Master Thesis Data Science(4589)
20664A
2024-2025

Master of Statistics and Data Science - Year 2
Hasselt University

Assessing Effectiveness of Causal Transformers in Heterogeneous Treatment Effect Estimation Over Time

Student Name: Charles Muiruri (2365043)

Submission Date: June 17th, 2025

Supervisor:
Prof. Dr. Inigo Bermejo Delgado

Abstract

Background: Children recovering from acute illness in low, and middle-income countries are at high risk of poor post-discharge outcomes, including inadequate weight gain. Acute illness can lead to nutritional depletion, reduced appetite, and impaired growth, making early recovery a critical window for intervention. While nutritional care is commonly provided during and after hospitalization, its effectiveness in promoting post-illness weight gain under routine conditions remains unclear.

Objectives: To assess the effectiveness of Causal Transformers in estimating heterogeneous treatment effects (HTEs) of nutrition care on weight gain in young children over time, and to benchmark their performance against established causal inference models.

Methods: Data from a longitudinal study (CHAIN) involving 3,101 children aged 2–23 months with acute illness and four follow-up visits were used to assess weight progression based on the nutrition care received. Exploratory data analysis (EDA) evaluated missingness patterns, covariate distributions, treatment exposure, and weight outcomes. A synthetic dataset with known counterfactuals was also generated to enable model validation. The Causal Transformer (CT) was implemented to model time-varying treatment effects using self-attention and temporal masking, with an extended version incorporating Inverse Probability of Censoring Weighting (CT-IPCW) to handle informative dropout. Comparator models included the Recurrent Marginal Structural Network (RMSN), Counterfactual Recurrent Network (CRN), and baseline Marginal Structural Model (MSM). Model performance was evaluated using Precision in Estimation of Heterogeneous Effect (PEHE), Root Mean Squared Error (RMSE), policy risk, and prediction loss, at both overall and timepoint-specific levels.

Results and Conclusions: On simulated data, deep learning models (CT-IPCW, CRN, RMSN) consistently outperformed MSM across all metrics, with CT-IPCW showing improved robustness to censoring. Sample size scaling experiments confirmed improved model stability and accuracy with larger sample sizes. Individual Treatment Effect scatterplots revealed clustering artifacts in some models, warranting further refinement. On real-world data, deep learning models again outperformed MSMs, though performance degraded due to data complexity. IPCW was particularly beneficial on real data, reducing policy risk and improving estimation at later timepoints. These findings suggest that Causal Transformers, particularly in combination with IPCW, offer a promising approach for informing personalized interventions such as nutritional care for acutely ill children.

Key Words: Heterogeneous Treatment Effects, Acute Illness, Nutrition, Deep Learning, Causal Transformer, Longitudinal Data, Informative Censoring, Causal Inference

---

Acknowledgments

We gratefully acknowledge the CHAIN Network and the research teams across study sites in sub-Saharan Africa and South Asia for their contribution in designing, collecting, and curating the longitudinal dataset used in this study. Their work, as reported in the CHAIN cohort study, has provided a vital resource for understanding child recovery and post-discharge outcomes.

Contents

1 Introduction

1.1 Background and Motivation

Recent years have seen a surge in large observational healthcare datasets capturing longitudinal patient trajectories. These data offer significant potential for informing clinical decisions beyond one-size-fits-all treatment paradigms, particularly through estimation of heterogeneous treatment effects (HTEs), which quantify how treatment responses vary across individuals and contexts [1, 2].

In child nutrition and global health, HTEs are especially relevant given treatment efficacy is influenced by age, sex, comorbidities, and socioeconomic context [3]. Average treatment effects (ATEs) often obscure such differences, leading to potential overtreatment or undertreatment [4]. Machine learning (ML) methods, particularly deep learning architectures like the Transformer [5], offer new tools to model complex temporal dependencies and counterfactual outcomes. However, challenges such as informative censoring and time-dependent confounding remain.

1.2 Problem Statement

Despite their promise, the performance of Transformer-based models for HTE estimation under realistic longitudinal healthcare settings remains underexplored. Handling of censoring is an open methodological question, and the sample size requirements for reliable HTE estimation are poorly characterized, critical gaps for child nutrition cohorts such as CHAIN [3].

1.3 Objectives of the Study

The overarching aim of this study is to benchmark the performance of modern ML architectures for estimating HTEs in longitudinal child nutrition data. Specifically, this study pursues the following objectives:

- To implement and compare multiple ML-based and regression-based models for HTE estimation under longitudinal data settings;
- To investigate the impact of **censoring** on model performance and evaluate methods for handling censoring (e.g., inverse probability of censoring weights);
- To assess the **sample size requirements** and sensitivity of Causal Transformers and related models for reliable HTE estimation;
- To provide a reproducible simulation framework aligned with the structure of real-world child nutrition cohorts such as CHAIN.

1.4 Research Questions

The study is guided by the following research questions:

1. **Primary question**: How well do Causal Transformers estimate Heterogeneous Treatment Effects (HTEs)?
2. **Secondary question 1**: How do Causal Transformers deal with censoring in longitudinal data?
3. **Secondary question 2**: Do different sample sizes affect the performance of the Causal Transformer?

1.5 Significance of the Study

By systematically evaluating the capabilities and limitations of Causal Transformers and comparator models (e.g., RMSN [6], CRN [6], Marginal Structural Models [7]), this study aims to advance the practical understanding of ML-based HTE estimation in healthcare.
The findings will inform both methodological development and applied practice in child nutrition and other domains with similar data challenges. Given the critical importance of optimising interventions for vulnerable children worldwide [3], reliable estimation of HTEs is a key step towards precision public health.

Quote: "Identifying which children benefit most from interventions could substantially improve child survival and growth" [3].

Overall, this thesis contributes to a growing body of work at the intersection of causal inference, machine learning, and global child health.

2 Data Description

2.1 Study Design and Population

This study utilizes observational longitudinal data from the CHAIN (Childhood Acute Illness and Nutrition) Network cohort, specifically the sub-cohort analyzed in the CHAIN Follow-Up Study [3]. The CHAIN cohort consists of children recruited at nine hospitals across sub-Saharan Africa and South Asia between 2016 and 2019. The overarching objective was to investigate pathways of child mortality and recovery following hospitalization for acute illness in low- and middle-income country (LMIC) settings.

Children were eligible if they were aged 2–23 months at admission and hospitalized for a broad range of common childhood illnesses. Children with known terminal illness, recent hospitalization, or those discharged against medical advice were excluded. The primary CHAIN cohort followed children for 6 months post-discharge to assess mortality risk and morbidity trajectories [3].

For this thesis, we focus on a sub-cohort of children enrolled in the CHAIN Follow-Up Study [3], which further examined growth outcomes and nutritional recovery post-discharge. Data used in this project include detailed anthropometric measurements, nutritional status, treatment exposures, and health outcomes over time. Sites contributing data span diverse LMIC contexts in Kenya, Uganda, Burkina Faso, Bangladesh, Pakistan, and Malawi.

2.2 Variables and Time Structure

Children were followed longitudinally at predefined intervals: baseline (hospital discharge, referred to as "time-point 0"), day 45, day 90, and day 180. At each visit, information was collected on:

- **Demographics and baseline characteristics:** age in months, sex, country, site, and region.
- **Anthropometric indicators:** weight gain as percentage change from baseline (`pct_weight_gain`), height-for-age, and mid-upper arm circumference (MUAC).
- **Health status:** presence of poor feeding, acute illness, and missed visits.
- **Treatment exposure:** participation in outpatient feeding programs (`onoutpfeed_prog`) and binary treatment indicator (`binary_treatment`).

- **Nutritional counseling and care practices:** whether caregivers received nutritional counseling at discharge.
- **Mortality and censoring:** follow-up was right-censored if a child missed a visit or was lost to follow-up.

The dataset includes both observed outcomes and counterfactual placeholders, enabling the application of causal inference methods such as Marginal Structural Models (MSMs) and Causal Transformers [8].

2.3 Outcome and Treatment Definition

The primary continuous outcome is **percent weight gain from baseline** (`pct_weight_gain`) over follow-up visits. This serves as a proxy for nutritional recovery and growth, consistent with established literature [9, 10].

The primary binary treatment variable (`binary_treatment`) encodes whether the child received a targeted outpatient feeding program (e.g., RUTF/RUSF) during follow-up. This variable is constructed based on treatment program records harmonized across study sites.

2.4 Real vs Simulated Data

To benchmark causal inference models and validate estimation procedures, both **real** and **simulated** datasets were used:

- **Real data:** The `df_trial` dataset includes approximately 10,000 child-level trajectories derived from the CHAIN Follow-Up Study [3]. It reflects realistic patterns of treatment assignment, outcome evolution, and censoring.
- **Simulated data:** A custom simulation pipeline was developed to generate synthetic longitudinal data aligned with the CHAIN cohort structure. Simulated datasets (`sim_5000`, `sim_10000`, `sim_50000`) were constructed to have known ground-truth Individual Treatment Effects (ITEs) and explicitly controlled time-varying confounding.

This dual data approach allows us to assess model performance on data where the true counterfactual outcomes are known (simulation) and where clinical relevance and real-world complexity are preserved (real data).

Table 1: Overall Summary by Timepoint

Variable	Missing	Overall	0	45	90	180
n		10966	2919	2743	2662	2642
sex = Female		4726 (43.1)	1262 (43.2)	1185 (43.2)	1147 (43.1)	1132 (42.8)
sex = Male		6240 (56.9)	1657 (56.8)	1558 (56.8)	1515 (56.9)	1510 (57.2)
site = Banfora		1541 (14.1)	398 (13.6)	387 (14.1)	382 (14.4)	374 (14.2)
site = Blantyre		1044 (9.5)	312 (10.7)	270 (9.8)	229 (8.6)	233 (8.8)
site = Dhaka		1509 (13.8)	386 (13.2)	376 (13.7)	374 (14.0)	373 (14.1)
site = Kampala		1716 (15.6)	447 (15.3)	430 (15.7)	423 (15.9)	416 (15.7)
site = Karachi		1263 (11.5)	343 (11.8)	311 (11.3)	308 (11.6)	301 (11.4)
site = Kilifi		863 (7.9)	233 (8.0)	215 (7.8)	204 (7.7)	211 (8.0)
site = Matlab		1220 (11.1)	311 (10.7)	303 (11.0)	303 (11.4)	303 (11.5)
site = Migori		877 (8.0)	233 (8.0)	224 (8.2)	215 (8.1)	205 (7.8)
site = Nairobi		933 (8.5)	256 (8.8)	227 (8.3)	224 (8.4)	226 (8.6)
region = South East Asia		3992 (36.4)	1040 (35.6)	990 (36.1)	985 (37.0)	977 (37.0)
region = Sub-Saharan Africa		6974 (63.6)	1879 (64.4)	1753 (63.9)	1677 (63.0)	1665 (63.0)
agemons, mean (SD)	0	14.0 (6.1)	11.5 (5.7)	13.0 (5.7)	14.5 (5.6)	17.4 (5.7)
weight, mean (SD)	860	7.7 (1.9)	6.8 (1.9)	7.5 (1.7)	8.0 (1.7)	8.7 (1.7)
height, mean (SD)	867	71.0 (7.2)	68.5 (7.6)	70.0 (6.8)	71.6 (6.5)	74.6 (6.1)
muac, mean (SD)	859	12.8 (1.6)	12.1 (1.7)	12.8 (1.5)	13.1 (1.4)	13.5 (1.3)

Table 2: Summary by Binary Treatment and Timepoint

Variable	Missing	Overall	0		45		90		180	
			T=0	T=1	T=0	T=1	T=0	T=1	T=0	T=1
n			1608	131	1226	513	1359	380	1492	247
sex = Female			712 (44.3)	58 (44.3)	544 (44.4)	226 (44.1)	593 (43.6)	177 (46.6)	650 (43.6)	120 (48.6)
sex = Male			896 (55.7)	73 (55.7)	682 (55.6)	287 (55.9)	766 (56.4)	203 (53.4)	842 (56.4)	127 (51.4)
site = Banfora			348	39	271	116	319	68	335	52
site = Blantyre			228	31	184	75	191	68	214	45
site = Kampala			424	6	293	137	355	75	402	28
site = Karachi			9	NA	9	NA	9	NA	9	NA
site = Kilifi			189	16	163	42	170	35	176	29
site = Migori			220	4	178	46	181	43	195	29
site = Nairobi			190	35	128	97	134	91	161	64
region = South East Asia			9	NA	9	NA	9	NA	9	NA
region = Sub-Saharan Africa			1599	131	1217	513	1350	380	1483	247
age_group = 0-6 months			279	3	155	13	57	4	0	0
age_group = 12-18 months			468	47	363	186	464	135	516	90
age_group = 18-24 months			309	29	257	138	324	98	437	75
age_group = 24-30 months			0	0	44	17	120	21	294	37
age_group = 6-12 months			552	52	407	159	394	122	245	45
feed_outcome = MAM			410	27	202	191	164	162	151	102
feed_outcome = Normal			662	4	940	146	1116	108	1285	91
feed_outcome = SAM			536	100	84	176	79	110	56	54

3 Methodology

3.1 Exploratory Data Analysis (EDA)

A comprehensive exploratory data analysis (EDA) was conducted to assess the structure of the data, handle missingness, and evaluate the plausibility of assumptions required for causal inference using longitudinal observational data [1, 6, 11]. The EDA followed a structured approach comprising initial exploration, handling of missing data, definition of the outcome variable, outcome-aware analyses, and causal inference–guided diagnostics.

3.1.1 Initial Data Exploration

Initial data exploration focused on characterizing the cohort and understanding key data characteristics:

- Distribution of demographic, clinical, and outcome variables.
- Temporal structure: completeness and spacing of repeated measurements.
- Patterns of missingness across covariates and outcomes.

Summary statistics, univariate distributions, and bivariate relationships were examined visually and numerically.

3.1.2 Handling Missing Data and Imputation

Patterns of missingness were systematically assessed. The majority of missingness occurred in baseline covariates and longitudinal outcome measurements.
Multiple Imputation by Chained Equations (MICE) [12, 13] was applied to handle missing baseline covariates. MICE assumes a Missing at Random (MAR) mechanism [14], whereby the probability of missingness depends only on observed data:

$$\mathbb{P}(R_j = 1 \mid X_{\text{obs}}, X_{\text{mis}}) = \mathbb{P}(R_j = 1 \mid X_{\text{obs}})$$

where R_j is the missingness indicator for variable X_j, and $X_{\text{obs}}, X_{\text{mis}}$ denote observed and missing values of the covariates, respectively.
MICE proceeds by iteratively fitting models for each variable conditional on the others, updating missing values via:

$$X_j^{(t+1)} \sim \mathbb{P}(X_j \mid X_{\setminus j}^{(t)})$$

where X_j is the variable being imputed, and $X_{\setminus j}$ are all other variables.
Importantly, outcome variables subject to right censoring were not imputed, to avoid introducing bias. Outcomes were left missing beyond the point of censoring.

3.1.2.1 Missing Data Patterns and Definition of Censoring Patterns of missing outcome data across visits were systematically explored. These patterns were used to inform the definition of right censoring: participants with intermittent missingness (e.g., missing intermediate visits but returning later) were excluded to preserve a monotonic censoring structure suitable for temporal modeling. Participants were treated as right-censored at the first visit they missed, with all subsequent observations excluded from analysis.

To evaluate the robustness of the MAR assumption, a sensitivity analysis was performed to assess potential violations of MAR, specifically, to explore the impact of a Missing Not At Random (MNAR) mechanism.

