Causal Transformer for Estimating Counterfactual Outcomes

Valentyn Melnychuk [1] **Dennis Frauen** [1] **Stefan Feuerriegel** [1]

Abstract

Estimating counterfactual outcomes over time from observational data is relevant for many applications (e.g., personalized medicine). Yet, state-of-the-art methods build upon simple long short-term memory (LSTM) networks, thus rendering inferences for complex, long-range dependencies challenging. In this paper, we develop a novel *Causal Transformer* for estimating counterfactual outcomes over time. Our model is specifically designed to capture complex, long-range dependencies among time-varying confounders. For this, we combine three transformer subnetworks with separate inputs for time-varying covariates, previous treatments, and previous outcomes into a joint network with in-between cross-attentions. We further develop a custom, end-to-end training procedure for our *Causal Transformer*. Specifically, we propose a novel counterfactual domain confusion loss to address confounding bias: it aims to learn adversarial balanced representations, so that they are predictive of the next outcome but non-predictive of the current treatment assignment. We evaluate our *Causal Transformer* based on synthetic and real-world datasets, where it achieves superior performance over current baselines. To the best of our knowledge, this is the first work proposing transformer-based architecture for estimating counterfactual outcomes from longitudinal data.

1. Introduction

Decision-making in medicine requires precise knowledge of individualized health outcomes over time after applying different treatments (Huang & Ning, 2012; Hill & Su, 2013). This then informs the choice of treatment plans and thus ensures effective care personalized to individual patients. Traditionally, the gold standard for estimating the effects of treatments are randomized controlled trials (RCTs). However, RCTs are costly, often impractical, or even unethical. To address this, there is a growing interest in estimating health outcomes over time from observational data, such as, e. g., electronic health records.

Numerous methods have been proposed for estimating (counterfactual) outcomes from observational data in the static setting (van der Laan & Rubin, 2006; Chipman et al., 2010; Johansson et al., 2016; Curth & van der Schaar, 2021; Kuzmanovic et al., 2022). Different from that, we focus on longitudinal settings, that is, *over time*. In fact, longitudinal data are nowadays paramount in medical practice. For example, almost all electronic health records (EHRs) nowadays store sequences of medical events over time (Allam et al., 2021). However, estimating counterfactual outcomes over time is challenging. One reason is that counterfactual outcomes are generally never observed. On top of that, directly estimating counterfactual outcomes with traditional machine learning methods in the presence of (time-varying) confounding has a larger generalization error of estimation (Alaa & van der Schaar, 2018a), or is even biased (in case of multiple-step-ahead prediction) (Robins & Hernán, 2009; Frauen et al., 2022). Instead, tailored methods are needed.

To estimate counterfactual outcomes over time, state-of-the-art methods make nowadays use of machine learning. Prominent examples are: recurrent marginal structural networks (RMSNs) (Lim et al., 2018), counterfactual recurrent network (CRN) (Bica et al., 2020), and G-Net (Li et al., 2021). However, these methods build upon simple long short-term memory (LSTM) networks, because of which their ability to model complex, long-range dependencies in observational data is limited. Long-range dependencies are omnipresent in medical data; e. g., long-term treatment effects have been observed for obesity (Latner et al., 2000), multiple sclerosis (Sormani & Bruzzi, 2015), or diabetes (Jacobson et al., 2013). To address this, we develop a *Causal Transformer* (CT) for estimating counterfactual outcomes over time. It is carefully designed to capture complex, long-range dependencies in medical data that are nowadays common in EHRs.

In this paper, we aim at estimating counterfactual outcomes over time, that is, for one- and multi-step-ahead predictions.

[1] LMU Munich, Munich, Germany. Correspondence to: Valentyn Melnychuk <melnychuk@lmu.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

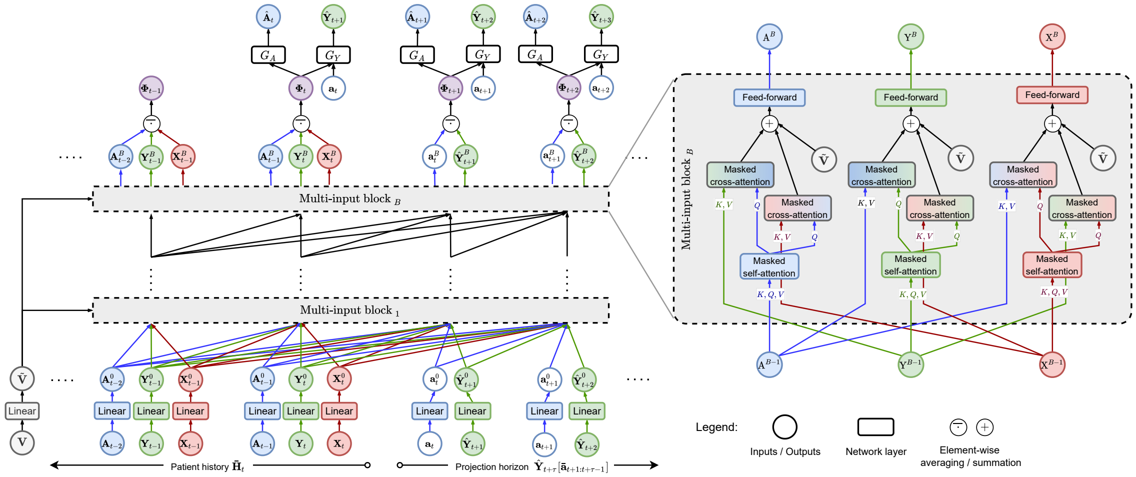


Figure 1. Overview of our CT. We distinguish two timelines: time steps $1, \ldots, t$ refer to observational data (patient trajectories) and thus input; time steps $t+1, \ldots, t+\tau$ is the projection horizon and thus output. Three separate transformers are used in parallel for encoding observational data as input: treatments $\mathbf{A}_t$ / treatment interventions $\mathbf{a}_t$ (blue), outcomes $\mathbf{Y}_t$ / outcome predictions $\hat{\mathbf{Y}}_t$ (green), and time-varying covariates $\mathbf{X}_t$ (red). These are fused via B stacked multi-input blocks. Additional static covariates $\mathbf{V}$ (gray) are fed into all multi-input blocks. Each multi-input block further makes use of cross-attentions. Afterward, the three respective representation for treatments, outcomes, and time-varying covariates are averaged, giving the (balanced) representation $\mathbf{\Phi}_t$ (purple). On top of that are two additional networks G_Y (outcome prediction network) and G_A (treatment classifier network) for learning balanced representations in our CDC loss. Layer normalizations and residual connections are omitted for clarity.

For this, we develop a novel *Causal Transformer* (CT). It combines two innovations: (1) a tailored transformer-based architecture to capture complex, long-range dependencies in the observational data; and (2) a novel counterfactual domain confusion (CDC) loss for end-to-end training.

For (1), we combine three separate transformer subnetworks for processing time-varying covariates, past treatments, and past outcomes, respectively, into a joint network with in-between cross-attentions. Here, each transformer subnetwork is further extended by (i) masked multi-head self-attention, (ii) shared trainable relative positional encoding, and (iii) attentional dropout.

For (2), we develop a custom end-to-end training procedure based on our CDC loss. This allows us to solve an adversarial balancing objective in which we balance representations to be (a) predictive of outcomes and (b) non-predictive of the current treatment assignment. The latter is crucial to address confounding bias and thus reduces the generalization error of counterfactual prediction. Importantly, this objective is different from previously proposed gradient reversal balancing (Ganin & Lempitsky, 2015; Bica et al., 2020), as it aims to minimize a reversed KL-divergence to build balanced representations.

We demonstrate the effectiveness of our CT over state-of-the-art methods using an extensive series of experiments with synthetic and real-world data. Our ablation study (e.g., against a single-subnetwork architecture) shows that neither (1) nor (2) alone are sufficient for learning. Rather, it is crucial to combine our transformer-based architecture based on three subnetworks *and* our novel CDC loss.

Overall, our **main contributions** are as follows:[1]

1. We propose a new end-to-end model for estimating counterfactual outcomes over time: the *Causal Transformer* (CT). To the best of our knowledge, this is the first transformer tailored to causal inference.
2. We develop a custom training procedure for our CT based on a novel counterfactual domain confusion (CDC) loss.
3. We use synthetic and real-world data to demonstrate that our CT achieves state-of-the-art performance. We further achieve this both for one- and multi-step-ahead predictions.

2. Related Work

Estimating counterfactual outcomes in static setting. Extensive literature has focused on estimating counterfactual outcomes (or, analogously, individual treatment effects [ITE]) in static settings (Johansson et al., 2016; Alaa & van der Schaar, 2018b; Wager & Athey, 2018; Yoon et al., 2018; Curth & van der Schaar, 2021). Several works have adapted deep learning for that purpose (Johansson et al., 2016; Yoon et al., 2018). In the static setting, the input is given by cross-sectional data, and, as such, there are *no* time-varying covariates, treatments, and outcomes. However, we are interested in counterfactual outcome estimation over time.

[1] Code is available online: `https://github.com/Valentyn1997/CausalTransformer`

Estimating counterfactual outcomes over time. Methods for estimating time-varying outcomes were originally introduced in epidemiology and make widespread use of simple linear models. Here, the aim is to estimate average (non-individual) effects of time-varying treatments. Examples of such methods include G-computation, marginal structural models (MSMs), and structural nested models (Robins, 1986; Robins et al., 2000; Hernán et al., 2001; Robins & Hernán, 2009). To address the limited expressiveness of linear models, several Bayesian non-parametric methods were proposed (Xu et al., 2016; Schulam & Saria, 2017; Soleimani et al., 2017). However, these make strong assumptions regarding the data generation mechanism, and are not designed for multi-dimensional outcomes as well as static covariates. Other methods build upon recurrent neural networks (Qian et al., 2021; Berrevoets et al., 2021) but these are restricted to single-time treatments or make stronger assumptions for identifiability, which do not hold for our setting (see Appendix B).

There are several methods that build upon the potential outcomes framework (Rubin, 1978; Robins & Hernán, 2009), and, thus, ensure identifiability by making the same assumptions as we do (see Sec. 3). Here, state-of-the-art methods are recurrent marginal structural networks (RMSNs) (Lim et al., 2018), counterfactual recurrent network (CRN) (Bica et al., 2020), and G-Net (Li et al., 2021). These methods address bias due to time-varying confounding in different ways. RMSNs combine two propensity networks and use the predicted inverse probability of treatment weighting (IPTW) scores for training the prediction networks. CRN uses an adversarial objective to produce a sequence of balanced representations, which are simultaneously predictive of the outcome but non-predictive of the current treatment assignment. G-Net aims to predict both outcomes and time-varying covariates, and then performs G-computation for multiple-step-ahead prediction. All of three aforementioned methods are built on top of one/two-layer LSTM encoder-decoder architectures. Because of that, they are limited in their ability to capture long-range, complex dependencies between time-varying confounders (i. e., time-varying covariates, previous treatments, and previous outcomes). However, such complex data are nowadays widespread in medical practice (e. g., EHRs) (Allam et al., 2021), which may impede the performance of the previous methods for real-world medical data. As a remedy, we develop a *deep* transformer network for counterfactual outcomes estimation over time.

Transformers. Transformers refer to deep neural networks for sequential data that typically adopt a custom self-attention mechanism (Vaswani et al., 2017). This makes transformers both flexible and powerful in modeling long-range associative dependencies for sequence-to-sequence tasks. Prominent examples come from natural language processing (e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and GPT-3 (Brown et al., 2020)). Other examples include image understanding tasks (Dosovitskiy et al., 2020), multi-modal tasks (image/video captioning) (Liu et al., 2021), math problem solving (Schlag et al., 2019), and time-series forecasting (Tang & Matteson, 2021; Zhou et al., 2021). However, to the best of our knowledge, no paper has developed transformers specifically for causal inference. This presents our novelty.

3. Problem Formulation

We build upon the standard setting for estimating counterfactual outcomes over time as in (Robins & Hernán, 2009; Lim et al., 2018; Bica et al., 2020; Li et al., 2021). Let i refer to some patient and with health trajectories that span time steps $t = 1, \ldots, T^{(i)}$. For each time step t and each patient i, we have the following: d_x time-varying covariates $\mathbf{X}_t^{(i)} \in \mathbb{R}^{d_x}$; d_a categorical treatments $\mathbf{A}_t^{(i)} \in \{a_1, \ldots, a_{d_a}\}$; and d_y outcomes $\mathbf{Y}_t^{(i)} \in \mathbb{R}^{d_y}$. For example, data from critical care units of COVID-19 patients would involve blood pressure and heart rate as time-varying covariates, ventilation as treatment, and respiratory frequency as outcome. Treatments are modeled as categorical variables as this relates to the question of whether to apply a treatment or not, and is thus consistent with prior works (Lim et al., 2018; Bica et al., 2020; Li et al., 2021). Further, we record static covariates describing a patient $\mathbf{V}^{(i)}$ (e. g., gender, age, or other risk factors). For notation, we omit patient index (i) unless needed.

For learning, we have access to i.i.d. observational data $\mathcal{D} = \Big\{\{\mathbf{x}_t^{(i)}, \mathbf{a}_t^{(i)}, \mathbf{y}_t^{(i)}\}_{t=1}^{T^{(i)}} \cup \mathbf{v}^{(i)}\Big\}_{i=1}^{N}$. In clinical settings, such data are nowadays widely available in form of EHRs (Allam et al., 2021). Here, we summarize the patient trajectory by $\bar{\mathbf{H}}_t = \{\bar{\mathbf{X}}_t, \bar{\mathbf{A}}_{t-1}, \bar{\mathbf{Y}}_t, \mathbf{V}\}$, where $\bar{\mathbf{X}}_t = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$, $\bar{\mathbf{Y}}_t = (\mathbf{Y}_1, \ldots, \mathbf{Y}_t)$, and $\bar{\mathbf{A}}_{t-1} = (\mathbf{A}_1, \ldots, \mathbf{A}_{t-1})$.

We build upon the potential outcomes framework (Neyman, 1923; Rubin, 1978) and its extension to time-varying treatments and outcomes (Robins & Hernán, 2009). Let $\tau \geq 1$ denote projection horizon for a τ-step-ahead prediction. Further, let $\bar{\mathbf{a}}_{t:t+\tau-1} = (\mathbf{a}_t, \ldots, \mathbf{a}_{t+\tau-1})$ denote a given (non-random) treatment intervention. Then, we are interested in the potential outcomes, $\mathbf{Y}_{t+\tau}[\bar{\mathbf{a}}_{t:t+\tau-1}]$, under the treatment intervention. However, the potential outcomes for a specific treatment intervention are typically never observed for a patient but must be estimated. Formally, the potential counterfactual outcomes over time are identifiable from factual observational data $\mathcal{D}$ under three standard assumptions: (1) consistency, (2) sequential ignorability, and (3) sequential overlap (see Appendix A for details).

Our task is thus to estimate future counterfactual outcomes $\mathbf{Y}_{t+\tau}$, after applying a treatment intervention $\bar{\mathbf{a}}_{t:t+\tau-1}$ for a given patient history $\bar{\mathbf{H}}_t$. Formally, we aim to estimate:

$$\mathbb{E}\big(\mathbf{Y}_{t+\tau}[\bar{\mathbf{a}}_{t:t+\tau-1}] \mid \bar{\mathbf{H}}_t\big). \tag{1}$$

To do so, we learn a function $g(\tau, \bar{\mathbf{a}}_{t:t+\tau-1}, \bar{\mathbf{H}}_t)$. Simply estimating $g(\cdot)$ with traditional machine learning is biased (Robins & Hernán, 2009). For example, one reason is that treatment interventions not only influence outcomes but also future covariates. To address this, we develop a tailored model for estimation.

4. Causal Transformer

Input. Our *Causal Transformer* (CT) is a single multi-input architecture, which combines three separate transformer subnetworks. Each subnetwork processes a different sequence as input: (i) past time-varying covariates $\bar{\mathbf{X}}_t$; (ii) past outcomes $\bar{\mathbf{Y}}_t$; and (iii) past treatments before intervention $\bar{\mathbf{A}}_{t-1}$. Since we aim at estimating the counterfactual outcome after treatment intervention, we further input the future treatment assignment that a medical practitioners wants to intervene on. Also, we autoregressively feed predictions of outcomes $\bar{\hat{\mathbf{Y}}}_{t+1:t+\tau-1}$, starting at the intervention time step (prediction origin). Thus, we concatenate two treatment sequences $\bar{\mathbf{A}}_{t-1} \cup \bar{\mathbf{a}}_{t:t+\tau-1}$, and two outcome sequences $\bar{\mathbf{Y}}_t \cup \bar{\hat{\mathbf{Y}}}_{t+1:t+\tau-1}$ for input. Additionally, (iv) the static covariates $\mathbf{V}$ are fed into all subnetworks.

4.1. Model architecture

Our CT yields a sequence of treatment-invariant (balanced) *representations* $\bar{\mathbf{\Phi}}_{t+\tau-1} = (\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_{t+\tau-1})$. To do so, we stack B identical *transformer blocks*. The first transformer block receives the three input sequences. The B-th transformer block outputs a sequence of representations $\bar{\mathbf{\Phi}}_{t+\tau-1}$. The architecture is shown in Fig. 1.

Transformer blocks. Let $b = 1, \ldots, B$ index the different transformer blocks. Each transformer block receives three parallel sequences of hidden states as input (for each of the input sequences). For time step t, we denote the respective hidden state by $\mathbf{A}_t^b$ or $\mathbf{a}_t^b$; $\mathbf{Y}_t^b$ or $\hat{\mathbf{Y}}_t^b$; and $\mathbf{X}_t^b$. We denote size of the hidden states by d_h. Further, each transformer block receives a representation vector of the static covariates $\tilde{\mathbf{V}}$ as additional input.

