.3.2 Population-Level and Individual-Level RMSE 26
4.4 Datasets 27

5 Experiments **29**
5.1 Evaluation of Uncertainty Predictions 29
5.1.1 G-Net and G-Transformer Implementation 29
5.1.2 Training 31
5.1.3 Simulation 31
5.1.4 Evaluation 31
5.2 Clinical Decision Making 31
5.2.1 Decision Rules 32
5.2.2 Methodology 32
5.2.3 Evaluation 33

6 CVSim Results **35**
6.1 Evaluation of Uncertainty Predictions 35
6.1.1 G-Net Based Model Calibrations and RMSEs 35
6.1.2 G-Transformer Based Model Calibrations and RMSEs 36
6.2 Clinical Decision-Making 37
6.2.1 G-Net Based Decision Making 38
6.2.2 G-Transformer Based Decision Making 38
6.3 Ensemble Size Comparison 41
6.4 Case Study 42

7 Cancer Growth Results **47**
7.1 Evaluation of Uncertainty Predictions 47
7.1.1 G-Net Based Model Calibrations and RMSEs 47
7.1.2 G-Transformer Based Model Calibrations and RMSEs 47
7.2 Clinical Decision-Making 49

8 Conclusion **51**

A **53**
A.1 CVSim Hyperparameter Settings 53
A.2 Cancer Growth Hyperparameter Settings 54
A.3 G-Net with Dropout Ensemble Results 55
A.4 G-Transformer with Dropout Ensemble Results 56
A.5 Alternative Implementations for Deep Ensemble 57

List of Figures

4.1 Depiction of an RNN with the standard dropout masks (left) and G-Net with Dropout (right). Colored arrows indicate dropout masks and dotted arrows indicate their absence. Arrows that are colored identically are identical dropout masks. Figure from [4]. 25

6.1 Per time-step calibration for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 36

6.2 Per time-step calibration for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 37

6.3 Decision making tasks for G-Net based models for varying MAP thresholds and a fixed PVP threshold at 20. Lower is better. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. . . . 38

6.4 Percent difference in performance on decision making tasks between G-Net Ensemble and the base G-Net model. Lower (greener) is better. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 39

6.5 Percent difference in performance on decision making tasks between G-Net w/ Dropout and the base G-Net model. Lower (greener) is better. 39

6.6 Decision making tasks for G-Transformer based models for varying MAP thresholds and a fixed PVP threshold at 20. Lower is better. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 40

6.7 Percent difference in performance on decision making tasks between G-Transformer Ensemble and the base G-Net model. Lower (greener) is better. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 40

6.8 Percent difference in performance on decision making tasks between G-Transformer w/ Dropout and the base G-Net model. Lower (greener) is better. 41

6.9 Per time-step calibration for G-Net Ensemble for various ensemble sizes on CVSim under counterfactual regimes g_{c1} and g_{c2}. Darker lines are larger ensembles. Ideal is 0.90. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 42

6.10 Per time-step calibration for G-Transformer Ensemble for various ensemble sizes on CVSim under counterfactual regimes g_{c1} and g_{c2}. Darker lines are larger ensembles. Ideal is 0.90. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 42
6.11 MAP Simulations from G-Net and G-Net with Dropout for patient 56 under counterfactual treatment strategy g_{c1}. VD=variational dropout. 43
6.12 MAP Simulations from G-Net and G-Net with Dropout for patient 123 under counterfactual treatment strategy g_{c1}. VD=variational dropout. 44
6.13 PVP Simulations from G-Net and G-Net with Dropout for patient 350 under counterfactual treatment strategy g_{c1}. VD=variational dropout. 44
6.14 PVP Simulations from G-Net and G-Net with Dropout for patient 386 under counterfactual treatment strategy g_{c1}. VD=variational dropout. 45

A.1 Per time-step calibration for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. 55
A.2 Percent difference in proportion of time-steps performance between G-Net w/ Dropout Ensemble and the base G-Net model. Lower (greener) is better. . . 55
A.3 Per time-step calibration for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. 56
A.4 Percent difference in proportion of time-steps performance between G-Transformer w/ Dropout Ensemble and the base G-Net model. Lower (greener) is better. 56
A.5 Per time-step calibration for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. 57
A.6 Per time-step calibration for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. 57

List of Tables

6.1 Individual-level RMSEs for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Bolded is best for each of g_{c1} and g_{c2}. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 36
6.2 Individual-level RMSEs for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Bolded is best for each of g_{c1} and g_{c2}. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 37

7.1 Percent RMSEs for G-Net based models on Cancer Growth data for various prediction horizons. Best performing models in bold. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 48
7.2 Calibrations for G-Net based models on Cancer Growth under the four counterfactual regimes. Calculated using the 5th and 95th quantiles; 0.90 is ideal. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 48
7.3 Percent RMSEs for G-Transformer based models on Cancer Growth data for various prediction horizons. Best performing models in bold. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 49
7.4 Calibrations for G-Transformer based models on Cancer Growth under the four counterfactual regimes. Calculated using the 5th and 95th quantiles; 0.90 is ideal. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 49
7.5 Frequency that the tumor volume threshold is crossed on the final time-step if we decide with the above G-Net based models. Lower is better. Best performing models in bold. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise. 50
7.6 Frequency that the tumor volume threshold is crossed on the final time-step if we decide with the above G-Transformer based models. Lower is better. Best performing models in bold. G-Transformer Ensemble is trained on MSE and simulated with empirical noise. 50

A.1 Hyperparameter search space in CVSim experiments. We perform no tuning for G-Transformer Ensemble because we can copy hyperparameters from the base G-Transformer model. 53

A.2 Hyperparameter search space in Cancer Growth experiments. We perform no tuning for G-Transformer Ensemble because we can copy hyperparameters from the base G-Transformer model. 54

Chapter 1

Introduction

In the context of clinical decision-making, counterfactual prediction is the problem of predicting how a patient would have responded to a different treatment strategy than what they actually received. In the real world, clinicians only have access to the adopted treatment strategies and the observed patient outcomes, which we call the *observational regime*. However, they would also like to know what would have occurred under a counterfactual treatment strategy. This is especially relevant for clinical decision-making settings in which doctors do not have the option to try alternative candidate treatments on a patient before making a decision. Counterfactual prediction is also especially important for settings in which different patients often respond differently to the same treatment. With better counterfactual predictions, clinicians can make more informed treatment decisions for their patients that can lead to better outcomes.

While methods for generating counterfactual predictions have been developed, clinicians need more than just predictions of a patient's treatment response. Equally critical is the ability to understand and quantify the uncertainty surrounding these predictions. By accurately modeling the probability distributions of a patient's potential treatment response, future trajectories, and the risks of adverse outcomes under different treatment plans, clinicians can make more informed decisions. With uncertainty quantification, clinicians would be better-equipped to navigate the difficult risk-reward trade-off inherent in many clinical treatment decisions. Thus, our work focuses on predicting patient responses to treatment in clinical settings as well as the uncertainty around these predictions. We investigate multiple uncertainty quantification approaches, including deep ensembles and variational dropout, to approximate the probability distributions of patients' future trajectories under alternative dynamic treatment strategies. We compare their performance in estimating the conditional treatment effect and when used for downstream decision making tasks.

1.1 G-Computation

In clinical settings, treatment strategies are often complex, involving decisions at multiple time-steps (*time varying*), where decision-making at each time-step depends on past patient history (*dynamic*). These properties of clinical treatment strategies further complicate the inherently difficult task of counterfactual prediction.

G-computation is a causal inference framework that is well-suited to this task [14, 13]. G-computation works by first learning the distribution of a patient's covariates (like heart rate and blood pressure) conditioned on the patient's history of treatments and covariates. It then can iteratively produce predictions of patient outcomes under counterfactual treatment strategies. Every time-step, g-computation decides treatments based on the desired counterfactual treatment strategy and then draws an estimate of that time-step's covariates from the learned conditional distribution. Simulating many trajectories gives us an estimate of the distribution of patient outcomes at every time-step.

1.2 G-Net and G-Transformer

Though conventional implementations of g-computation have used generalized linear models to model the conditional distribution of covariates, G-Net [10] and G-Transformer **cite** have recently demonstrated that g-computation can also be implemented through deep learning models. G-Net is a deep learning framework that implements g-computation with recurrent neural networks, and G-Transformer implements g-computation with transformers. Since the base models used have much greater expressivity than linear models, they can make much more accurate predictions of expected patient trajectories under counterfactual treatment strategies.

1.3 Thesis Overview and Contributions

Though the use of neural networks in G-Net and G-Transformer for g-computation offer performance boosts over more traditional implementations using linear models, their current implementations do not provide adequate estimates for the uncertainty over model parameters. Our work builds upon G-Net and G-Transformer by enhancing them to support uncertainty quantification. We evaluated the quality of these models by checking their calibration and predictive accuracy. We also evaluated their performance on two clinical decision-making task using two simulated datasets based on mechanistic models. In particular, we study the problem of administering treatment in the form of fluids and vasopressors to sepsis patients using a simulated data set generated by CVSim. We use a simulated dataset from a pharmacokinetics and pharmacodynamics (PK/PD) model of tumor growth to study the usage of chemotherapy and radiotherapy to treat cancer patients.

Our contributions are the following:

1. **Enhanced G-Net and G-Transformer to support uncertainty quantification.** We use variational dropout [5] and deep ensemble [9], two commonly-used methods for uncertainty quantification. We adapt the G-Net and G-Transformer models to be more compatible with their respective uncertainty quantification methods.

2. **Benchmarked the performance of different uncertainty quantification approaches in counterfactual prediction under dynamic treatment regimes.** We use the enhanced G-Net and G-Transformer based models for counterfactual prediction on two simulated datasets based on mechanistic models of the human body.

We evaluate their performance on an individual level using calibration and root mean squared error (RMSE) as our metrics. Since the goal of this research is primarily to accurately estimate predictive uncertainty, we prioritize improving calibration while maintaining similar levels of predictive performance. We show that the enhanced G-Net and G-Transformer models are better calibrated than their baseline counterparts while maintaining or even improving predictive accuracy.

3. **Characterized impact to downstream decision making tasks under different uncertainty quantification approaches.** We design a clinical decision-making task that emulates the decision-making process that a doctor may use to prevent choosing treatment strategies that could lead to adverse outcomes such as hypotension and pulmonary edema. We demonstrate that the enhanced G-Net and G-Transformer models outperform their baseline counterparts in these tasks, meaning that they do better in predicting and preventing rare adverse clinical outcomes under alternative dynamic treatment regimes.

1.4 Thesis Outline

Chapter 2 provides a summary of related works in g-computation methods and counterfactual prediction. Chapter 3 describes the theory behind g-computation. Chapter 4 presents in detail the G-Net and G-Transformer frameworks, the uncertainty quantification methods, and the simulated datasets. Chapter 5 describes the implementation and design of our experiments. Chapter 6 presents the results of our experiments. Chapter 7 summarizes our results and discusses their implications and limitations.

Chapter 2

Background and Related Works

Recent works presented deep learning approaches to estimate time-varying treatment effects [11, 2, 1, 12]. However, most previous approaches focus on estimating counterfactual outcomes under static time-varying treatment strategies where treatments are not dependent on past covariate history. Recent works have leveraged machine learning for counterfactual prediction under **dynamic and time-varying** treatment regimes. In particular, G-Net and G-Transformer are two deep learning approaches that support g-computation, a causal inference framework that is well-suited conditional treatment effect estimation under dynamic treatment regimes [14, 13]. However, none of these prior efforts support uncertainty quantification in a **dynamic and time-varying** treatment setting.