Following the delta-adjustment sensitivity framework outlined in [15, 16], artificial shifts were applied to the imputed values of the outcome variable (e.g., weight gain), simulating MNAR scenarios of the form:

$$X_{\text{shifted}} = X_{\text{imputed}} + \delta$$

where δ is a fixed shift parameter applied to all imputed values. Multiple values of δ were explored, and resulting treatment effect estimates were compared across scenarios to assess sensitivity of conclusions to plausible MNAR deviations.

3.1.3 Outcome Variable Definition

The primary outcome variable, percent weight gain, was defined as the percent change in weight from baseline to each subsequent visit. The variable was computed as:

$$\text{Percent Weight Gain} = \frac{W_{\text{visit}} - W_{\text{baseline}}}{W_{\text{baseline}}} \times 100$$

where W_{visit} is the weight of the participant at the given visit, and W_{baseline} is the participant's baseline weight. This derived outcome was computed after imputation of baseline covariates and served as the primary outcome in all causal modeling analyses.

3.1.4 Outcome-aware EDA

We conducted outcome-aware EDA to explore heterogeneity in outcome evolution and relationships between outcomes and covariates:

- Trajectories of primary and secondary outcomes over time.
- Outcome distributions stratified by treatment, site, region, age group, and clinical status.
- Visualization of time-varying covariates in relation to outcomes.

These analyses informed model specification and interpretation of treatment effect heterogeneity.

3.1.5 Causal Inference–Guided EDA

The plausibility of key causal inference assumptions was assessed via a dedicated Causal Inference–Guided EDA framework [1]. The workflow followed the sequence:

1. Specification of the target trial emulation.
2. Assessment of positivity (overlap).
3. Assessment of covariate balance to inform exchangeability.
4. Assessment of censoring and dropout.
5. Sensitivity analysis.

3.1.5.1 Target Trial Emulation To emulate a hypothetical target trial [1], the data were explicitly structured as follows:

- The analytic cohort was restricted to participants aged between 2 and 24 months at baseline.
- Certain covariates were engineered to reflect the desired target trial specification:
  - *Age group* was defined as a categorical variable derived from continuous age.
  - *Acutely ill* was defined as an indicator variable constructed from multiple existing clinical variables.
  - Additional variables were similarly derived to align with the trial's intended baseline covariate structure.
- Only participants with complete visit sequences at the predefined timepoints were retained for analysis.
- Covariates were defined prior to treatment initiation.
- Outcome trajectories and censoring patterns were explicitly evaluated to ensure consistency with the target trial estimand.

The target estimand was the average causal effect of treatment on outcomes over time, conditional on baseline covariates, under assumptions of no unmeasured confounding, no informative censoring conditional on covariates, and sufficient positivity.

3.1.5.2 Positivity Propensity scores (PS) [1,17] were estimated using logistic regression models, where the probability of treatment assignment was modeled as a function of measured confounders:

$$e(X) = \mathbb{P}(T = 1 \mid X)$$

where T is the binary treatment indicator, and X represents baseline confounders.
The positivity assumption requires that:

$$0 < e(X) < 1 \quad \forall X$$

Distributional overlap in propensity scores between treatment groups was examined visually. Observations with extreme PS values were trimmed (typically < 0.005 or > 0.995) to restrict analysis to regions of sufficient overlap.

3.1.5.3 Covariate Balance as a diagnostic for Exchangeability: Inverse Probability of Treatment Weighting (IPTW) [7] was used to balance confounders across treatment groups. Stabilized IPTW weights were computed as:

$$w_i = \frac{p_T}{e(X_i)} \cdot \mathbb{I}(T_i = 1) + \frac{1 - p_T}{1 - e(X_i)} \cdot \mathbb{I}(T_i = 0)$$

In this stabilized inverse probability weighting scheme, p_T denotes the marginal probability of treatment in the sample, while $e(X_i) = \mathbb{P}(T_i = 1 \mid X_i)$ is the propensity score, the probability of receiving treatment conditional on covariates X_i. The indicator function $\mathbb{I}(T_i = t)$ selects the appropriate term based on whether the individual received treatment ($t = 1$) or control ($t = 0$). The stabilization by p_T helps reduce the variance of the weights compared to the unstabilized IPTW estimator.
Covariate balance was assessed using standardized mean differences (SMDs), defined as:

$$\text{SMD} = \frac{\bar{X}_1 - \bar{X}_0}{\sqrt{\frac{Var_1 + Var_0}{2}}}$$

where $\bar{X}_1, \bar{X}_0$ and Var_1, Var_0 are the weighted means and variances of the covariate in the treated and untreated groups, respectively. A threshold of 0.1 was used to indicate acceptable balance.

3.1.5.4 Censoring and Dropout Analysis: Censoring was assessed using a binary indicator for missed scheduled visits. Logistic regression was used to model the probability of censoring conditional on baseline [`site`, `region`] and time-varying covariates [`age_group`, `poor_feeding`, `acutely_ill`]:

$$\log\left(\frac{\mathbb{P}(C = 1 \mid X)}{1 - \mathbb{P}(C = 1 \mid X)}\right) = \beta_0 + \beta^T X$$

where C is the censoring indicator.

Strong associations between censoring and covariates indicated the potential for informative censoring. Consequently, inverse probability of censoring weights (IPCW) [7] were computed and incorporated in causal models where appropriate.

3.1.5.5 Sensitivity Analysis: A sensitivity analysis was performed to evaluate the robustness of results to potential violations of the Missing at Random (MAR) assumption underlying the multiple imputation process. Specifically, artificial shifts were applied to the imputed outcome values to simulate plausible Missing Not At Random (MNAR) scenarios, following [15, 16]:

$$X_{\text{shifted}} = X_{\text{imputed}} + \delta$$

where δ is a fixed shift parameter applied to all imputed values. Multiple values of δ were explored, and treatment effect estimates were examined across this range to assess sensitivity to MNAR deviations.

This sensitivity analysis provided an additional layer of robustness, ensuring that the causal conclusions were not unduly dependent on the MAR assumption.

3.2 Data Loading and Preparation

Both simulated and real datasets were formatted as **longitudinal panels**, where each row corresponds to an observation for an individual at a specific timepoint. Datasets included patient-level covariates, a binary treatment indicator, a continuous outcome, and corresponding counterfactual outcome columns.

Following the **Target Trial Emulation** framework [1], the true counterfactual outcomes were generated in simulation, and left as missing (`NaN`) in the real data to preserve a consistent structure.

Handling of counterfactuals.

For each individual i at time t, both the **factual outcome** $y_{i,t}^{\text{factual}}$ and the **counterfactual outcome** $y_{i,t}^{\text{counterfactual}}$ were stored in the dataset, such that:

$$y_{i,t}^{\text{factual}} = \begin{cases} y_{i,t}(1) & \text{if } T_{i,t} = 1 \\ y_{i,t}(0) & \text{if } T_{i,t} = 0 \end{cases}$$

$$y_{i,t}^{\text{counterfactual}} = \begin{cases} y_{i,t}(0) & \text{if } T_{i,t} = 1 \\ y_{i,t}(1) & \text{if } T_{i,t} = 0 \end{cases}$$

where $T_{i,t}$ is the binary treatment indicator. This allowed the models to be trained and evaluated on both factual and counterfactual targets when available.

Masking of censored observations.
As the datasets exhibit **right censoring**, a binary **mask** $\mathbf{1}_{\text{observed}_{i,t}}$ was computed for each individual and timepoint:

$$\mathbf{1}_{\text{observed}_{i,t}} = \begin{cases} 1 & \text{if the visit at time } t \text{ was observed} \\ 0 & \text{if the visit at time } t \text{ was censored or missing} \end{cases}$$

The mask was applied to all loss computations and evaluation metrics, ensuring that only observed data points contributed to optimization and reporting. This follows established best practices for modeling longitudinal data with informative dropout or right censoring [7, 12].

Data splitting and loading.
For each experiment, the dataset was split into **training**, **validation**, and **test** sets at the individual level, ensuring that no leakage occurred across splits. Data loaders provided full **patient trajectories**, maintaining temporal ordering across timepoints, enabling the use of sequential models such as RNNs and Transformers [6, 8, 18].

Categorical outcomes.
When modeling **categorical outcomes**, such as feeding outcome classes, outcomes were automatically encoded using label encoding [19]. Both factual and counterfactual categorical outcomes were prepared accordingly, maintaining consistency across model inputs and loss functions.

3.3 Simulation Design and Justification

To evaluate the performance of the Causal Transformer and comparator models, we simulate longitudinal data that mimics key patterns observed in real-world child nutrition cohorts. The simulation framework is grounded in established scientific literature in pediatrics [3, 20], epidemiology [1], and causal inference methodologies for longitudinal data with time-varying treatments, missing data, and simulation-based evaluation [7, 12, 21].

3.3.1 Time-Varying Treatment Effects

Treatment effects in the simulated data are heterogeneous and evolve over time, reflecting real-world variability. For instance, younger children or those with higher morbidity burdens may respond differently to nutritional interventions. Time-varying treatment effects are supported by Marginal Structural Models [7, 22], which allow for proper causal interpretation when both treatment and confounders change over time.
The simulated Individual Treatment Effect (ITE) for the continuous outcome `percent weight gain` is defined as:

$$\begin{aligned} ITE_{\text{weight},i,t} = {} & 0.1 \cdot t + 0.5 \cdot \mathbb{1}_{\text{Female}} + 0.3 \cdot \mathbb{1}_{\text{Sub-Saharan Africa}} \\ & + 0.4 \cdot \mathbb{1}_{\text{Nutrition counseling}} + 0.6 \cdot \mathbb{1}_{\text{Poor feeding}} + 0.7 \cdot \mathbb{1}_{\text{Acutely ill}} + \epsilon_{i,t} \end{aligned}$$

$$\epsilon_{i,t} \sim \mathcal{N}(0, 0.5^2)$$

A similar ITE formula is used for the categorical outcome `feed_outcome` based on a latent index with additional noise:

$$\begin{aligned} ITE_{\text{feed},i,t} = {} & -1 + 0.15 \cdot t + 0.2 \cdot \mathbb{1}_{\text{Sub-Saharan Africa}} \\ & + 0.4 \cdot \mathbb{1}_{\text{Nutrition counseling}} + 0.8 \cdot \mathbb{1}_{\text{Poor feeding}} + 1.0 \cdot \mathbb{1}_{\text{Acutely ill}} + \nu_{i,t} \end{aligned}$$

$$\nu_{i,t} \sim \mathcal{N}(0, 0.3^2)$$

This ensures explicitly known heterogeneity in treatment effects, critical for HTE benchmarking [2, 6, 8].

3.3.2 Artificial Right Censoring

The simulation includes artificial right censoring, implemented as random dropout between timepoints 6 and 10. This mirrors patterns in longitudinal health studies where loss to follow-up, mobility, or mortality lead to incomplete observations. Proper handling of right censoring is essential to avoid biased effect estimates [12, 13] and enables the evaluation of models like the Causal Transformer under realistic, non-ideal conditions.

3.3.3 Categorical Outcome Generation

The categorical outcome, `feed_outcome`, is simulated using a logistic function over a latent health index. This index is influenced by covariates such as acute illness and poor feeding. Using thresholds on the latent score, we classify participants as Severely Malnourished (SAM), Moderately Malnourished (MAM), or Normal. This strategy aligns with ordinal regression approaches commonly used in nutritional surveillance [11, 19] and WHO guidelines on malnutrition [20].

3.3.4 Continuous Outcome Generation

The continuous outcome, `pct_weight_gain`, is modeled as a normally distributed random variable with a subject-specific treatment effect. This is consistent with linear growth models in pediatric epidemiology [10, 23]. It also facilitates evaluation of metrics like RMSE and PEHE in HTE estimation frameworks.

3.3.5 Covariate Influence on Treatment Effects

Covariates such as sex, region, nutritional counseling, poor feeding, and acute illness are used to model heterogeneity in treatment effects. These covariates are widely recognized in the literature as effect modifiers and risk factors [9, 24]. For instance, poor feeding is a direct barrier to nutritional recovery, while region captures structural inequalities in access to care and food.

3.3.6 Outcome Consistency with Real Data

To enable unified evaluation across real and synthetic data, the simulation aligns in structure with the real dataset. Both datasets contain factual outcomes and include placeholders for counterfactuals and individual treatment effects (ITEs), which are either simulated or estimated.

3.4 Modeling Approaches

In this study, we benchmark four modeling approaches for estimating time-varying heterogeneous treatment effects (HTEs) from longitudinal data: the Causal Transformer (CT), Recurrent Marginal Structural Network (RMSN), Counterfactual Recurrent Network (CRN), and a neural Marginal Structural Model (MSM). All models predict individual outcomes under both observed (factual) and counterfactual treatment trajectories and are evaluated using a common framework.

3.4.1 Causal Transformer (CT)

The Causal Transformer architecture [8] adapts the Transformer framework [25] for causal inference in longitudinal settings. It models the entire patient history through stacked self-attention layers, enabling the capture of complex temporal dependencies and interactions between covariates, treatments, and outcomes.
At each time point t, the model outputs predictions for the outcome under both treatment and control:

$$\widehat{Y}_{i,t}(1), \quad \widehat{Y}_{i,t}(0)$$

where $\widehat{Y}_{i,t}(1)$ and $\widehat{Y}_{i,t}(0)$ denote the predicted outcomes under treatment and control, respectively. The individual treatment effect (ITE) is computed as:

$$\widehat{\mathrm{ITE}}_{i,t} = \widehat{Y}_{i,t}(1) - \widehat{Y}_{i,t}(0)$$

Additionally, we incorporate the **Counterfactual Domain Confusion (CDC) loss**, which encourages alignment of the latent representations of the factual and counterfactual domains:

$$\mathcal{L}_{\mathrm{CDC}} = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathrm{cosine\_similarity}(h_i^{\mathrm{factual}}, h_i^{\mathrm{counterfactual}})$$

where h_i^{factual} and $h_i^{\mathrm{counterfactual}}$ are the learned representations under observed and counterfactual treatments. This regularization helps improve generalization to unseen treatment trajectories.

3.4.2 Recurrent Marginal Structural Network (RMSN)

The Recurrent Marginal Structural Network (RMSN) [26] models the patient history using a recurrent neural network (RNN) architecture, capturing the evolution of covariates and treatments over time. At each time point t, it produces outcome predictions under treatment and control:

$$\widehat{Y}_{i,t}(1), \quad \widehat{Y}_{i,t}(0)$$

Like CT, we optionally include the CDC loss to encourage domain alignment in the latent space.
RMSN provides a flexible architecture capable of handling time-varying confounding and complex temporal dynamics, consistent with the causal inference literature on longitudinal data [7, 22].

3.4.3 Counterfactual Recurrent Network (CRN)

The Counterfactual Recurrent Network (CRN) [6] builds on the RNN architecture used in RMSN but explicitly emphasizes learning representations that generalize across factual and counterfactual domains. It does so by incorporating domain confusion regularization, such as the CDC loss, directly into training.

The architecture and outputs are analogous to those of RMSN:

$$\widehat{Y}_{i,t}(1), \quad \widehat{Y}_{i,t}(0)$$

CRN aims to produce representations $h_{i,t}$ that are invariant to treatment assignment, improving counterfactual prediction performance in the presence of selection bias and time-varying confounding.

3.4.4 Marginal Structural Model (MSM)

The Marginal Structural Model (MSM) [7, 13] provides a principled statistical framework for estimating causal effects under time-varying confounding. In this study, we implement a neural version of MSM, applying a feedforward model independently at each time point to predict outcomes under treatment and control:

$$\widehat{Y}_{i,t}(1) = f_1(X_{i,t}), \quad \widehat{Y}_{i,t}(0) = f_0(X_{i,t})$$

where $X_{i,t}$ denotes the covariates at time t, and $f_1(\cdot)$ and $f_0(\cdot)$ are neural networks corresponding to treatment and control regimes.