For the first transformer block ($b = 1$), we use linearly-transformed time-series as input:

$$\begin{aligned} &\mathbf{A}_t^0, \mathbf{a}_t^0 = \mathrm{Linear}_A(\mathbf{A}_t, \mathbf{a}_t), \qquad \mathbf{X}_t^0 = \mathrm{Linear}_X(\mathbf{X}_t), \\ &\mathbf{Y}_t^0, \hat{\mathbf{Y}}_t^0 = \mathrm{Linear}_Y(\mathbf{Y}_t, \hat{\mathbf{Y}}_t), \qquad \tilde{\mathbf{V}} = \mathrm{Linear}_V(\mathbf{V}), \end{aligned} \tag{2}$$

where parameters of fully-connected linear layers are shared for all time steps. All blocks ≥ 2 use the output sequence of the previous block $b-1$ as inputs. For notation, we denote sequences of hidden states after block b by three tensors $\mathrm{A}^b = \big(\bar{\mathbf{A}}_{t-1}^b \cup \bar{\mathbf{a}}_{t:t+\tau-1}^b\big)^\top, \mathrm{X}^b = \big(\bar{\mathbf{X}}_t^b\big)^\top$, and $\mathrm{Y}^b = \big(\bar{\mathbf{Y}}_t^b \cup \bar{\hat{\mathbf{Y}}}_{t+1:t+\tau-1}^b\big),^\top$.

Following (Dong et al., 2021; Lu et al., 2021), each transformer block combines a (i) multi-head self-/cross-attention, (ii) feed-forward layer, and (iii) layer normalization. Details are in Appendix C.

(i) Multi-head self-/cross-attention uses a scaled dot-product attention with several parallel attention heads. Each attention head requires a 3-tuple of keys, queries, and values, i. e., $K, Q, V \in \mathbb{R}^{T \times d_{qkv}}$, respectively. These are obtained from a sequence of hidden states $\mathrm{H}^b = \big(\mathbf{h}_1^b, \ldots, \mathbf{h}_t^b\big)^\top \in \mathbb{R}^{T \times d_h}$ (H^b is one of A^b, X^b or Y^b, depending on the subnetwork). Formally, we compute

$$\mathrm{Attn}^{(i)}(Q^{(i)}, K^{(i)}, V^{(i)}) = \mathrm{softmax}\Big(\frac{Q^{(i)} K^{(i)\top}}{\sqrt{d_{qkv}}}\Big) V^{(i)}, \tag{3}$$

$$Q^{(i)} = Q^{(i)}(\mathrm{H}^b) = \mathrm{H}^b\, W_Q^{(i)} + \mathbf{1} b_Q^{(i)\top}, \tag{4}$$

$$K^{(i)} = K^{(i)}(\mathrm{H}^b) = \mathrm{H}^b\, W_K^{(i)} + \mathbf{1} b_K^{(i)\top}, \tag{5}$$

$$V^{(i)} = V^{(i)}(\mathrm{H}^b) = \mathrm{H}^b\, W_V^{(i)} + \mathbf{1} b_V^{(i)\top}, \tag{6}$$

where $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)} \in \mathbb{R}^{d_h \times d_{qkv}}$ and $b_Q^{(i)}, b_Q^{(i)}, b_V^{(i)} \in \mathbb{R}^{d_{qkv}}$ are parameters of a single attention head i, where $\mathrm{softmax}(\cdot)$ operates separately on each row, and where $\mathbf{1} \in \mathbb{R}^{d_{qkv}}$ is a vector of ones. We set the dimensionality of keys and queries to $d_{qkv} = d_h / n_h$, where n_h is the number of heads.

The output of a multi-head attention is a concatenation of the different heads, i. e.,

$$\mathrm{MHA}(Q, K, V) = \mathrm{Concat}(\mathrm{Attn}^{(1)}, \ldots, \mathrm{Attn}^{(n_h)}). \tag{7}$$

Here, we simplified the original multi-head attention in (Vaswani et al., 2017) by omitting the final output projection layer after concatenation to reduce risk of overfitting.

In our CT, self-attention uses the sequence of hidden states from the same transformer subnetwork to infer keys, queries, and values, while cross-attention uses the sequence of hidden states of the other two transformer subnetworks as keys and values. We use multiple cross-attentions to exchange the information between parallel hidden states.[2] These are placed on top of the self-attention layers (see subdiagram in Fig. 1). We add the representation vector of static covariates, $\tilde{\mathbf{V}}$ when pooling different cross-attention outputs. We mask hidden states for self- and cross-attentions by setting the

[2] Different variants of combining multiple-input information with self- and cross-attentions were already studied in the context of multi-source translation, e.g., in (Libovický et al., 2018). Our implementation is closest to parallel attention combination.

attention logits in Eq. (3) to $-\infty$. This ensures that information flows only from the current input to future hidden states (and not the other way around).

(ii) Feed-forward layer (FF) with ReLU activation is applied time-step-wise to the sequence of hidden states, i. e.,

$$\mathrm{FF}(\mathbf{h}_t) = \mathrm{Linear}\left(\mathrm{ReLU}(\mathrm{Linear}(\mathbf{h}_t))\right),$$

where fully-connected linear layers are followed by dropout.

(iii) Layer normalization (LN) (Ba et al., 2016) and residual connections are added after each self- and cross-attention. We compute the layer normalization via

$$\mathrm{LN}(\mathbf{h}_t) = \frac{\gamma}{\sigma} \odot (\mathbf{h}_t - \mu) + \beta, \tag{8}$$

$$\mu = \frac{1}{d_h} \sum_{j=1}^{d_h} (\mathbf{h}_t)_j, \quad \sigma = \sqrt{\frac{1}{d_h} \sum_{j=1}^{d_h} \left((\mathbf{h}_t)_j - \mu\right)^2}, \tag{9}$$

where $\gamma, \beta \in \mathbb{R}^{d_h}$ are scale and shift parameters and where $\odot$ is an element-wise product.

Balanced representations. The (balanced) representations are then constructed via average pooling over three (or two) parallel hidden states of the B-th transformer block. Thereby, we use a fully-connected linear layer and an exponential linear unit (ELU) non-linearity; i. e.,

$$\tilde{\mathbf{\Phi}}_i = \begin{cases} \frac{1}{3}(\mathbf{A}^B_{i-1} + \mathbf{X}^B_i + \mathbf{Y}^B_i), & i \in \{1, \ldots, t\}, \\ \frac{1}{2}(\mathbf{a}^B_{i-1} + \hat{\mathbf{Y}}^B_i), & i \in \{t+1, \ldots, t+\tau-1\}, \end{cases}$$

$$\mathbf{\Phi}_t = \mathrm{ELU}(\mathrm{Linear}(\tilde{\mathbf{\Phi}}_t)) \tag{10}$$

where fully-connected linear layer is followed by dropout, $\mathbf{\Phi}_t \in \mathbb{R}^{d_r}$ and d_r is the dimensionality of the balanced representation.

4.2. Positional encoding

In order to preserve information about the order of hidden states, we make use of position encoding (PE). This is especially relevant for clinical practice as it allows us to distinguish sequences such as, e. g., ⟨treatment A ↦ side-effect S ↦ treatment B⟩ from ⟨treatment A ↦ treatment B ↦ side-effect S⟩.

We model information about relative positions in the input at time steps j and i with $0 \le j \le i \le t$ by a set of vectors $a^V_{ij}, a^K_{ij} \in \mathbb{R}^{d_{qkv}}$ (Shaw et al., 2018). Specifically, they are shaped in the form of Toeplitz matrices

$$a^V_{ij} = w^V_{\mathrm{clip}(j-i, l_{\max})}, \qquad a^K_{ij} = w^K_{\mathrm{clip}(j-i, l_{\max})}, \tag{11}$$

$$\mathrm{clip}(x, l_{\max}) = \max\{-l_{\max}, \min\{l_{\max}, x\}\} \tag{12}$$

with trainable weights $w^K_l, w^V_l \in \mathbb{R}^{d_{qkv}}$, for $l \in \{-l_{\max}, \ldots, 0\}$, and where $l_{\max}$ is the maximum distinguishable distance in the relative PE. The above formalization ensures that we obtain *relative* encodings, that is, our CT considers the distance between past or current position j and current position i, but not the actual location. Furthermore, the current position i attends only to past information or itself, and, thus, we never use a^V_{ij} and a^K_{ij} where $i < j$. As a result, there are only $(l_{\max} + 1) \times d_{qkv}$ parameters to estimate.

We then use the relative PE to modify the self-attention operation (Eq. (3)). Formally, we compute the attention scores via (indices of heads are dropped for clarity)

$$(\mathrm{Attn}(Q, K, V))_i = \sum_{j=1}^{t} \alpha_{ij} (V_j + a^V_{ij}), \tag{13}$$

$$\alpha_{ij} = \mathrm{softmax}_j \left(\frac{Q_i^\top (K_j + a^K_{ij})}{\sqrt{d_{qkv}}} \right), \tag{14}$$

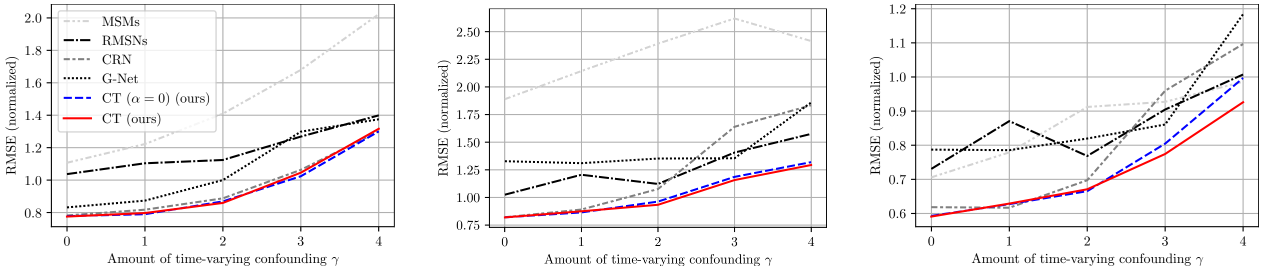
with attention scores α_{ij} and where K_j, V_j, and Q_i are columns of corresponding matrices and where $\mathrm{softmax}_j$ operates with respect to index j. Cross-attention with PE is defined in an analogous way. In our CT, the attention scores are shared across all the heads and blocks, as well as the three different subnetworks.

In our CT, we use relative positional encodings (Shaw et al., 2018) that are incorporated in every self- and cross-attention. This is different from the original transformer (Vaswani et al., 2017), which used absolute positional encodings with fixed weights for the initial hidden states of the first transformer block (see Appendix D for details). However, relative PE is regarded as more robust and, further, suited for patient trajectories where the order of treatments and diagnoses is informative (Allam et al., 2021), but not the absolute time step. Additionally, it allows for better generalization to unseen sequence lengths: for the ranges beyond the maximal distinguishable distance $l_{\max}$, CT stops to distinguish the precise relative location of states and considers everything as distant past information. In line with this, our experiments later also confirm relative PE to be superior over absolute PE.

4.3. Training of our *Causal Transformer*

In our CT, we aim at two simultaneous objectives to address confounding bias: we aim at learning representations that are (a) predictive of the next outcome and (b) are non-predictive of the current treatment assignment. This thus naturally yields an adversarial objective. For this purpose, we make use of balanced representations, which we train via a novel *counterfactual domain confusion (CDC) loss.*

Adversarial balanced representations. As in (Bica et al., 2020), we build *balanced* representations that allow us to achieve the adversarial objectives (a) and (b). For this, we put two fully-connected networks on top of the represen-



(a) One-step-ahead prediction

(b) τ-step-ahead prediction (single sliding treatment).

(c) τ-step-ahead prediction (random trajectories)

Figure 2. Results for fully-synthetic data based on tumor growth simulator (lower values are better). Shown is the mean performance averaged over five runs with different seeds. Here: $\tau = 6$.

Table 1. Results for semi-synthetic data for τ-step-ahead prediction based on real-world medical data (MIMIC-III). Shown: RMSE as mean ± standard deviation over five runs. Here: random trajectory setting. MSMs struggle for long prediction horizons with values $>$ 10.0 (due to linear modeling of IPTW scores).

	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 6$	$\tau = 7$	$\tau = 8$	$\tau = 9$	$\tau = 10$
MSMs (Robins et al., 2000)	0.37 ± 0.01	0.57 ± 0.03	0.74 ± 0.06	0.88 ± 0.03	1.14 ± 0.10	1.95 ± 1.48	3.44 ± 4.57	$>$ 10.0	$>$ 10.0	$>$ 10.0
RMSNs (Lim et al., 2018)	0.24 ± 0.01	0.47 ± 0.01	0.60 ± 0.01	0.70 ± 0.02	0.78 ± 0.04	0.84 ± 0.05	0.89 ± 0.06	0.94 ± 0.08	0.97 ± 0.09	1.00 ± 0.11
CRN (Bica et al., 2020)	0.30 ± 0.01	0.48 ± 0.02	0.59 ± 0.02	0.65 ± 0.02	0.68 ± 0.02	0.71 ± 0.01	0.72 ± 0.01	0.74 ± 0.01	0.76 ± 0.01	0.78 ± 0.02
G-Net (Li et al., 2021)	0.34 ± 0.01	0.67 ± 0.03	0.83 ± 0.04	0.94 ± 0.04	1.03 ± 0.05	1.10 ± 0.05	1.16 ± 0.05	1.21 ± 0.06	1.25 ± 0.06	1.29 ± 0.06
EDCT w/ GR ($\lambda = 1$) (*ours*)	0.29 ± 0.01	0.46 ± 0.01	0.56 ± 0.01	0.62 ± 0.01	0.67 ± 0.01	0.70 ± 0.01	0.72 ± 0.01	0.74 ± 0.01	0.76 ± 0.01	0.78 ± 0.01
CT ($\alpha = 0$) (*ours*) *	**0.20 ± 0.01**	**0.38 ± 0.01**	**0.45 ± 0.01**	0.50 ± 0.02	**0.52 ± 0.02**	0.55 ± 0.02	0.56 ± 0.02	0.58 ± 0.02	0.60 ± 0.02	0.61 ± 0.02
CT (*ours*)	**0.20 ± 0.01**	**0.38 ± 0.01**	**0.45 ± 0.01**	**0.49 ± 0.01**	**0.52 ± 0.02**	**0.53 ± 0.02**	**0.55 ± 0.02**	**0.56 ± 0.02**	**0.58 ± 0.02**	**0.59 ± 0.02**

Lower = better (best in bold)
* Identical hyperparameters as proposed CT for comparability

tation $\mathbf{\Phi}_t$, corresponding to the respective objectives: (a) an outcome prediction network G_Y and (b) a treatment classifier network G_A. Both receive the representation $\mathbf{\Phi}_t$ as input; the outcome prediction network additionally receives the current treatment $\mathbf{A}_t$. We implement both as single hidden layer fully-connected networks with number of units n_{FC} and ELU activation. For notation, let θ_Y and θ_A denote the trainable parameters in G_Y and G_A, respectively. Further, let θ_R denote all trainable parameters in CT for generating the representation $\mathbf{\Phi}_t$.

Factual outcome loss. For objective (a), we fit the outcome prediction network G_Y, and thus $\mathbf{\Phi}_t$, by minimizing the factual loss of the next outcome. This can be done, e. g., via the mean squared error (MSE). We then yield

$$\mathcal{L}_{G_Y}(\theta_Y, \theta_R) = \left\| \mathbf{Y}_{t+1} - G_Y\left(\mathbf{\Phi}_t(\theta_R), \mathbf{A}_t; \theta_Y\right) \right\|^2 . \tag{15}$$

CDC loss. For objective (b), we want to fit the treatment classifier network G_A, and thus the representation $\mathbf{\Phi}_t$, in way that it is non-predictive of the current treatment $\mathbf{A}_t$. To achieve this, we develop a novel CDC loss tailored for counterfactual inference. Our idea builds upon the domain confusion loss (Tzeng et al., 2015) for handling adversarial objectives, which was previously used for unsupervised domain adaptation, whereas we adapt it specifically for counterfactual inference.

Then, we fit G_A so that it can predict the current treatment, i. e., via

$$\mathcal{L}_{G_A}(\theta_A, \theta_R) = -\sum_{j=1}^{d_a} \mathbb{1}_{[\mathbf{A}_t = a_j]} \log G_A(\mathbf{\Phi}_t(\theta_R); \theta_A), \tag{16}$$

where $\mathbb{1}_{[\cdot]}$ is the indicator function. This thus minimizes a classification loss of the current treatment assignment given $\mathbf{\Phi}_t$. However, while G_A can predict the current treatment, the actual representation $\mathbf{\Phi}_t$ should not, and should rather be non-predictive. For this, we propose to minimize the cross-entropy between a uniform distribution over treatment categorical space and predictions of G_A via

$$\mathcal{L}_{\text{conf}}(\theta_A, \theta_R) = -\sum_{j=1}^{d_a} \frac{1}{d_a} \log G_A(\mathbf{\Phi}_t(\theta_R); \theta_A), \tag{17}$$

thus achieving domain confusion.

Overall adversarial objective. Using the above, CT is trained via

$$(\hat{\theta}_Y, \hat{\theta}_R) = \underset{\theta_Y, \theta_R}{\arg\min}\, \mathcal{L}_{G_Y}(\theta_Y, \theta_R) + \alpha \mathcal{L}_{\text{conf}}(\hat{\theta}_A, \theta_R), \tag{18}$$

$$\hat{\theta}_A = \underset{\theta_A}{\arg\min}\, \alpha \mathcal{L}_{G_A}(\theta_A, \hat{\theta}_R), \tag{19}$$

where α is a hyperparameter for domain confusion. Thereby, optimal values of $\hat{\theta}_Y$, $\hat{\theta}_R$ and $\hat{\theta}_A$ achieve an equilibrium between factual outcome prediction and domain confusion. In CT, we implement this by performing iterative updates of the parameters (rather than optimizing globally). Details are in Appendix E.

Previous work (Bica et al., 2020) has addressed the above adversarial objective through gradient reversal (Ganin & Lempitsky, 2015). However, this has two shortcomings: (i) If the parameter λ of gradient reversal becomes too large, the representation may be predictive of opposite treatment (Atan et al., 2018a). (ii) If the treatment classifier network learns too fast, gradients vanish and are not passed to representations, leading to poor fit (Tzeng et al., 2017). Different from that, we propose a novel CDC loss. As we see later, our loss is highly effective: it even improves CRN (Bica et al., 2020), when replacing gradient reversal with our loss.