2.1 Uncertainty Quantification Methods

There exist many machine learning frameworks for uncertainty quantification. Deep ensemble [9] is a machine learning framework that captures uncertainty in both model parameters and predicted outputs. It enhances a base model by ensembling it with bagging, capturing model uncertainty. For continuous outputs, it predicts both a point estimate and the variance around that estimate, capturing uncertainty in the model output. Bayesian neural networks can be used to obtain model uncertainty, but using them for inference is computationally expensive. Other methods attempt to approximate Bayesian inference at lower computational costs. For example, variational dropout [5] utilizes dropout, a regularization technique used commonly in training neural networks. Turning on dropout masks during test time and using them to simulate multiple stochastic forward passes through the network enables approximate Bayesian inference and estimates of predictive uncertainty. Stochastic Weight Averaging-Gaussian (SWAG) also approximates the posterior distribution of neural network weights by collecting multiple snapshots of one model during its training and using them to fit a Gaussian to the weights. While these methods have enjoyed success in many applications in which predicting uncertainty is important, they have not been used for counterfactual prediction under dynamic time-varying interventions.

2.2 Counterfactual Prediction

There exist many frameworks for counterfactual prediction under dynamic time-varying interventions. G-computation [14, 13] is a framework that uses generalized linear models (GLMs) to learn covariate distributions conditioned on patient history. It then uses the conditional covariate distributions to iteratively predict outcomes under counterfactual regimes. G-Net [10] implements g-computation with neural networks, replacing the GLMs used in traditional g-computation methods with recurrent neural networks (RNNs) for their strength in expressing long-term dependencies. G-Transformer [15] also implements g-computation by using transformer encoders instead of GLMs, leveraging the expressivity of the more modern model type. While these frameworks have been shown to produce accurate counterfactual predictions, neither G-Net nor G-Transformer have adequate measures of predictive uncertainty.

2.3 Counterfactual Prediction with Uncertainty Quantification

Classic implementations of g-computation using linear models have used bootstrapping for uncertainty quantification. For complex prediction settings, 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 predicted conditional counterfactual distributions.

In "A Bayesian approach to the g-formula" [8], the authors compare the frequentist g-formula (“standard g-formula”) using bootstrap and the Bayesian g-formula. However, their goal was regularization. They were interested in average treatment effect while we focus instead on the conditional treatment effect, which is a more difficult problem.

Prior works have noted that machine learning approaches to treatment effect estimation could lead to highly biased estimates, and straightforward application of conventional uncertainty quantification approaches such as bootstrap do not work with machine learning technique in these settings. To get average treatment effect estimation with ML approaches, we need to use doubly-robust estimation and cross fitting when using bootstrap to get confidence intervals [3].

Chapter 3

Problem Statement and G-Computation

In this work, we use the g-computation framework to tackle the problem of counterfactual prediction under dynamic and time-varying treatment strategies. G-computation is a causal inference technique that can be used for treatment effect estimation conditioned on patient history [14, 13]. The g-computation algorithm is composed primarily of two stages. It first uses patient data to estimate the conditional distribution of patient outcomes given history. It then uses the learned distribution to estimate possible patient trajectories via Monte Carlo simulation.

This chapter presents an overview of g-computation and provides the foundation for deep-learning implementations such as G-Net and G-Transformer. We define the notation we use and list the technical assumptions we need.

3.1 Problem Statement

G-computation is a framework used to make counterfactual predictions according to time varying and dynamic treatment strategies [14]. These strategies involve making decisions at multiple time-steps, each conditioned on the patient history up until that point. In this work, we represent patient history with patient covariates and treatment actions measured at discrete time-steps.

For ease of description, we define variables as follows:

- $t \in \{1, \ldots, K\}$: time-steps, assumed discrete
- A_t: the observed treatment action at time t
- L_t: the vector of covariates at time t
- Y_t: the outcome(s) of interest at time t. For simplicity, we model the outcome(s) as covariates and include them in the vector L_t.
- $Y_t(S_m)$: the outcome of following counterfactual strategy S from time-steps m to t
- $H_m \triangleq (L_{1:m}, A_{1:m-1})$: the patient history at time t

3.2 G-computation

The goal of counterfactual prediction is then to estimate the expected outcome of a patient at time t if some counterfactual strategy had started at time m: $\mathbb{E}[Y_t(S)|H_m]$. Of equal importance to us is the conditional distribution of patient outcomes: $p(Y_t(S)|H_m)$. G-computation is well-suited for this task under the following technical assumptions [14]:

- **Consistency:** when the counterfactual strategy is the same as the observed strategy, their outcomes are equivalent.
- **Sequential Exchangeability:** there is no observed confounding at any time-step. Any covariate that factors into treatment decision making is observed.
- **Positivity:** the counterfactual strategy S has a positive probability of being administered.

The g-computation method is as follows. When $t = m$ and under the above assumptions, the treatment action A_m is a direct function of the counterfactual strategy S, so we have

$$p(Y_m(S)|H_m) = p(Y_m|H_m, A_m = S(H_m)). \tag{3.1}$$

When $t > m$ and under the above assumptions, the conditional distribution becomes complex due to time-varying confounding. The closed form becomes

$$\begin{aligned} p(Y_t(S) = y|H_m) = &\int_{l_{m+1:t}} p(Y_t = y|H_m, L_{m+1:t} = l_{m+1:t}, A_{m:t} = S(H_m, l_{m+1:t})) \\ &\times \prod_{j=m+1}^{t} p(L_j = l_j|H_m, L_{m+1:j-1} = l_{m+1:j-1}, A_{m:j-1} = S(H_m, l_{m+1:j-1})). \end{aligned} \tag{3.2}$$

This expression is intractable, so we use Monte-Carlo simulation to estimate it. At each time step, the simulation strategy involves sampling from the distribution of covariates conditioned on all actual and simulated observations until that time-step. The simulation procedure requires an estimate of this conditional distribution:

$$p(L_t|L_{1:t-1}, A_{1:t-1}). \tag{3.3}$$

The conditional distribution is estimated in g-computation, and is trained in G-Net [10] and G-Transformer [15]. To simulate one time-step given the tuple $(L_{1:m}, A_{1:m}, S)$, we sample $\hat{L}_{m+1}$ from our trained conditional distribution $p(L_{m+1}|L_{1:m}, A_{1:m})$ and then decide A_{m+1} according to $S(L_{1:m}, A_{1:m}, \hat{L}_{m+1})$. This process can be repeated iteratively to simulate any number of time-steps. If we simulate multiple trajectories, we get an empirical estimate of the conditional distribution from (3.3). If we average these trajectories, we get an empirical estimate of the conditional expectation $\mathbb{E}[L_t|L_{1:t-1}, A_{1:t-1}]$.

Chapter 4

Methods

In this work, we enhance the G-Net and G-Transformer models by using two uncertainty quantification methods: *variational dropout* and *deep ensemble*. We evaluate the models' performances on two datasets that cover different domains: Cancer Growth [6], which simulates cancerous tumor growths, and CVSim [7], which simulates the human cardiovascular system. Calibration is our main performance metric of interest, though we also would like to maintain predictive accuracy measured by root mean-squared error.

This chapter begins with an overview of the G-Net and G-Transformer frameworks. We move on to a summary of the uncertainty quantification methods used in this work and how we implemented them for our specific use cases. Lastly, we discuss our choice of model evaluation methods and introduce the datasets we use for evaluation.

4.1 G-Net and G-Transformer

G-Net [10] and G-Transformer [15] 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 models that do not have adequate measures of uncertainty.

4.1.1 G-Net and G-Transformer Framework

The goal of G-Net and G-Transformer is the same as that of g-computation: to approximate the conditional distribution of covariates $p(L_t|L_{1:t-1}, A_{1:t-1})$. 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

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

In this work, we use $G = 2$ and set the groups to be the categorical and continuous covariates, respectively. This is called the *two-box* architecture, with one classifier for the categorical covariates and one regressor for the continuous covariates. With this framework, we have the ability to group the covariates however we like. One option used previously sets G to be the number of covariates, which we call the *one-variable-per-box* architecture.

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 offers impressive flexibility because we never assume the model architectures of R^0 and R^1. We can essentially use any model that conforms to the desired input and output format. 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.

4.1.2 Training Procedure

For the two-box model, we train the two boxes jointly, 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, as we will discuss in section 4.2.2.

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 $\hat{L}_{t+1}^0$ to R_1, we pass the observed L_{t+1}^0. Since we evaluate performance on multiple time-step ahead prediction, teacher-forcing notably optimizes a different loss function than our true objective. However, we've found that teacher-forcing performs better than our alternative, *student forcing*.

4.1.3 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 would 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:

$$L_t^g|L_t^0 \dots L_t^{g-1}, L_{1:t-1}, A_{1:t-1} \sim \hat{\mathbb{E}}[L_t^0 \dots L_t^{g-1}, L_t|L_{1:t-1}, A_{1:t-1}] + \epsilon_t^g \tag{4.1}$$

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, so we simulate 32 time-steps ahead in that case.

4.2 Uncertainty Quantification

One thing to note about the base G-Net and G-Transformer models is that while their simulation procedure accounts for the uncertainty in the covariate distribution (by using empirical residuals), they do not take into account the uncertainty in the model parameters. To address this, we enhance G-Net and G-Transformer with two different methods to produce uncertainty estimates: *variational dropout* [5] and *deep ensemble* [9].

This section describes the theory behind the two uncertainty quantification methods, their implementations, and how we adapted them to work well with the existing G-Net and G-Transformer models.

4.2.1 Variational Dropout

Dropout is traditionally used in training neural networks as a form of regularization. In such use cases, inputs to certain network layers are randomly masked during training time, but no masking occurs during test time to maximize model performance. However, Gal and Ghahramani showed that dropout, if used during test time, can also be used to produce a variational approximation to the posterior distribution of a Bayesian neural network [5]. Simulating multiple forward passes of a trained model with independent and identically distributed dropout masks would give us samples that take into account the uncertainty over model parameters.

This corresponds very nicely with the G-Net framework because G-Net and G-Transformer already perform Monte Carlo simulations during test time. To implement variational dropout for G-Transformer, for example, we can take a trained G-Transformer model and (with minor modifications) sample trajectories from it with dropout masks turned on. With dropout masks turned off, the G-Transformer model captures uncertainty in the covariate distribution; with dropout masks turned on, G-Transformer captures uncertainty in the model parameters as well.

To prevent confusion, we will refer to the variational dropout versions of G-Net and G-Transformer *G-Net with Dropout* and *G-Transformer with Dropout*, respectively.

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-Net with Dropout

G-Transformer with Dropout can be implemented simply as the base G-Transformer model along with the modifications discussed earlier in this chapter. 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 [4] 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.

4.2.2 Deep Ensemble

Deep ensemble [9] is a flexible method of producing uncertainty estimates, and it can be applied to many different neural architectures. The deep ensemble algorithm starts with a base model — in our case, a recurrent neural network or a transformer encoder. It trains an ensemble of these models through *bagging*, with randomization coming from just from the initialization of model parameters and the order in which data is fed into the model during training. Furthermore, each individual model within the ensemble models the conditional distribution of the continuous covariates as a Gaussian with some mean and variance. Thus, each model outputs not only a predicted mean but also a variance estimate. Then for each model, deep ensemble treats the observed continuous data as samples from a Gaussian distribution parametrized by the predicted mean and variance and optimizes the negative log-likelihood of this sample as its loss function. With θ representing a model's parameters, we have:

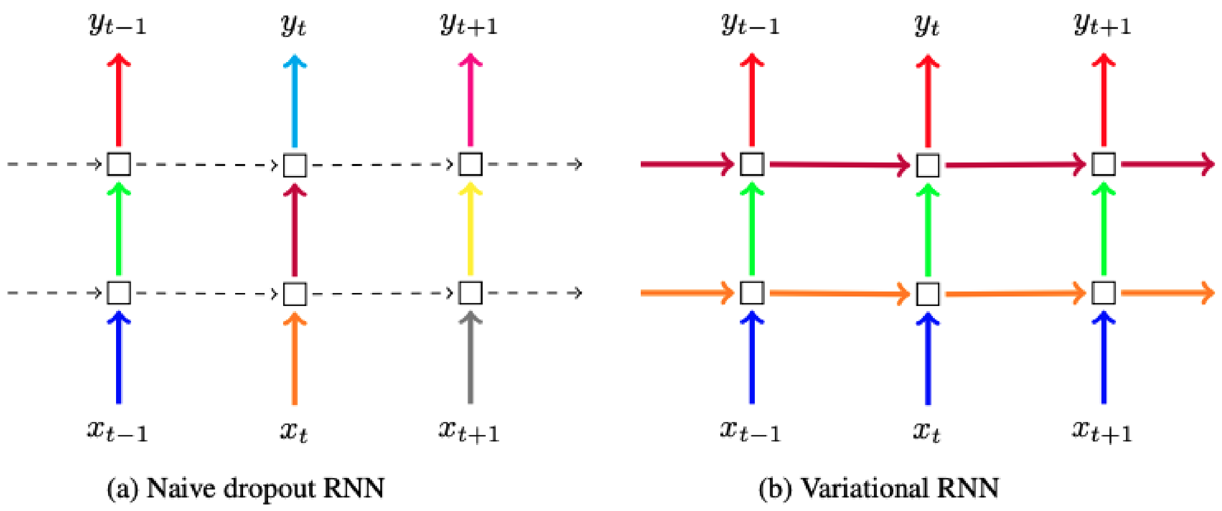


Figure 4.1: Depiction of an RNN with the standard dropout masks (left) and G-Net with Dropout (right). Colored arrows indicate dropout masks and dotted arrows indicate their absence. Arrows that are colored identically are identical dropout masks. Figure from [4].

$$\text{Loss}(\theta, \mathbf{x}, y) \triangleq -\log p_\theta(y|\mathbf{x}) = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{(y - \mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} + \text{constant}. \tag{4.2}$$

The entire ensemble is then interpreted as a mixture of Gaussians with equal weights. As the number of models, M, increases, the expressivity of the ensemble increases accordingly; even a highly multimodal distribution can be approximated well with a large enough ensemble. We found that $M = 20$ worked well for our purposes.