MSM does not explicitly model temporal dependencies across time points, but instead relies on conditioning on the current covariate history. This is consistent with the original MSM framework where inverse probability weights (IPW) adjust for time-varying confounding [7, 22]. In our neural implementation, we train the model end-to-end without explicit weighting but provide this model as a baseline for comparison.

3.4.5 Common Modeling Framework

All models produce the following outputs for each individual i and time point t:

- Predicted factual outcome: $\widehat{Y}_{i,t}^{\text{factual}}$
- Predicted counterfactual outcome: $\widehat{Y}_{i,t}^{\text{counterfactual}}$
- Predicted individual treatment effect (ITE): $\widehat{\text{ITE}}_{i,t}$
- (Optionally) Latent representations for CDC loss regularization.

By ensuring this common output structure, we enable consistent computation of evaluation metrics across all models, including PEHE, ATE, Policy Risk, and RMSE (see Section 3.5).

3.4.6 Implementation Details

All models were implemented in PyTorch [27] following the guidelines of the official Causal Transformer repository [28] and corresponding papers [6, 8, 26]. We used a unified training pipeline and data loaders for all models to ensure fair and consistent evaluation.

3.5 Training and Evaluation Framework

All models were trained using a custom PyTorch-based training loop specifically adapted for the longitudinal causal inference setting [8, 18, 28]. At each iteration, mini-batches consisted of full individual patient trajectories across multiple timepoints. Observed and censored timepoints were identified through a binary mask, $\mathbf{1}_{\text{observed}_{i,t}}$, which was constructed as part of data preprocessing. The mask was applied to all loss computations as:

$$\text{Masked Loss} = \frac{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}} \cdot \text{loss}(\hat{y}_{i,t}, y_{i,t})}{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}}}$$

3.5.0.1 Loss functions. The models used **Mean Squared Error (MSE)** loss for continuous outcomes (e.g., percentage weight gain), which is appropriate for this regression task and ensures that prediction errors are penalized quadratically:

$$\text{MSE Loss} = \frac{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}} \cdot (\hat{y}_{i,t} - y_{i,t})^2}{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}}}$$

For models incorporating **Inverse Probability of Censoring Weights (IPCW)** [7, 12], an adjusted MSE was computed to correct for informative censoring:

$$\text{IPCW-MSE Loss} = \frac{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}} \cdot w_{i,t} \cdot (\hat{y}_{i,t} - y_{i,t})^2}{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}} \cdot w_{i,t}}$$

Additionally, for models with **latent representations** such as the **Causal Transformer** [8] and **CRN** [6], a **Counterfactual Domain Confusion (CDC)** loss [18] was used to encourage alignment between treated and untreated latent trajectories. The CDC loss penalizes dissimilarity in the latent representations of the factual and counterfactual domains:

$$\text{CDC Loss} = 1 - \frac{1}{N} \sum_i \text{cosine\_similarity}(h_i^{\text{factual}}, h_i^{\text{counterfactual}})$$

The total loss for these models was computed as:

$$\text{Total Loss} = \text{MSE Loss} + \lambda_{\text{CDC}} \cdot \text{CDC Loss}$$

with $\lambda_{\text{CDC}} = 0.1$ in all experiments, following the Causal Transformer reference implementation [28].

3.5.0.2 Evaluation metrics. To evaluate model performance in estimating individual and average treatment effects, we used the following metrics at both the epoch level and per timepoint:

- **Precision in Estimation of Heterogeneous Effects (PEHE):** measures the root mean squared error between predicted and true Individual Treatment Effects (ITE), providing a direct measure of HTE estimation accuracy:

$$\text{PEHE} = \sqrt{\frac{1}{N} \sum_{i,t} (\hat{\tau}_{i,t} - \tau_{i,t})^2}$$

- **Average Treatment Effect (ATE) error:** measures the absolute difference between the estimated and true average treatment effect across the population:

$$\text{ATE Error} = |\mathbb{E}[\hat{\tau}] - \mathbb{E}[\tau]|$$

  This evaluates the model's ability to capture global treatment effects.

- **Root Mean Squared Error (RMSE):** measures the accuracy of predicted factual outcomes compared to true factual outcomes:

$$\text{RMSE} = \sqrt{\frac{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}} \cdot (\hat{y}_{i,t} - y_{i,t})^2}{\sum_{i,t} \mathbf{1}_{\text{observed}_{i,t}}}}$$

  This allows comparison of outcome prediction accuracy across models.

- **Policy Risk:** measures the proportion of individuals for whom the model would recommend the wrong treatment based on the sign of the predicted ITE compared to the true ITE:

$$\text{Policy Risk} = \frac{1}{N} \sum_{i,t} \mathbf{1} \left[\text{sign}(\hat{\tau}_{i,t}) \neq \text{sign}(\tau_{i,t}) \right]$$

  Policy risk is particularly important from a clinical perspective, as it quantifies how often model-based treatment recommendations would be incorrect.

The training and evaluation pipeline allows for direct comparison of all models (Causal Transformer, CRN, RMSN, MSM) under a unified framework, controlling for right-censoring and temporal structure. Losses and evaluation metrics are consistent with prior work on **counterfactual outcome prediction** and **individual treatment effect estimation** in longitudinal data [1, 6–8, 18].

3.6 Hyperparameter Tuning and Cross-Validation

Hyperparameter tuning plays a crucial role in training deep learning models for causal inference. Poorly tuned models can lead to either underfitting or overfitting, both of which distort heterogeneous treatment effect (HTE) estimation. We therefore perform structured hyperparameter optimization for all deep learning models evaluated.

We employ **Bayesian Optimization**, implemented via the `Optuna` framework [29], to efficiently explore the hyperparameter space. Bayesian optimization adaptively selects promising hyperparameter configurations based on the outcomes of previous trials, in contrast to naive grid search which explores all configurations uniformly. This is especially important given the computational demands of evaluating models on large, longitudinal datasets with right censoring.

Each trial of Bayesian optimization evaluates a specific hyperparameter configuration using **10-fold cross-validation** on the simulated data. The primary objective metric is the **Precision in Estimation of Heterogeneous Effects (PEHE)** on the validation folds. We seek to minimize PEHE, as lower PEHE corresponds to more accurate estimation of individual-level treatment effects.

3.6.1 Choice of 10-fold Cross-Validation:

We adopt 10-fold cross-validation rather than 5-fold or leave-one-out cross-validation for several reasons. First, 10-fold cross-validation provides a robust balance between computational efficiency and low variance in the estimated generalization performance [30]. While 5-fold CV is faster, it introduces higher variance due to smaller training sets per fold—this is problematic in longitudinal data where modeling individual heterogeneity is critical.

Conversely, leave-one-out CV is computationally prohibitive given the large size of the simulated dataset ($N = 10,000$ individuals) and does not provide substantial gains in bias reduction compared to 10-fold CV [30].

Using 10-fold CV allows us to use 90% of individuals for training and 10% for validation in each fold, ensuring that models are evaluated across diverse patient trajectories. Since the goal is to estimate heterogeneous treatment effects across the population, having a larger number of folds helps average out the variation due to treatment timing, covariate shifts, and right censoring patterns.

3.6.2 Why Bayesian Optimization Instead of Grid Search:

We choose Optuna's Bayesian optimization framework over grid search for two reasons:

1. **Efficiency in High-dimensional Search Spaces:** Grid search suffers from the curse of dimensionality. Even with 5 hyperparameters, a reasonable grid resolution leads to thousands of combinations. This is impractical given that each trial involves 10-fold CV.

2. **Adaptivity:** Optuna's sampler leverages past trial results to prioritize promising regions of the hyperparameter space. This adaptive exploration is more sample-efficient than grid search, which treats all configurations equally. As a result, we can achieve better performance within a limited budget of tuning trials [29].

3.6.3 Hyperparameters Tuned

For the Causal Transformer (CT), we tune:

- Learning rate (η)
- Embedding dimension (d_{model})
- Number of Transformer layers
- Number of attention heads
- Dropout rate

For RNN-based models (RMSN, CRN), we tune:

- Learning rate
- Hidden dimension size
- Number of GRU layers
- Dropout rate

The Marginal Structural Model (MSM) baseline does not require tuning as it is implemented as a fixed linear model across time.

3.6.4 Objective Function:

The Optuna objective function performs the following steps:

1. Samples a hyperparameter configuration from the search space.
2. Runs 10-fold cross-validation using this configuration.
3. Returns the mean PEHE across the validation folds.

This ensures that hyperparameters are selected to optimize model performance on unseen patient trajectories.

4 Results

4.1 Exploratory Data Analysis

4.1.1 Individual profile

Figure 1 illustrates the longitudinal trajectories of percentage weight gain for a random sample of 100 participants in the real dataset. The plot highlights substantial heterogeneity in individual growth patterns across the follow-up period.

While the majority of children exhibited gradual positive weight gain, a notable subset showed stagnation or even negative weight gain at certain timepoints. This heterogeneity underscores the importance of estimating individualized treatment effects rather than relying solely on average trends.

Moreover, the wide dispersion of trajectories by day 180 reflects varying responses to nutritional and clinical interventions, as well as the potential influence of unmeasured factors such as household food security or comorbid conditions. These patterns motivate the use of flexible modeling approaches, such as the Causal Transformer, that can capture complex time-varying heterogeneity in treatment effects.

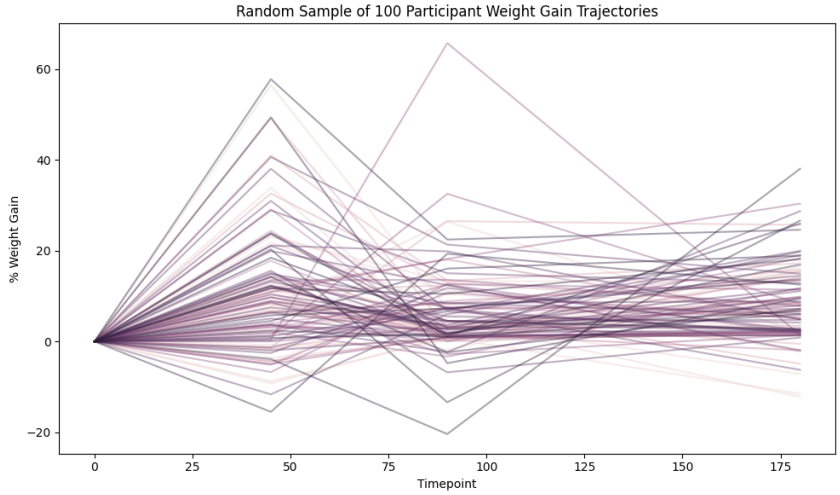

Figure 1: Individual % Weight Gain Trajectories(random 100 records)

4.1.2 Mean Structure

Figure 2 presents the mean percentage weight gain over time, stratified by treatment group (`binary_treatment`). Clear differences in growth trajectories between treated and untreated children are observed.

Children receiving nutritional treatment (`binary_treatment` = 1) exhibited faster initial weight gain during the early phase of follow-up (by day 45), achieving a higher peak in mean weight gain compared to untreated peers. This effect appears to attenuate between day 45 and day 90, followed by a more gradual increase toward day 180.

In contrast, untreated children demonstrated slower early weight gain and a flatter recovery trajectory. The persistence of this gap across timepoints suggests both short-term and potentially longer-term benefits of nutritional interventions, although the differential effect may vary across subgroups (to be further explored in HTE analysis).

Importantly, the non-parallel evolution of mean weight gain curves supports the existence of time-varying treatment effects. This motivates the use of modeling approaches, such as the Causal Transformer, that can

flexibly capture temporal heterogeneity in treatment response.

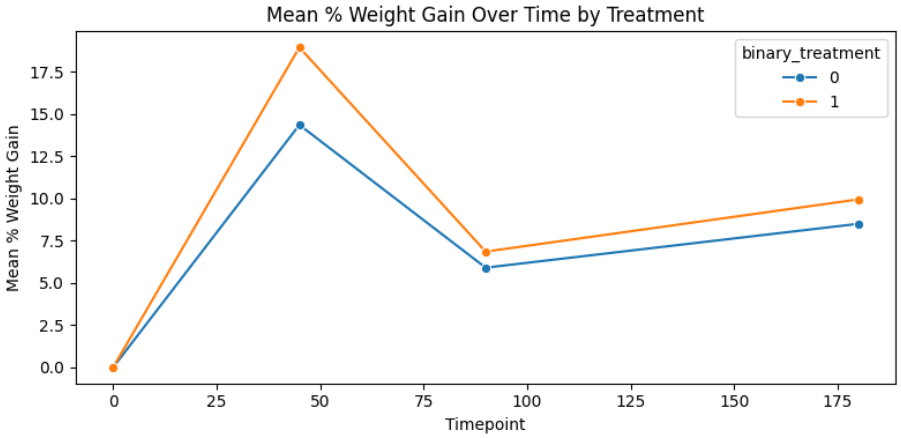

Figure 2: Mean structure of % Weight gain over Follow up timepoints

4.1.3 Missing Data Mechanism

Missing data patterns were carefully explored to assess their potential impact on causal inference analyses. Two complementary strategies were applied: (i) formal modeling of missingness predictors to assess the plausibility of a Missing At Random (MAR) mechanism, and (ii) sensitivity analysis to evaluate robustness of results under plausible Missing Not At Random (MNAR) scenarios.

4.1.3.1 Predictors of Missingness (MAR Modeling). For key variables subject to missingness, including anthropometric and clinical variables, logistic regression models were fitted to predict the probability of missingness as a function of observed covariates:

$$\text{logit}\big(\Pr(R_j = 1 \mid X)\big) = \beta_0 + \beta^T X$$

where R_j is the missingness indicator for variable j, and X includes `timepoint`, `sex`, `site`, `region`, `agemons`, and `group_adm`.

Significant associations between missingness and observed covariates were detected for most variables. Missingness strongly increased at later timepoints (e.g., coefficients for `timepoint` 45, 90, 180 all $p < 0.001$). Site and clinical severity (`group_adm`, `oedema`) were also associated with missingness in several variables.

These results are consistent with the hypothesis that dropout and missing data are driven by observable factors such as illness severity, geographic context, and study visit timing. No strong evidence was found of additional unobserved drivers of missingness beyond those included in the models. Thus, the MAR assumption appears plausible for this dataset.

4.1.3.2 MNAR Sensitivity Analysis. To further test robustness, an MNAR sensitivity analysis was conducted following the approach in [15, 16]. Specifically, imputed values of weight were artificially shifted in the range $\delta \in \{-0.5, -0.25, 0, +0.25, +0.5\}$, with downstream treatment effect models re-estimated under each scenario. This emulates violations of the MAR assumption by simulating a range of plausible bias magnitudes in the imputed outcomes.

Key findings:

- Across all δ values, treatment effect estimates remained consistent in both direction and magnitude.

- Associations between treatment and early weight gain (pct_gain_0_45) were robust, with coefficients stable across MNAR shifts.
- Covariate effect estimates (e.g., `oedema`, `vomiting`, `muac`) also demonstrated stability under MNAR perturbations.
- No reversal of key conclusions was observed even under extreme $\delta = \pm 0.5$ scenarios.

These results suggest that conclusions regarding treatment effects and covariate relationships are unlikely to be artifacts of the MAR assumption. The analysis provides additional confidence that observed results are robust to plausible deviations from MAR.

In summary, modeling of missingness mechanisms and MNAR sensitivity analyses both support the conclusion that missingness in this dataset is consistent with a MAR mechanism conditional on observed covariates. While non-random missingness cannot be definitively ruled out, causal inference results appear robust to plausible violations. This justifies the use of standard MAR-based multiple imputation and right censoring handling in the subsequent modeling stages.

4.1.4 Missing Data Pattern

Patterns of missingness across longitudinal timepoints were explored to inform censoring definitions and model specification. Table 3 summarizes observed visit completion patterns across the four scheduled follow-up timepoints (0, 45, 90, and 180 days).