Stabilization. We further stabilize the above adversarial training by employing exponential moving average (EMA) of model parameters during training (Yaz et al., 2018). EMA helps to limit cycles of model parameters around the equilibrium with vanishing amplitude and thus accelerates overall convergence. We apply EMA to all trainable parameters (i. e., θ_Y, θ_R, θ_A). Formally, we update parameters during training via

$$\theta^{(i)}_{\text{EMA}} = \beta\, \theta^{(i-1)}_{\text{EMA}} + (1-\beta)\, \theta^{(i)}, \tag{20}$$

where superscripts (i) refers to the different steps of the optimization algorithm, where β is a exponential smoothing parameter, and where we initialize $\theta^{(0)}_{\text{EMA}} = \theta^{(0)}$. We provide pseudocode for an iterative gradient update in CT via EMA in Appendix E.

Attentional dropout. To reduce the risk of overfitting between time steps, we implement attentional dropout via DropAttention (Zehui et al., 2019). During training, attention scores α_{ij} in Eq. (14) are element-wise randomly set to zero with probability p (i. e., the dropout rate). However, we make a small simplification. We do not perform normalized rescaling (Zehui et al., 2019) of attention scores but opt for traditional dropout rescaling (Srivastava et al., 2014), as this resulted in more stable training for short-length sequences.

Mini-batch augmentation with masking. For training data $\mathcal{D}$, we always have access to the full time-series, that is, including all time-varying covariates $\mathbf{x}^{(i)}_1, \ldots, \mathbf{x}^{(i)}_{T^{(i)}}$. However, upon deployment, these are no longer observable for τ-step-ahead predictions with $\tau \geq 2$. To reflect this during training, we perform data augmentation at the mini-batch level. For this, we duplicate the training samples: We uniformly sample the length $1 \leq t_s \leq T^{(i)}$ of the masking window, and then create a duplicate data sample where the last t_s time-varying covariates $\mathbf{x}^{(i)}_{t_s}, \ldots, \mathbf{x}^{(i)}_{T^{(i)}}$ are masked by setting the corresponding attention logits of $\mathrm{H}^b = \mathrm{X}^b$ in Eq. (3) to $-\infty$.

Mini-batch augmentation with masking allows us to train a single model for both one- and multiple-step-ahead prediction in end-to-end fashion. This distinguishes our CT from RMSNs and CRN, which are built on top of encoder-decoder architectures and trained in a multiple-stage procedure. Later, we also experiment with an encoder-decoder version of CT (i.e., a single-subnetwork variant) but find that it is inferior performance to our end-to-end model.

4.4. Theoretical insights

The following result provides a theoretical justification that our CDC loss indeed leads to balanced representations, and, thus, removes the bias induced by time-varying confounders.[3]

Theorem 4.1. *We fix $t \in \mathbb{N}$ and define P as the distribution of $\bar{\mathbf{H}}_t$, P_j as the distribution of $\bar{\mathbf{H}}_t$ given $\mathbf{A}_t = a_j$, and P^{Φ}_j as the distribution of $\mathbf{\Phi}_t = \Phi(\bar{\mathbf{H}}_t)$ given $\mathbf{A}_t = a_j$ for all $j \in \{1, \ldots, d_a\}$. Here, $\Phi(\cdot) = \Phi(\cdot; \theta_R)$ denotes any network that generates representations. Let G^j_A denote the output of G_A corresponding to treatment a_j. Then, there exists an optimal pair (Φ^*, G^*_A) such that*

$$\Phi^* = \arg\max_{\Phi} \sum_{j=1}^{d_a} \mathbb{E}_{\bar{\mathbf{H}}_t \sim P}\left[\log {G^*}^j_A(\Phi(\bar{\mathbf{H}}_t)\right] \tag{21}$$

$$G^*_A = \arg\max_{G_A} \sum_{j=1}^{d_a} \mathbb{E}_{\bar{\mathbf{H}}_t \sim P_j}\left[\log G^j_A(\Phi^*(\bar{\mathbf{H}}_t)\right] \mathbb{P}(\mathbf{A}_t = a_j) \tag{22}$$

$$\textit{subject to} \sum_{i=1}^{d_a} G^i_A(\Phi^*(\bar{\mathbf{H}}_t)) = 1. \tag{23}$$

Furthermore, Φ^* satisfies Eq. (21) if and only if it induces balanced representations across treatments, i.e., $P^{\Phi^*}_1 = \ldots = P^{\Phi^*}_{d_a}$.

Proof. See Appendix F. □

Further, it can be easily shown that objectives (16) and (17) are exactly finite sample versions of (22) and (21) from Theorem 4.1, respectively.

4.5. Implementation

Training. We implemented CT in PyTorch Lightning. We trained CT using Adam (Kingma & Ba, 2015) with learning rate η and number of epochs n_e. The dropout rate p was

[3] Importantly, our loss is different from gradient reversal (GR) in (Ganin & Lempitsky, 2015; Bica et al., 2020). It builds balanced representations by minimizing *reversed KL divergence* between the treatment-conditional distribution of representation and mixture of all treatment-conditional distributions.

Table 2. Results for experiments with real-world medical data (MIMIC-III). Shown: RMSE as mean ± standard deviation over five runs.

	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$
MSMs	6.37 ± 0.26	9.06 ± 0.41	11.89 ± 1.28	13.12 ± 1.25	14.44 ± 1.12
RMSNs	5.20 ± 0.15	9.79 ± 0.31	10.52 ± 0.39	11.09 ± 0.49	11.64 ± 0.62
CRN	4.84 ± 0.08	9.15 ± 0.16	9.81 ± 0.17	10.15 ± 0.19	10.40 ± 0.21
G-Net	5.13 ± 0.05	11.88 ± 0.20	12.91 ± 0.26	13.57 ± 0.30	14.08 ± 0.31
CT (*ours*)	**4.59 ± 0.09**	**8.99 ± 0.21**	**9.59 ± 0.22**	**9.91 ± 0.26**	**10.14 ± 0.29**

Lower = better (best in bold)

Table 3. Ablation study for proposed CT (with CDC loss, $\alpha = 0.01, \beta = 0.99$). Reported: normalized RMSE of CT with relative changes.

		$\tau = 1$		$\tau = 6$	
		$\gamma = 1$	$\gamma = 4$	$\gamma = 1$	$\gamma = 4$
CT (proposed)		0.80	1.32	0.63	0.93
a	w/ non-trainable PE*	±0.00	−0.02	+0.01	−0.03
	w/ absolute PE*	+0.04	+0.16	+0.15	+1.00
	w/o attentional dropout*	±0.00	+0.07	+0.00	+0.09
	w/o cross-attention*	+0.03	+0.16	+0.06	+0.10
b	w/o EMA ($\beta = 0$)*	+0.03	+0.38	+0.03	+0.33
	w/o balancing ($\alpha = 0$; $\beta = 0.99$)*	−0.01	−0.02	±0.00	+0.07
	w/ GR ($\lambda = 1$)	+0.02	+0.17	+0.08	+0.33
c	EDCT w/ GR ($\lambda = 1$)	+0.16	+0.08	+0.05	+0.23
	EDCT w/ DC ($\alpha = 0.01$; $\beta = 0.99$)	−0.03	+0.10	−0.03	+0.23

Lower = better;
Improvement over CT in green, worse performance in red
* Identical hyperparameters as proposed CT for comparability

kept the same for both feed-forward layers and DropAttention (we call it sequential dropout rate). We employed the teacher forcing technique (Williams & Zipser, 1989). During evaluation of multiple-step-ahead prediction, we switch off teacher forcing and autoregressively feed model predictions. For the parameters α and β of adversarial training, we choose values $\beta = 0.99$ and $\alpha = 0.01$ as in the original works (Tzeng et al., 2015; Yaz et al., 2018), which also performed well in our experiments. We additionally perform an exponential rise of α during training.

Hyperparameter tuning. p, η, and all other hyperparameters (number of blocks B, minibatch size, number of attention heads n_h, size of hidden units d_h, size of balanced representation d_r, size of hidden units in fully-connected networks n_{FC}) are subject to hyperparameter tuning. Details are in Appendix H.

5. Experiments

To demonstrate the effectiveness of our CT, we make use of synthetic datasets. Thereby, we follow common practice in benchmarking for counterfactual inference (Lim et al., 2018; Bica et al., 2020; Li et al., 2021). For real datasets, the true counterfactual outcomes are typically unknown. By using (semi-)synthetic datasets, we can compute the true counterfactuals and thus validate our CT.

Baselines. The chosen baselines are identical to those in previous, state-of-the-art literature for estimating counterfactual outcomes over time (Lim et al., 2018; Bica et al., 2020; Li et al., 2021). These are: **MSMs** (Robins et al., 2000; Hernán et al., 2001), **RMSNs** (Lim et al., 2018), **CRN** (Bica et al., 2020), and **G-Net** (Li et al., 2021). Details are in Appendix G. For comparability, we use the same hyperparameter tuning for the baselines as for CT (see Appendix H).

5.1. Experiments with fully-synthetic data

Data. We build upon the pharmacokinetic-pharmacodynamic model of tumor growth (Geng et al., 2017). It provides a state-of-the-art biomedical model to simulate the effects of lung cancer treatments over time. The same model was previously used for evaluating RMSNs (Lim et al., 2018) and CRN (Bica et al., 2020). For τ-step-ahead prediction, we distinguish two settings: (i) "single sliding treatment" where trajectories involve only a single treatment as in (Bica et al., 2020); and (ii) "random trajectories" where one or more treatments are assigned. We simulate patient trajectories for different amounts of confounding γ. Further details are in Appendix J. Here, and in all following experiments, we apply hyperparameter tuning (see Appendix H).

Results. Fig. 2 shows the results. We see a notable performance gain for our CT over the state-of-the-art baselines, especially pronounced for larger confounding γ and larger τ. Overall, CT is superior by a large margin.

Fig. 2 also shows a CT variant in which we removed the CDC loss by setting α to zero, called CT ($\alpha = 0$). For comparability, we keep the hyperparameters as in the original CT. The results demonstrate the effectiveness of the proposed CDC loss, especially for multi-step-ahead prediction. CT also provides a significant runtime speedup in comparison to other neural network methods, mainly due to faster processing of sequential data with self- and cross-attentions, and single-stage end-to-end training (see exact runtime and model size comparison in Appendix M). We plotted t-SNE embeddings of the balanced representations (Appendix N) to exemplify how balancing works.

5.2. Experiments with semi-synthetic data

Data. We create a semi-synthetic dataset based on real-world medical data from intensive care units. This allows us to validate our CT with high-dimensional, long-range patient trajectories. For this, we use the MIMIC-III dataset (Johnson et al., 2016). Building upon the ideas of (Schulam & Saria, 2017), we then generate patient trajectories with outcomes under endogeneous and exogeneous dependencies while considering treatment effects. Thereby, we can again control for the amount of confounding. Details are in Appendix K. Importantly, we again have access to the ground-truth counterfactuals for evaluation.

Table 4. CRN with different training procedures. Results for fully-synthetic data based on tumor growth simulator (here: $\gamma = 4$).

	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 6$
CRN + original GR ($\lambda = 1$) as in (Bica et al., 2020)	**1.30 ± 0.14**	**1.12 ± 0.25**	1.23 ± 0.32	1.23 ± 0.34	1.17 ± 0.34	1.10 ± 0.32
CRN + our counterfactual DC loss ($\alpha = 0.01$; $\beta = 0.99$)	1.33 ± 0.21	1.18 ± 0.31	**1.19 ± 0.36**	**1.12 ± 0.35**	**1.03 ± 0.33**	**0.93 ± 0.31**

Lower = better (best in bold)

Results. Table 1 shows the results. Again, CT has a consistent and large improvement across all projection horizons τ (average improvement over baselines: 38.5%). By comparing our CT against CT ($\alpha = 0$), we see clear performance gains, demonstrating the benefit of our CDC loss. Additionally, we separately fitted an encoder-decoder architecture, namely *Encoder-Decoder Causal Transformer* (EDCT). This approach leverages a single-subnetwork architecture, where all three sequences are fed into a single subnetwork (as opposed to three separate networks as in our CT). Further, the EDCT leverages the existing GR loss from (Bica et al., 2020) and the similar encoder-decoder two-stage training. Details on this EDCT model are in Appendix I. Here, we find that, for superior performance, it is crucial to combine both three-subnetwork architecture and our CDC loss.

Semi-synthetic data is also used for a case study, where we study the importance of each subnetwork. See Appendix O.

5.3. Experiments with real-world data

Data. We now demonstrate the applicability of our CT to real-world data and, for this, use intensive care unit stays in MIMIC-III (Johnson et al., 2016). We use the same 25 vital signs and 3 static features. We use (diastolic) blood pressure as an outcome and consider two treatments: vasopressors and mechanical ventilation, similar to (Kuzmanovic et al., 2021; Hatt & Feuerriegel, 2021). Prediction of blood pressure is crucial for critical care, e. g., to avoid tissue hypoperfusion (Vincent et al., 2018). The application of vasopressors is highly confounded by previous and current levels of blood pressure, as they aim to raise low blood pressure. So far, an optimal administration of vasopressors is not fully understood (Subramanian et al., 2008), and, hence, it is important for medical practitioners to have individualized counterfactual predictions. Experiment details are in Appendix L.

Results. Because we no longer have access to the true counterfactuals, we now report the performance of predicting factual outcomes; see Table 2. All state-of-the-art baselines are outperformed by our CT. This demonstrates the superiority of our proposed model.

5.4. Ablation study

We performed an extensive ablation study (Table 3) using full-synthetic data (setting: random trajectories) to confirm the effectiveness of the different components inside the subnetworks, the CDC loss, and the subnetwork architecture. We grouped these into categories. a varies different components within the subnetworks. Here, we replace trainable relative positional encoding (PE) with non-trainable relative PE, generated as described in Appendix D. Further, we replace our PE with a trainable absolute PE as in the original transformer (Vaswani et al., 2017). Finally, we remove attentional dropout as well as cross-attention layers for all subnetworks. b varies the loss. Here, we remove EMA of model weights; switch off adversarial balancing, but not EMA; and replace our CDC loss with gradient reversal (GR) as in (Bica et al., 2020). c evaluates a single-subnetwork version of CT. We refer to this as EDCT (see Appendix I for details). It thus has an encoder-decoder architecture which we train with either our CDC loss or GR.

Overall, we see that the combination of both our novel architecture based three-subnetworks and our novel DC loss is crucial. This observation is particularly pronounced for a long prediction horizon ($\tau = 6$), where our proposed CT achieves the best performance. Notably, the main insight here is: simply switching the backbone from LSTM to transformer and using gradient reversal as in (Bica et al., 2020) gives unstable results (see "EDCT w/ GR (λ = 1)"). Furthermore, our three-subnetworks CT with GR loss performs even worse (see ablation "w/ GR (λ = 1)").

To further demonstrate the effectiveness of our novel CDC loss, we perform an additional test based on the fully-synthetic dataset (Table 4). We use (i) a CRN with GR as in (Bica et al., 2020). We compare it with (ii) a CRN trained with our proposed CDC loss (implementation details in Appendix G). Evidently, our loss also helps the CRN to achieve a better RMSE.

6. Conclusion

For personalized medicine, estimates of the counterfactual outcomes for patient trajectories are needed. Here, we proposed a novel, state-of-the-art method: the *Causal Transformer* which is designed to capture complex, long-range patient trajectories. It combines a custom subnetwork architecture to process the input together with a new counterfactual domain confusion loss for end-to-end training. Across extensive experiments, our *Causal Transformer* achieves state-of-the-art performance.

References

Alaa, A. and van der Schaar, M. Limits of estimating heterogeneous treatment effects: Guidelines for practical algorithm design. In *International Conference on Machine Learning*, 2018a.

Alaa, A. M. and van der Schaar, M. Bayesian nonparametric causal inference: Information rates and learning algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 12(5):1031–1046, 2018b.

Allam, A., Feuerriegel, S., Rebhan, M., Krauthammer, M., et al. Analyzing patient trajectories with artificial intelligence. *Journal of Medical Internet Research*, 23(12): e29812, 2021.

Atan, O., Zame, W. R., and van der Schaar, M. Counterfactual policy optimization using domain-adversarial neural networks. In *ICML CausalML Workshop*, 2018a.

Atan, O., Zame, W. R., and van der Schaar, M. Learning optimal policies from observational data. *arXiv preprint arXiv:1802.08679*, 2018b.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Berrevoets, J., Curth, A., Bica, I., McKinney, E., and van der Schaar, M. Disentangled counterfactual recurrent networks for treatment effect inference over time. *arXiv preprint arXiv:2112.03811*, 2021.

Bica, I., Alaa, A. M., Jordon, J., and van der Schaar, M. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Chipman, H. A., George, E. I., and McCulloch, R. E. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Curth, A. and van der Schaar, M. Nonparametric estimation of heterogeneous treatment effects: From theory to learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Dong, Y., Cordonnier, J.-B., and Loukas, A. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Frauen, D., Hatt, T., Melnychuk, V., and Feuerriegel, S. Estimating average causal effects from patient trajectories. *arXiv preprint arXiv:2203.01228*, 2022.

Ganin, Y. and Lempitsky, V. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, 2015.

Geng, C., Paganetti, H., and Grassberger, C. Prediction of treatment response for combined chemo-and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model. *Scientific Reports*, 7(1):Article 13542, 2017.

Hatt, T. and Feuerriegel, S. Sequential deconfounding for causal inference with unobserved confounders. *arXiv preprint arXiv:2104.09323*, 2021.

Hensman, J., Durrande, N., Solin, A., et al. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(1):5537–5588, 2017.

Hernán, M. A., Brumback, B., and Robins, J. M. Marginal structural models to estimate the joint causal effect of nonrandomized treatments. *Journal of the American Statistical Association*, 96(454):440–448, 2001.

Hill, J. and Su, Y.-S. Assessing lack of common support in causal inference using Bayesian nonparametrics: Implications for evaluating the effect of breastfeeding on children's cognitive outcomes. *The Annals of Applied Statistics*, 7:1386–1420, 2013.