The deep ensemble method quantifies uncertainty in both the model output and in the model parameters. Ensembling with random initializations represents uncertainty in model parameters, and parametrizing the model output as Gaussian represents uncertainty in the model output. In some of our applications of deep ensemble, we simulate using empirical residuals instead of the predicted Gaussian, in which case the empirical residuals represent uncertainty in the model output.

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 5.1.1.

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 4.1. 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 were 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.

4.2.3 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. Results for this type of model are included in the appendix.

4.3 Evaluation Methods

We have two performance metrics of interest for the G-Net and G-Transformer models both with and without uncertainty quantification: *calibration* and *root mean-squared error*. The latter can be calculated two ways, by individual level and by population level. The goal of this thesis on uncertainty estimation is to improve the G-Net and G-Transformer calibrations with uncertainty quantification while maintaining similar levels of individual-level and population-level RMSEs.

4.3.1 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 model's observed covariates fall between the α_{low} and α_{high} quantiles of the predicted covariates. The frequency is calculated across all patients, time-steps, and covariates for a single number. If the frequency is close to $\alpha_{high} - \alpha_{low}$, the model is well-calibrated.

4.3.2 Population-Level and Individual-Level RMSE

Performance of a trained model can be evaluated by *population-level* RMSE or by *individual-level* RMSE. To compute population-level RMSE, the simulated trajectories for every patient are averaged into a single population-level trajectory, which is then compared to the ground-truth population-level trajectory. To compute individual-level RMSE, trajectories are averaged per patient and compared to the ground-truth individual-level trajectories. 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

$$\sqrt{\frac{1}{N_c(K-m)d}\sum_{i=1}^{N_c}\sum_{t=m}^{K}\sum_{h=1}^{d}(L_{ti}^{h,CF} - \hat{L}_{ti}^{h,CF}(f))^2}. \tag{4.3}$$

For the Cancer Growth and CVSim datasets, we focus on the individual-level RMSE because it offers a more granular representation of the predictive power of our models.

4.4 Datasets

We perform experiments on two separate clinical datasets: Cancer Growth, CVSim. Each has different properties that affect how we model the distribution of covariates given patient history.

Cancer-Growth Dataset. The Cancer Growth dataset is simulated from a pharmacokinetic-pharmacodynamic (PK/PD) model of tumor growth [6]. We have a data generated under an observational treatment in which chemotherapy and radiotherapy (the two treatments) are applied to the patient through a strategy that is conditioned on their tumor volume history [10]. We also have four testing datasets, each generated with a counterfactual treatment strategy applied in the last four time-steps of each trajectory. The counterfactual strategies are no treatment, radiotherapy, chemotherapy, and both (chem-rad). Treatment strategies are time-varying but static.

CVSim Dataset. The CVSim dataset is simulated from a model of the human cardiovascular system [7]. We have data generated under an observational treatment g_0 and two testing counterfactual datasets generated under treatments g_{c1} and g_{c2}. g_0 is a treatment strategy in which the fluid and vasopressor treatment at each time-step is *stochastic*. On the other hand, g_{c1} and g_{c2} are *deterministic* treatment strategies, where g_{c2} is strictly more aggressive than g_{c1}. Treatment strategies are time-varying and dynamic.

Chapter 5

Experiments

To evaluate the performance of our models with uncertainty quantification, we conduct two types of experiments. The first is in line with experiments that have already been conducted for G-Net and G-Transformer: measuring the calibration and RMSEs of predicted patient outcomes on simulated datasets. The second evaluates a model's usefulness for decision-making in clinical settings.

In this chapter, we describe the motivations and implementation details of these two experiments.

5.1 Evaluation of Uncertainty Predictions

In this section, we describe our methods for evaluating how well our models quantify uncertainty. We train, simulate, and 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 4.2.2.

The methodologies for the CVSim and the Cancer Growth datasets are similar, so the following descriptions apply for both. Any dataset-specific differences are noted in their respective subsections.

5.1.1 G-Net and G-Transformer Implementation

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 4.1.

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. The G-Net with Dropout model is implemented nearly identically, but with the dropout masks implemented as described in 4.2.1.

At each time-step, the base G-Net model theoretically 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.

Implementing 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 as well.

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 as in 4.2.2 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. To ensure numerical stability, we use PyTorch's MultivariateNormal distribution to compute negative log-likelihood.

5.1.2 Training

Our training procedure for each of our models begins with a grid search over multiple hyperparamter settings. After selecting hyperparameters, we train the two boxes of the model on the data collected from the observational regime. The two boxes are couple and are trained with teacher forcing. 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 simply 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. For G-Net Ensemble, we trained 20 G-Net models on seeds 0 to 19; for G-Transformer Ensemble, we trained 20 G-Transformer models on seeds 0 to 19.

5.1.3 Simulation

After training our models, we simulate once once for each of the counterfactual treatment strategies. For the CVSim dataset, we simulate under g_{c1} and g_{c2}; for the Cancer Growth dataset, we simulate under no treatment, radiotherapy, chemotherapy, and both. Simulation happens as described in 4.1. For a fixed counterfactual treatment strategy, we simulate 100 trajectories per patient.

For the sake of fair comparison, we require that the ensemble models produce exactly 100 trajectories per patient as well. Since our ensembles are composed of 20 models, each model contributes 5 trajectories per patient to the overall ensemble's collection of simulations.

5.1.4 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 in 4.3.

For the Cancer Growth dataset, we only simulate under the counterfactual treatment strategies for four time-steps. This allows us to analyze RMSEs with more granularity, so we calculate per-time-step individual-level RMSEs in this case. We also divide by the maximum tumor volume ($1150cm^3$) for percentage RMSEs.

At this stage, we use these metrics to choose between the different versions of the ensemble models. We choose primarily using calibration because calibration reflects the quality of the model's predictive uncertainty better than RMSE does.

5.2 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 dataset, 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.

5.2.1 Decision Rules

Before we go into detail about the methodology of this experiment, we must define objective functions that emulate those of a doctor. For the CVSim dataset, doctors want to avoid worst-case scenarios in which a patient's MAP drops too low and also cases in which a patient's PVP rises too high. For the Cancer Growth dataset, doctors want to avoid scenarios in which a patient's cancer volume rises too high.

For the CVSim dataset, the *proportion of time-steps* function is the proportion of time-steps in which a patient's MAP drops below some threshold *or* their PVP rises above some threshold. On the other hand, the *proportion of patients* function is the proportion of patients in which a patient's MAP drops below some threshold *or* their PVP rises above some threshold at some point in their trajectory. We include results for both functions, but we focus on the proportion of time-steps function because it better represents the duration of time during which the patients are unwell.

For the Cancer Growth dataset, the *final tumor volume* function is proportion of patients in which a patient's cancer volume rises above some threshold on the final time-step. The function is designed this way because in the real world, we care more about the patient's final cancer volume than the patient's history of cancer before it, and we also hope that this volume doesn't get too large.

5.2.2 Methodology

Our methodology emulates the decision-making process that a doctor may use when deciding a treatment strategy for a patient. If a doctor wanted to choose between multiple treatment strategies, they would choose the one that they predict would result in the best patient outcome according to some utility function that gives importance to avoiding especially poor outcomes. Such functions are described in the above section.

For the CVSim dataset, our decision-making task is as follows. We pre-select MAP and PVP thresholds and either *proportion of time-steps* or *proportion of patients* as our objective function. Note that since we want to avoid time-steps in which MAP is too low or PVP is too high, we want to *minimize* the objective. For each patient, our task is to decide between g_{c1} and g_{c2} for treatment.

For the Cancer Growth dataset, we pre-select cancer volume thresholds for our *final tumor volume* function. We want to minimize this objective, and for each patient, we decide between no treatment, radiotherapy, chemotherapy, or chem-rad for treatment.

We decide with some trained G-Net or G-Transformer based model. Using the model, we simulate 100 trajectories for the patient under each of the counterfactual treatment regimes. We compute the objective functions between all sets of simulations, and we choose the treatment strategy that minimizes the objective. This process is done for each patient on an individualized basis.

5.2.3 Evaluation

With the choice of treatment strategy for every patient, we evaluate our model's performance by computing the objective function for the patient's *ground truth trajectory* corresponding to the chosen treatment strategy. We compute this result across all our G-Net and G-Transformer based models, across multiple sets of (PVP and MAP, or cancer volume) thresholds, and across all our objective functions. We also compute the results that would occur if we naively chose one of the counterfactual treatment strategies for every single patient; we use these as baselines.

Chapter 6

CVSim Results

In this chapter, we present the results of the CVSim experiments described in chapter 5. The experiments for calibration and RMSE quantify the quality of the uncertainty estimation and the predictive accuracy of the models. The experiments for decision-making quantify the usefulness of uncertainty estimation in real settings. We include a brief analysis on how ensemble size affects model calibration. Finally, we include a brief case study that illustrates the value of adding uncertainty quantification to our models.

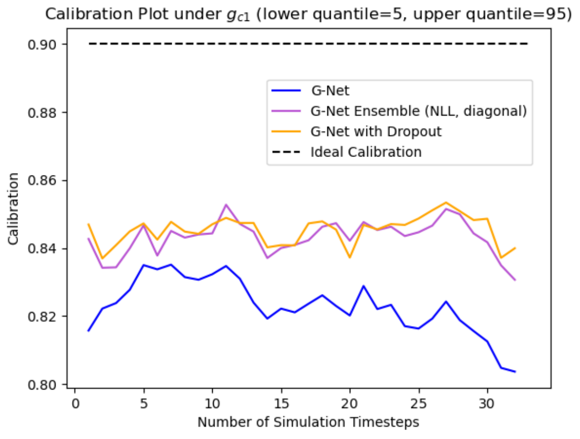
6.1 Evaluation of Uncertainty Predictions

For each of the G-Net and G-Transformer based models, we performed experiments as described in 5.1. 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. To compare between uncertainty quantification methods, we group results from the G-Net based models separately from the G-Transformer based results.