Table 3: Summary of Visit Completion Patterns(O=Observed, M=Missed)

Category	**0**	**45**	**90**	**180**	**Count**
Completers	O	O	O	O	2597
Dropout pattern	O	M	M	M	151
Dropout pattern	O	O	M	M	73
Dropout pattern	O	O	O	M	50
Non-monotone pattern	O	M	M	O	10
Non-monotone pattern	O	M	O	M	3
Non-monotone pattern	O	M	O	O	12
Non-monotone pattern	O	O	M	O	23

4.1.4.1 Interpretation. The majority of participants (2597; ˜81%) completed all scheduled visits. A substantial fraction exhibited monotonic dropout patterns, consistent with classical right censoring:

- 151 participants (4.7%) attended only baseline and were lost before day 45.
- Additional 73 and 50 participants dropped out after day 45 or day 90, respectively.

In contrast, a smaller subset (non-monotone patterns; $n = 48$) demonstrated intermittent missingness, i.e., missed intermediate visits but returned at later ones.

4.1.4.2 Modeling decision. To preserve a well-defined censoring process suitable for longitudinal causal modeling, participants with **non-monotone patterns** were excluded from analysis. This ensures that right censoring is handled consistently, with outcome trajectories censored at first missing visit and subsequent data points not included.

This modeling choice aligns with the study's explicit focus on handling right censoring, while simplifying assumptions required by the Causal Transformer and comparator models. Participants censored at day 45, 90, or 180 were correctly handled via censoring indicators and appropriate weighting during model training and evaluation.

4.1.5 Outcome-aware EDA

4.1.5.1 Outcome Distributions and Covariate Relationships Outcome-aware exploratory data analysis was performed to characterize patterns of the primary outcome (`pct_weight_gain`) across key covariates and subgroups, and to inform subsequent causal modeling.

4.1.5.2 Distribution of % Weight Gain The distribution of percent weight gain exhibited substantial right skew, with most participants achieving modest positive gains and a smaller proportion showing large gains or weight loss (Figure 3). The median was slightly above zero, with a heavy tail of positive outliers.

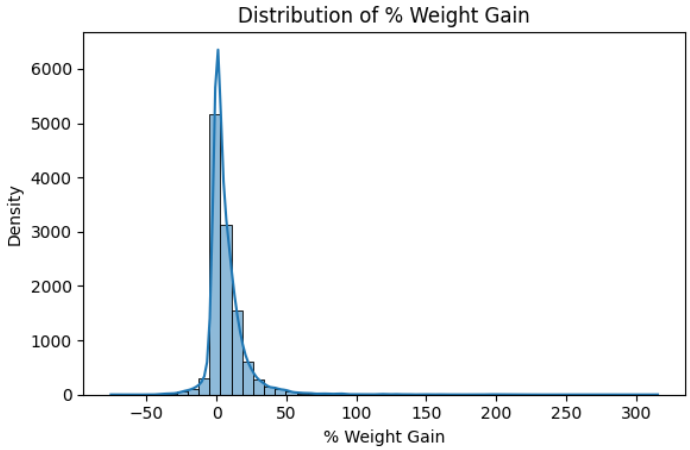

Figure 3: Distribution of % Weight Gain.

4.1.5.3 Treatment Group Differences Boxplots of % weight gain stratified by treatment group (Figure 4) suggested a positive shift in outcomes for treated participants (RUTF/RUSF), albeit with substantial overlap between groups. The median and interquartile range were slightly higher for the treated group, indicating potential treatment benefit but also substantial outcome heterogeneity.

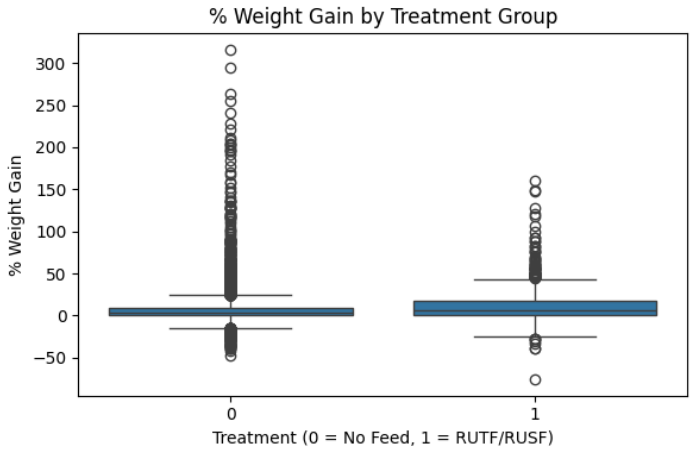

Figure 4: % Weight Gain by Treatment Group.

4.1.5.4 Site-wise Outcome Patterns Substantial between-site heterogeneity in % weight gain was observed (Figure 5). Some sites (e.g., Matlab, Banfora) had wider distributions and higher median gains, while others showed more modest gains. This reinforces the need to control for site effects and account for potential effect modification by site when estimating heterogeneous treatment effects (HTEs).

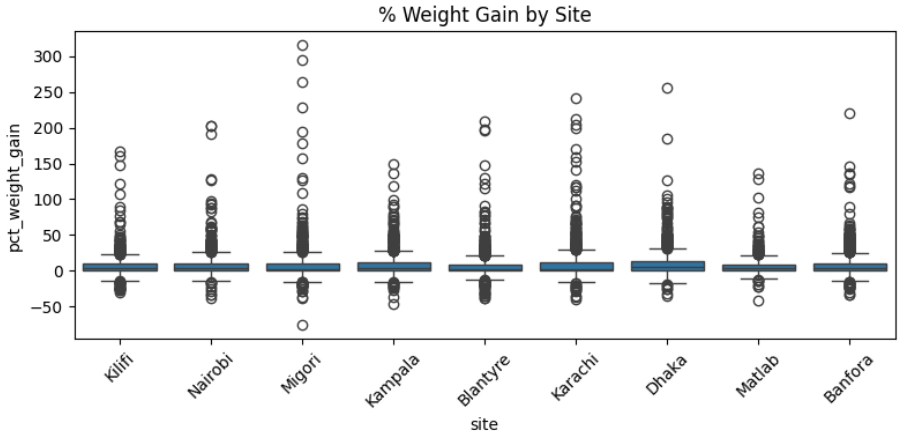

Figure 5: % Weight Gain by Site.

4.1.5.5 Relationship with Key Covariates Scatterplots with local regression overlays (Figure 6) revealed nuanced relationships between % weight gain and key continuous covariates:

- **Age (agemons):** Younger children tended to exhibit higher weight gains, consistent with greater catch-up growth potential.
- **MUAC:** Participants with lower MUAC at baseline showed more variable gains, with some achieving substantial catch-up growth.
- **Height and Weight:** Lower baseline anthropometric measures were associated with wider outcome dispersion, while higher measures were linked to more stable outcomes.

Treatment group trends were broadly parallel across covariate levels, but exploratory interactions were apparent, motivating the use of models capable of capturing such heterogeneity.

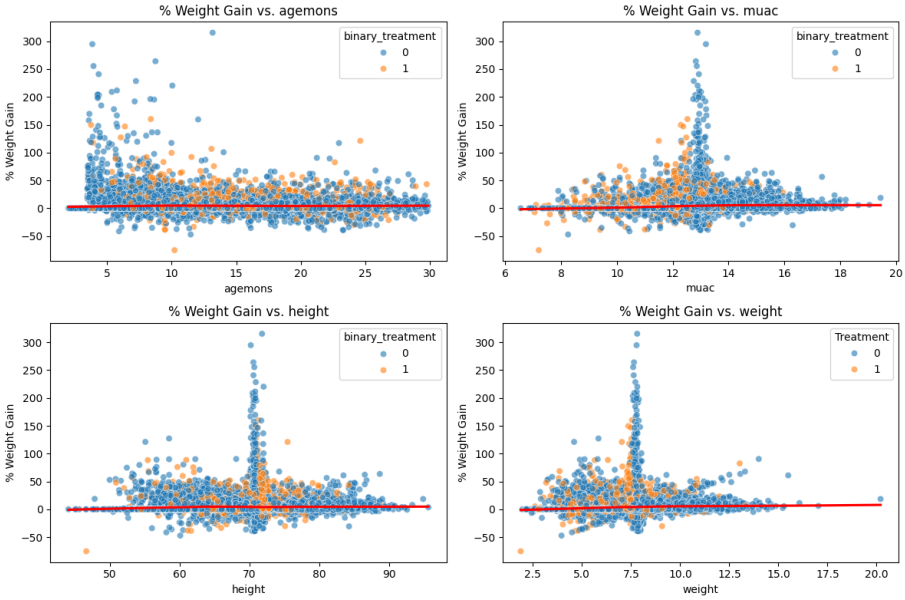

Figure 6: Relationships between % Weight Gain and Key Covariates by Treatment Group.

4.1.5.6 Preliminary Regression Model A simple ordinary least squares (OLS) model of weight as a function of treatment, covariates, site, and timepoint confirmed significant associations with multiple predictors. The model explained a large portion of outcome variance ($R^2 = 0.926$). Treatment was associated with a modest but statistically significant reduction in weight, though this likely reflects confounding by indication and is not causally interpretable. These results underscored the need for more sophisticated causal models.

Table 4: Summary of OLS Regression on Weight Gain.

Statistic	**Value**
Number of observations	11,676
R-squared (R^2)	0.926
Adjusted R^2	0.926
F-statistic	9709
p-value (F-statistic)	< 0.001
Treatment effect estimate	-0.0861 (significant, $p < 0.001$)

Outcome-aware EDA highlighted substantial outcome heterogeneity across participants, sites, and covariates. Visual and regression-based analyses pointed to complex interactions and non-linear relationships that motivated the use of flexible, temporally aware models such as the Causal Transformer and RNN-based architectures for HTE estimation.

4.2 Causal Inference EDA

4.2.1 Eligibility and Target Trial Emulation

To align the observational dataset with a target trial specification, a series of eligibility restrictions and cohort definitions were applied:

- Participants aged 2–23 months at baseline (Day 0) were included.

- Only participants with valid Day 0 and Day 45 visits were retained to allow valid calculation of the primary outcome (`pct_weight_gain`).
- Baseline covariates were merged to the longitudinal data, and follow-up was restricted to the four target timepoints (Day 0, 45, 90, 180).

4.2.2 Positivity

Propensity scores (PS) for receipt of RUTF/RUSF treatment were estimated using logistic regression with the following covariates: `sex`, `site`, `region`, `age_group`, `nutri_counsel_disch`, `poor_feeding`, and `acutely_ill`. Figure 7 shows the initial PS overlap. After trimming extreme PS values (PS ¡ 0.005 or ¿ 0.995) and restricting to participants with complete follow-up visits, overlap improved markedly (Figure 8).

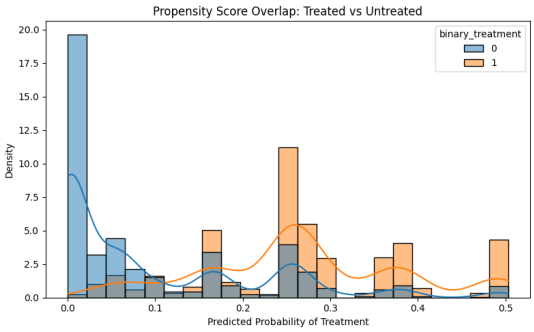

Figure 7: Initial Propensity Score Overlap: Treated vs Untreated.

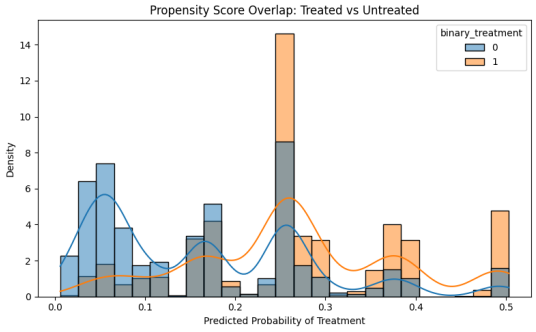

Figure 8: Propensity Score Overlap After Trimming and Restricting to Full Visits.

4.2.2.1 Diagnostics before and after trimming: The positivity assumption was reasonably met after trimming and restriction to participants with full follow-up.

Statistic	Before trimming	After trimming
Min	0.0001	0.0060
Max	0.5032	0.5032
Mean	0.1173	0.1823
25% percentile	0.0023	0.0626
75% percentile	0.2449	0.2593
% below 0.05	45.6%	14.9%
% above 0.95	0.0%	0.0%

4.2.3 Exchangeability and Covariate Balance

Stabilized inverse probability of treatment weights (IPTW) were computed using the estimated propensity scores. Covariate balance was assessed using standardized mean differences (SMD) computed both before and after IPTW.

As shown in Figure 9, all covariates achieved excellent balance after weighting (SMD < 0.1), except for *region*, whose post-weighting SMD remained marginally above the 0.1 threshold. In contrast, several covariates exhibited large imbalances before IPTW.

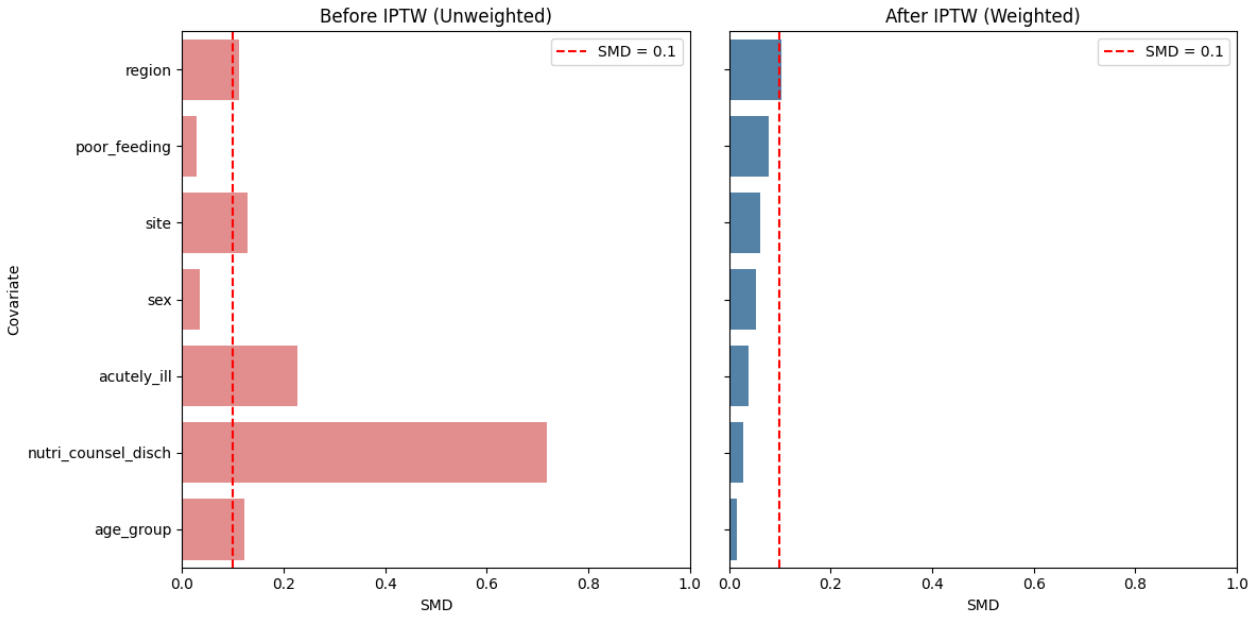

Figure 9: Standardized Mean Differences Before and After IPTW.

The exchangeability assumption is likely satisfied, conditional on the measured covariates, given the substantial improvement in balance achieved after weighting.