Huang, X. and Ning, J. Analysis of multi-stage treatments for recurrent diseases. *Statistics in Medicine*, 31(24): 2805–2821, 2012.

Jacobson, A. M., Braffett, B. H., Cleary, P. A., Gubitosi-Klug, R. A., Larkin, M. E., and research group, D. The long-term effects of type 1 diabetes treatment and complications on health-related quality of life: a 23-year follow-up of the diabetes control and complications/epidemiology of diabetes interventions and complications cohort. *Diabetes Care*, 36(10):3131–3138, 2013.

Johansson, F., Shalit, U., and Sontag, D. Learning representations for counterfactual inference. In *International Conference on Machine Learning*, 2016.

Johnson, A. E., Pollard, T. J., Shen, L., Li-Wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. MIMIC-III: A freely accessible critical care database. *Scientific Data*, 3(1):Article 160035, 2016.

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

Kuzmanovic, M., Hatt, T., and Feuerriegel, S. Deconfounding temporal autoencoder: estimating treatment effects over time using noisy proxies. In *Machine Learning for Health*, 2021.

Kuzmanovic, M., Hatt, T., and Feuerriegel, S. Estimating conditional average treatment effects with missing treatment information. *arXiv preprint arXiv:2203.01422*, 2022.

Latner, J., Stunkard, A., Wilson, G., Jackson, M., Zelitch, D., and Labouvie, E. Effective long-term treatment of obesity: a continuing care model. *International Journal of Obesity*, 24(7):893–898, 2000.

Li, R., Shahn, Z., Li, J., Lu, M., Chakraborty, P., Sow, D., Ghalwash, M., and Lehman, L.-W. H. G-Net: A deep learning approach to G-computation for counterfactual outcome prediction under dynamic treatment regimes. In *Machine Learning for Health*, 2021.

Libovický, J., Helcl, J., and Mareček, D. Input combination strategies for multi-source transformer decoder. In *Conference on Machine Translation*, 2018.

Lim, B., Alaa, A., and van der Schaar, M. Forecasting treatment responses over time using recurrent marginal structural networks. *Advances in Neural Information Processing Systems*, 2018.

Liu, W., Chen, S., Guo, L., Zhu, X., and Liu, J. CPTR: Full transformer network for image captioning. *arXiv preprint arXiv:2101.10804*, 2021.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

Neyman, J. S. On the application of probability theory to agricultural experiments. *Annals of Agricultural Sciences*, 10:1–51, 1923.

Qian, Z., Zhang, Y., Bica, I., Wood, A., and van der Schaar, M. SyncTwin: Treatment effect estimation with longitudinal outcomes. In *Advances in Neural Information Processing Systems*, 2021.

Robins, J. A new approach to causal inference in mortality studies with a sustained exposure period: Application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7(9-12):1393–1512, 1986.

Robins, J. M. and Hernán, M. A. *Estimation of the causal effects of time-varying exposures*. CRC Press, Boca Raton, FL, 2009.

Robins, J. M., Hernan, M. A., and Brumback, B. Marginal structural models and causal inference in epidemiology, 2000.

Rubin, D. B. Bayesian inference for causal effects: The role of randomization. *The Annals of Statistics*, pp. 34–58, 1978.

Schlag, I., Smolensky, P., Fernandez, R., Jojic, N., Schmidhuber, J., and Gao, J. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611*, 2019.

Schulam, P. and Saria, S. Reliable decision support using counterfactual models. *Advances in Neural Information Processing Systems*, 2017.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

Soleimani, H., Subbaswamy, A., and Saria, S. Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions. In *Uncertainty in Artificial Intelligence*, 2017.

Sormani, M. P. and Bruzzi, P. Can we measure long-term treatment effects in multiple sclerosis? *Nature Reviews Neurology*, 11(3):176–182, 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Subramanian, S., Yilmaz, M., Rehman, A., Hubmayr, R. D., Afessa, B., and Gajic, O. Liberal vs. conservative vasopressor use to maintain mean arterial blood pressure during resuscitation of septic shock: an observational study. *Intensive Care Medicine*, 34(1):157–162, 2008.

Tang, B. and Matteson, D. Probabilistic transformer for time series analysis. *Advances in Neural Information Processing Systems*, 2021.

Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. Simultaneous deep transfer across domains and tasks. In *IEEE International Conference on Computer Vision*, 2015.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

van der Laan, M. J. and Rubin, D. Targeted maximum likelihood learning. *The International Journal of Biostatistics*, 2(1):Article 11, 2006.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Vincent, J.-L., Nielsen, N. D., Shapiro, N. I., Gerbasi, M. E., Grossman, A., Doroff, R., Zeng, F., Young, P. J., and Russell, J. A. Mean arterial pressure and mortality in patients with distributive shock: a retrospective analysis of the MIMIC-III database. *Annals of Intensive Care*, 8 (1):1–10, 2018.

Wager, S. and Athey, S. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Wang, S., McDermott, M. B., Chauhan, G., Ghassemi, M., Hughes, M. C., and Naumann, T. MIMIC-extract: A data extraction, preprocessing, and representation pipeline for MIMIC-III. In *ACM Conference on Health, Inference, and Learning*, 2020.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

Xu, Y., Xu, Y., and Saria, S. A non-parametric Bayesian approach for estimating treatment-response curves from sparse time series. In *Machine Learning for Health*, 2016.

Yaz, Y., Foo, C.-S., Winkler, S., Yap, K.-H., Piliouras, G., Chandrasekhar, V., et al. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*, 2018.

Yoon, J., Jordon, J., and van der Schaar, M. GANITE: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*, 2018.

Zehui, L., Liu, P., Huang, L., Chen, J., Qiu, X., and Huang, X. DropAttention: A regularization method for fully-connected self-attention networks. *arXiv preprint arXiv:1907.11065*, 2019.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Association for the Advancement of Artificial Intelligence*, 2021.

A. Assumptions for Causal Identification

We build upon the potential outcomes framework (Neyman, 1923; Rubin, 1978) and its extension to time-varying treatments and outcomes (Robins & Hernán, 2009). The potential outcomes framework has been widely used in earlier works with a similar objective as ours (Robins & Hernán, 2009; Lim et al., 2018; Bica et al., 2020). To this end, three standard assumptions for data generating mechanism are needed to identify a counterfactual outcome distribution over time, or, specifically, average τ-step-ahead potential outcome conditioned on history from Eq. (1):

Assumption A.1. (Consistency). If $\bar{\mathbf{A}}_t = \bar{\mathbf{a}}_t$ is a given sequence of treatments for some patient, then $\mathbf{Y}_{t+1}[\bar{\mathbf{a}}_t] = \mathbf{Y}_{t+1}$. This means that the potential outcome under treatment sequence $\bar{\mathbf{a}}_t$ coincides for the patient with the observed (factual) outcome, conditional on $\bar{\mathbf{A}}_t = \bar{\mathbf{a}}_t$.

Assumption A.2. (Sequential Overlap). There is always a non-zero probability of receiving/not receiving any treatment for all the history space over time: $0 < \mathbb{P}(\mathbf{A}_t = \mathbf{a}_t \mid \bar{\mathbf{H}}_t = \bar{\mathbf{h}}_t) < 1$, if $\mathbb{P}(\bar{\mathbf{H}}_t = \bar{\mathbf{h}}_t) > 0$, where $\bar{\mathbf{h}}_t$ is some realization of a patient history.

Assumption A.3. (Sequential Ignorability) or No Unobserved Confounding. The current treatment is independent of the potential outcome, conditioning on the observed history: $\mathbf{A}_t \perp\!\!\!\perp \mathbf{Y}_{t+1}[\mathbf{a}_t] \mid \bar{\mathbf{H}}_t, \quad \forall \mathbf{a}_t$. This implies that there are no unobserved confounders that affect both treatment and outcome.

The data generating mechanism for $\mathcal{D}$ is shown in Figure 3.

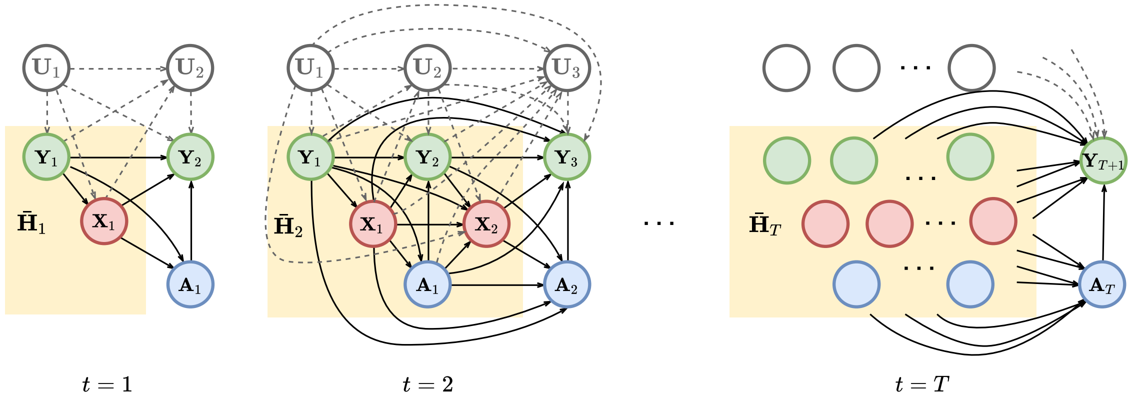


Figure 3. Causal diagram for data generating mechanism, depicted for different time steps t. $\mathbf{U}_t$ is unobserved exogenous noise, which only affects time-varying covariates and outcomes, but not treatments. All time-varying confounders up to time t are included in the observed history $\bar{\mathbf{H}}_t$. Static covariates are ignored for the simplicity.

Corollary A.4. *(G-computation (Robins, 1986)). Assumptions A.1–A.3 provide sufficient identifiability conditions for Eq. (1), e.g., with the G-computation*

$$\mathbb{E}\left(\mathbf{Y}_{t+\tau}[\bar{\mathbf{a}}_{t:t+\tau-1}] \mid \bar{\mathbf{H}}_t\right) = \int_{\mathbb{R}^{d_x}\times\cdots\times\mathbb{R}^{d_x}} \mathbb{E}\left(\mathbf{Y}_{t+\tau} \mid \bar{\mathbf{H}}_t, \bar{\mathbf{x}}_{t+1:t+\tau-1}, \bar{\mathbf{y}}_{t+1:t+\tau-1}, \bar{\mathbf{a}}_{t:t+\tau-1}\right) \times \\ \prod_{j=t+1}^{t+\tau-1} \mathbb{P}\left(\mathbf{x}_j\mathbf{y}_j \mid \bar{\mathbf{H}}_t, \bar{\mathbf{x}}_{t+1:j-1}, \bar{\mathbf{y}}_{t+1:j-1}, \bar{\mathbf{a}}_{t:j-1}\right) \, \mathrm{d}\bar{\mathbf{x}}_{t+1:t+\tau-1}\, \mathrm{d}\bar{\mathbf{y}}_{t+1:t+\tau-1} \tag{24}$$

Empirical G-computation is used by G-Net (Li et al., 2021), but requires the estimation of conditional distributions of time-varying covariates. This could be particularly challenging, given a finite dataset size and high dimensionality of covariates. Thus, we refrain from explicit usage of G-computation.

B. Methods for Estimating Counterfactual Outcomes over Time

In Table 5, we provide an overview of the machine-learning-based methods for estimating counterfactual outcomes over time. For our experiments, we selected **MSMs** (Robins et al., 2000; Hernán et al., 2001), as the simplest linear baseline, and three state-of-the-art methods: **RMSNs** (Lim et al., 2018), **CRN** (Bica et al., 2020), and **G-Net** (Li et al., 2021)). Importantly, our choice of baselines is thus analogous to the those in the state-of-the-art literature (Lim et al., 2018; Bica et al., 2020; Li et al., 2021). Below, we provide details why certain works are not of fit for our setting and are thus not applicable as baselines. Here, we again emphasize that this selection is again consistent with the literature (Lim et al., 2018; Bica et al., 2020; Li et al., 2021).

One stream of the literature focuses on non- or semi-parametric methods (Xu et al., 2016; Schulam & Saria, 2017; Soleimani et al., 2017). These are, for example, based on Gaussian processes (GPs). However, the aforementioned methods have three limitations: (1) They are not designed to handle static covariates. As such, risk factors (e.g., age, gender) that are standard in any electronic health record must be excluded. This omits a substantial heterogeneity in any patient cohort, and is thus impractical. (2) These methods cannot handle multiple outcomes. (3) Due to the non-parametric nature of their estimation, these methods typically cannot scale to large-scale datasets. Contrary to that, several methods have been built that overcome these limitations, namely **RMSNs** (Lim et al., 2018), **CRN** (Bica et al., 2020), and **G-Net** (Li et al., 2021), which we included as baselines.

We excluded two additional methods as these do not match our setting:

- SyncTwin (Qian et al., 2021) is a semi-parametric method using synthetic control, but is limited to a single-time binary treatment and, therefore, not applicable to our setting.
- DCRN (Berrevoets et al., 2021) may appear relevant at first glance; however, it only works with sequences of binary treatments. More importantly, it requires a stronger version of the sequential ignorability assumption: *Sequential Ignorability conditional on current covariates. The current treatment is independent from the potential outcome, conditional on current time-varying covariates:* $\mathbf{A}_t \perp\!\!\!\perp \mathbf{Y}_{t+1}[\mathbf{a}_t] \mid \mathbf{X}_t, \quad \forall \mathbf{a}_t$. Therefore, this setting and ours are different, as in our setting, past time-varying covariates may also serve as confounders.

We further make an important remark. The problem of estimating counterfactual outcomes over time differs also from reinforcement learning: different from reinforcement learning, we assume a non-Markovian data generation mechanism. This impedes the applicability of such approaches as the size of the state space (history) grows typically with time (see Appendix A, Fig. 3).

Table 5. Overview of methods for estimating counterfactual outcomes over time.

Method	Setting	Model type (backbone)	Time	Treatments	Framework
HITR (Xu et al., 2016)	DGM (✗)	NP (GP)	Disc & Cont	Seq, Cat	G-computation
CGP (Schulam & Saria, 2017)	C, SO, SI, CSI (✗)	NP (GP)	Cont	Seq, Cat	G-computation
MOGP (Soleimani et al., 2017)	DGM (✗)	SP (GP)	Disc & Cont	Seq, Cont	G-computation
SyncTwin (Qian et al., 2021)	DGM (✗)	SP (GRU-D, LSTM)	Disc	Single-time, Bin	Synthetic control
DCRN (Berrevoets et al., 2021)	C, SO, Cov (✗)	P (3 LSTMs)	Disc	Seq, Bin	Disentangled representation
* MSMs (Robins et al., 2000)	C, SO, SI (✓)	P (Logistic & linear regressions)	Disc	Seq, Cat	IPTW weighted loss
* RMSNs (Lim et al., 2018)	C, SO, SI (✓)	P (LSTM)	Disc	Seq, Cat	IPTW weighted loss
* CRN (Bica et al., 2020)	C, SO, SI (✓)	P (LSTM)	Disc	Seq, Cat	BR (gradient reversal)
* G-Net (Li et al., 2021)	C, SO, SI (✓)	P (LSTM)	Disc	Seq, Cat	G-computation
* *Causal Transformer* (this paper)	C, SO, SI	P (3 transformers)	Disc	Seq, Cat	BR (CDC)

* = Methods with the same assumptions as ours (and thus included in our baselines)

Legend:

- Setting: consistency (C), sequential overlap (SO), sequential ignorability (SI), sequential ignorability but conditional on covariates (Cov), continuous sequential ignorability (CSI), assumed data generating model (DGM)
- Model: parametric (P), semi-parametric (SP), and non-parametric (NP)
- Time: discrete (Disc) or continuous (Cont) time steps
- Treatments: sequential (Seq), binary (Bin), categorical (Cat), continuous (Cont).
- Framework: inverse probability of treatment weights (IPTW), balanced representations (BR)

C. Details for Transformer Block

Here, we provide the detailed formalization of the multi-input transformer block. Recall that our multi-input transformer block builds on top of three intertwined transformer subnetworks (see Fig. 1). First, we incorporate three separate self-attentions:

$$\tilde{\mathrm{A}}^{b-1} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\mathrm{A}^{b-1}), K(\mathrm{A}^{b-1}), V(\mathrm{A}^{b-1})\right) + \mathrm{A}^{b-1}\Big), \tag{25}$$

$$\tilde{\mathrm{X}}^{b-1} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\mathrm{X}^{b-1}), K(\mathrm{X}^{b-1}), V(\mathrm{X}^{b-1})\right) + \mathrm{X}^{b-1}\Big), \tag{26}$$

$$\tilde{\mathrm{Y}}^{b-1} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\mathrm{Y}^{b-1}), K(\mathrm{Y}^{b-1}), V(\mathrm{Y}^{b-1})\right) + \mathrm{Y}^{b-1}\Big). \tag{27}$$

Further, we incorporate cross-attentions:

$$\tilde{\mathrm{A}}^{b-1}_{\mathrm{X}} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\tilde{\mathrm{A}}^{b-1}), K(\mathrm{X}^{b-1}), V(\mathrm{X}^{b-1})\right) + \tilde{\mathrm{A}}^{b-1}\Big), \tag{28}$$

$$\tilde{\mathrm{A}}^{b-1}_{\mathrm{Y}} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\tilde{\mathrm{A}}^{b-1}), K(\mathrm{Y}^{b-1}), V(\mathrm{Y}^{b-1})\right) + \tilde{\mathrm{A}}^{b-1}\Big), \tag{29}$$

$$\tilde{\mathrm{X}}^{b-1}_{\mathrm{A}} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\tilde{\mathrm{X}}^{b-1}), K(\mathrm{A}^{b-1}), V(\mathrm{A}^{b-1})\right) + \tilde{\mathrm{X}}^{b-1}\Big), \tag{30}$$

$$\tilde{\mathrm{X}}^{b-1}_{\mathrm{Y}} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\tilde{\mathrm{X}}^{b-1}), K(\mathrm{Y}^{b-1}), V(\mathrm{Y}^{b-1})\right) + \tilde{\mathrm{X}}^{b-1}\Big), \tag{31}$$

$$\tilde{\mathrm{Y}}^{b-1}_{\mathrm{X}} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\tilde{\mathrm{Y}}^{b-1}), K(\mathrm{X}^{b-1}), V(\mathrm{X}^{b-1})\right) + \tilde{\mathrm{Y}}^{b-1}\Big), \tag{32}$$

$$\tilde{\mathrm{Y}}^{b-1}_{\mathrm{A}} = \mathrm{LN}\Big(\mathrm{MHA}\left(Q(\tilde{\mathrm{Y}}^{b-1}), K(\mathrm{A}^{b-1}), V(\mathrm{A}^{b-1})\right) + \tilde{\mathrm{Y}}^{b-1}\Big). \tag{33}$$