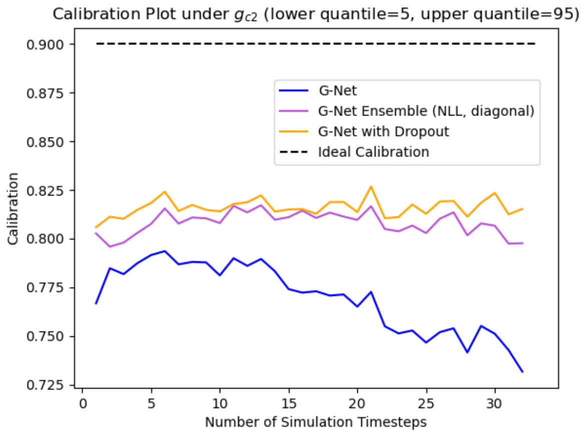
6.1.1 G-Net Based Model Calibrations and RMSEs

For the G-Net based models, both methods of uncertainty quantification succeeded in improving calibration uniformly across all time-steps. Of the many training and simulation methods available for the ensemble model, training with NLL with respect to a diagonal Gaussian and simulating with empirical noise performed the best, so we present those results. While the calibration results don't meet the ideal of 0.90, improving by a couple percent as shown in the plots is still a significant improvement. Interestingly, while the base G-Net model's calibration decays into further time-steps, the calibrations of the enhanced models don't exhibit that effect as much.

As for the RMSE results, we see that the individual-level RMSEs of the models are all at similar levels. Adding uncertainty quantification even has the potential to improve RMSE under some counterfactual regimes, as with G-Net with Dropout under g_{c2}.



(a) CVSim under g_{c1}



(b) CVSim under g_{c2}

Figure 6.1: Per time-step calibration for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.

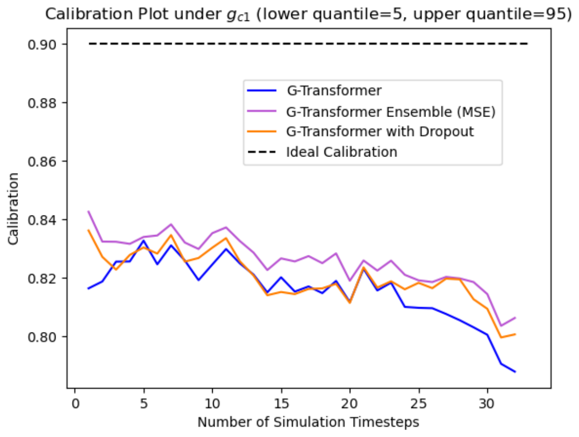
	Individual Level	
Model	**RMSE**$_{gc1}$	**RMSE**$_{gc2}$
G-Net	**1.022**	1.114
G-Net Ensemble (NLL)	1.027	1.119
G-Net with Dropout	1.033	**1.111**
G-Net with Dropout Ensemble	1.027	1.115

Table 6.1: Individual-level RMSEs for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Bolded is best for each of g_{c1} and g_{c2}. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.

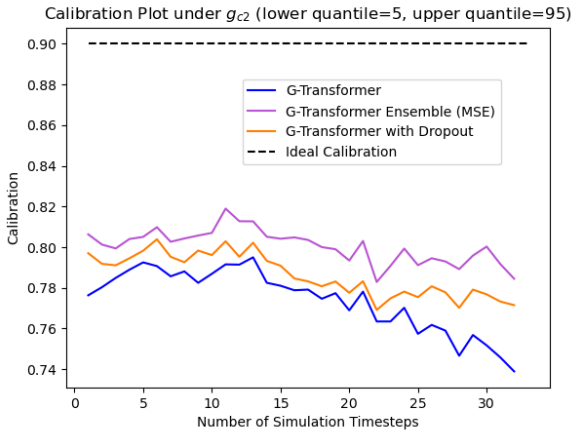
6.1.2 G-Transformer Based Model Calibrations and RMSEs

For the G-Transformer based models, both methods of uncertainty quantification improved calibration, though not by significant amounts. Of the many training and simulation methods available for the ensemble model, training with MSE and simulating with empirical noise performed the best, so we present those results. Under g_{c1}, all three models have similar calibrations, with G-Transformer Ensemble performing marginally better than the base model. Under g_{c2}, G-Transformer Ensemble's edge in calibration over the base model is more significant. Under both counterfactual regimes, G-Transformer with Dropout offers only marginal benefits in calibration over the base model, if at all.

We again observe that the individual-level RMSEs of the models are all at similar levels. As before, adding uncertainty quantification sometimes improves RMSE, as with G-Transformer Ensemble under g_{c1}.



(a) CVSim under g_{c1}



(b) CVSim under g_{c2}

Figure 6.2: Per time-step calibration for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

Model	Individual Level RMSE$_{gc1}$	Individual Level RMSE$_{gc2}$
G-Transformer	1.017	1.102
G-Transformer Ensemble (MSE)	**1.013**	1.105
G-Transformer with Dropout	1.017	1.109
G-Transformer with Dropout Ensemble	1.014	**1.099**

Table 6.2: Individual-level RMSEs for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Bolded is best for each of g_{c1} and g_{c2}. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

6.2 Clinical Decision-Making

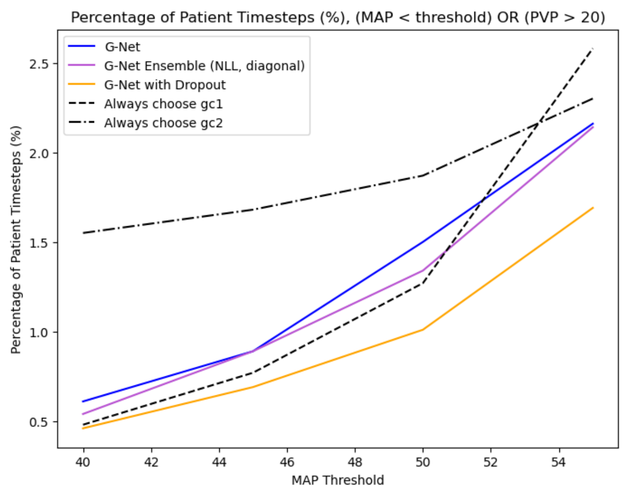
We now present the results for the clinical decision making experiment as described in 5.2. We present results from both the G-Net and G-Transformer based models on the proportion of time-steps and proportion of patients function tasks. We select the same ensemble variants that we used for evaluating calibration and RMSE in the previous section. For the proportion of time-steps task, we fixed the PVP threshold to 20 and analyzed results with MAP thresholds from the set $\{40, 45, 50, 55\}$. For the proportion of patients task, we used MAP thresholds from $\{35, 40, 45, 50, 55\}$. We compared the model performances with the baselines of always choosing g_{c1} or always choosing g_{c2} for every patient.

Note that we care more about the proportion of time-steps task because we care about the duration of time a patient's condition is poor.

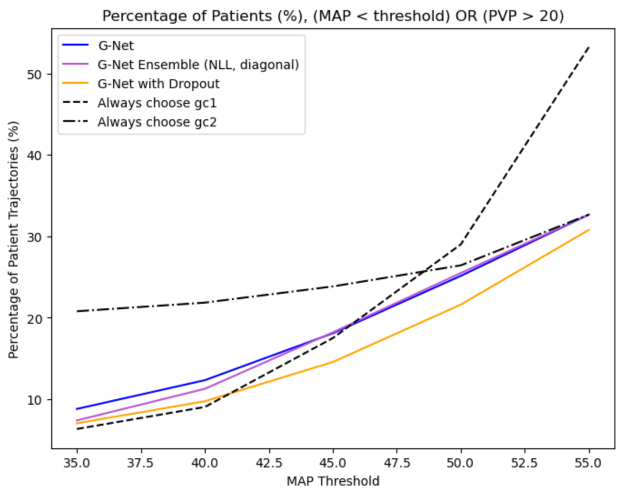
6.2.1 G-Net Based Decision Making

For the G-Net based models, both uncertainty quantification methods improved the base model's performance on the decision-making task. G-Net Ensemble at least matches G-Net's performance across all thresholds and tasks, while G-Net with Dropout performs significantly better. Notably, while G-Net often fails to outperform always choosing g_{c1} on the proportion of time-steps task, G-Net with Dropout outperforms g_{c1} across all the thresholds. The same nearly holds true for the proportion of patients task, where G-Net with Dropout very nearly matches the performance of always choosing g_{c1}. This suggests that for the G-Net based models, uncertainty quantification not only improves clinical decision making performance. It can also make G-Net based models more viable for individual-level treatment rather than population-level treatment.

Another point to note is that uncertainty quantification methods improve G-Net's ability to predict *rare events*. For example, the event of a patient's MAP falling below 40 or their PVP rising above 20 happens in just 0.5% of time-steps and in roughly 10% of patients if g_{c1} is always chosen as the treatment strategy. As the MAP threshold lowers (indicating more rare events), the base G-Net model begins performing worse than always choosing g_{c1}. However, G-Net with Dropout still outperforms g_{c1} in those tasks, suggesting that G-Net with Dropout is better than the base G-Net model at predicting rare events. This property is especially useful for our task of clinical decision-making because those rare events are often the ones that clinicians want to prevent the most.



(a) Proportion of time-steps task

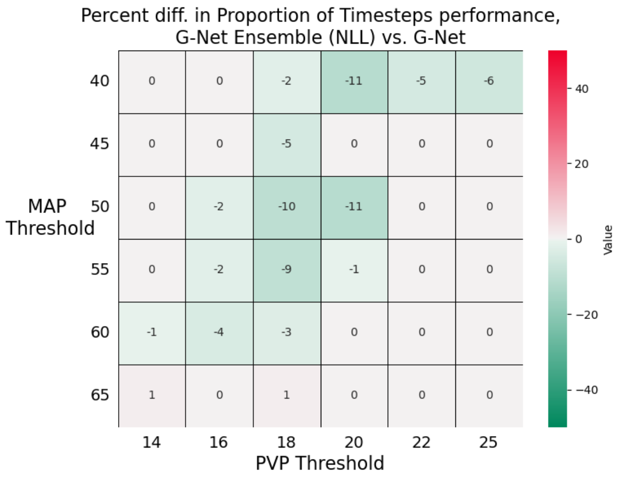


(b) Proportion of patients task

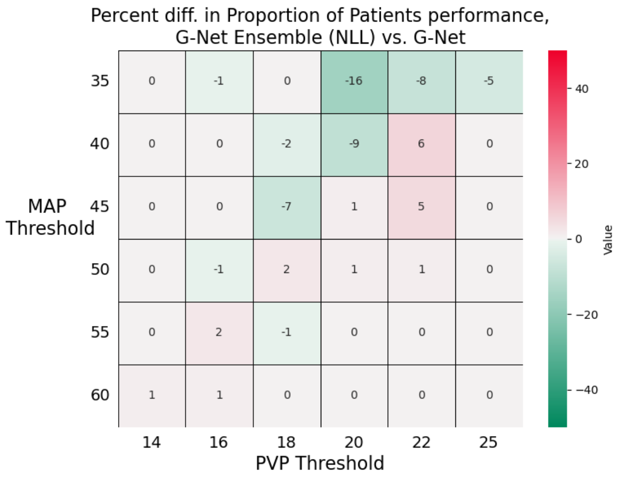
Figure 6.3: Decision making tasks for G-Net based models for varying MAP thresholds and a fixed PVP threshold at 20. Lower is better. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.

6.2.2 G-Transformer Based Decision Making

For the G-Transformer based models, only deep ensemble improved the base model's performance on the decision-making task. G-Net with Dropout performs slightly worse than

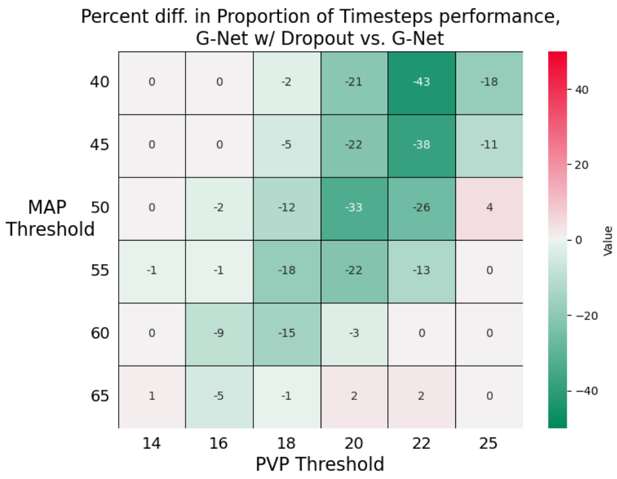


(a) Proportion of time-steps task

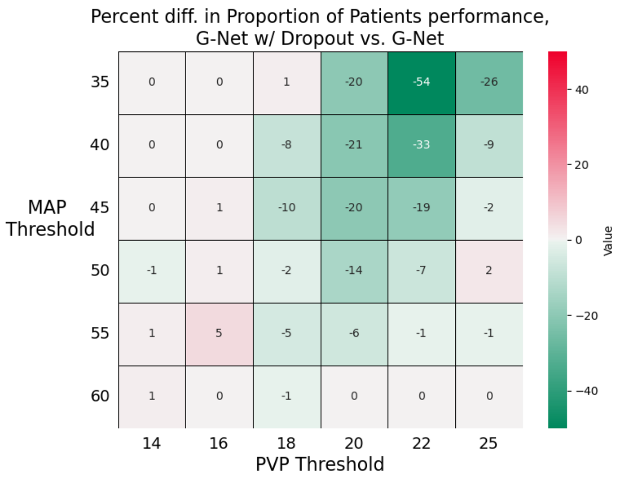


(b) Proportion of patients task

Figure 6.4: Percent difference in performance on decision making tasks between G-Net Ensemble and the base G-Net model. Lower (greener) is better. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.