4.2.4 Censoring and Dropout Analysis

Censoring was operationalized as having a missed visit at a given timepoint. Patterns of missed visits were explored across key baseline covariates (site, region, age group, poor feeding, acutely ill).

4.2.4.1 Censoring patterns:

- Higher censoring rates were observed in certain sites (e.g., Blantyre, Nairobi, Migori).
- Younger age groups exhibited greater censoring risk.
- Poor feeding and acute illness were associated with *lower* censoring probability, potentially due to higher clinical follow-up needs.

4.2.4.2 Logistic regression of censoring risk: Key odds ratios (OR) from the model predicting missed visits:

- Site effects: Strong (OR up to 4.75 for Blantyre vs. reference).
- Age group effects: Increasing OR for younger age groups.
- Clinical condition effects: Protective (OR = 0.036 for acutely ill; OR = 0.211 for poor feeding).

Feature	Log Odds	OR
site_Blantyre	1.557	4.746
site_Nairobi	0.784	2.190
site_Migori	0.715	2.044
age_group_6-12 months	0.632	1.882
age_group_24-30 months	0.566	1.761
age_group_12-18 months	0.517	1.677
site_Kilifi	0.256	1.292
age_group_18-24 months	0.249	1.283
region_Sub-Saharan Africa	0.103	1.108
site_Kampala	-0.055	0.946
site_Karachi	-0.704	0.494
poor_feeding_Yes	-1.556	0.211
acutely_ill_Yes	-3.321	0.036

Table 5: Predictors of Missed Visit (Logistic Regression Odds Ratios).

Censoring is *not random*; it is strongly associated with site, age, and clinical condition. - This motivates the need to use inverse probability of censoring weights (IPCW) or alternative approaches in subsequent causal analyses.

4.2.5 Summary of Causal Inference EDA

- **Positivity:** Sufficient overlap of propensity scores was achieved after trimming and restricting to complete cases.
- **Exchangeability:** Excellent covariate balance achieved via IPTW; assumption likely holds conditional on observed covariates.
- **Censoring:** Non-random censoring patterns detected; models will need to address informative censoring to mitigate bias.

The dataset is well-prepared for causal modeling of heterogeneous treatment effects, with sufficient positivity, exchangeability under IPTW, and a clear plan to address censoring.

4.3 Estimating Heterogeneous Treatment Effects with Causal Transformers

4.3.1 Performance on Simulated Data

4.3.1.1 Training Dynamics

Model convergence was assessed by tracking training loss across epochs and across timepoints (Figure 10). Most models converged stably, with clear improvements over the first 10–15 epochs. Some divergence in training dynamics was observed across architectures:

- `CRN`, `RMSN`, and `CT-IPCW` converged quickly and exhibited stable loss across timepoints.
- `CT-Large` and `CT` showed slightly slower convergence.
- `MSM` had persistently higher loss, reflecting its more limited modeling capacity.

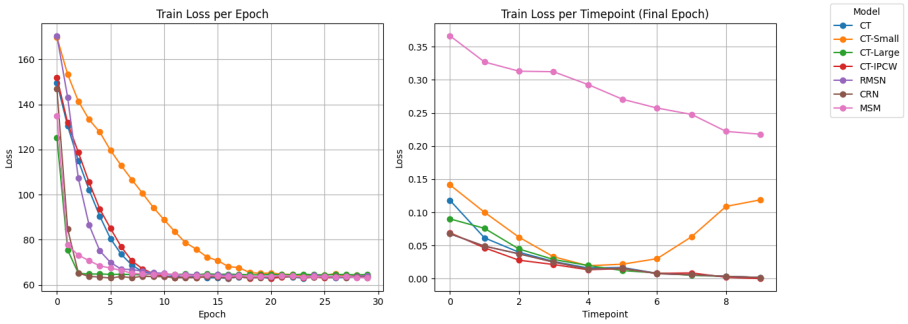

Figure 10: Training Loss per Epoch (left) and per Timepoint (right) on Simulated Data.

4.3.1.2 Performance over Epochs

Model performance was evaluated on the test set using four key metrics: PEHE, ATE, RMSE, and Policy Risk. Figure 11 shows the evolution of these metrics across epochs.

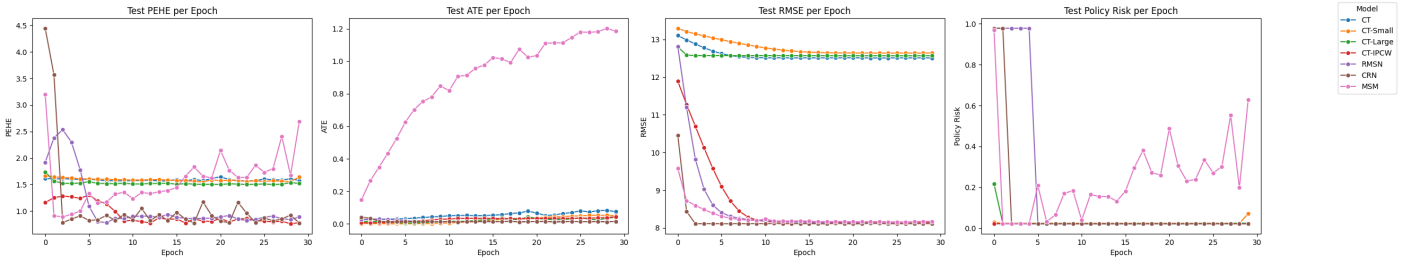

Figure 11: Test PEHE, ATE, RMSE, and Policy Risk across Epochs for Simulated Data.

4.3.1.3 Key observations: The following patterns were observed across epochs:

- `CRN`, `RMSN`, and `CT-IPCW` achieved the lowest PEHE and RMSE values.
- `MSM` showed substantially worse performance across all metrics.
- `Policy Risk` remained low for most deep learning models, while `MSM` had high and unstable policy risk.

4.3.1.4 Final Epoch Metrics

Table 6 summarizes model performance on train, validation, and test sets at the final epoch.

Model	Train PEHE	Val PEHE	Test PEHE	Train ATE	Val ATE	Test ATE	Train Policy Risk	Val Policy Risk	Test Policy Risk	Train RMSE	Val RMSE	Test RMSE
`CRN`	0.9238	0.7644	0.7742	0.0008	0.0026	0.0153	0.0228	0.0246	0.0222	7.9901	7.9588	8.1168
`MSM`	1.8445	2.6687	2.6903	1.1948	1.1457	1.1838	0.2828	0.6318	0.6279	8.0304	8.0307	8.1710
`RMSN`	0.8368	0.8929	0.8884	0.0002	0.0046	0.0141	0.0228	0.0246	0.0222	7.9920	7.9479	8.1178
`CT`	2.3134	1.5559	1.5771	0.3179	0.0556	0.0748	0.0297	0.0279	0.0253	8.3952	12.3911	12.5075
`CT-IPCW`	0.7576	0.7927	0.7816	0.0230	0.0684	0.0433	0.0211	0.0226	0.0216	7.9648	8.1295	8.1555
`CT-Large`	1.7063	1.5166	1.5209	0.3487	0.0597	0.0412	0.0290	0.0244	0.0237	8.0924	12.4441	12.5672
`CT-Small`	1.2093	1.6832	1.6409	0.4536	0.1116	0.0499	0.0699	0.0735	0.0715	8.0871	12.6246	12.6410

Table 6: Performance comparison of models on Train, Validation, and Test sets using PEHE, ATE, Policy Risk, and RMSE metrics for the simulated data. Columns are grouped by metric.

4.3.1.5 Temporal Performance: Timepoint-Wise Metrics

Figure 12 presents model performance by timepoint.

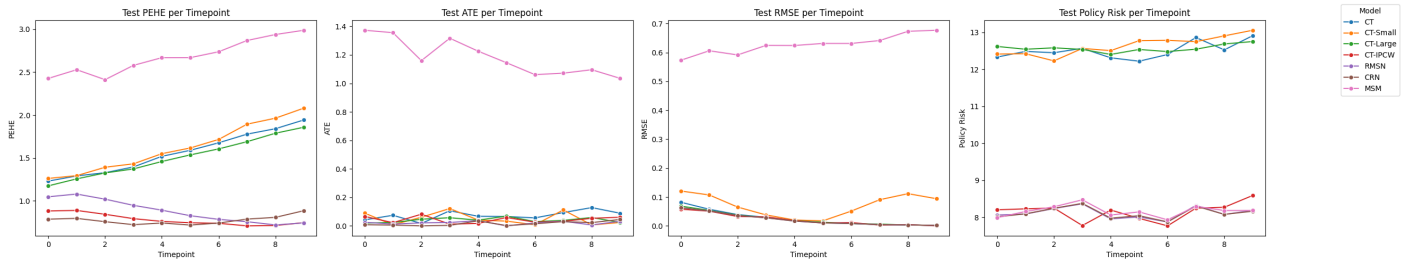

Figure 12: Test evaluation Metrics per Timepoint for Simulated Data.

4.3.1.6 Key Patterns:

The following patterns were observed across timepoints:

- `CRN` and `RMSN` maintained low PEHE and RMSE across timepoints.
- Transformer variants (`CT`, `CT-Large`, `CT-IPCW`) showed slightly increasing PEHE at later timepoints but remained competitive.
- `MSM` was outperformed by all deep learning models on all metrics and across all timepoints.

4.3.1.7 ITE Estimation Accuracy

ITE scatter plots comparing predicted vs. true ITE are shown in Figure 13.

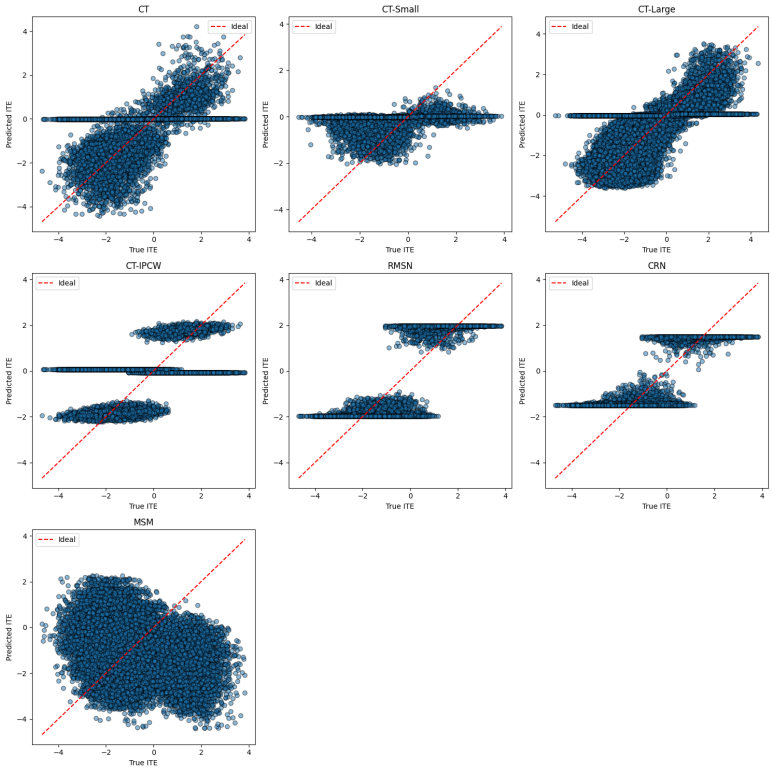

Figure 13: Predicted vs. True ITE: Scatterplots for All Models on Simulated Data.

4.3.1.8 Observations:

- `CRN`, `RMSN`, and `CT-IPCW` achieved the best alignment to the diagonal, indicating superior ITE estimation.
- The baseline `MSM` exhibited poor ITE estimation and systematic bias.
- Transformer variants learned non-trivial treatment heterogeneity but exhibited higher variance at extreme ITE values.

4.3.1.9 Summary of Simulation Results

- Deep learning models, especially `CRN`, `RMSN`, and `CT-IPCW`, significantly outperformed `MSM` on PEHE, ATE, RMSE, and Policy Risk.
- All models demonstrated increased difficulty at later timepoints, consistent with increasing outcome uncertainty over time.
- The addition of IPCW loss appeared beneficial for Transformer variants, improving both PEHE and Policy Risk.
- `CRN` and `RMSN` remained particularly strong across all dimensions, setting a high bar for ITE estimation in this complex longitudinal setting.

4.3.2 Performance on Real-World Data

4.3.2.1 Training Dynamics

Model convergence was tracked via training loss over epochs and across timepoints (Figure 14). Training dynamics were more noisy compared to the simulated setting, reflecting greater heterogeneity and missingness patterns in the real data.

- All models exhibited variable loss trajectories across epochs.
- `CT-IPCW` and `RMSN` showed the most stable and lowest per-timepoint training loss.
- `MSM` and `CT` had higher and more volatile training loss, consistent with their inferior ITE performance in this setting.

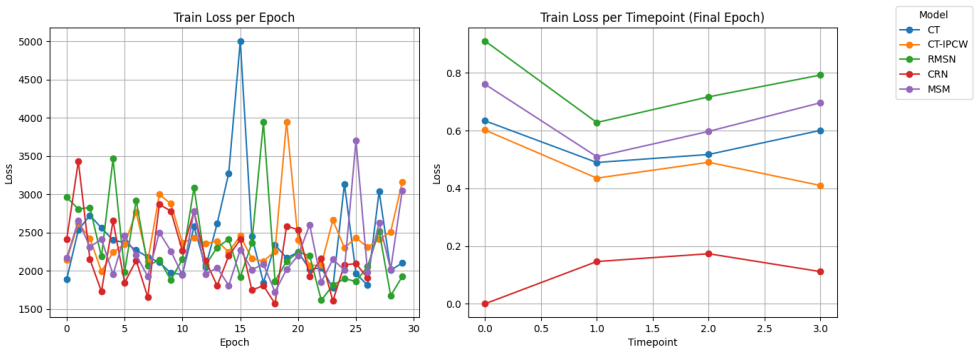

Figure 14: Training Loss per Epoch (left) and per Timepoint (right) on Real Data.

4.3.2.2 Performance over Epochs

Test RMSE and Policy Risk were tracked across epochs (Figure **??**).

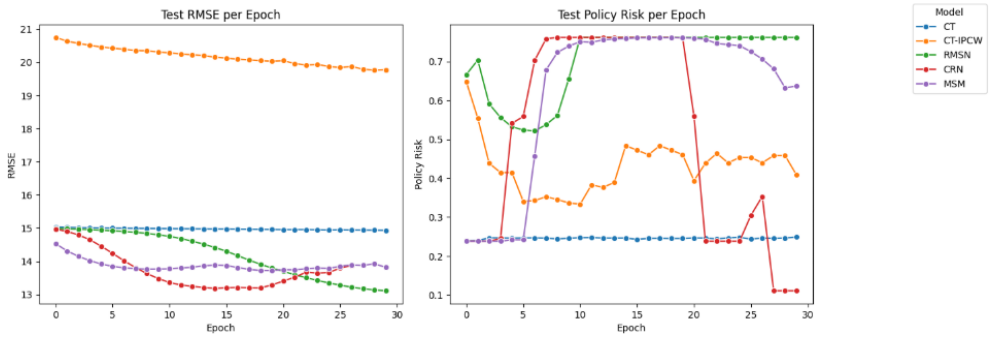

Figure 15: RMSE, and Policy Risk across Epochs for Real Data.

4.3.2.3 Key observations:

- `CT-IPCW` achieved the lowest RMSE overall, demonstrating the benefit of incorporating IPCW in this real-world setting with informative censoring.
- `RMSN` and `CRN` performed well but were less stable across epochs.
- `MSM` and `CT` had higher RMSE and unstable Policy Risk, particularly for `MSM`.

4.3.2.4 Final Epoch Metrics

Table 7 summarizes model performance at the final epoch for Train, Validation, and Test sets — focusing on RMSE and Policy Risk.