Notably, the tensors of treatment representations A^b and $\tilde{\mathrm{A}}^b$ are left-shifted with respect to covariates and outcomes representation tensors. Next, we pool the intermediate outputs using linearly transformed static features:

$$\tilde{\tilde{\mathrm{A}}}^{b-1} = \tilde{\mathrm{A}}^{b-1}_{\mathrm{X}} + \tilde{\mathrm{A}}^{b-1}_{\mathrm{Y}} + \mathbf{1}\tilde{\mathbf{V}}^{\top}, \tag{34}$$

$$\tilde{\tilde{\mathrm{X}}}^{b-1} = \tilde{\mathrm{X}}^{b-1}_{\mathrm{A}} + \tilde{\mathrm{X}}^{b-1}_{\mathrm{Y}} + \mathbf{1}\tilde{\mathbf{V}}^{\top}, \tag{35}$$

$$\tilde{\tilde{\mathrm{Y}}}^{b-1} = \tilde{\mathrm{Y}}^{b-1}_{\mathrm{X}} + \tilde{\mathrm{Y}}^{b-1}_{\mathrm{A}} + \mathbf{1}\tilde{\mathbf{V}}^{\top}. \tag{36}$$

Finally, the hidden states are processed in parallel by feed-forward layers:

$$\mathrm{A}^b = \mathrm{LN}\left(\mathrm{FF}(\tilde{\tilde{\mathrm{A}}}^{b-1}) + \tilde{\tilde{\mathrm{A}}}^{b-1}\right) \tag{37}$$

$$\mathrm{X}^b = \mathrm{LN}\left(\mathrm{FF}(\tilde{\tilde{\mathrm{X}}}^{b-1}) + \tilde{\tilde{\mathrm{X}}}^{b-1}\right), \tag{38}$$

$$\mathrm{Y}^b = \mathrm{LN}\left(\mathrm{FF}(\tilde{\tilde{\mathrm{Y}}}^{b-1}) + \tilde{\tilde{\mathrm{Y}}}^{b-1}\right). \tag{39}$$

D. Absolute Positional Encoding

For completeness, we briefly summarize *absolute* positional encoding (Vaswani et al., 2017) in the following. We thereby hope that readers can better understand the differences in the use of *relative* positional encoding as used in our *Causal Transformer*.

In (Vaswani et al., 2017), absolute positional encoding $\mathrm{PE}(t) \in \mathbb{R}^{d_h}$ was introduced to uniquely encode each time step $t \in \{1, \ldots, T\}$. Absolute positional encodings were added to the initial hidden states right before the first transformer block via

$$\hat{\mathbf{h}}_t^0 = \mathbf{h}_t^0 + \mathrm{PE}(t). \tag{40}$$

In addition, the authors used fixed (non-trainable) weights, produced by sine and cosine functions with differing frequencies; i. e.,

$$\big(\mathrm{PE}(t)\big)_{2j} = \sin \frac{t}{10000^{2j/d_h}}, \tag{41}$$

$$\big(\mathrm{PE}(t)\big)_{2j+1} = \cos \frac{t}{10000^{2j/d_h}}. \tag{42}$$

This encoding scheme ensures continuity between neighboring time steps and that time-delta shifts are equivalent to linear transformations. Alternatively, one could use trainable absolute positional encodings, which would require learning $T \times d_h$ parameters, where T is a maximum sequence length from the training subset. For our CT, this limits the ability to generalize to unseen sequence lengths. Hence, this is the main reason why we opted for clipped relative positional encodings for our CT instead.

Notably, we used the same fixed encoding scheme from Eq. (41)–(42) to produce non-trainable relative PE.

E. Details on Adversarial Training

In our CT, we implement the adversarial training for Eq. (18) and Eq. (19) by performing iterative gradient descent updates (rather than optimizing globally). Algorithm 1 provides the pseudocode. Recall that we further make use of exponential moving average (EMA) for stabilization.

Algorithm 1 Adversarial training in CT via iterative gradient descent

```
**Input:** number of iterations $n_{\text{iter}}$, smoothing parameter $\beta$, CDC coefficient $\alpha$, learning rate $\eta$
Initialize $\theta_Y^{(0)}, \theta_A^{(0)}, \theta_R^{(0)}$
**for** $i = 1$ **to** $n_{\text{iter}}$ **do**
    Update gradient descent $\theta_Y^{(i)} \leftarrow \theta_Y^{(i-1)} - \eta \nabla_{\theta_Y} \left[ \mathcal{L}_{G_Y}(\theta_Y^{(i-1)}, \theta_R^{(i-1)}) \right]$
    Update gradient descent $\theta_R^{(i)} \leftarrow \theta_R^{(i-1)} - \eta \nabla_{\theta_R} \left[ \mathcal{L}_{G_Y}(\theta_Y^{(i-1)}, \theta_R^{(i-1)}) + \alpha \mathcal{L}_{\text{conf}}(\theta_{A,\text{EMA}}^{(i-1)}, \theta_R^{(i-1)}) \right]$
    Update EMA $\theta_{Y,\text{EMA}}^{(i)} \leftarrow \beta \theta_{Y,\text{EMA}}^{(i-1)} + (1-\beta)\theta_Y^{(i)}$
    Update EMA $\theta_{R,\text{EMA}}^{(i)} \leftarrow \beta \theta_{R,\text{EMA}}^{(i-1)} + (1-\beta)\theta_R^{(i)}$
    Update gradient descent $\theta_A^{(i)} \leftarrow \theta_A^{(i-1)} - \eta \nabla_{\theta_A} \left[ \mathcal{L}_{G_A}(\theta_A^{(i-1)}, \theta_{R,\text{EMA}}^{(i)}) \right]$
    Update EMA $\theta_{A,\text{EMA}}^{(i)} \leftarrow \beta \theta_{A,\text{EMA}}^{(i-1)} + (1-\beta)\theta_A^{(i)}$
**end for**
```

F. Proof of Theorem F.2

We begin by stating a lemma similar to *Lemma 1* (Bica et al., 2020), yet ours includes the treatment probabilities $\mathbb{P}(\mathbf{A}_t = a_i)$ of Eq. (22).

Lemma F.1. *Let $\alpha_i = \mathbb{P}(\mathbf{A}_t = a_i)$ and $x' = \Phi(\bar{\mathbf{H}}_t)$ for some fixed representation network $\Phi(\cdot)$. Then, it holds that*

$$G^{*j}_{A}(x') = \frac{\alpha_j P_j^{\Phi}(x')}{\sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x')}. \tag{43}$$

Proof. The objective in Eq. (46) is obtained for fixed Φ by maximizing the following objective pointwise for any x':

$$G_A^* = \arg\max_{G_A} \sum_{j=1}^{d_a} \alpha_j \log\left(G_A^j(x')\right) P_j^{\Phi}(x') \quad \text{subject to} \quad \sum_{i=1}^{d_a} G_A^i(x') = 1. \tag{44}$$

The result can now be obtained by applying Lagrange multipliers as done in (Bica et al., 2020). □

We now derive our theorem.

Theorem F.2. *We fix $t \in \mathbb{N}$ and define P as the distribution of $\bar{\mathbf{H}}_t$, P_j as the distribution of $\bar{\mathbf{H}}_t$ given $\mathbf{A}_t = a_j$, and P_j^{Φ} as the distribution of $\Phi(\bar{\mathbf{H}}_t)$ given $\mathbf{A}_t = a_j$ for all $j \in \{1, \ldots, d_a\}$. Let G_A^j denote the output of G_A corresponding to treatment a_j. Then, there exists an optimal pair (Φ^*, G_A^*) such that*

$$\Phi^* = \arg\max_{\Phi} \sum_{j=1}^{d_a} \mathbb{E}_{\bar{\mathbf{H}}_t \sim P}\left[\log G^{*j}_{A}(\Phi(\bar{\mathbf{H}}_t))\right] \tag{45}$$

$$G_A^* = \arg\max_{G_A} \sum_{j=1}^{d_a} \mathbb{E}_{\bar{\mathbf{H}}_t \sim P_j}\left[\log G_A^j(\Phi^*(\bar{\mathbf{H}}_t))\right] \mathbb{P}(\mathbf{A}_t = a_j) \tag{46}$$

$$\textit{subject to} \sum_{i=1}^{d_a} G_A^i(\Phi^*(\bar{\mathbf{H}}_t)) = 1. \tag{47}$$

Furthermore, Φ^* satisfies Eq. (45) if and only if it induces balanced representations across treatments, i.e., $P_1^{\Phi^*} = \ldots = P_{d_a}^{\Phi^*}$.

Proof of Theorem F.2. We plug the optimal prediction probabilities provided by Lemma F.1 into the objective Eq. (45) and obtain

$$\sum_{j=1}^{d_a} \mathbb{E}_{x' \sim P^{\Phi}}\left[\log \frac{\alpha_j P_j^{\Phi}(x')}{\sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x')}\right] = \sum_{j=1}^{d_a} \int \log\left(\frac{P_j^{\Phi}(x')}{\sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x')}\right) P^{\Phi}(x') \, \mathrm{d}x' + \underbrace{\sum_{j=1}^{d_a} \log(\alpha_j)}_{=C} \tag{48}$$

$$= \sum_{j=1}^{d_a} \int \log\left(\frac{P_j^{\Phi}(x')}{\sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x')}\right) \sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x') \, \mathrm{d}x' + C \tag{49}$$

$$= -\sum_{j=1}^{d_a} \mathrm{KL}\left(\sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x') \,\middle\|\, P_j^{\Phi}(x')\right) + C. \tag{50}$$

Hence, the objective becomes

$$\min_{\Phi} \sum_{j=1}^{d_a} \mathrm{KL}\left(\sum_{i=1}^{d_a} \alpha_i P_i^{\Phi}(x') \,\middle\|\, P_j^{\Phi}(x')\right). \tag{51}$$

For balanced representations $P_1^\Phi = \cdots = P_{d_a}^\Phi$, we obtain a global minimum because

$$\mathrm{KL}\left(\sum_{i=1}^{d_a} \alpha_i P_i^\Phi(x') \,\middle\|\, P_j^\Phi(x')\right) = \mathrm{KL}\left(P_1^\Phi(x') \,\middle\|\, P_1^\Phi(x')\right) = 0 \tag{52}$$

for all $j \in \{1, \ldots, d_a\}$.

Let us now assume that there exists an optimal Φ that satisfies Eq. (52) and that induces unbalanced representations, i.e., there exists an $j \neq \ell$ with $P_j^\Phi \neq P_\ell^\Phi$. This implies

$$\sum_{i=1}^{d_a} \alpha_i P_i^\Phi(x') = P_j^\Phi \neq P_\ell^\Phi = \sum_{i=1}^{d_a} \alpha_i P_i^\Phi(x'), \tag{53}$$

which is a contradiction. Hence, Φ attains the global optimum if and only if it induces balanced representations. □

G. Baseline Methods

We select four methods as baselines, which make use of the same setting as our work (see Sec. 2). These are: (1) marginal structural models (**MSMs**) (Robins et al., 2000; Hernán et al., 2001), (2) recurrent marginal structural networks (**RMSNs**) (Lim et al., 2018), (3) counterfactual recurrent network (**CRN**) (Bica et al., 2020), and (4) **G-Net** (Li et al., 2021). We provide details for each in the following.

G.1. Marginal Structural Models (MSMs)

Marginal structural models (MSMs) (Robins et al., 2000; Hernán et al., 2001) are a standard baseline from epidemiology, which aim at counterfactual outcomes estimation with inverse probability of treatment weights (IPTW) via linear modeling. Time-varying confounding bias is removed with the help of stabilized weights

$$SW(t,\tau) = \frac{\prod_{n=t}^{t+\tau} f\left(\mathbf{A}_n \mid \bar{\mathbf{A}}_{n-1}\right)}{\prod_{n=t}^{t+\tau} f\left(\mathbf{A}_n \mid \bar{\mathbf{H}}_n\right)}, \tag{54}$$

where τ ranges from 1 to $\tau_{\max}$, and $f\left(\mathbf{A}_n \mid \bar{\mathbf{A}}_{n-1}\right)$ and $f\left(\mathbf{A}_n \mid \bar{\mathbf{H}}_n\right)$ denote the conditional probabilities mass functions for discrete treatments of $\mathbf{A}_n$ given $\bar{\mathbf{A}}_{n-1}$ and $\bar{\mathbf{H}}_n$, respectively. Both are estimated with logistic regressions, which depend on the sum of previous treatment applications, two previous time-varying covariates and static covariates

$$f\left(\mathbf{A}_t \mid \bar{\mathbf{A}}_{t-1}\right) = \sigma\left(\sum_{j=1}^{d_A} \omega_j \sum_{n=1}^{t-1} \mathbb{1}_{[\mathbf{A}_n = a_j]}\right), \tag{55}$$

$$f\left(\mathbf{A}_t \mid \bar{\mathbf{H}}_t\right) = \sigma\left(W_{1,x}\mathbf{X}_t + W_{2,x}\mathbf{X}_{t-1} + W_{1,y}\mathbf{Y}_t + W_{2,y}\mathbf{X}_{t-1} + W_v\mathbf{V} + \sum_{j=1}^{d_A} \phi_j \sum_{n=1}^{t-1} \mathbb{1}_{[\mathbf{A}_n = a_j]}\right), \tag{56}$$

where $\sigma(\cdot)$ is a sigmoid function and where $\omega_\cdot, \phi_\cdot, W_\cdot$ are logistic regression parameters. After the stabilized weights are estimated, they are normalized and truncated at their 1-st and 99-th percentiles as done in (Lim et al., 2018).

Counterfactual outcome regressions are fit for each prediction horizon τ separately. For a specific τ, we split dataset into smaller chunks with a rolling origin and calculate stabilized weights for each chunk. Outcome regressions use the same history inputs, as $f\left(\mathbf{A}_n \mid \bar{\mathbf{H}}_n\right)$ (Eq. (56)).

MSMs do not contain hyperparameters; thus, we have merged train and validation subsets for all the experiments.

G.2. Recurrent Marginal Structural Networks (RMSNs)

RMSNs refer to sequence-to-sequence architectures consisting of four LSTM subnetworks: propensity treatment network, propensity history network, encoder, and decoder. RMSNs are designed to handle multiple binary treatments. The encoder first learns a representation of the observed history $\bar{\mathbf{H}}_t$ to perform one-step-ahead prediction. The decoder then uses this representation for estimating τ-step-ahead counterfactual outcomes. A fully-connected linear layer (memory adapter) is used to match the size of the representation of the encoder and the hidden units of the decoder.

In RMSNs, time-varying confounding is addressed by re-weighting the objective with the IPTW (Robins et al., 2000) during training. IPTW creates a pseudo-population that mimics a randomized controlled trial. As done in (Lim et al., 2018), we use the stabilized weights (Eq. (54)). Both $f\left(\mathbf{A}_n \mid \bar{\mathbf{A}}_{n-1}\right)$ and $f\left(\mathbf{A}_n \mid \bar{\mathbf{H}}_n\right)$ are learned from the data using LSTM networks, which are called propensity treatment network (nominator) and propensity history network (denominator).

During training, the propensity networks are trained first to estimate the stabilized weights $SW(t,\tau)$. Afterward, the encoder is trained using a mean squared error (MSE) weighted with $SW(\cdot, 1)$. Similarly to MSMs, stabilized weights are normalized and truncated.

Finally, the decoder is trained by minimizing the loss using the full stabilized weights $SW(\cdot, \tau_{\max})$. For this purpose, the dataset is processed into smaller chunks with rolling origins, and, for each rolling origin, a representation is built using the trained encoder. We refer to (Lim et al., 2018) for details on the training algorithm.

We tuned the same hyperparameters, as in the original paper (Lim et al., 2018) (see details in Appendix H).

G.3. Counterfactual Recurrent Network (CRN)

CRN consists of an encoder-decoder architecture. In contrast to RMSNs, which use IPTW to address time-varying confounding, CRN builds balanced representations which are non-predictive of the treatment assignment. This is achieved by adopting an adversarial learning technique, namely gradient reversal (Ganin & Lempitsky, 2015).

In CRN, both encoder and decoder consist of a single LSTM-layer. Unlike RMSNs, the authors and we did not use a memory adapter. Thus, the number of LSTM hidden units d_h of decoder is set to the size of the balanced representation of the encoder.

At each time step t, the hidden states $\mathbf{h}_t$ are fed into a fully-connected linear layer that builds a representation $\mathbf{\Phi}_t$. Then, two fully-connected networks G_Y and G_A, put on top of $\mathbf{\Phi}_t$, aim to predict the next outcome $\mathbf{Y}_{t+1}$ and the current treatment $\mathbf{A}_t$, respectively. For this, both encoder and decoder are trained by minimizing the loss

$$\mathcal{L} = \left\| \mathbf{Y}_{t+1} - G_Y\left(\mathbf{\Phi}_t, \mathbf{A}_t\right) \right\|^2 - \lambda \sum_{j=1}^{d_a} \mathbb{1}_{[\mathbf{A}_t = a_j]} \log G_A(\mathbf{\Phi}_t) \tag{57}$$

with hyperparameter λ. The loss $\mathcal{L}$ is based on a gradient reversal layer (Ganin & Lempitsky, 2015), which forces G_A to minimize cross-entropy between predicted and current treatment, but $\mathbf{\Phi}_t$ to maximize it. In our experiments, we kept $\lambda = 1$, as it was used by (Atan et al., 2018b; Bica et al., 2020).