(a) Proportion of time-steps task



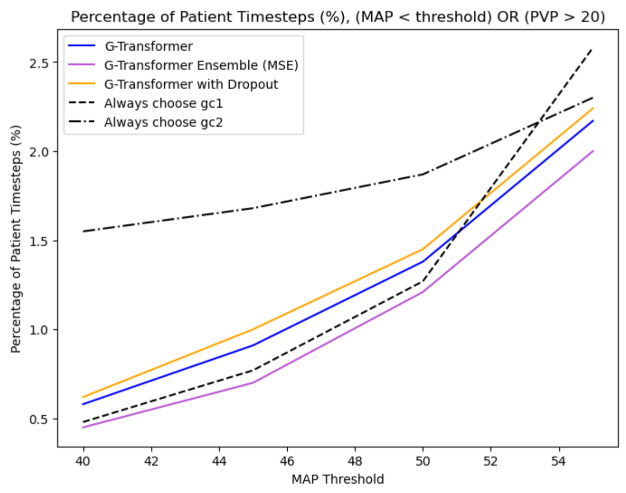
(b) Proportion of patients task

Figure 6.5: Percent difference in performance on decision making tasks between G-Net w/ Dropout and the base G-Net model. Lower (greener) is better.

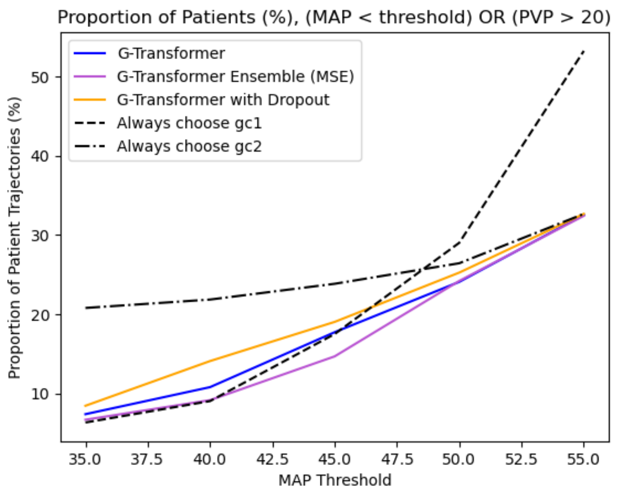
the base G-Transformer model. As we saw before, one of the uncertainty quantification methods outperforms always choosing g_{c1} (and g_{c2}), though this time it's deep ensemble instead of variational dropout. As was the case with G-Net, this suggests that uncertainty quantification can make G-Transformer more effective at individual-level predictions.

Again as we saw before with G-Net, uncertainty quantification for G-Transformer can make it more effective at predicting rare events. While the base G-Transformer model is outperformed by always choosing g_{c1} for the MAP < 50 threshold (which happens just 1% of the time under g_{c1}), the G-Transformer Ensemble model outperforms always choosing g_{c1}

across all thresholds.

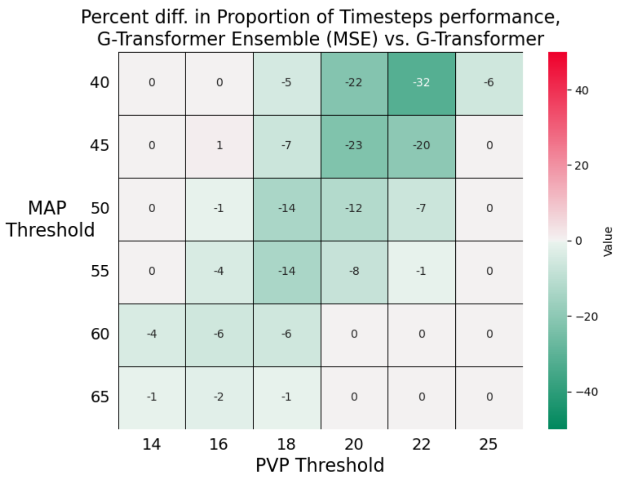


(a) Proportion of time-steps task

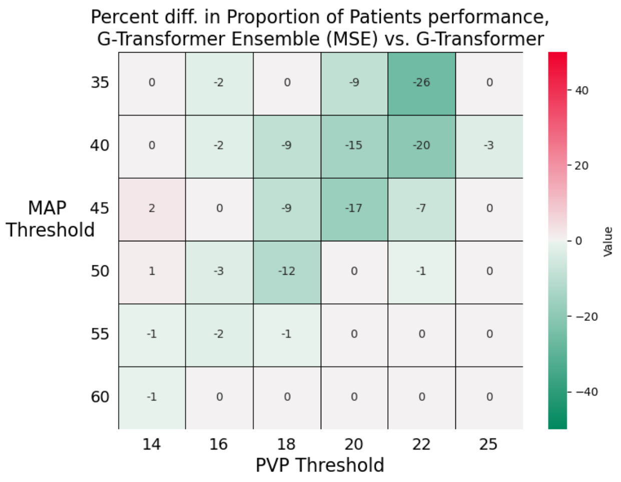


(b) Proportion of patients task

Figure 6.6: Decision making tasks for G-Transformer based models for varying MAP thresholds and a fixed PVP threshold at 20. Lower is better. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

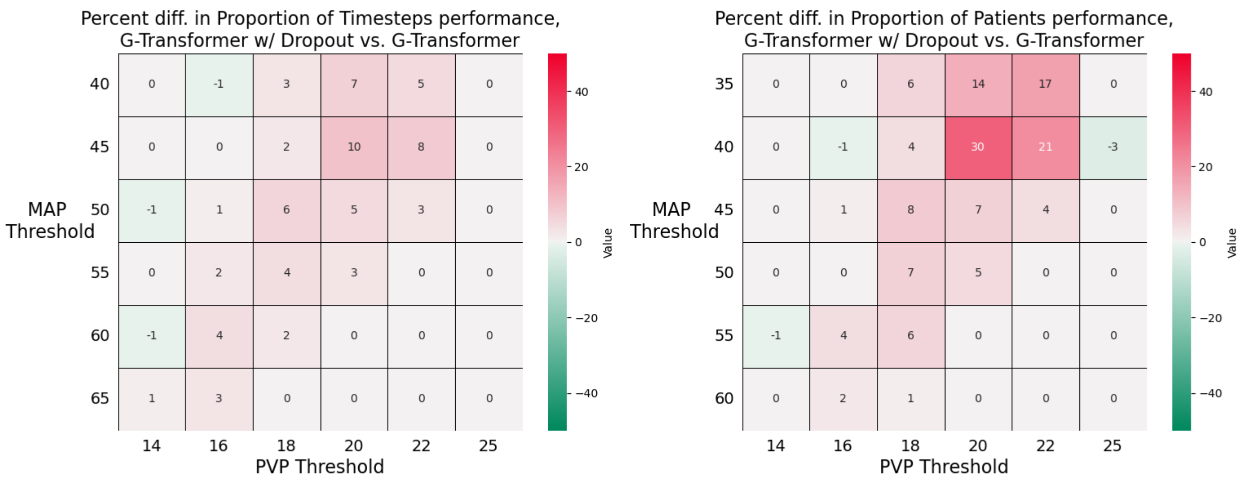


(a) Proportion of time-steps task



(b) Proportion of patients task

Figure 6.7: Percent difference in performance on decision making tasks between G-Transformer Ensemble and the base G-Net model. Lower (greener) is better. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.



(a) Proportion of time-steps task

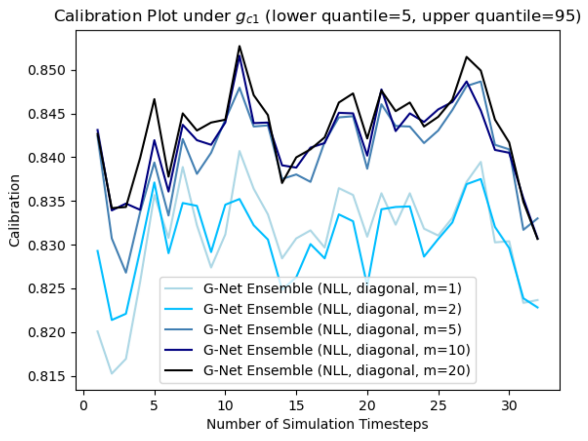
(b) Proportion of patients task

Figure 6.8: Percent difference in performance on decision making tasks between G-Transformer w/ Dropout and the base G-Net model. Lower (greener) is better.

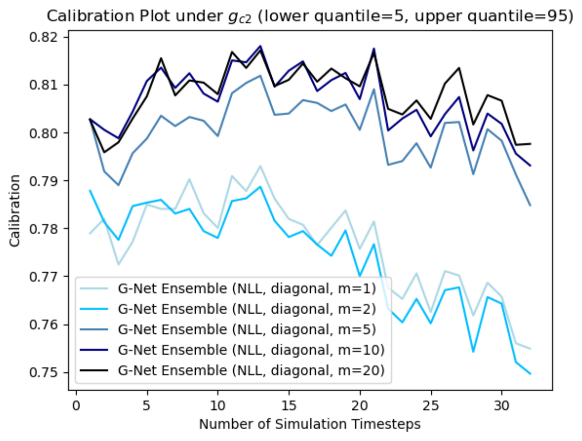
6.3 Ensemble Size Comparison

As an aside to the calibration and RMSE experiments, we also compared calibration results for various ensemble sizes to determine how additional ensemble members 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 made sure to include 100 simulated trajectories for each of the ensembles to compare fairly.

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.

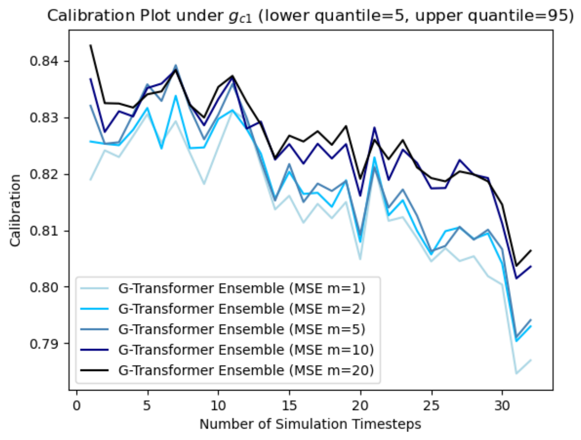


(a) CVSim under g_{c1}

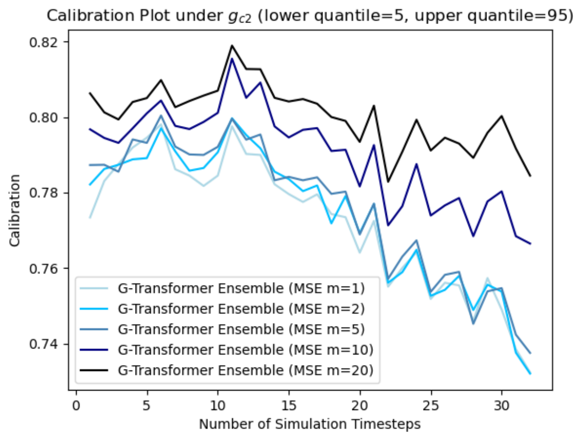


(b) CVSim under g_{c2}

Figure 6.9: Per time-step calibration for G-Net Ensemble for various ensemble sizes on CVSim under counterfactual regimes g_{c1} and g_{c2}. Darker lines are larger ensembles. Ideal is 0.90. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.