Model	Train		Validation		Test	
	RMSE	**Policy Risk**	**RMSE**	**Policy Risk**	**RMSE**	**Policy Risk**
`CRN`	NA	0.1085	NA	0.0999	NA	0.1107
`MSM`	18.1726	0.6400	16.0499	0.6501	13.8222	0.6368
`RMSN`	17.6972	0.7609	15.4438	0.7737	13.1125	0.7619
`CT`	18.4124	0.5599	17.1345	0.2443	14.9347	0.2487
`CT-IPCW`	47.6313	0.4859	25.1378	0.3971	19.7741	0.4088

Table 7: Performance comparison of models on Train, Validation, and Test sets (Real Data) using RMSE and Policy Risk metrics at final epoch.

4.3.2.5 Temporal Performance: Timepoint-Wise Metrics

Figure 16 presents model performance by timepoint.

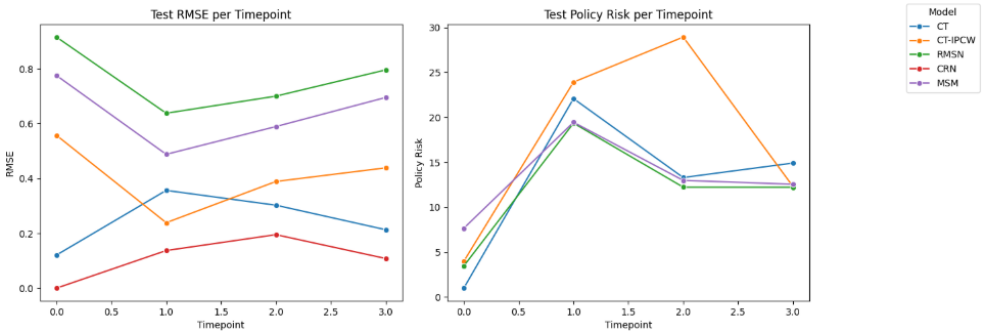

Figure 16: Test evaluation Metrics per Timepoint for Real Data.

4.3.2.6 Key Patterns:

- `CT-IPCW` outperformed `CT` at early timepoints but suffered some instability at later timepoints.
- `RMSN` and `CRN` maintained relatively stable and competitive Policy Risk across timepoints.
- `MSM` consistently underperformed in both RMSE and Policy Risk.

4.3.2.7 ITE Estimation Accuracy

ITE scatter plots comparing predicted vs. true ITE are shown in Figure 17.

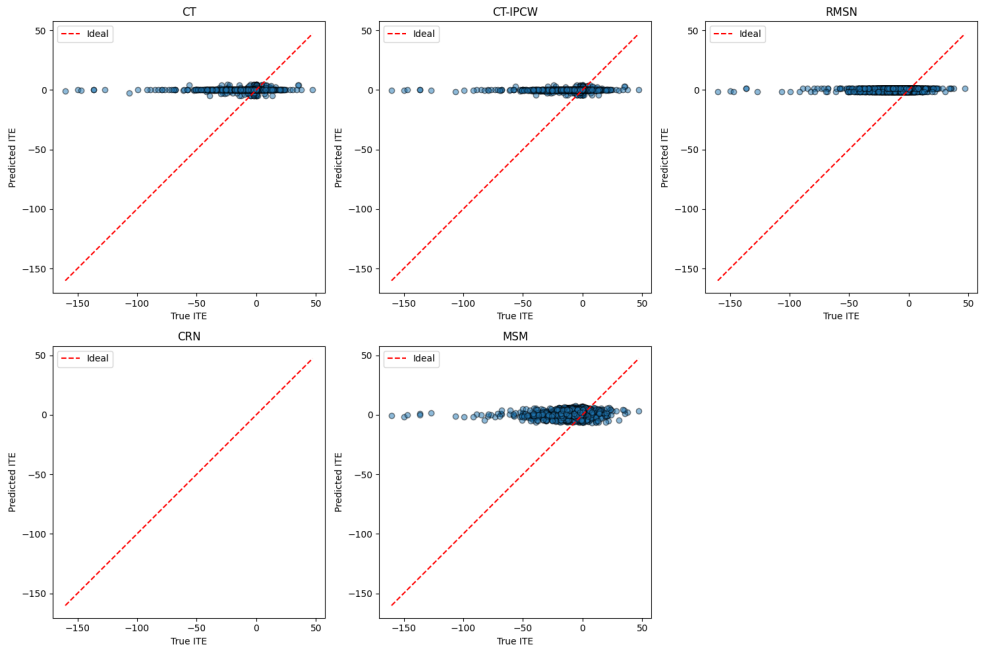

Figure 17: Predicted vs. True ITE: Scatterplots for All Models on Real Data.

4.3.2.8 Observations:

- All models struggled to capture the true ITE signal in this more challenging setting.
- `CT-IPCW` provided some improvement over `CT`, although the gap was less pronounced than in simulated data.

- `RMSN` and `CRN` continued to show relatively good alignment on non-extreme ITEs.
- `MSM` exhibited poor and highly biased ITE estimates.

4.3.2.9 Summary of Real data Results

- Adding IPCW loss to the Causal Transformer improved robustness to censoring, particularly in reducing Policy Risk at early timepoints.
- `CT-IPCW`, `RMSN`, and `CRN` provided the best overall performance on real data.
- `MSM` and standard `CT` struggled, confirming the need for advanced architectures and censoring correction in practical applications.
- Real-world data exposed greater instability and model sensitivity, highlighting the complexity of real-world HTE estimation compared to simulated settings.

4.3.3 Comparison to Other Models

The Transformer architecture, enhanced with IPCW loss, approached the performance of specialized temporal models (`CRN`, `RMSN`), validating its potential for flexible, scalable modeling of HTEs. However, classic `MSM` performed poorly, reinforcing the value of modern deep learning architectures in this domain.

4.4 Handling Censoring in Causal Transformer Models

The baseline Causal Transformer (CT) masks censored outcomes during training. We enhanced this via inverse probability of censoring weighting (CT-IPCW), which reduced potential bias.

CT-IPCW achieved consistently lower PEHE and Policy Risk across epochs and timepoints. Notably, benefits were largest at later timepoints where censoring was more severe. IPCW augmentation improved model robustness and is recommended when informative censoring is present.

4.5 Effect of Sample Size on Causal Transformer Performance

To assess the impact of sample size, we trained CT-Small (N=5,000), CT (N=10,000), and CT-Large (N=50,000) variants using identical hyperparameters. While this isolates size effects, it may limit optimal tuning for each size.

Results showed that larger sample sizes substantially improved PEHE, RMSE, and ITE alignment. CT-Large outperformed smaller variants and exhibited greater stability, underscoring the importance of sufficient data when deploying Causal Transformer models.

5 Discussion and Interpretation

This thesis systematically evaluated the performance of modern machine learning (ML) models, particularly the Causal Transformer (CT), for estimating Heterogeneous Treatment Effects (HTEs) in longitudinal child nutrition and health data. Both simulated and real-world datasets were used to ensure robustness and clinical relevance of the findings.

On simulated data, deep learning models (CT-IPCW, RMSN, CRN) consistently outperformed traditional Marginal Structural Models (MSM) across all key metrics (PEHE, RMSE, Policy Risk), confirming their superior capacity to capture complex time-varying treatment heterogeneity. The ITE scatterplots revealed that CRN and RMSN achieved the best alignment to ground-truth ITEs, while CT also showed strong performance, especially when IPCW loss was incorporated. Notably, ITE scatterplots revealed visible clustering effects in some models, particularly for the CT variants, which warrant further investigation.

On real-world data, results were more nuanced. While CT-IPCW, RMSN, and CRN again outperformed MSM and CT, overall model performance degraded compared to simulation, highlighting the inherent difficulty of estimating ITEs under real-world conditions, including unmeasured confounding, noisy outcomes, and informative censoring. IPCW augmentation provided measurable gains in CT, reducing Policy Risk at early timepoints and achieving lower RMSE overall, emphasizing its practical value in health data applications where dropout is informative.

Sample size scaling experiments demonstrated that larger sample sizes substantially improved model stability and ITE estimation accuracy, especially at later timepoints. However, it is important to note that all sample size variants were trained using identical hyperparameters to isolate the effect of size. This likely penalized CT-Small and CT-Large slightly but provided a fair baseline comparison.

It is also important to highlight that the models used in this thesis were not exact replicas of those in the Causal Transformer repository; rather, they were strongly inspired by the original repository and adapted to suit the structure and specific needs of the CHAIN dataset.

Finally, while this thesis focused on advancing HTE estimation through deep learning, the ultimate motivation remains clinical: to inform nutritional care and feeding protocols for children recovering from acute illness. By identifying heterogeneous treatment responses, this work contributes toward the design of more personalized interventions that can improve nutritional recovery and child survival outcomes.

6 Possible Drawbacks of the Used Methods

Several limitations merit discussion. The Optuna tuning budget was limited (5–20 trials), potentially leaving models sub-optimally tuned. CT variants exhibited ITE clustering, suggesting the need for improved regularization. Identical hyperparameters across sample size variants isolated size effects but likely limited performance. IPCW was applied only to CT; extending it to other models could enhance robustness. Real-world evaluation lacked ground-truth ITEs, so causal interpretability remains indirect. Finally, models were adapted from, but not exact replicas of, the Causal Transformer repository.

7 Ethical Thinking, Societal Relevance, and Stakeholder Awareness

This thesis is motivated by the need to improve nutritional care and feeding protocols for acutely ill children. Ethical safeguards were maintained throughout: CHAIN data were collected under informed consent and appropriate approvals, and all data handlers were trained in Human Subjects Research and Good Clinical Practice.

The CHAIN dataset remains restricted-access, ensuring compliance with data privacy standards. Methodologically, this thesis addressed fairness by explicitly modeling informative censoring through IPCW.

Findings have potential to inform personalized care strategies. For clinicians, the models can support individualized treatment decisions; for children and families, improved HTE estimation may yield more effective

interventions. Future deployment will require further work on model explainability and collaboration with clinicians to ensure trustworthiness and usability.

8 Conclusion

This thesis demonstrates that Causal Transformers, especially when enhanced with IPCW loss, offer a promising, scalable approach for estimating HTEs in longitudinal healthcare data. The models outperformed traditional regression-based MSM and performed competitively with state-of-the-art temporal neural architectures (CRN, RMSN).

Handling informative censoring is critical in real-world settings: IPCW-augmented Causal Transformers achieved notably better robustness and lower Policy Risk. Sample size scaling further improved stability and accuracy.

Importantly, the deep learning models explored here can help inform nutrition care and feeding protocols for acutely ill children, contributing to improved recovery and survival. The models used were inspired by, but not exact replicas of, the Causal Transformer repository, allowing flexibility to better suit the CHAIN dataset.

Challenges remain: tuning is computationally demanding, clustering artifacts require further investigation, and model transparency needs to be enhanced for clinical use. Nonetheless, this thesis provides a robust foundation for further work in applying modern ML methods to support child health and nutrition in LMIC settings.

9 Ideas for Future Research

Future directions include: (1) more extensive hyperparameter tuning for CRN and RMSN; (2) mitigating ITE clustering artifacts; (3) extending IPCW loss to CRN and RMSN; (4) dynamic hyperparameter tuning across sample sizes; (5) improving interpretability of Causal Transformer outputs; (6) testing models in diverse clinical settings; and (7) collaborating with clinicians to ensure practical usability.

10 Conflict of Interest

The author declares that there is no conflict of interest regarding the content of this thesis. The work was conducted independently as part of the MSc in Statistics and Data Science program at Hasselt University. No commercial entities were involved in the design, execution, funding, or reporting of this research. The study received no external funding or sponsorship. The clinical data used (CHAIN cohort) were accessed under appropriate ethical approvals and data use agreements, and are not publicly available. All data were collected with informed consent from participants, and all individuals handling the data were trained in Good Clinical Practice (GCP) and Human Subjects Research ethics.

11 Supplementary Material

The GitHub repository containing all code, data simulations, and additional resources related to this thesis is available at:

- **GitHub Repository:** `github.com/Muirur1/MasterThesis`

Table 8: Summary of Weight and On feeding program observations per timepoint Pattern

Table 9: Weight observations

Category	0	45	90	180	Count
Completers	O	O	O	O	2143
Dropout pattern	O	M	M	M	317
Dropout pattern	O	O	M	M	102
Dropout pattern	O	O	O	M	129
Non-monotone pattern	M	O	O	O	1
Non-monotone pattern	O	M	M	O	41
Non-monotone pattern	O	M	O	M	17
Non-monotone pattern	O	M	O	O	92
Non-monotone pattern	O	O	M	O	77

Table 10: On feeding program observations

Category	0	45	90	180	Count
Completers	O	O	O	O	2252
Dropout pattern	O	M	M	M	267
Dropout pattern	O	O	M	M	90
Dropout pattern	O	O	O	M	104
Non-monotone pattern	M	O	M	M	2
Non-monotone pattern	M	O	O	O	2
Non-monotone pattern	O	M	M	O	28
Non-monotone pattern	O	M	O	M	18
Non-monotone pattern	O	M	O	O	83
Non-monotone pattern	O	O	M	O	73

Model	TP 0	TP 1	TP 2	TP 3	TP 4	TP 5	TP 6	TP 7	TP 8	TP 9
CRN	0.7847	0.7940	0.7553	0.7205	0.7398	0.7159	0.7395	0.7856	0.8057	0.8836
CT	1.2291	1.2892	1.3269	1.3939	1.5180	1.5880	1.6756	1.7766	1.8401	1.9422
CT-IPCW	0.8804	0.8866	0.8415	0.7907	0.7587	0.7428	0.7361	0.7065	0.7109	0.7439
CT-Large	1.1732	1.2550	1.3240	1.3709	1.4576	1.5346	1.6043	1.6887	1.7873	1.8573
CT-Small	1.2600	1.2923	1.3901	1.4305	1.5487	1.6153	1.7148	1.8931	1.9629	2.0800
MSM	2.4264	2.5282	2.4121	2.5792	2.6686	2.6691	2.7376	2.8693	2.9387	2.9887
RMSN	1.0458	1.0777	1.0179	0.9459	0.8895	0.8249	0.7825	0.7551	0.7174	0.7412

Table 11: Test PEHE per Timepoint for Simulated Data.

Model	TP 0	TP 1	TP 2	TP 3	TP 4	TP 5	TP 6	TP 7	TP 8	TP 9
CRN	0.0085	0.0059	0.0007	0.0040	0.0359	0.0025	0.0152	0.0304	0.0234	0.0457
CT	0.0426	0.0747	0.0174	0.1067	0.0677	0.0661	0.0565	0.0939	0.1287	0.0893
CT-IPCW	0.0668	0.0229	0.0823	0.0126	0.0177	0.0594	0.0266	0.0331	0.0536	0.0612
CT-Large	0.0206	0.0240	0.0473	0.0577	0.0405	0.0690	0.0305	0.0388	0.0591	0.0244
CT-Small	0.0903	0.0077	0.0611	0.1220	0.0413	0.0338	0.0082	0.1123	0.0069	0.0245
MSM	1.3724	1.3554	1.1592	1.3164	1.2256	1.1444	1.0612	1.0715	1.0956	1.0349
RMSN	0.0259	0.0128	0.0217	0.0243	0.0375	0.0006	0.0210	0.0328	0.0069	0.0333

Table 12: Test ATE per Timepoint for Simulated Data.