In our ablation study (Sec. 5.4), we combined CRN with our CDC loss. For that, we applied our adversarial training procedure (introduced in Sec. 4.3) to representations of LSTM-based encoder and decoder, and feed-forward networks G_Y and G_A. Here, EMA of model parameters ($\beta = 0.99$) was also accompanying the CDC loss.

G.4. G-Net

G-Net is based the G-computation formula from Eq. (24), which expresses the average counterfactual outcome $\mathbf{Y}_{t+\tau}[\bar{\mathbf{a}}_{t,t+\tau-1}]$ conditioned on the history $\bar{\mathbf{H}}_t$ in terms of the observational data distribution.

G-Net performs counterfactual outcomes prediction in two steps: First, the conditional distributions $\mathbb{P}(\mathbf{X}_j \mid \bar{\mathbf{H}}_t, \bar{\mathbf{x}}_{t+1:j-1}, \bar{\mathbf{a}}_{t:j-1})$ are estimated. Then, Monte Carlo simulations are performed via Eq. (24), by sampling from the estimated distributions. Afterward, $\mathbf{Y}_{t+\tau}[\bar{\mathbf{a}}_{t,t+\tau-1}]$ is predicted by taking the empirical mean over the Monte Carlo samples ($M = 50$ in our experiments).

The conditional distributions $\mathbb{P}(\mathbf{X}_j \mid \bar{\mathbf{H}}_t, \bar{\mathbf{x}}_{t+1:j-1}, \bar{\mathbf{a}}_{t:j-1})$ are learned by estimating the respective conditional expectations $\mathbb{E}(\mathbf{X}_j \mid \bar{\mathbf{H}}_t, \bar{\mathbf{x}}_{t+1:j-1}, \bar{\mathbf{a}}_{t:j-1})$, which are learned via a single LSTM jointly with outcome prediction. One can then sample from $\mathbb{P}(\mathbf{X}_{t+j} \mid \bar{\mathbf{H}}_t, \bar{\mathbf{x}}_{t+1:j-1}, \bar{\mathbf{a}}_{t:j-1})$ by drawing from the empirical distributions of the residuals on some holdout set not used to estimate the conditional expectations. We used 10% of the training data for the holdout dataset.

For better comparability with other baselines, we used one or two-layered LSTMs (as in the original papers) with an extra fully-connected linear representation layer and a network with hidden units on top of the latter (analogous to G_Y in CT or CRN).

H. Hyperparameter Tuning

We performed hyperparameter optimization for all benchmarks via random grid search with respect to the factual RMSE of the validation set. We list the ranges of hyperparameter grids in Table 6. We report additional information on model-specific hyperparameters in Table 7 (here we used the same ranges for all experiments). For reproducibility, we make the selected hyperparameters public: they can be found in YAML format in our GitHub[4].

We aimed for a fair comparison and thus kept the number of parameters and layers similar across datasets and models. Nevertheless, the hyperparameter ranges differ slightly for each dataset and model, as the size of inputs is different (see Table 7). Thus, e.g., the range of sizes of hidden units (sequential, representational, or fully-connected) is decreased for the MIMIC-III-based experiments. In specific cases (LSTM hidden units propensity treatment network of RMSNs or transformer units of CT), we discarded unrealistically small values for synthetic datasets. For the fully-synthetic dataset based on the tumor growth simulator, we use one layer sequential models. For MIMIC-III, we also include two-layered LSTMs/transformers. The number of epochs (n_e) is also chosen differently for each dataset to reflect its complexity. CT generally requires more epochs to converge due to the EMA of model weights. Therefore, we used approximately 60 % more epochs for CT than other models. Note that CT still outperforms the baselines when EMA is omitted, as shown in our ablation study. Due to the high memory usage of self-attention for long sequences and batch augmentation with masked vitals of CT, we also use smaller ranges of minibatch sizes for CT. Notably, as in CT, we omitted the final projection layer after concatenation of the attention heads, as we need the size of hidden units (which always depends on the input size while tuning) to always be divisible by the number of heads n_h. Thus, we have chosen the closest larger divisible by the number of hidden units.

Training of baselines: All baseline models are implemented in PyTorch Lightning and, as our CT, trained with Adam (Kingma & Ba, 2015). The number of epochs (n_e) is varied across datasets for a better fit.

We perform exponential rise of both α (in the CDC loss) and λ (in gradient reversal). This is given by

$$\alpha_e = \alpha \cdot \Big(\frac{2}{1 + \exp(-10 * e/n_e)} - 1\Big), \qquad \lambda_e = \lambda \cdot \Big(\frac{2}{1 + \exp(-10 * e/n_e)} - 1\Big), \tag{58}$$

where $e \in 1, \ldots, n_e$ is an index of current epoch.

For all baselines, we also used the teacher forcing technique (Williams & Zipser, 1989) when training the models for multiple-step-ahead prediction. During evaluation of multiple-step-ahead prediction, we switch off teacher forcing and autoregressively feed model predictions.

[4] https://github.com/Valentyn1997/CausalTransformer/

Table 6. Ranges for hyperparameter tuning across experiments. Here, we distinguish (1) data using the tumor growth (TG) simulator (=experiments with fully-synthetic data), (2) data from the semi-synthetic (SS) benchmark, and (3) real-world (RW) MIMIC-III data. C is the input size. d_r is the size of balanced representation (BR) or the output of LSTM (in the case of G-Net).

Model	Sub-model	Hyperparameter	Range (TG simulator)	Range (SS data)	Range (RW data)
RMSNs	Propensity treatment network	LSTM layers (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	64, 128, 256		
		LSTM hidden units (d_h)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		LSTM dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Max gradient norm	0.5, 1.0, 2.0		
		Number of epochs (n_e)	100	400	200
	Propensity history network Encoder	LSTM layers (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	64, 128, 256		
		LSTM hidden units (d_h)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		LSTM dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Max gradient norm	0.5, 1.0, 2.0		
		Number of epochs (n_e)	100	400	200
	Decoder	LSTM layers (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	256, 512, 1024		
		LSTM hidden units (d_h)	1C, 2C, 4C, 8C, 16C	1C, 2C, 4C	1C, 2C, 4C
		LSTM dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Max gradient norm	0.5, 1.0, 2.0, 4.0		
		Number of epochs (n_e)	100	400	200
CRN	Encoder	LSTM layers (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	64, 128, 256		
		LSTM hidden units (d_h)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		BR size (d_r)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		FC hidden units (n_{FC})	$0.5d_r$, $1d_r$, $2d_r$, $3d_r$, $4d_r$	$0.5d_r$, $1d_r$, $2d_r$	$0.5d_r$, $1d_r$, $2d_r$
		LSTM dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Number of epochs (n_e)	100	400	200
	Decoder	LSTM layers (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	256, 512, 1024		
		LSTM hidden units (d_h)	BR size of encoder		
		BR size (d_r)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		FC hidden units (n_{FC})	$0.5d_r$, $1d_r$, $2d_r$, $3d_r$, $4d_r$	$0.5d_r$, $1d_r$, $2d_r$	$0.5d_r$, $1d_r$, $2d_r$
		LSTM dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Number of epochs (n_e)	100	400	200
G-Net	—	LSTM layers (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	64, 128, 256		
		LSTM hidden units (d_h)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		LSTM output size (d_r)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		FC hidden units (n_{FC})	$0.5d_r$, $1d_r$, $2d_r$, $3d_r$, $4d_r$	$0.5d_r$, $1d_r$, $2d_r$	$0.5d_r$, $1d_r$, $2d_r$
		LSTM dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Number of epochs (n_e)	50	400	200
CT	—	Transformer blocks (B)	1	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001		
		Minibatch size	64, 128, 256	32, 64	32, 64
		Attention heads (n_h)	2	2, 3	2, 3
		Transformer units (d_h)	1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		BR size (d_r)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C	0.5C, 1C, 2C
		FC hidden units (n_{FC})	$0.5d_r$, $1d_r$, $2d_r$, $3d_r$, $4d_r$	$0.5d_r$, $1d_r$, $2d_r$	$0.5d_r$, $1d_r$, $2d_r$
		Sequential dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5		
		Max positional encoding ($l_{\max}$)	15	20	30
		Number of epochs (n_e)	150	400	300

Table 7. Additional information on model-specific hyperparameters (kept the same for all experiments).

Model	Sub-model	Hyperparameter	Value
RMSNs	Propensity treatment network	Random search iterations Input size (C) Output size	50 d_a d_a
	Propensity history network	Random search iterations Input size (C) Output size	50 $d_a + d_y + d_x + d_v$ d_a
	Encoder	Random search iterations Input size (C) Output size	50 $d_a + d_y + d_x + d_v$ d_y
	Decoder	Random search iterations Input size (C) Output size	20 $d_a + d_y + d_v$ d_y
CRN	Encoder	Random search iterations Input size (C) Output size Gradient reversal coefficient (λ)	50 $d_a + d_y + d_x + d_v$ $d_a + d_y$ 1.0
	Decoder	Random search iterations Input size (C) Output size Gradient reversal coefficient (λ)	30 $d_a + d_y + d_v$ $d_a + d_y$ 1.0
G-Net	—	Random search iterations Input size (C) Output size MC samples (M) Number of covariate groups Holdout dataset ratio (empirical residuals)	50 $d_a + d_y + d_x + d_v$ $d_y + d_x$ 50 1 10%
CT	—	Random search iterations Input size (C) Output size CDC coefficient (α) EMA of model weights (β) Positional encoding	50 $\max\{d_a, d_y, d_x, d_v\}$ $d_a + d_y$ 0.01 0.99 relative, trainable

I. Encoder-Decoder Causal Transformer

I.1. Overview

Here, we summarize the *Encoder-Decoder Causal Transformer* (EDCT) from our ablation study, namely a single-subnetwork version of out CT. The EDCT consists of transformer-based encoder and decoder (see Fig. 4). The encoder builds a treatment-invariant sequence of representations of the history $\bar{\mathbf{\Phi}}_t = (\mathbf{\Phi}_1, \dots, \mathbf{\Phi}_t)$, balanced with a custom adversarial objective. The decoder subsequently uses $\bar{\mathbf{\Phi}}_t$ as cross-attention keys and values for estimating outcomes of future treatments.

We start by mapping the concatenated time-varying covariates, left-shifted treatments, outcomes and static covariates to a hidden state space of dimensionality d_h via fully-connected linear layer:

$$\mathbf{h}_t^0 = \text{Linear}(\text{Concat}(\mathbf{A}_{t-1}, \mathbf{Y}_t, \mathbf{X}_t, \mathbf{V})). \tag{59}$$

In the case of decoder, we apply a similar input transformation

$$\mathbf{h}_t^0 = \text{Linear}(\text{Concat}(\mathbf{a}_{t-1}, \hat{\mathbf{Y}}_t, \mathbf{V})), \tag{60}$$

where $\hat{\mathbf{Y}}_t$ are autoregressively-fed model outputs.

We then stack of B identical encoder/decoder blocks or layers, which transform the whole sequence of hidden states $\left(\mathbf{h}_1^0, \dots, \mathbf{h}_t^0\right)$ in quadratic time, depending on sequence length t. This is given by

$$\mathrm{H}^b = \left(\mathbf{h}_1^b, \dots, \mathbf{h}_t^b\right)^\top \in \mathbb{R}^{t \times d_h}, \tag{61}$$

$$\mathrm{H}^b = \text{Block}_b(\mathrm{H}^{b-1}), \qquad b \in \{1, \dots, B\}, \tag{62}$$

where B is the total number of blocks.

The encoder uses its hidden states to infer keys, queries, and values (thus: self-attention). In contrast, the decoder has both self- and cross-attentions. For later, we use keys and values, inferred from the sequence of balanced representations of the history. Note that the dimensionality of hidden decoder state is set such that it matches the size of the balanced representations of the encoder, i.e., $d_h = d_r$.

Lastly, we take the balanced representations from the last transformer block as outputs. Here, we apply an additional fully-connected linear layer and exponential linear unit (ELU) non-linearity as in Eq. (10).

We make slightly adaptations to the relative positional encoding for the cross-attention of the decoder (Eq. (66)). The decoder works a priori with the continuation of the same sequence. However, it is here beneficial to use the sequence for construction of a_{ij}^V and a_{ij}^K, so that $j \in \{1, \dots, t\}$ and $i \in \{t+1, \dots, t+\tau-1\}$. Notably, relative positional encodings are shared neither between encoder and decoder, nor between self-attention and the cross-attention of the decoder.

We show the EDCT in Fig. 4. We further formalize the encoder/decoder transformer blocks in the following section.

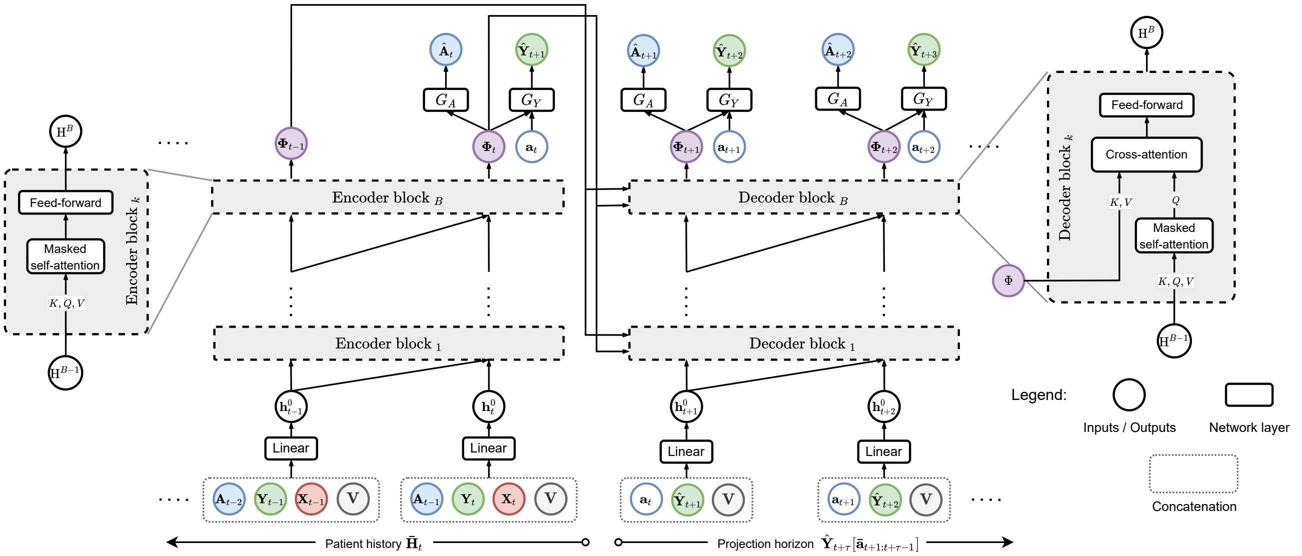


Figure 4. Architecture of the encoder-decoder causal transformer (EDCT). Residual connections with layer normalizations are omitted for clarity. The encoder is trained to perform one-step-ahead prediction $\hat{\mathbf{Y}}_{t+1}[\mathbf{a}_t]$, whereas the decoder uses the pretrained balanced representations of history from the encoder. Based on them, the decoders makes predictions for the projection horizon $\tau \geq 2$ via $\hat{\mathbf{Y}}_{t+\tau}[\bar{\mathbf{a}}_{t+1:t+\tau-1}]$.

I.2. Transformer Blocks in EDCT

The EDCT encoder block is defined in a following way:

$$\tilde{\mathrm{H}}^{b-1} = \mathrm{LN}\Big(\mathrm{MHA}\big(Q(\mathrm{H}^{b-1}), K(\mathrm{H}^{b-1}), V(\mathrm{H}^{b-1})\big) + \mathrm{H}^{b-1}\Big), \tag{63}$$

$$\mathrm{H}^{b} = \mathrm{LN}\big(\mathrm{FF}(\tilde{\mathrm{H}}^{b-1}) + \tilde{\mathrm{H}}^{b-1}\big). \tag{64}$$

The EDCT decoder block adds a cross-attention layer after the self-attention (i.e., between Eq. (63) and Eq. (64)). This is formalized by

$$\tilde{\mathrm{H}}^{b-1} = \mathrm{LN}\Big(\mathrm{MHA}\big(Q(\mathrm{H}^{b-1}), K(\mathrm{H}^{b-1}), V(\mathrm{H}^{b-1})\big) + \mathrm{H}^{b-1}\Big), \tag{65}$$

$$\tilde{\tilde{\mathrm{H}}}^{b-1} = \mathrm{LN}\Big(\mathrm{MHA}\big(Q(\tilde{\mathrm{H}}^{b-1}), K((\bar{\mathbf{\Phi}}_t)^\top), V((\bar{\mathbf{\Phi}}_t)^\top)\big) + \tilde{\mathrm{H}}^{b-1}\Big), \tag{66}$$

$$\mathrm{H}^{b} = \mathrm{LN}\big(\mathrm{FF}(\tilde{\tilde{\mathrm{H}}}^{b-1}) + \tilde{\tilde{\mathrm{H}}}^{b-1}\big), \tag{67}$$

where $\bar{\mathbf{\Phi}}_t$ is a sequence of the encoder representations (i.e., the encoded history $\bar{\mathbf{H}}_t$), as transformed according to Eq. (61)).

I.3. Hyperparameter Tuning for EDCT

We performed a hyperparameter selection for EDCT in a similar manner to CRN, see Appendix H. We provide hyperparameter ranges for both encoder and decoder in Table 8. Other hyperparameters are kept fixed, the same as for CRN in Table 7. For the fully-synthetic dataset based on the tumor growth simulator, we add a two-layer (two-block) architecture to the search range. This was done to keep the number of total trainable parameters similar to other baselines. We employed trainable relative positional encodings. Notably, the decoder has l_{max} set to τ_{max} for the self-attention and to l_{max} of the encoder for the cross-attention.

Table 8. Ranges for hyperparameter tuning for EDCT across experiments. Here, we distinguish (1) data using the tumor growth (TG) simulator (=experiments with fully-synthetic data) and (2) semi-synthetic (SS) data. C is the input size. d_r is the size of balanced representation (BR) or the output of LSTM (in the case of G-Net).