(a) CVSim under g_{c1}



(b) CVSim under g_{c2}

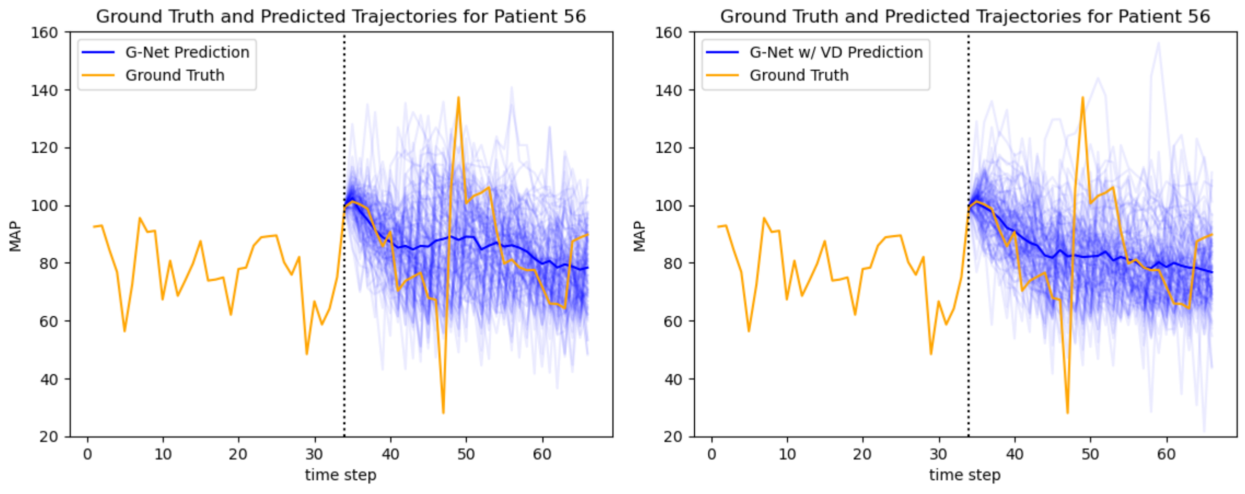
Figure 6.10: Per time-step calibration for G-Transformer Ensemble for various ensemble sizes on CVSim under counterfactual regimes g_{c1} and g_{c2}. Darker lines are larger ensembles. Ideal is 0.90. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

6.4 Case Study

To further illustrate the value added by uncertainty quantification, we present a case study of patients in the CVSim datasets. For both patients, we compare counterfactual predictions for g_{c1} produced by the G-Net and G-Net with Dropout models. We purposefully chose patients that encountered rare, adverse outcomes in their ground truth trajectories to see how well the two models predicted such an event in their simulations. For each model, we plotted the patient's ground truth trajectories, the model's 100 simulated patient trajectories, and the model's prediction (obtained by taking a per time-step average of the 100 simulations).

We specifically looked at trajectories of MAP and PVP because those were the relevant covariates in the decision making experiments. More specifically, we focused on patients whose MAP values fell very low or patients whose PVP values rose very high.

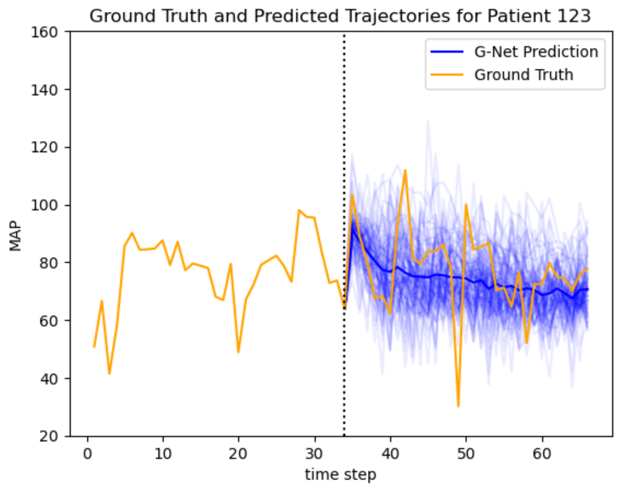
We can see that with patients 56 and 123, whose MAP values dipped below 30, G-Net with Dropout was able to predict dips of such magnitude while the base G-Net model wasn't. With patients 207 and 386, whose PVP values rose above 25, the difference in the simulations between the two models is even more pronounced. For patient 386, for example, G-Net didn't come close to predicting that the patient's PVP might spike, whereas G-Net with Dropout had many simulations indicating that it might happen. This indicates that G-Net with Dropout is much better at predicting for rare events than G-Net, which may not predict for such events at all.

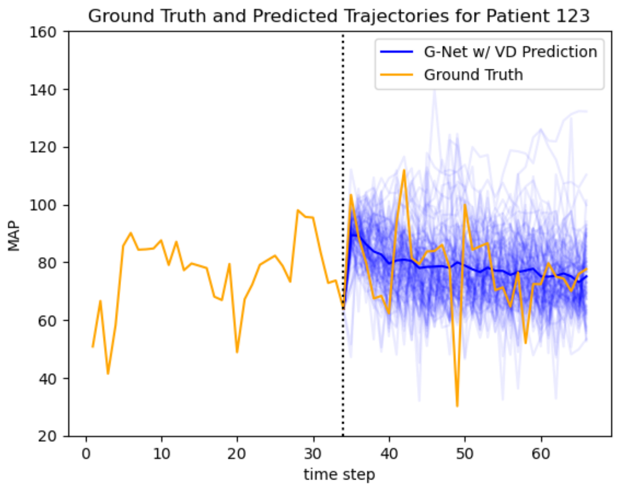


(a) G-Net Simulations

(b) G-Net with Dropout Simulations

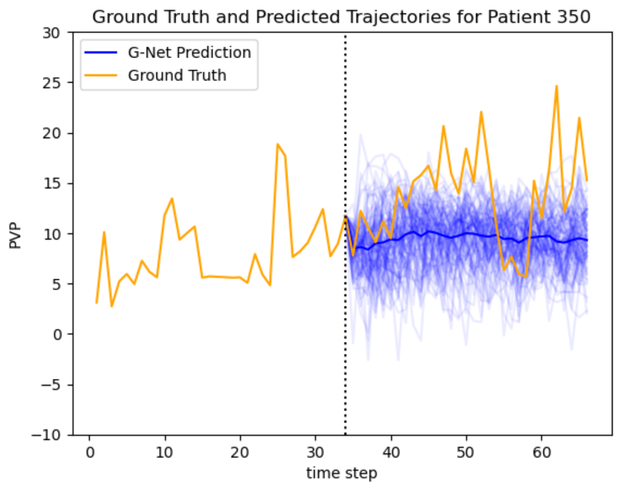
Figure 6.11: MAP Simulations from G-Net and G-Net with Dropout for patient 56 under counterfactual treatment strategy g_{c1}. VD=variational dropout.

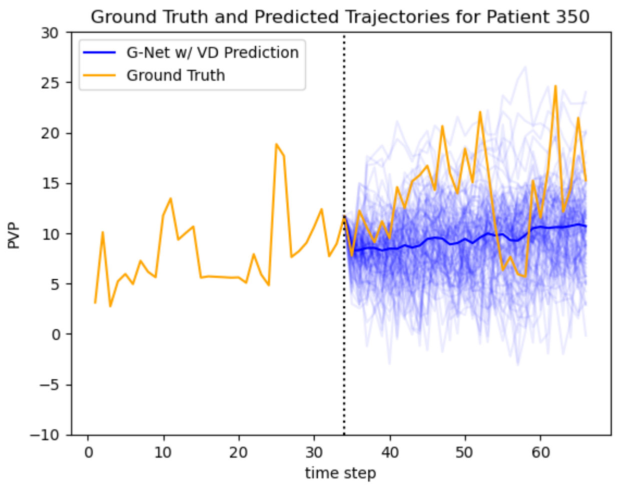




(a) G-Net Simulations (b) G-Net with Dropout Simulations

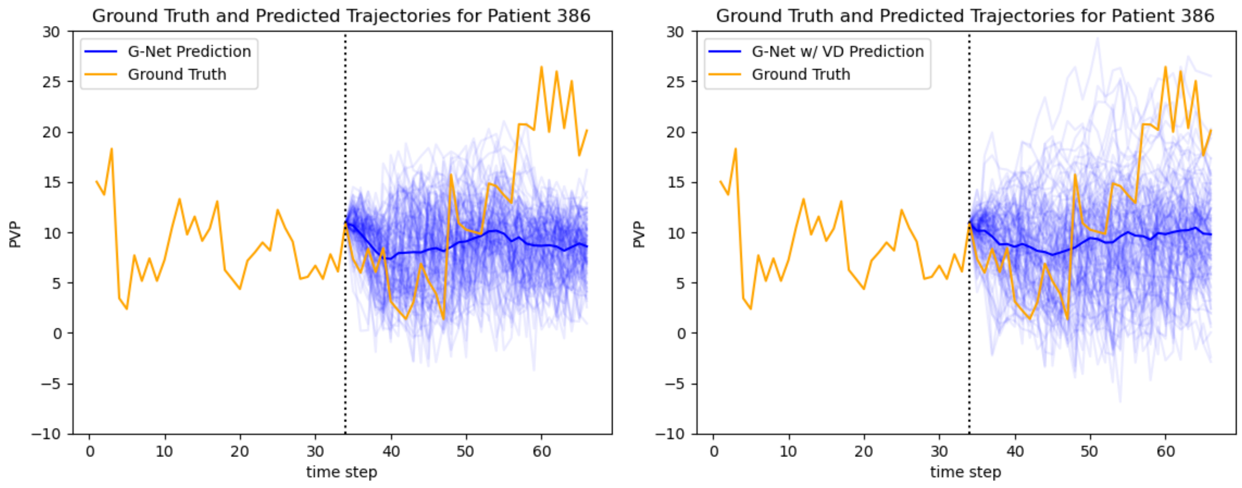
Figure 6.12: MAP Simulations from G-Net and G-Net with Dropout for patient 123 under counterfactual treatment strategy g_{c1}. VD=variational dropout.





(a) G-Net Simulations (b) G-Net with Dropout Simulations

Figure 6.13: PVP Simulations from G-Net and G-Net with Dropout for patient 350 under counterfactual treatment strategy g_{c1}. VD=variational dropout.



(a) G-Net Simulations

(b) G-Net with Dropout Simulations

Figure 6.14: PVP Simulations from G-Net and G-Net with Dropout for patient 386 under counterfactual treatment strategy g_{c1}. VD=variational dropout.

Chapter 7

Cancer Growth Results

Our experiments for the Cancer Growth dataset are twofold: we analyze the calibration and RMSE of every model, and we evaluate the models on a decision-making task.

7.1 Evaluation of Uncertainty Predictions

We calculated calibrations and RMSEs for all models and for each of our four counterfactual regimes. Since the Cancer Growth dataset has just four time-steps of counterfactual treatment, we present individual-level RMSEs for each time-step for greater granularity. Similarly to CVSim, we group results around the base model: G-Net and G-Transformer.

7.1.1 G-Net Based Model Calibrations and RMSEs

Of the ensemble variants for G-Net, we found that training with NLL and simulating with empirical noise performed the best, so we present those results. We found that the models with uncertainty quantification had better RMSEs than the base model under no treatment and chemotherapy, while the base model performed better on the other two counterfactual regimes.

For calibration, we focus our analysis on the no treatment and chemotherapy treatment strategies because all the models produced greatly biased estimates for radiotherapy and chem-rad. With such biased estimates, calibration becomes less meaningful — simply adding noise to biased simulations will increase calibration. Focusing on no treatment and chemotherapy, we see that G-Net Ensemble is better calibrated than G-Net under chemotherapy but worse under no treatment. G-Net with Dropout's simulations are too noisy, with calibrations close to 1.