Model	TP 0	TP 1	TP 2	TP 3	TP 4	TP 5	TP 6	TP 7	TP 8	TP 9
CRN	0.0626	0.0519	0.0359	0.0280	0.0167	0.0102	0.0083	0.0043	0.0032	0.0016
CT	0.0819	0.0574	0.0390	0.0296	0.0181	0.0106	0.0083	0.0043	0.0032	0.0016
CT-IPCW	0.0577	0.0509	0.0323	0.0313	0.0181	0.0100	0.0115	0.0027	0.0045	0.0000
CT-Large	0.0684	0.0535	0.0374	0.0285	0.0181	0.0126	0.0079	0.0060	0.0030	0.0019
CT-Small	0.1206	0.1070	0.0650	0.0379	0.0202	0.0173	0.0511	0.0905	0.1115	0.0941
MSM	0.5733	0.6062	0.5918	0.6248	0.6244	0.6315	0.6315	0.6415	0.6736	0.6772
RMSN	0.0626	0.0519	0.0359	0.0280	0.0167	0.0102	0.0083	0.0043	0.0032	0.0016

Table 13: Test RMSE per Timepoint for Simulated Data.

Model	TP 0	TP 1	TP 2	TP 3	TP 4	TP 5	TP 6	TP 7	TP 8	TP 9
CRN	8.0100	8.0921	8.2416	8.3748	7.9744	8.0403	7.8745	8.3016	8.0769	8.1627
CT	12.3302	12.4828	12.4473	12.5660	12.3096	12.2184	12.3985	12.8579	12.5243	12.9088
CT-IPCW	8.2004	8.2263	8.2484	7.7750	8.1942	7.9695	7.7661	8.2455	8.2690	8.5883
CT-Large	12.6176	12.5415	12.5801	12.5372	12.4012	12.5351	12.4787	12.5424	12.6845	12.7497
CT-Small	12.4116	12.4209	12.2269	12.5634	12.5039	12.7740	12.7833	12.7466	12.9033	13.0553
MSM	7.9888	8.1548	8.2827	8.4703	8.0592	8.1444	7.9286	8.2956	8.1805	8.1822
RMSN	8.0590	8.0861	8.2350	8.3621	7.9558	8.0082	7.8713	8.3204	8.0745	8.1856

Table 14: Test Policy Risk per Timepoint for Simulated Data.

Table 15: Best hyperparameters and performance across models (based on 10-fold CV, optimized using Optuna, 5 trials[RMSN and CRN], 20 trials[CT]).

Hyperparameter / Metric	**CT**	**RMSN**	**CRN**
Learning Rate (lr)	7.68×10^{-5}	4.63×10^{-5}	7.88×10^{-4}
Hidden Dimension	128	256	64
Number of Layers	5	3 (GRU)	3
Dropout	0.1211	0.15	0.249
Average PEHE	1.4703	0.7722	0.82

References

[1] Miguel A. Hernán and James M. Robins. *Causal Inference: What If*. Chapman & Hall/CRC, 2020.

[2] Uri Shalit, Fredrik Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[3] The Childhood Acute Illness and Nutrition (CHAIN) Network. Childhood mortality during and after acute illness in africa and south asia: a prospective cohort study. *The Lancet Global Health*, 10(5):e673–e684, 2022.

[4] Susan A. Murphy. Optimal dynamic treatment regimes. *Annual Review of Statistics and Its Application*, 6:321–345, 2019.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[6] Ioana Bica, Ahmed M. Alaa, James Jordon, and Mihaela van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. *arXiv preprint arXiv:2002.04083*, 2020.

[7] James M Robins, Miguel A Hernan, and Babette Brumback. Marginal structural models and causal inference in epidemiology. *Epidemiology*, pages 550–560, 2000.

[8] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. *arXiv preprint arXiv:2204.07258*, 2022. Also presented at ICML 2022, PMLR 162.

[9] Zulfiqar A Bhutta, Jai K Das, Arjumand Rizvi, et al. Evidence-based interventions for improvement of maternal and child nutrition: what can be done and at what cost? *The Lancet*, 382(9890):452–477, 2013.

[10] JL Leroy, M Ruel, JP Habicht, and EA Frongillo. Using height-for-age differences (had) instead of height-for-age z-scores (haz) for the meaningful measurement of population-level catch-up in linear growth in children ¡5 years of age. *BMC Pediatrics*, 15(1):145, 2015.

[11] Geert Molenberghs and Geert Verbeke. *Models for Discrete Longitudinal Data*. Springer, New York, NY, 2005.

[12] Rhian M. Daniel, Simon N. Cousens, Bianca L. De Stavola, Michael G. Kenward, and Jonathan A. C. Sterne. Methods for dealing with time-dependent confounding. *Statistics in Medicine*, 32(9):1584–1618, Apr 2013. Epub 2012 Dec 3.

[13] Zoe Fewell, Miguel A Hernan, Frederick Wolfe, Kate Tilling, Hyon Choi, and Jonathan AC Sterne. Controlling for time-dependent confounding using marginal structural models. *Stata Journal*, 4(4):402–420, 2004.

[14] Roderick JA Little and Donald B Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Hoboken, NJ, 2nd edition edition, 2002.

[15] Joseph G. Ibrahim and Geert Molenberghs. Missing data methods in longitudinal studies: a review. *TEST*, 18:1–43, 2009.

[16] Geert Verbeke and Geert Molenberghs. *Linear Mixed Models for Longitudinal Data*. Springer Series in Statistics. Springer, New York, NY, 2000.

[17] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[18] Patrick Schwab, Lorenz Linhardt, Stefan Bauer, Joachim M Buhmann, and Walter Karlen. Learning counterfactual representations for estimating individual dose-response curves. *arXiv preprint arXiv:1902.00981*, 2020.

[19] Alan Agresti. *Categorical Data Analysis*. Wiley, 2010.

[20] World Health Organization. Updates on the management of severe acute malnutrition in infants and children. 2013.

[21] James Carpenter and John Bithell. Bootstrap confidence intervals: when, which, what? a practical guide for medical statisticians. *Statistics in Medicine*, 19(9):1141–1164, 2000.

[22] Miguel A. Hernán and James M. Robins. *Causal Inference: What If*. Chapman Hall/CRC, Boca Raton, 2020.

[23] Cesar G Victora, Linda Adair, Caroline Fall, Pedro C Hallal, Reynaldo Martorell, Linda Richter, and Harshpal Singh Sachdev. Maternal and child undernutrition: consequences for adult health and human capital. *The Lancet*, 371(9609):340–357, 2008.

[24] Robert E Black, Cesar G Victora, Susan P Walker, et al. Maternal and child undernutrition and overweight in low-income and middle-income countries. *The Lancet*, 382(9890):427–451, 2013.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[26] Bryan Lim, Ahmed M. Alaa, and Mihaela van der Schaar. Forecasting treatment responses over time using recurrent marginal structural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 7494–7504, 2018.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[28] Valentyn Melnychuk. Causaltransformer github repository. `https://github.com/Valentyn1997/CausalTransformer`, 2022.

[29] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[30] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

Listing 1: Data simulation Python Code Used

```
def simulate_data(n_individuals=10000, n_timepoints=10, seed=42):
    """ Data simulation code: Emulating Acute illness and Nutrition status
    across different regions """
    np.random.seed(seed)

    # Constants
    sites = [
        "Kampala", "Banfora", "Dhaka", "Karachi", "Blantyre",
        "Matlab", "Nairobi", "Kilifi", "Migori"
    ]
    country_map = {
        "Kilifi": "Kenya", "Nairobi": "Kenya", "Migori": "Kenya",
        "Dhaka": "Bangladesh", "Matlab": "Bangladesh",
        "Karachi": "Pakistan",
        "Kampala": "Uganda",
        "Banfora": "Burkina␣Faso",
        "Blantyre": "Malawi"
    }
    region_map = {
        "Kenya": "Sub-Saharan␣Africa", "Uganda": "Sub-Saharan␣Africa", "Malawi": "
            Sub-Saharan␣Africa",
        "Bangladesh": "South␣East␣Asia", "Pakistan": "South␣East␣Asia",
        "Burkina␣Faso": "Sub-Saharan␣Africa"
    }
    age_bins = [0, 6, 12, 18, 24, 30, 36.1]
    age_labels = [
        "0-6␣months", "6-12␣months", "12-18␣months", "18-24␣months",
        "24-30␣months", "30-36␣months"
    ]

    # Generate baseline data
    record_ids = np.arange(1, n_individuals + 1)
    sexes = np.random.choice(["Male", "Female"], size=n_individuals)
    sites_sampled = np.random.choice(sites, size=n_individuals)
```

```
    agemons_baseline = np.random.uniform(2, 23, size=n_individuals)

    data = []

    for i in range(n_individuals):
        # Determine dropout (right censoring)
        dropout_time = np.random.randint(6, n_timepoints + 1)  # 6 to 10
        censored_flag = int(dropout_time == n_timepoints)
        timepoint_censored = np.nan if censored_flag else dropout_time

        for t in range(n_timepoints):
            missed_visit = int(t >= dropout_time)
            agemons = agemons_baseline[i] + t
            age_group = pd.cut([agemons], bins=age_bins, labels=age_labels, right=
                False)[0]
            sex = sexes[i]
            site = sites_sampled[i]
            country = country_map[site]
            region = region_map[country]

            # Time-varying covariates
            nutri_counsel = np.random.choice(["Yes", "No"], p=[0.7, 0.3])
            poor_feeding = np.random.choice(["Yes", "No"], p=[0.2, 0.8])
            acutely_ill = np.random.choice(["Yes", "No"], p=[0.3, 0.7])
            onoutpfeed_prog = np.random.choice(["No␣Feeding␣program", "RUTF␣for␣SAM"
                , "RUSF␣for␣MAM"])

            # Compute pscore
            logit_pscore = (
                -0.5 +
                0.05 * agemons +
                0.4 * (sex == "Female") +
                0.3 * (region == "Sub-Saharan␣Africa") +
                0.6 * (nutri_counsel == "Yes") +
                0.7 * (poor_feeding == "Yes") +
                0.8 * (acutely_ill == "Yes")
            )
            pscore = 1 / (1 + np.exp(-logit_pscore))

            # Sample treatment
            binary_treatment = np.random.binomial(1, pscore)

            # Compute IPTW
            iptw = 1 / pscore if binary_treatment == 1 else 1 / (1 - pscore)

            # Simulated ITE for weight gain
            ite_weight = (
                0.1 * t
                + 0.5 * (sex == "Female")
                + 0.3 * (region == "Sub-Saharan␣Africa")
                + 0.4 * (nutri_counsel == "Yes")
                + 0.6 * (poor_feeding == "Yes")
                + 0.7 * (acutely_ill == "Yes")
                + np.random.normal(0, 0.5)
            )

            ite_feed = (
```

```
            -1 + 0.15 * t
            + 0.2 * (region == "Sub-Saharan Africa")
            + 0.4 * (nutri_counsel == "Yes")
            + 0.8 * (poor_feeding == "Yes")
            + 1.0 * (acutely_ill == "Yes")
            + np.random.normal(0, 0.3)
        )

        # Continuous outcome
        base_weight_gain = np.random.normal(loc=10, scale=8)
        pct_weight_gain_0 = base_weight_gain
        pct_weight_gain_1 = base_weight_gain + ite_weight
        pct_weight_gain_factual = pct_weight_gain_1 if binary_treatment else
            pct_weight_gain_0
        pct_weight_gain_counterfactual = pct_weight_gain_0 if binary_treatment
            else pct_weight_gain_1

        # Categorical outcome
        def feed_outcome_logit(val):
            if val < -3:
                return "SAM"
            elif -3 <= val < -2:
                return "MAM"
            else:
                return "Normal"

        feed_outcome_0 = feed_outcome_logit(np.random.normal(-1 + ite_feed / 2,
            1))
        feed_outcome_1 = feed_outcome_logit(np.random.normal(-1 + ite_feed / 2 +
             0.5, 1))
        feed_outcome_factual = feed_outcome_1 if binary_treatment else
            feed_outcome_0
        feed_outcome_counterfactual = feed_outcome_0 if binary_treatment else
            feed_outcome_1

        data.append({
            "record_id": record_ids[i],
            "timepoint": t,
            "agemons": round(agemons, 2),
            "age_group": age_group,
            "sex": sex,
            "site": site,
            "country": country,
            "region": region,
            "nutri_counsel_disch": nutri_counsel,
            "poor_feeding": poor_feeding,
            "acutely_ill": acutely_ill,
            "binary_treatment": binary_treatment,
            "pscore": pscore,
            "iptw": iptw,
            "onoutpfeed_prog": onoutpfeed_prog,
            "missed_visit": missed_visit,
            "censored": censored_flag,
            "timepoint_censored": timepoint_censored,
            "ite_weight": ite_weight,
            "ite_feed": ite_feed,
            "pct_weight_gain_factual": pct_weight_gain_factual,
```

```
                "pct_weight_gain_counterfactual": pct_weight_gain_counterfactual,
                "feed_outcome_factual": feed_outcome_factual,
                "feed_outcome_counterfactual": feed_outcome_counterfactual
            })

    return pd.DataFrame(data)
```

Listing 2: Data preparation for modeling Python Code Used

```
class TimeSeriesDataset(Dataset):
    def __init__(self, df, covariates, outcome_col, treatment_col, time_col="timepoint", id_col="record_id"):
        self.df = df.copy()
        self.covariates = covariates
        self.outcome_col = outcome_col
        self.treatment_col = treatment_col
        self.time_col = time_col
        self.id_col = id_col

        # Automatically encode categorical outcome
        self.label_encoder = None
        if df[outcome_col].dtype == "object" or df[outcome_col].dtype.name == "category":
            self.label_encoder = LabelEncoder()
            self.df[outcome_col] = self.label_encoder.fit_transform(self.df[outcome_col])
            # Also encode counterfactual if it exists
            cf_col = outcome_col.replace("factual", "counterfactual")
            if cf_col in df.columns and df[cf_col].notna().all():
                self.df[cf_col] = self.label_encoder.transform(self.df[cf_col])

        # Group by individual
        self.groups = self.df.groupby(id_col)
        self.ids = list(self.groups.groups.keys())

    def __len__(self):
        return len(self.ids)

    def __getitem__(self, idx):
        rid = self.ids[idx]
        group = self.groups.get_group(rid).sort_values(by=self.time_col)

        x = torch.tensor(group[self.covariates].values, dtype=torch.float32)
        t = torch.tensor(group[self.treatment_col].values, dtype=torch.float32)
        y = torch.tensor(group[self.outcome_col].values, dtype=torch.float32)
        time = torch.tensor(group[self.time_col].values, dtype=torch.long)
        mask = torch.tensor(group["censored"].values, dtype=torch.bool)

        iptw = torch.tensor(group["iptw"].values, dtype=torch.float32)

        # Handle counterfactuals safely
        y_cf_col = self.outcome_col.replace("factual", "counterfactual")
        if y_cf_col in group.columns:
            y_cf_values = group[y_cf_col].values
            if np.all(np.isnan(y_cf_values)):
                y_cf = torch.zeros_like(y)
            else:
                y_cf_values = np.nan_to_num(y_cf_values, nan=0.0)
```

```
                y_cf = torch.tensor(y_cf_values, dtype=torch.float32)
        else:
            y_cf = torch.zeros_like(y)

        # Ensure record_id is Tensor[1]
        rid_tensor = torch.tensor(rid, dtype=torch.long)

        return {
            "x": x,
            "t": t,
            "y": y,
            "y_cf": y_cf,
            "time": time,
            "record_id": rid_tensor,
            "mask": mask,
            "weight": iptw
        }

def create_dataloaders(df: pd.DataFrame,
                       covariates: list,
                       outcome_col: str = "pct_weight_gain_factual",
                       treatment_col: str = "binary_treatment",
                       batch_size: int = 64,
                       test_size: float = 0.4,
                       val_size: float = 0.2,
                       num_workers: int = 4,
                       seed: int = 42):

    np.random.seed(seed)
    record_ids = df["record_id"].unique()
    train_ids, test_ids = train_test_split(record_ids, test_size=test_size,
        random_state=seed)
    train_ids, val_ids = train_test_split(train_ids, test_size=val_size,
        random_state=seed)

    df_train = df[df["record_id"].isin(train_ids)]
    df_val = df[df["record_id"].isin(val_ids)]
    df_test = df[df["record_id"].isin(test_ids)]

    train_set = TimeSeriesDataset(df_train, covariates, outcome_col, treatment_col)
    val_set = TimeSeriesDataset(df_val, covariates, outcome_col, treatment_col)
    test_set = TimeSeriesDataset(df_test, covariates, outcome_col, treatment_col)

    return train_set, val_set, test_set, \
           DataLoader(train_set, batch_size=batch_size, shuffle=True, num_workers=
               num_workers), \
           DataLoader(val_set, batch_size=batch_size, shuffle=False, num_workers=
               num_workers), \
           DataLoader(test_set, batch_size=batch_size, shuffle=False, num_workers=
               num_workers)
```