Model	Sub-model	Hyperparameter	Range (TG simulator)	Range (SS data)
EDCT	Encoder	Transformer blocks (B)	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001	
		Minibatch size	64, 128, 256	32, 64, 128
		Attention heads (n_h)	2	2, 3
		Transformer units (d_h)	1C, 2C, 3C, 4C	0.5C, 1C, 2C
		BR size (d_r)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C
		FC hidden units (n_{FC})	0.5d_r, 1d_r, 2d_r, 3d_r, 4d_r	0.5d_r, 1d_r, 2d_r
		Sequential dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5	
		Max positional encoding (self-attention) (l_{max})	15	20
		Number of epochs (n_e)	100	400
	Decoder	Transformer blocks (B)	1, 2	1, 2
		Learning rate (η)	0.01, 0.001, 0.0001	
		Minibatch size	256, 512, 1024	128, 256, 512
		Attention heads (n_h)	2	2, 3
		Transformer units (d_h)	BR size of encoder	
		BR size (d_r)	0.5C, 1C, 2C, 3C, 4C	0.5C, 1C, 2C
		FC hidden units (n_{FC})	0.5d_r, 1d_r, 2d_r, 3d_r, 4d_r	0.5d_r, 1d_r, 2d_r
		Sequential dropout rate (p)	0.1, 0.2, 0.3, 0.4, 0.5	
		Max positional encoding (self-attention) (l_{max})	τ_{max}	
		Max positional encoding (cross-attention) (l_{max})	15	20
		Number of epochs (n_e)	100	400

J. Details on Experiments with Synthetic Data

J.1. Summary of Tumor Growth Simulator

The tumor growth (TG) simulator (Geng et al., 2017) models the volume of tumor $\mathbf{Y}_{t+1}$ for $t+1$ days after cancer diagnosis (so that the outcome is one-dimensional, i. e., $d_y = 1$). The model has two binary treatments: (i) radiotherapy ($\mathbf{A}_t^r$) and (ii) chemotherapy ($\mathbf{A}_t^c$). These are modeled as follows: (i) Radiotherapy when assigned to a patient has an immediate effect $d(t)$ on the next outcome. (ii) Chemotherapy affects several future outcomes with exponentially decaying effect $C(t)$ via

$$\mathbf{Y}_{t+1} = \left(1 + \rho \log\left(\frac{K}{\mathbf{Y}_t}\right) - \beta_c C_t - (\alpha_r d_t + \beta_r d_t^2) + \varepsilon_t\right)\mathbf{Y}_t, \tag{68}$$

where $\rho, K, \beta_c, \alpha_r, \beta_r$ are parameters in the simulation and where $\varepsilon_t \sim N(0, 0.01^2)$ is independently sampled noise. Here, the parameters $\beta_c, \alpha_r, \beta_r$ describe the individual response of each patient and are sampled from a mixture of truncated normal distributions with three mixture components. For exact values of parameters, refer to the code implementation[5]. The indices of mixture components are considered as static covariates ($d_v = 1$). Time-varying confounding is introduced by a biased treatments assignment, identical for both treatments; i. e.,

$$\mathbf{A}_t^c, \mathbf{A}_t^r \sim \text{Bernoulli}\left(\sigma\Big(\frac{\gamma}{D_{\text{max}}}(\bar{D}_{15}(\bar{\mathbf{Y}}_{t-1}) - D_{\text{max}}/2)\Big)\right), \tag{69}$$

where $\sigma(\cdot)$ is a sigmoid activation, D_{max} is the maximum tumor diameter, $\bar{D}_{15}(\bar{\mathbf{Y}}_{t-1})$ is the average tumor diameter over the last 15 days, and γ is a confounding parameter. We can control the level of confounding via γ. For $\gamma = 0$, the treatment assignment is fully randomized. For increasing values, the the amount of time-varying confounding becomes also larger.

In our implementation, we proceed as follows. For RMSNs, we insert two binary treatments directly. For all other methods, we use a single-categorical treatment out of the set $\{(\mathbf{A}_t^c = 0, \mathbf{A}_t^r = 0), (\mathbf{A}_t^c = 1, \mathbf{A}_t^r = 0), (\mathbf{A}_t^c = 0, \mathbf{A}_t^r = 1), (\mathbf{A}_t^c = 1, \mathbf{A}_t^r = 1)\}$.

For each patient in the test set and each time step, we simulate several counterfactual trajectories, depending on τ. For one-step-ahead prediction, we simulate all four combinations of one-step-ahead counterfactual outcomes $\mathbf{Y}_{t+1}$. This corresponds to the tumor volume under all possible combinations of treatment assignments. For multi-step-ahead prediction, the number of all potential outcomes of $\mathbf{Y}_{t+2}, \ldots, \mathbf{Y}_{t+\tau_{\text{max}}}$ growths exponentially with the projection horizon τ_{max}. Therefore, we adopt two alternative schemes:

1. *Single sliding treatment*. To test that the correct timing of a treatment is chosen, we simulate trajectories with a single treatment but where the treatments are iteratively moved over a window ranging from t to $t + \tau_{\text{max}} - 1$. This effectively results in $2(\tau_{\text{max}} - 1)$ trajectories.

2. *Random trajectories*. Here, we simulate a fixed number of trajectories, i. e., $2\,(\tau_{\text{max}} - 1)$, each with random treatment assignments.

The former setting is identical to the one in (Bica et al., 2020). We additionally included the latter setting, as it may also involve more diverse trajectories with multiple treatments. Thereby, we hope to make our analysis more realistic with respect to clinical practice.

For each level of confounding γ, we simulate 10,000 patient trajectories for training, 1,000 for validation, and 1,000 trajectories for testing. We limit the length for trajectories to max. 60 time steps (some patients have shorter trajectories due to recovery or death). Here, and in all following experiments, we apply hyperparameter tuning.

J.2. Experimental Details

Hyperparameter tuning. We perform hyperparameter tuning separately for all models as well as all the different values of the confounding amount γ. For this, we use the 1,000 factual patient time-series from the validation set. Details are in H.

[5] Code is available online: https://github.com/Valentyn1997/CausalTransformer/blob/main/src/data/cancer_sim/cancer_simulation.py

Performance measurement: We retrain the models on five simulated datasets with different random seeds. We then report the averaged root mean square error (RMSE) on the test set, that is, for hold-out data. We report a normalized RMSE, where we normalize by the maximum tumor volume $V_{\text{max}} = 1150\,\text{cm}^3$.

We acknowledge that our results are slightly different from those reported in (Lim et al., 2018; Bica et al., 2020). The aforementioned papers calculate the test RMSE based on both counterfactual trajectories after rolling origin **and** historical factual trajectories *before* rolling origin. However, the latter biases evaluation towards factual performance. For that reason, we opted for a more challenging evaluation that directly matches our aim, namely predicting counterfactuals over time. Therefore, we only measure performance with respect to counterfactual outcomes after rolling origin (and thus without considering historical factual patient trajectories).

J.3. Additional Results

In the following, we provide additional results for one-step-ahead prediction (Table 9), τ-step-ahead prediction in a setting with single sliding treatment (Table 11), and τ-step-ahead prediction with random trajectories (Table 10). Note that CT ($\alpha = 0$) uses the same model and hyperparameters as CT. The only difference is that we switched off our CDC loss.

In the setting of random trajectories (Table 10), RMSEs become lower for increasing projection horizons. This can be expected as the application of treatment should decrease the tumor volume. This results in a lower error of estimation. Importantly, the results confirm the superiority of our *Causal Transformer*.

Table 9. Normalized RMSE for one-step-ahead prediction. Shown: mean and standard deviation over five runs (lower is better). Parameter γ is the the amount of time-varying confounding: higher values mean larger treatment assignment bias.

	$\gamma = 0$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
MSMs	1.107 ± 0.113	1.222 ± 0.108	1.410 ± 0.089	1.680 ± 0.118	2.023 ± 0.230
RMSNs	1.037 ± 0.123	1.104 ± 0.116	1.124 ± 0.115	1.268 ± 0.116	1.399 ± 0.196
CRN	0.782 ± 0.053	0.817 ± 0.050	0.887 ± 0.072	1.063 ± 0.124	1.301 ± 0.144
G-Net	0.832 ± 0.052	0.873 ± 0.080	1.000 ± 0.062	1.299 ± 0.303	1.375 ± 0.250
CT ($\alpha = 0$) (ours)	0.778 ± 0.065	**0.790 ± 0.081**	0.869 ± 0.075	**1.024 ± 0.148**	**1.300 ± 0.220**
CT (ours)	**0.775 ± 0.063**	0.797 ± 0.066	**0.859 ± 0.070**	1.046 ± 0.147	1.316 ± 0.229

Lower = better (best in bold).

Table 10. Normalized RMSE for τ-step-ahead prediction (here: random trajectories setting). Shown: mean and standard deviation over five runs (lower is better). Parameter γ is the the amount of time-varying confounding: higher values mean larger treatment assignment bias.

		$\gamma = 0$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
$\tau = 2$	MSMs	1.04 ± 0.04	1.21 ± 0.13	1.50 ± 0.23	1.73 ± 0.43	1.85 ± 0.71
	RMSNs	1.01 ± 0.09	1.03 ± 0.12	1.00 ± 0.10	1.13 ± 0.16	1.09 ± 0.22
	CRN	0.77 ± 0.04	**0.76 ± 0.05**	**0.81 ± 0.07**	0.94 ± 0.13	1.12 ± 0.25
	G-Net	0.94 ± 0.13	0.95 ± 0.09	1.01 ± 0.05	1.10 ± 0.13	1.20 ± 0.26
	CT ($\alpha = 0$) (ours)	0.76 ± 0.06	**0.76 ± 0.05**	0.82 ± 0.07	0.92 ± 0.21	1.09 ± 0.28
	CT (ours)	**0.75 ± 0.06**	0.77 ± 0.06	**0.81 ± 0.08**	**0.90 ± 0.18**	**1.06 ± 0.27**
$\tau = 3$	MSMs	1.00 ± 0.04	1.14 ± 0.12	1.38 ± 0.22	1.54 ± 0.38	1.51 ± 0.59
	RMSNs	0.96 ± 0.05	1.02 ± 0.09	0.98 ± 0.10	1.11 ± 0.20	1.17 ± 0.29
	CRN	0.78 ± 0.03	**0.78 ± 0.06**	**0.83 ± 0.09**	1.05 ± 0.16	1.23 ± 0.32
	G-Net	1.01 ± 0.15	1.03 ± 0.12	1.07 ± 0.07	1.15 ± 0.20	1.35 ± 0.32
	CT ($\alpha = 0$) (ours)	0.76 ± 0.04	**0.78 ± 0.06**	**0.83 ± 0.10**	0.95 ± 0.25	1.16 ± 0.37
	CT (ours)	**0.75 ± 0.04**	0.79 ± 0.06	**0.83 ± 0.11**	**0.93 ± 0.23**	**1.12 ± 0.32**
$\tau = 4$	MSMs	0.90 ± 0.06	1.02 ± 0.11	1.22 ± 0.21	1.31 ± 0.31	1.25 ± 0.51
	RMSNs	0.89 ± 0.06	0.98 ± 0.08	0.92 ± 0.10	1.06 ± 0.22	1.15 ± 0.31
	CRN	0.74 ± 0.03	**0.74 ± 0.07**	0.80 ± 0.10	1.07 ± 0.17	1.23 ± 0.34
	G-Net	0.97 ± 0.15	0.97 ± 0.13	1.01 ± 0.08	1.07 ± 0.21	1.33 ± 0.34
	CT ($\alpha = 0$) (ours)	0.72 ± 0.03	0.75 ± 0.06	**0.79 ± 0.11**	0.93 ± 0.28	1.14 ± 0.39
	CT (ours)	**0.71 ± 0.03**	0.75 ± 0.06	0.80 ± 0.12	**0.90 ± 0.26**	**1.07 ± 0.35**
$\tau = 5$	MSMs	0.80 ± 0.07	0.89 ± 0.10	1.06 ± 0.20	1.10 ± 0.27	1.08 ± 0.47
	RMSNs	0.81 ± 0.06	0.93 ± 0.07	0.85 ± 0.10	0.99 ± 0.22	1.09 ± 0.30
	CRN	0.68 ± 0.04	**0.68 ± 0.07**	0.75 ± 0.10	1.03 ± 0.16	1.17 ± 0.34
	G-Net	0.88 ± 0.14	0.88 ± 0.14	0.92 ± 0.08	0.97 ± 0.21	1.26 ± 0.36
	CT ($\alpha = 0$) (ours)	**0.66 ± 0.03**	0.69 ± 0.06	**0.73 ± 0.11**	0.88 ± 0.29	1.08 ± 0.38
	CT (ours)	**0.66 ± 0.03**	0.70 ± 0.06	0.74 ± 0.12	**0.84 ± 0.26**	**1.01 ± 0.34**
$\tau = 6$	MSMs	0.71 ± 0.07	0.78 ± 0.09	0.91 ± 0.18	0.93 ± 0.23	0.99 ± 0.44
	RMSNs	0.73 ± 0.05	0.87 ± 0.06	0.77 ± 0.09	0.90 ± 0.21	1.01 ± 0.28
	CRN	0.62 ± 0.04	**0.62 ± 0.07**	0.70 ± 0.09	0.96 ± 0.15	1.10 ± 0.32
	G-Net	0.79 ± 0.12	0.79 ± 0.13	0.82 ± 0.09	0.86 ± 0.20	1.18 ± 0.35
	CT ($\alpha = 0$) (ours)	**0.59 ± 0.02**	0.63 ± 0.06	**0.67 ± 0.11**	0.80 ± 0.29	1.00 ± 0.36
	CT (ours)	**0.59 ± 0.02**	0.63 ± 0.06	**0.67 ± 0.12**	**0.77 ± 0.25**	**0.93 ± 0.32**

Lower = better (best in bold).

Table 11. Normalized RMSE for τ-step-ahead prediction (here: single sliding treatment setting). Shown: mean and standard deviation over five runs (lower is better). Parameter γ is the the amount of time-varying confounding: higher values mean larger treatment assignment bias.

		$\gamma = 0$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
$\tau = 2$	MSMs	1.33 ± 0.13	1.59 ± 0.20	1.88 ± 0.36	2.23 ± 0.63	2.51 ± 0.91
	RMSNs	0.98 ± 0.12	1.10 ± 0.25	0.98 ± 0.08	1.18 ± 0.10	**0.94 ± 0.09**
	CRN	0.71 ± 0.07	0.75 ± 0.06	0.77 ± 0.04	0.94 ± 0.14	1.11 ± 0.22
	G-Net	0.99 ± 0.16	0.99 ± 0.06	1.03 ± 0.09	1.10 ± 0.08	1.18 ± 0.16
	CT ($\alpha = 0$) (ours)	**0.70 ± 0.09**	**0.71 ± 0.09**	0.76 ± 0.08	**0.90 ± 0.21**	1.00 ± 0.21
	CT (ours)	**0.70 ± 0.09**	0.72 ± 0.09	**0.74 ± 0.07**	**0.90 ± 0.13**	1.01 ± 0.23
$\tau = 3$	MSMs	1.61 ± 0.15	1.90 ± 0.24	2.20 ± 0.42	2.53 ± 0.72	2.64 ± 0.95
	RMSNs	0.98 ± 0.10	1.16 ± 0.21	1.00 ± 0.09	1.23 ± 0.12	**1.06 ± 0.14**
	CRN	**0.73 ± 0.06**	0.78 ± 0.06	0.85 ± 0.06	1.16 ± 0.26	1.34 ± 0.37
	G-Net	1.15 ± 0.20	1.16 ± 0.11	1.20 ± 0.15	1.24 ± 0.12	1.47 ± 0.22
	CT ($\alpha = 0$) (ours)	**0.73 ± 0.08**	**0.75 ± 0.08**	0.82 ± 0.09	0.99 ± 0.25	1.13 ± 0.28
	CT (ours)	**0.73 ± 0.08**	0.76 ± 0.07	**0.79 ± 0.08**	**0.98 ± 0.19**	1.12 ± 0.27
$\tau = 4$	MSMs	1.79 ± 0.16	2.08 ± 0.26	2.37 ± 0.45	2.67 ± 0.76	2.62 ± 0.94
	RMSNs	0.99 ± 0.10	1.18 ± 0.17	1.03 ± 0.11	1.28 ± 0.15	**1.21 ± 0.19**
	CRN	**0.76 ± 0.05**	0.81 ± 0.07	0.93 ± 0.08	1.35 ± 0.38	1.55 ± 0.50
	G-Net	1.25 ± 0.24	1.24 ± 0.14	1.27 ± 0.21	1.29 ± 0.14	1.64 ± 0.28
	CT ($\alpha = 0$) (ours)	**0.76 ± 0.07**	**0.79 ± 0.06**	0.87 ± 0.11	1.06 ± 0.27	**1.21 ± 0.32**
	CT (ours)	**0.76 ± 0.07**	0.80 ± 0.06	**0.85 ± 0.09**	**1.05 ± 0.22**	**1.21 ± 0.30**
$\tau = 5$	MSMs	1.88 ± 0.17	2.15 ± 0.27	2.42 ± 0.45	2.69 ± 0.75	2.54 ± 0.90
	RMSNs	1.00 ± 0.10	1.19 ± 0.14	1.08 ± 0.13	1.34 ± 0.19	1.39 ± 0.31
	CRN	**0.79 ± 0.04**	0.85 ± 0.07	1.01 ± 0.11	1.51 ± 0.47	1.72 ± 0.58
	G-Net	1.29 ± 0.26	1.28 ± 0.18	1.32 ± 0.25	1.33 ± 0.15	1.76 ± 0.37
	CT ($\alpha = 0$) (ours)	**0.79 ± 0.06**	**0.83 ± 0.06**	0.92 ± 0.12	1.12 ± 0.30	1.28 ± 0.34
	CT (ours)	**0.79 ± 0.07**	0.84 ± 0.07	**0.89 ± 0.11**	**1.11 ± 0.24**	**1.26 ± 0.31**
$\tau = 6$	MSMs	1.89 ± 0.17	2.14 ± 0.26	2.39 ± 0.44	2.62 ± 0.73	2.41 ± 0.85
	RMSNs	1.03 ± 0.10	1.21 ± 0.12	1.12 ± 0.14	1.41 ± 0.25	1.58 ± 0.45
	CRN	**0.82 ± 0.04**	0.89 ± 0.07	1.08 ± 0.13	1.64 ± 0.54	1.83 ± 0.62
	G-Net	1.33 ± 0.27	1.31 ± 0.22	1.35 ± 0.29	1.35 ± 0.16	1.86 ± 0.47
	CT ($\alpha = 0$) (ours)	**0.82 ± 0.04**	**0.86 ± 0.05**	0.96 ± 0.12	1.19 ± 0.33	1.32 ± 0.34
	CT (ours)	**0.82 ± 0.05**	0.88 ± 0.06	**0.93 ± 0.11**	**1.16 ± 0.25**	**1.29 ± 0.29**

Lower = better (best in bold).