7.1.2 G-Transformer Based Model Calibrations and RMSEs

Of the ensemble variants for G-Transformer, we found that training with MSE and simulating with empirical noise performed the best, so we present those results. We found that the models with uncertainty quantification had better RMSEs than the base model under every treatment strategy, with G-Transformer Ensemble having the best RMSE under no

	t	G-Net	G-Net Ensemble (NLL)	G-Net with Dropout
No Treat	1	**0.25**	**0.25**	0.26
	2	0.52	**0.41**	0.50
	3	0.77	**0.49**	0.73
	4	1.03	**0.59**	0.98
	A	0.64	**0.44**	0.62
Radio	1	**3.60**	5.31	4.41
	2	**3.64**	5.14	4.39
	3	**3.64**	4.82	4.27
	4	**3.73**	4.60	4.24
	A	**3.65**	4.97	4.33
Chemo	1	0.36	**0.26**	0.29
	2	0.65	**0.41**	0.53
	3	0.87	**0.49**	0.72
	4	0.97	**0.56**	0.81
	A	0.71	**0.43**	0.59
Radio Chemo	1	**3.51**	5.03	4.02
	2	**2.31**	3.64	2.87
	3	**1.52**	2.56	1.97
	4	**1.17**	2.02	1.41
	A	**2.31**	3.31	2.57

Table 7.1: Percent RMSEs for G-Net based models on Cancer Growth data for various prediction horizons. Best performing models in bold. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.

	G-Net	G-Net Ensemble (NLL)	G-Net with Dropout
No Treat	0.90	0.95	0.98
Radio	0.55	0.59	0.48
Chemo	0.66	0.93	0.99
ChemRad	0.36	0.74	0.84

Table 7.2: Calibrations for G-Net based models on Cancer Growth under the four counterfactual regimes. Calculated using the 5th and 95th quantiles; 0.90 is ideal. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.

treatment and G-Transformer with Dropout having the best under radiotherapy, chemotherapy, and chem-rad. It's uncommon to see one model outperforming another under all four counterfactual regimes, but G-Transformer with Dropout achieves just that against the base G-Transformer model.

For calibration, we again focus our analysis on the no treatment and chemotherapy treatment strategies. G-Transformer Ensemble and G-Transformer with Dropout are better calibrated than G-Net 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.

	t	G-Transformer	G-Transformer Ensemble (MSE)	G-Transformer with Dropout
No Treat	1	0.42	**0.38**	0.38
	2	0.73	**0.63**	0.67
	3	0.95	**0.79**	0.84
	4	1.19	**0.98**	1.08
	A	0.82	**0.70**	0.74
Radio	1	3.60	3.73	**3.55**
	2	3.54	3.62	**3.36**
	3	3.56	3.47	**3.24**
	4	3.74	3.40	**3.30**
	A	3.61	3.56	**3.36**
Chemo	1	**0.40**	**0.40**	**0.40**
	2	1.33	1.35	**1.07**
	3	2.44	2.48	**2.04**
	4	3.31	3.37	**2.71**
	A	1.87	1.90	**1.56**
Radio Chemo	1	3.25	3.42	**3.17**
	2	2.44	2.47	**2.40**
	3	1.82	**1.72**	1.78
	4	1.45	**1.22**	1.33
	A	2.24	2.21	**2.17**

Table 7.3: Percent RMSEs for G-Transformer based models on Cancer Growth data for various prediction horizons. Best performing models in bold. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

	G-Transformer	G-Transformer Ensemble (MSE)	G-Transformer with Dropout
No Treat	0.87	1.00	0.98
Radio	0.60	0.76	0.74
Chemo	0.72	0.88	0.92
ChemRad	0.69	0.83	0.78

Table 7.4: Calibrations for G-Transformer based models on Cancer Growth under the four counterfactual regimes. Calculated using the 5th and 95th quantiles; 0.90 is ideal. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

7.2 Clinical Decision-Making

We now present results for the clinical decision-making experiment for the Cancer Growth dataset. We present results from the G-Net and G-Transformer based models. We select the same ensemble variants that we used for evaluating calibration and RMSE in the previous section. For the decision making task, we analyzed results with cancer volume thresholds from the set $\{20, 40, 60, 80\}$.

Across every threshold and model type, the models with uncertainty quantification outperform the base models in the decision making task. The deep ensemble and variational dropout methods perform similarly on the decision making tasks. We see again that the models with uncertainty quantification perform better in predicting for rare events than their baseline counterparts.

Threshold	G-Net	G-Net Ensemble(NLL)	G-Net with Dropout
20	0.129	0.086	**0.081**
40	0.068	0.039	**0.038**
60	0.053	**0.026**	**0.026**
80	0.037	**0.015**	**0.015**

Table 7.5: Frequency that the tumor volume threshold is crossed on the final time-step if we decide with the above G-Net based models. Lower is better. Best performing models in bold. G-Net Ensemble is trained on NLL w.r.t. a diagonal Gaussian and simulated with empirical noise.

Threshold	G-Transformer	G-Transformer Ensemble (MSE)	G-Transformer with Dropout
20	0.121	0.088	**0.086**
40	0.070	0.054	**0.052**
60	0.043	**0.041**	0.044
80	0.040	**0.030**	0.034

Table 7.6: Frequency that the tumor volume threshold is crossed on the final time-step if we decide with the above G-Transformer based models. Lower is better. Best performing models in bold. G-Transformer Ensemble is trained on MSE and simulated with empirical noise.

Chapter 8

Conclusion

This thesis builds upon G-Net and G-Transformer, two neural network frameworks for producing counterfactual predictions under dynamic and time-varying clinical treatment strategies. While the base G-Net and G-Transformer models only take into account the uncertainty in patient outcomes, they do not account for model uncertainty. To address this, we enhance G-Net and G-Transformer using deep ensembles and variational dropout, two methods that take into account both types of uncertainty. With greater measures of uncertainty, doctors would be better-equipped to make safer and more accurate decisions in clinical settings.

To evaluate our models, we used two simulated datasets: CVSim, which simulates the human body's cardiovascular system, and Cancer Growth, which simulates human tumor growth. For each dataset, we measured each model's predictive performance using calibration and RMSE. We also evaluated our models on constructed decision making tasks that emulate the decision making process that doctors undergo in real-life clinical settings.

We found that on the CVSim dataset, both methods of uncertainty quantification improved the calibration of the G-Net and G-Transformer models while maintaining or even improving predictive accuracy as measured by RMSE. Additionally, per time-step calibration decayed more slowly for the models with uncertainty quantification, indicating that they may be useful in learning long-term dependencies. The results from the Cancer Growth dataset were less interpretable because all models were biased on the radiotherapy and chemrad counterfactual regimes. Under the no treatment and chemotherapy regimes, we found that uncertainty quantification improved predictive accuracy. On the decision making tasks, we found that the uncertainty quantification methods outperformed the base G-Net and G-Transformer models. Notably, they were especially strong at predicting for rare adverse events, which are exactly the events that doctors want to avoid in clinical settings.

There are a few extensions to this work that we may explore in the future. We'd like to apply our methods to real-world clinical datasets like MIMIC-IV. Evaluating the quality of our uncertainty predictions would be challenging in such settings where we only have access to the ground truth, so we would need to develop some predictive checks. Since neural network architectures are greatly improved by larger amounts of training data, it would be interesting to expand our dataset for future experiments. This would especially benefit the evaluation of the uncertainty quantification methods and how they perform on rare events. There is also some exploration to be done on parametric simulation methods for the deep ensemble based models. If we could use feature transformations to make the conditional distributions of covariates more closely resemble Gaussians, simulating parametrically from a learned Gaussian rather than from an emprical set of residuals could have promise.

We hope that the enhanced G-Net and G-Transformer models presented in this work can help doctors make more accurate and informed clinical decisions, resulting in better patient outcomes overall. Our methods are flexible, so they can be used even outside of clinical applications to any setting which involves counterfactual prediction under dynamic and time-varying interventions.

Appendix A

A.1 CVSim Hyperparameter Settings

Table A.1: Hyperparameter search space in CVSim experiments. We perform no tuning for G-Transformer Ensemble because we can copy hyperparameters from the base G-Transformer model.

	Hyperparameters	**Search Range**
G-Net	Number of Layers	**2**, 3
	Hidden Dimension (Categorical)	64, **128**
	Hidden Dimension (Continuous)	64, **128**
	Batch Size	**16**
	Learning Rate	**0.0001**
G-Net Ensemble (NLL, diagonal)	Batch Size	16, **32**
	Number of Layers	**1**, 2, 4
	Hidden Dimension (Continuous)	32, 64, **128**
	Learning Rate	**0.01**, 0.001
G-Net with Dropout	Batch Size	16, **32**
	Number of Layers	1, **2**, 4
	Hidden Dimension (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**

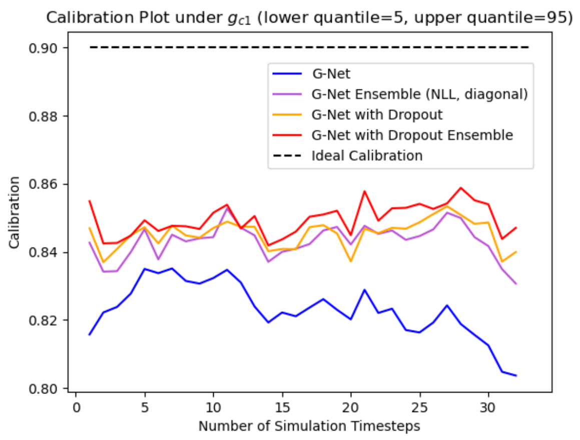
A.2 Cancer Growth Hyperparameter Settings

Table A.2: Hyperparameter search space in Cancer Growth experiments. We perform no tuning for G-Transformer Ensemble because we can copy hyperparameters from the base G-Transformer model.

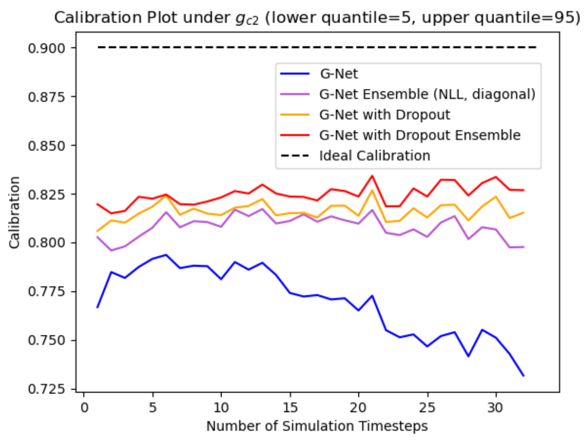
	Hyperparameters	**Search Range**
G-Net	Number of Layers	1, **2**, 4
	Hidden Dimension (Continuous)	**64**
	Batch Size	16, **32**
	Learning Rate	0.001, **0.01**
G-Net Ensemble (NLL, diagonal)	Number of Layers	**1**, 2, 4
	Hidden Dimension (Continuous)	**64**
	Batch Size	**16**, 32
	Learning Rate	**0.001**,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

A.3 G-Net with Dropout Ensemble Results

Calibration Results



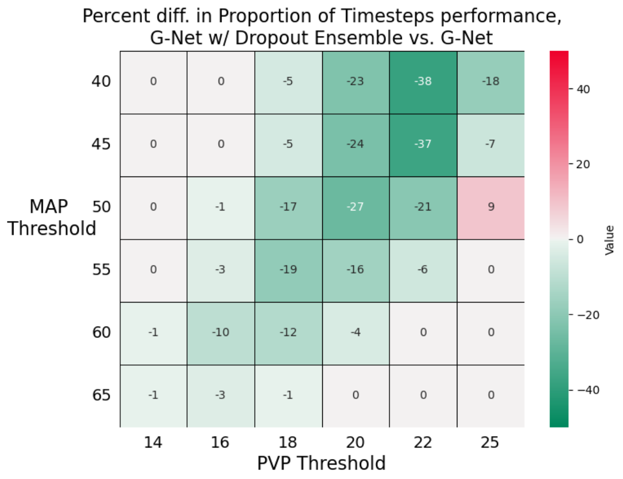
(a) CVSim under g_{c1}



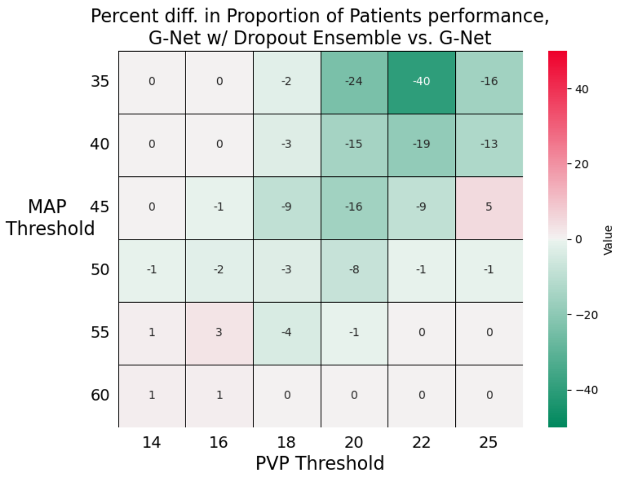
(b) CVSim under g_{c2}

Figure A.1: Per time-step calibration for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90.

Decision Making Results



(a) Proportion of time-steps task

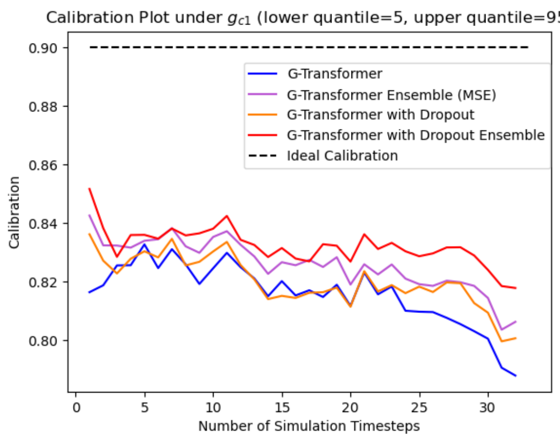


(b) Proportion of patients task

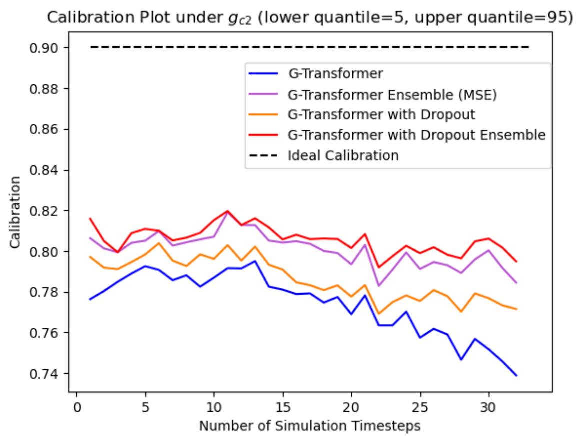
Figure A.2: Percent difference in proportion of time-steps performance between G-Net w/ Dropout Ensemble and the base G-Net model. Lower (greener) is better.

A.4 G-Transformer with Dropout Ensemble Results

Calibration Results



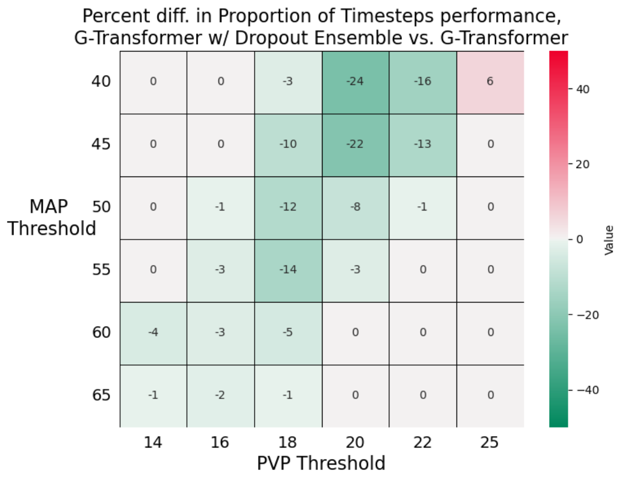
(a) CVSim under g_{c1}



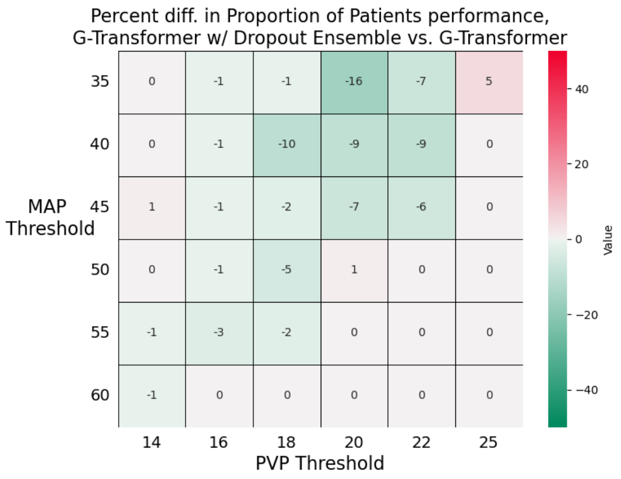
(b) CVSim under g_{c2}

Figure A.3: Per time-step calibration for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90.

Decision Making Results



(a) Proportion of time-steps task



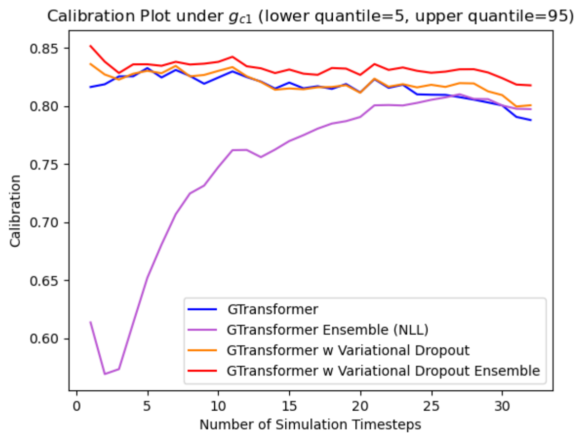
(b) Proportion of patients task

Figure A.4: Percent difference in proportion of time-steps performance between G-Transformer w/ Dropout Ensemble and the base G-Net model. Lower (greener) is better.

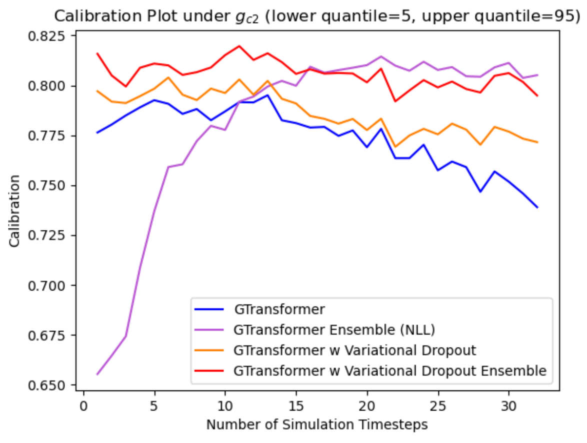
A.5 Alternative Implementations for Deep Ensemble

As mentioned in 4.2.2, the training and simulation process for deep ensemble can be implemented in many ways. We include two methods that we explored: training a diagonal Gaussian and simulating from it parametrically, and training a Gaussian with full covariance matrix and simulating from it.

Train Diagonal Gaussian, Simulate Parametrically



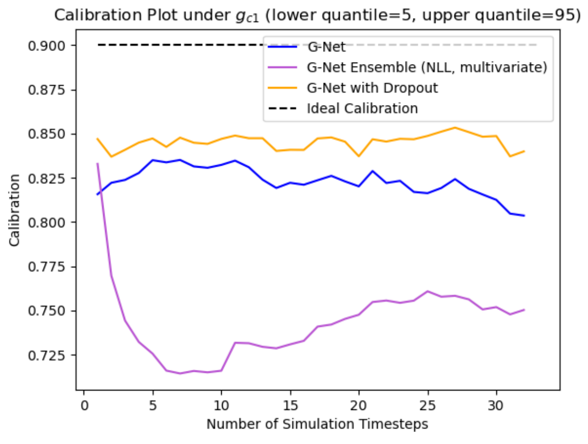
(a) CVSim under g_{c1}



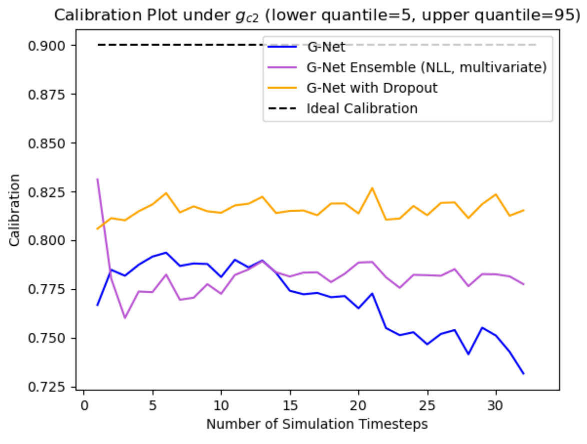
(b) CVSim under g_{c2}

Figure A.5: Per time-step calibration for G-Transformer based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90.

Train Covariance Matrix, Simulate Parametrically



(a) CVSim under g_{c1}



(b) CVSim under g_{c2}

Figure A.6: Per time-step calibration for G-Net based models on CVSim under counterfactual regimes g_{c1} and g_{c2}. Ideal is 0.90.

Bibliography

[1] Ioana Bica, Ahmed M Alaa, and Mihaela van der Schaar. "Time Series Deconfounder: Estimating Treatment Effects over Time in the Presence of Hidden Confounders". In: *International Conference on Machine Learning* (2020).

[2] Ioana Bica et al. "Estimating counterfactual treatment outcomes over time through adversarially balanced representations". In: *International Conference on Learning Representations* (2020).

[3] Victor Chernozhukov et al. "Double/debiased machine learning for treatment and structural parameters". In: *The Econometrics Journal* (June 2017). DOI: 10.3386/w23564.

[4] Yarin Gal and Zoubin Ghahramani. *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks*. 2016. arXiv: 1512.05287 `[stat.ML]`.

[5] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2016. arXiv: 1506.02142 `[stat.ML]`.

[6] 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 bio-mathematical model." In: *Scientific Reports* (2017).

[7] T Heldt et al. "CVSim: An Open-Source Cardiovascular Simulator for Teaching and Research". In: *Open Pacing, Electrophysiol & Ther J* 3 (2010), pp. 45–54.

[8] Alexander P Keil et al. "A Bayesian approach to the g-formula". In: *Statistical Methods in Medical Research* 27.10 (Mar. 2017), pp. 3183–3204. ISSN: 1477-0334. DOI: 10.1177/0962280217694665. URL: http://dx.doi.org/10.1177/0962280217694665.

[9] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. 2017. arXiv: 1612.01474 `[stat.ML]`.

[10] Rui Li et al. *G-Net: A Deep Learning Approach to G-computation for Counterfactual Outcome Prediction Under Dynamic Treatment Regimes*. 2020. arXiv: 2003.10551 `[cs.LG]`.

[11] 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.

[12] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. "Causal Transformer for Estimating Counterfactual Outcomes". In: *arXiv preprint arXiv:2204.07258* (2022).

[13] James Robins. “A Graphical Approach to the Identification and Estimation of Causal Parameters in Mortality Studies with Sustained Exposure Periods”. In: *Journal of Chronic Diseases* (1987).

[14] 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”. In: *Mathematical Modelling* (1986).

[15] Hong Xiong et al. *G-Transformer: Counterfactual Outcome Prediction under Dynamic and Time-varying Treatment Regimes*. 2024. arXiv: 2406.05504 [cs.LG]. URL: https://arxiv.org/abs/2406.05504.