K. Details on Experiments with Semi-Synthetic Data

K.1. Data

We used the MIMIC-extract (Wang et al., 2020) with a standardized preprocessing pipeline of MIMIC-III dataset (Johnson et al., 2016). MIMIC-extract provides intensive care unit (ICU) data aggregated at hourly levels. We used forward and backward filling for missing values and did standard normalization of all the continuous time-varying features.

For our semi-synthetic data, we then extract 25 different vital signs (as time-varying covariates) and 3 static covariates, i. e., gender, ethnicity, and age (as static covariates). The full list of features is given in the code of our GitHub repository for reproducibility. We one-hot-encode all static covariates (gender, ethnicity, and age) and use them later further for generating noise. Altogether, this results into a 44-dimensional feature vector ($d_v = 44$).

Our simulation of semi-synthetic data extends the basic idea of (Schulam & Saria, 2017). As such, we first generate untreated trajectories of outcomes under endogeneous and exogeneous dependencies and, then, sequentially apply treatments to the trajectory. Dependencies between treatments, outcomes, and time-varying covariates are assumed to be sparse, so outcomes are influenced by only a few treatments and time-varying covariates. Treatment assignment in turn depends on a few outcomes and time-varying covariates.

Our semi-synthetic simulator proceeds as follows. *First*, we select a cohort of 1,000 patients, which are randomly chosen from patients where the intensive care unit stay lasted at least 20 hours. For the selected cohort, we clip intensive care unit stays longer than 100 hours (so that $T^{(i)}$ ranges from 20 to 100).

Second, we simulate d_y untreated outcomes $\mathbf{Z}_t^{j,(i)}, j = 1 \ldots, d_y$, for each patient i from the cohort. Therein, we combine (1) an endogenous component (B-spline(t) and random function $g^{j,(i)}(t)$); (2) an exogenous dependency $f_Z^j(\mathbf{X}_t^{(i)})$ on a subset of current time-varying covariates; and (3) independent random noise ε_t. Formally, we generate the simulations via

$$\mathbf{Z}_t^{j,(i)} = \underbrace{\alpha_S^j \, \text{B-spline}(t) + \alpha_g^j \, g^{j,(i)}(t)}_{\text{endogenous}} + \underbrace{\alpha_f^j \, f_Z^j(\mathbf{X}_t^{(i)})}_{\text{exogenous}} + \underbrace{\varepsilon_t}_{\text{noise}} \tag{70}$$

with $\varepsilon_t \sim N(0, 0.005^2)$ and where α_S^j, α_g^j, and α_f^j are weight parameters. Further, B-spline(t) is sampled from a mixture of three cubic splines (one with rapid decline during all intensive care unit stay, one with a mild decline, and one stable); $g^{j,(i)}(\cdot)$ is sampled independently for each patient from Gaussian process with Matérn kernel; and $f_Z^j(\cdot)$ is sampled from a random Fourier features (RFF) approximation of an Gaussian process (Hensman et al., 2017). Here, we specifically use RFF as they circumvent the need for tedious Cholesky decomposition when sampling random functions at many points of time-varying feature space $\mathbb{R}^{d_x}$. By combining all three components, we aim to simulate outcomes, which have endogeneous dependencies with different resolutions (global trends of B-splines and local correlation structure of Gaussian processes) and arbitrarily chosen exogeneous dependencies on other time-varying features.

Third, we sequentially simulate synthetic d_a binary treatments $\mathbf{A}_t^l$, $l = 1, \ldots, d_a$. We add confounding to the treatments by a subset of current time-varying covariates via a random function $f_Y^l(\mathbf{X}_t)$. Subsequently, we average of the subset of previous T_l treated outcomes $\bar{A}_{T_l}(\bar{\mathbf{Y}}_{t-1})$. Formally, we compute $\mathbf{A}_t^l$ via

$$p_{\mathbf{A}_t^l} = \sigma\Big(\gamma_A^l \bar{A}_{T_l}(\bar{\mathbf{Y}}_{t-1}) + \gamma_X^l f_Y^l(\mathbf{X}_t) + b_l\Big), \tag{71}$$

$$\mathbf{A}_t^l \sim \text{Bernoulli}\left(p_{\mathbf{A}_t^l}\right), \tag{72}$$

where $\sigma(\cdot)$ is the sigmoid activation, γ_A^l and γ_X^l are confounding parameters, b_l is a fixed bias, and $f_Y^l(\cdot)$ is sampled from an RFF approximation of a Gaussian process (similar to $f_Z^j(\cdot)$).

Fourth, we apply treatments to the untreated outcomes. For this, we set $\mathbf{Y}_1 = \mathbf{Z}_1$. Each treatment l is modeled so that it has a long-lasting effect on some outcome j, with maximal additive effect β_{lj} right after application. Here, we assume that the treatment has an effect within a time window $t - w^l, \ldots, t$. We further assume that the effect size of treatments is subject to an inverse-square decay over time. We also scale the effect by the probability $p_{\mathbf{A}_t^l}$. Afterward, the effects of multiple treatments are aggregated by taking the minimum across the treatment effects. Formally, we model this via

$$E^j(t) = \sum_{i=t-w^l}^{t} \frac{\min_{l=1,\ldots,d_a} \mathbb{1}_{[\mathbf{A}_i^l=1]} p_{\mathbf{A}_i^l} \beta_{lj}}{(w^l - i)^2}, \tag{73}$$

where β_{lj} is the maximum effect size of treatment l. This is either constant for all the outcomes j, or zero, so that the treatment does not influence the outcome.

Fifth, we combine the above. That is, we simply add the simulated treatment effect $E^j(t)$ to untreated outcome; i. e.,

$$\mathbf{Y}_t^j = \mathbf{Z}_t^j + E^j(t). \tag{74}$$

Sixth, we generate our semi-synthetic dataset based on the above simulator. For exact values of all simulation parameters, we refer to code implementation. After simulating three synthetic binary treatments ($d_a = 3$) and two synthetic outcomes ($d_y = 2$), we split the cohort of 1,000 patients into train/validation/test subsets via a 60% / 20% / 20 % split. For one-step-ahead prediction, we then simulate all $2^3 = 8$ counterfactual outcomes. For multiple-step-ahead prediction with $\tau_{\text{max}} = 10$, we sample 10 random trajectories for each patient/time step.

K.2. Experimental Details

Hyperparameter tuning. We perform hyperparameter tuning separately for all models. For this, we use the 200 factual patient time-series from the validation subset. Details are in Appendix H.

Performance measurement: We retrain the models on five simulated datasets with different random seeds (random seeds for sampling from Gaussian processes are kept the same). We then report the averaged root mean square error (RMSE) on the test set, that is, for hold-out data. RMSE is calculated for standardized outcomes.

L. Details on Experiments with Real-World Data

L.1. Data

Similarly to the semi-synthetic data in Appendix K, we make use of the MIMIC-extract following a standardized preprocessing pipeline (Wang et al., 2020). The data gives measurements from intensive care units aggregated at hourly levels. We used forward and backward filling for missing values and did standard normalization of all the continuous time-varying features.

We use the same 25 vital signs ($d_x = 25$) and the 3 static features (also one-hot-encoded for categorical features, $d_v = 44$) as in the semi-synthetic experiments. Both time-varying covariates and static features serve as potential confounders. We use two binary treatments ($d_a = 2$): vasopressors and mechanical ventilation. We then estimate the factual outcome ($d_y = 1$): (diastolic) blood pressure. Here, it is known that this may be positively or negatively affected by vasopressors and mechanical ventilation, thus raising the question for clinical practitioners of how a patient trajectory may look like when such treatment is applied.

For our experiments, we selected a cohort of 5,000 patients, randomly chosen from the patients with intensive care unit (ICU) stays of at least 30 hours. For the selected cohort, we cut off ICU stays at 60 hours. We then split the cohort of 5,000 patients with a ratio of 70%/15%/15% into train/validation/test subsets. We varied the implementation according to the projection horizon τ. (i) For the one-step-ahead prediction, we used all trajectories in the test set. (ii) For a τ-step-ahead prediction with $\tau \geq 2$, we proceed as follows. Let $\tau_{\text{max}} \geq \tau$ denote the maximum projection horizon. In our experiments, $\tau_{\text{max}} = 5$. We then extract all sub-trajectories with a length of at least $\tau_{\text{max}} + 1$ with a rolling origin, where we remove vital signs from time steps $1, \ldots, T^{(i)} - \tau_{\text{max}} + 1$, respectively. We then make predictions but where a looking-ahead is prevented due to masking. Later, we report only the performance for the τ-step-ahead prediction.

L.2. Experimental Details

Hyperparameter tuning. We perform hyperparameter tuning separately for all models. For this, we use the 750 factual patient time-series from the validation subset. Details are in Appendix H.

Performance measurement: We retrain the models on five random sub-samples of the dataset with different random seeds. We then report the averaged root mean square error (RMSE) on the test set, that is, for hold-out data. RMSE is then unscaled to the original range with standard normalization parameters.

M. Runtime and Model Size Comparison

CT with a single-stage training also provides a decent speed up for training and inference in comparison to other methods. In Table 12, we compare the total runtime of experiments, averaged over all confounding levels $\gamma = 0, \ldots, 4$, for synthetic data. Among all neural models, our CT has the smallest runtime. Hence, our transformer architecture together with a single-stage training procedure with CDC loss not only improves the performance of counterfactual outcomes estimations but also achieves a substantial computational speed-up. In Table 13, we report the total number of trainable parameters for different models after hyperparameter tuning. For semi-synthetic and real-world data, CT turns out to be more parsimonious, than LSTM-based models.

Table 12. Runtime of experiments (all stages of training and inference) for both tasks of one- and τ-step-ahead prediction, averaged over different $\gamma = 0, \ldots, 4$ (lower is better). Total runtime includes data generation. Experiments are carried out on $1\times$ TITAN V GPU.

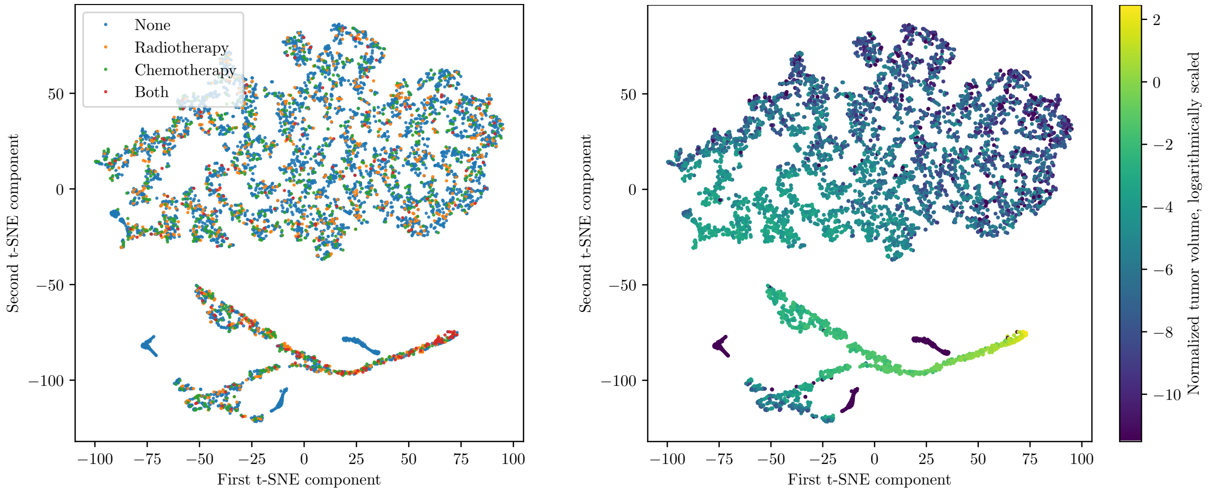
	Main stages of training & inference	Total runtime (in min)
MSMs	2 logistic regressions for IPTW & linear regression	3.5 ± 0.3
RMSNs	2 networks for IPTW & encoder & decoder	109.7 ± 2.3
CRN	encoder & decoder	75.3 ± 17.5
G-Net	single network & MC sampling for inference	118.0 ± 2.0
CT (ours)	single multi-input network	**13.5** ± **4.8**

Table 13. Total number of trainable parameters of models after hyperparameter tuning. Here, we distinguish (1) data using the tumor growth (TG) simulator (=experiments with fully-synthetic data), (2) data from semi-synthetic (SS) benchmark, and (3) real-world (RW) MIMIC-III data. The number is the sum of trainable parameters among all the sub-models for MSMs, RMSNs, and CRN.

	TG simulator					SS data	RW data
	$\gamma = 0$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$		
MSMs	<100					3K	1K
RMSNs	20K	4K	23K	21K	22K	477K	947K
CRN	4K	6K	8K	7K	8K	165K	219K
G-Net	3K	2K	3K	4K	3K	151K	310K
CT (ours)	11K	11K	10K	10K	10K	45K	69K

N. Visualization of Learned Representations

Figure 5(a,b) visualizes the t-SNE embeddings for the balanced representations of CT. Here, we use the validation set of the fully-synthetic data from the tumor growth simulator. Colors show the health outcome (tumor volume). As the plots show, we observe several regions where representations are indeed balanced, so that they appear non-predictive of the current treatment but expressive of the outcome. To this end, one can observe a continuous change in color (outcome). In severe cases, the points are colored in yellow when tumor size is comparatively large. As we can see, balancing then becomes challenging, as few patients with this condition receive no treatment.



(a) t-SNE embeddings of balanced representations with indicated current treatments

(b) t-SNE embeddings of balanced representations with indicated next outcomes

Figure 5. t-SNE embeddings of the balanced representations of CT. We display $N = 100$ patients from the fully-synthetic data (tumor growth simulator). Here: representations of the validation set ($\gamma = 4$), where each patient trajectory contains 60 time steps, thus displaying 6,000 embeddings. Note the logarithmic scale for the outcomes (in color).

O. Case Study: Importance of Subnetworks

In the following, we provide a case study for explainability. That is, we study the importance of different subnetworks. Such insights may help medical practitioners to ponder about the relevance of treatments for patient outcomes, or the relevance of time-varying covariates for patient outcomes.

To this end, we examine the role of using multiple cross-attention for information exchange between three subnetworks of CT. We informally define an importance score of subnetwork A, Y, or X as the difference in performance (e. g., with test RMSE) between full CT and CT with the correspondingly isolated subnetwork. Here, isolating a subnetwork means that we remove cross-attentions of the particular subnetwork. As such, it does not completely ignore the input sequence but only the interactions, as we still use sequences of all subnetworks representations at the latest stage of average pooling. Therefore, the importance score aims to explain how the connectivity of subnetworks via cross-attentions helps in estimating counterfactuals over time.

For our case study, we use the semi-synthetic benchmark and kept the same hyperparameters, as for the original CT. Figure 6 shows importance scores of each subnetwork for different prediction horizons τ. We observe that subnetwork processing time-varying covariates has the largest importance score (red bars). Interestingly, the importance score for the treatment subnetwork is close to zero for a small prediction horizon τ and grows only for larger prediction horizons. This thus has implications: it suggests *long-range* treatment effects.

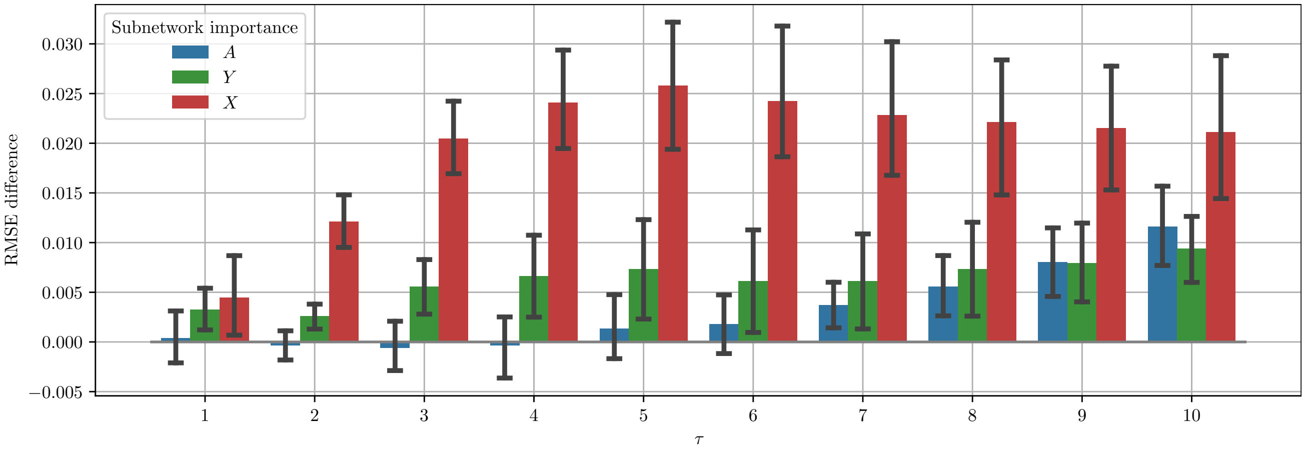


Figure 6. Subnetworks importance scores based on semi-synthetic benchmark (higher values correspond to higher importance of subnetwork connectivity via cross-attentions). Shown: RMSE differences between model with isolated subnetwork and full CT, means ± standard errors.