Listing 3: Training and Evaluation Python Code Used

```
def compute_step_metrics(ite_true, ite_pred, y_true, y_pred, weight):
    n_time = ite_true.shape[1]
    step_metrics = {"PEHE": [], "ATE": [], "RMSE": [], "Policy␣Risk": []}

    for t in range(n_time):
```

```
        pehe_t, ate_t, rmse_t, policy_risk_t = compute_all_metrics_full(
            ite_true[:, t], ite_pred[:, t],
            y_true[:, t], y_pred[:, t],
            weight[:, t],
            return_dict=False
        )
        step_metrics["PEHE"].append(pehe_t)
        step_metrics["ATE"].append(ate_t)
        step_metrics["RMSE"].append(rmse_t)
        step_metrics["Policy Risk"].append(policy_risk_t)

    return step_metrics

def training_loop(
    model,
    train_loader,
    val_loader,
    test_loader,
    optimizer,
    loss_type="mse",
    use_ipcw=False,
    use_cdc=False,
    lambda_cdc=0.1,
    device="cpu",
    epochs=50
):
    metrics_dict = {
        "train_steps": [], "val_steps": [], "test_steps": [],
        "train_epochs": [], "val_epochs": [], "test_epochs": [],
        "train_loss": [], "val_loss": [], "test_loss": [],
        "train_loss_steps": [], "val_loss_steps": [], "test_loss_steps": [],
        "final_metrics": {},
        "ites": {}
    }

    for epoch in trange(epochs, desc="Training Epochs"):
        model.train()
        train_preds, train_trues, train_cf_trues, train_weights, train_masks,
            train_ites, train_cf_preds = [], [], [], [], [], [], []
        epoch_losses = []

        for batch in train_loader:
            optimizer.zero_grad()

            # Only real tensors: move to device
            x, t, y, y_cf, time, mask = [batch[k].to(device) for k in ["x", "t", "y"
                , "y_cf", "time", "mask"]]
            record_id = batch["record_id"]  # integer, leave on CPU
            iptw = batch["weight"].to(device)

            weights = iptw * mask.float() if use_ipcw else iptw

            out = model(x, t, mask=mask)
            y_pred = out["factual_pred"]
            y_cf_pred = out["counterfactual_pred"]

            if loss_type == "mse":
```

```
                loss = weighted_mse_loss(y_pred, y, mask, weights)
            elif loss_type == "ce":
                loss = weighted_cross_entropy_loss(y_pred, y.long(), mask, weights)
            else:
                raise ValueError(f"Unknown loss_type: {loss_type}")

            latent_factual = out.get("latent_factual", None)
            latent_counterfactual = out.get("latent_counterfactual", None)
            if use_cdc and latent_factual is not None and latent_counterfactual is
                not None:
                cdc = cdc_loss(latent_factual, latent_counterfactual)
                loss += lambda_cdc * cdc

            epoch_losses.append(loss.item())

            loss.backward()
            optimizer.step()

            train_preds.append(y_pred.detach().cpu().numpy())
            train_cf_preds.append(y_cf_pred.detach().cpu().numpy())
            train_trues.append(y.detach().cpu().numpy())
            train_cf_trues.append(y_cf.detach().cpu().numpy())
            train_ites.append((y_cf - y).detach().cpu().numpy())
            train_weights.append(weights.detach().cpu().numpy())
            train_masks.append(mask.detach().cpu().numpy())

        # Flatten arrays
        y_true_all = np.concatenate(train_trues)
        y_cf_true_all = np.concatenate(train_cf_trues)
        y_pred_all = np.concatenate(train_preds)
        y_cf_pred_all = np.concatenate(train_cf_preds)
        ite_true_all = np.concatenate(train_ites)
        ite_pred_all = y_cf_pred_all - y_pred_all
        weights_all = np.concatenate(train_weights)

        train_metrics_step = compute_all_metrics_full(
            ite_true_all, ite_pred_all, y_true_all, y_pred_all, weights_all,
                return_dict=True
        )
        metrics_dict["train_epochs"].append({"epoch": epoch, **train_metrics_step})
        metrics_dict["ites"]["train"] = {"true": ite_true_all, "pred": ite_pred_all}

        step_metrics_epoch = compute_step_metrics(ite_true_all, ite_pred_all,
            y_true_all, y_pred_all, weights_all)
        metrics_dict["train_steps"].append(step_metrics_epoch)

        metrics_dict["train_loss"].append(np.mean(epoch_losses))
        metrics_dict["train_loss_steps"].append(step_metrics_epoch["RMSE"])

        # Evaluate val/test
        def evaluate(loader, name):
            model.eval()
            preds, trues, cf_trues, ites, cf_preds, weights, masks = [], [], [], [],
                [], [], []
            batch_losses = []

            with torch.no_grad():
```

```
                for batch in loader:
                    x, t, y, y_cf, time, mask = [batch[k].to(device) for k in ["x",
                        "t", "y", "y_cf", "time", "mask"]]
                    record_id = batch["record_id"]
                    iptw = batch["weight"].to(device)

                    weights_eval = iptw * mask.float() if use_ipcw else iptw

                    out = model(x, t, mask=mask)
                    y_pred = out["factual_pred"]
                    y_cf_pred = out["counterfactual_pred"]

                    if loss_type == "mse":
                        loss = weighted_mse_loss(y_pred, y, mask, weights_eval)
                    elif loss_type == "ce":
                        loss = weighted_cross_entropy_loss(y_pred, y.long(), mask,
                            weights_eval)
                    else:
                        raise ValueError(f"Unknown␣loss_type:␣{loss_type}")

                    latent_factual = out.get("latent_factual", None)
                    latent_counterfactual = out.get("latent_counterfactual", None)
                    if use_cdc and latent_factual is not None and
                        latent_counterfactual is not None:
                        cdc = cdc_loss(latent_factual, latent_counterfactual)
                        loss += lambda_cdc * cdc

                    batch_losses.append(loss.item())

                    preds.append(y_pred.cpu().numpy())
                    cf_preds.append(y_cf_pred.cpu().numpy())
                    trues.append(y.cpu().numpy())
                    cf_trues.append(y_cf.cpu().numpy())
                    ites.append((y_cf - y).cpu().numpy())
                    weights.append(weights_eval.cpu().numpy())

            y_true_all = np.concatenate(trues)
            y_cf_true_all = np.concatenate(cf_trues)
            y_pred_all = np.concatenate(preds)
            y_cf_pred_all = np.concatenate(cf_preds)
            ite_true_all = np.concatenate(ites)
            ite_pred_all = y_cf_pred_all - y_pred_all
            weight_all = np.concatenate(weights)

            metrics = compute_all_metrics_full(
                ite_true_all, ite_pred_all, y_true_all, y_pred_all, weight_all,
                    return_dict=True
            )
            metrics_dict[f"{name}_epochs"].append({"epoch": epoch, **metrics})
            metrics_dict["ites"][name] = {"true": ite_true_all, "pred": ite_pred_all
                }

            step_metrics_epoch = compute_step_metrics(ite_true_all, ite_pred_all,
                y_true_all, y_pred_all, weight_all)
            metrics_dict[f"{name}_steps"].append(step_metrics_epoch)

            metrics_dict[f"{name}_loss"].append(np.mean(batch_losses))
```

```
            metrics_dict[f"{name}_loss_steps"].append(step_metrics_epoch["RMSE"])

        evaluate(val_loader, "val")
        evaluate(test_loader, "test")

        print(f"Epoch {epoch}: Train PEHE={train_metrics_step['PEHE']:.4f}, ATE={
            train_metrics_step['ATE']:.4f}, RMSE={train_metrics_step['RMSE']:.4f}")

    # Final metrics
    final_train = metrics_dict["train_epochs"][-1]
    final_val = metrics_dict["val_epochs"][-1]
    final_test = metrics_dict["test_epochs"][-1]

    print(f"\n>>> FINAL TEST METRICS (EPOCH {epoch}):")
    print(f"Test PEHE={final_test['PEHE']:.4f}, ATE={final_test['ATE']:.4f}, "
        f"RMSE={final_test['RMSE']:.4f}, Policy Risk={final_test['Policy Risk']:.4f}
            ")

    metrics_dict["final_metrics"] = {
        "Final Train PEHE": final_train["PEHE"],
        "Final Val PEHE": final_val["PEHE"],
        "Final Test PEHE": final_test["PEHE"],
        "Final Train ATE": final_train["ATE"],
        "Final Val ATE": final_val["ATE"],
        "Final Test ATE": final_test["ATE"],
        "Final Train RMSE": final_train["RMSE"],
        "Final Val RMSE": final_val["RMSE"],
        "Final Test RMSE": final_test["RMSE"],
        "Final Train Policy Risk": final_train["Policy Risk"],
        "Final Val Policy Risk": final_val["Policy Risk"],
        "Final Test Policy Risk": final_test["Policy Risk"],
    }

    return metrics_dict
```

Listing 4: Training Loop for Simulation data Code Used

```
# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Define list of model configs
model_names = ["transformer", "transformer_small", "transformer_large", "
    transformer_ipcw", "rmsn", "crn", "msm"]

# Define best lr per model
lr_overrides = {
    "transformer": 7.68e-5,
    "transformer_ipcw": 7.68e-5,
    "transformer_small": 7.68e-5,
    "transformer_large": 7.68e-5,
    "rmsn": 4.6342578001718805e-5,
    "crn": 7.882919066627159e-4,
    "msm": 1e-3  # no tuning done, use default
}

results = {}

# Loop over models
```

```
for model_name in model_names:
    print(f"\n=== Running {model_name.upper()} ===")

    with initialize(version_base=None, config_path="config"):
        cfg = compose(config_name="config.yaml", overrides=[f"model={model_name}",
                                                            f"optimizer.lr={
                                                                lr_overrides[
                                                                model_name]}"])
        print(OmegaConf.to_yaml(cfg))

        # Select dataset
        if model_name.startswith("transformer"):
            df_used = df_simulated if model_name in ["transformer", "
                transformer_ipcw"] else \
                    df_sim_5000 if model_name == "transformer_small" else \
                    df_sim_50000
        else:
            # All other models: always use df_simulated
            df_used = df_simulated

        # Preprocess covariates
        covariate_cols = [
            'agemons',
            'binary_treatment',
            'sex', 'site', 'region', 'age_group',
            'nutri_counsel_disch', 'poor_feeding', 'acutely_ill'
        ]
        categorical_covs = [col for col in covariate_cols if col not in ["agemons",
            "binary_treatment"]]
        df_encoded = pd.get_dummies(df_used, columns=categorical_covs, drop_first=
            True)
        df_encoded["binary_treatment"] = df_used["binary_treatment"]

        encoded_covariates = [col for col in df_encoded.columns if any(base in col
            for base in covariate_cols)]
        df_encoded[encoded_covariates] = df_encoded[encoded_covariates].astype("
            float32")

        # Create dataloaders
        train_set, val_set, test_set, train_loader, val_loader, test_loader =
            create_dataloaders(
            df=df_encoded,
            covariates=encoded_covariates,
            outcome_col="pct_weight_gain_factual",
            treatment_col="binary_treatment",
            batch_size=cfg.batch_size,
        )

        # Infer correct input_dim
        input_dim = len(encoded_covariates)

        # Instantiate model and optimizer
        model = instantiate(cfg.model, input_dim=input_dim).to(device)
        optimizer = instantiate(cfg.optimizer, params=model.parameters())

        # For Transformer, CRN, RMSN, set _loss_type
        if hasattr(model, "_loss_type"):
```

```
            model._loss_type = "mse"

        # Train model
        metrics = training_loop(
            model=model,
            train_loader=train_loader,
            val_loader=val_loader,
            test_loader=test_loader,
            optimizer=optimizer,
            loss_type="mse",
            use_ipcw=cfg.model.use_ipcw,
            use_cdc=cfg.model.use_cdc,
            lambda_cdc=0.1,
            device=device,
            epochs=cfg.epochs,
        )

        # Store results
        results[model_name] = metrics
```

Listing 5: Visualization Python Code Used

```

def plot_epoch_summary(df_long, split):
    """
    Plot PEHE, ATE, RMSE, Policy Risk per epoch by model for a given split.
    One row of subplots with common legend.
    """
    df_split = df_long[df_long["split"] == split]
    metrics = df_split["metric"].unique()

    fig, axes = plt.subplots(1, len(metrics), figsize=(6 * len(metrics), 5), sharey=
        False)

    if len(metrics) == 1:
        axes = [axes]

    for i, metric in enumerate(metrics):
        df_plot = df_split[df_split["metric"] == metric]
        sns.lineplot(data=df_plot, x="epoch", y="value", hue="model", marker="o", ax
            =axes[i])
        axes[i].set_title(f"{split} {metric} per Epoch")
        axes[i].set_xlabel("Epoch")
        axes[i].set_ylabel(metric)

    # Single legend
    handles, labels = axes[0].get_legend_handles_labels()
    fig.legend(handles, labels, title="Model", bbox_to_anchor=(1.05, 1), loc="upper
        left")
    for ax in axes:
        ax.get_legend().remove()

    plt.tight_layout()
    plt.show()

def plot_step_summary(df_long, split):
    """
    Plot PEHE, ATE, RMSE, Policy Risk per timepoint by model for a given split.
```

```
    One row of subplots with common legend.
    """
    df_split = df_long[df_long["split"] == split]
    metrics = df_split["metric"].unique()

    fig, axes = plt.subplots(1, len(metrics), figsize=(6 * len(metrics), 5), sharey=
        False)

    if len(metrics) == 1:
        axes = [axes]

    for i, metric in enumerate(metrics):
        df_plot = df_split[df_split["metric"] == metric]
        sns.lineplot(data=df_plot, x="timepoint", y="value", hue="model", marker="o"
            , ax=axes[i])
        axes[i].set_title(f"{split} {metric} per Timepoint")
        axes[i].set_xlabel("Timepoint")
        axes[i].set_ylabel(metric)

    # Single legend
    handles, labels = axes[0].get_legend_handles_labels()
    fig.legend(handles, labels, title="Model", bbox_to_anchor=(1.05, 1), loc="upper
        left")
    for ax in axes:
        ax.get_legend().remove()

    plt.tight_layout()
    plt.show()

def plot_loss_curves(loss_dict, split="Train"):
    """
    Plot loss per epoch and per timepoint.
    """
    # Loss per epoch
    plt.figure(figsize=(8, 5))
    plt.plot(loss_dict["epoch"], marker="o")
    plt.title(f"{split} Loss per Epoch")
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    # Loss per timepoint (last epoch)
    last_timepoint_losses = loss_dict["per_timepoint"][-1] if isinstance(loss_dict["
        per_timepoint"][0], (list, np.ndarray)) else loss_dict["per_timepoint"]
    plt.figure(figsize=(8, 5))
    plt.plot(last_timepoint_losses, marker="o")
    plt.title(f"{split} Loss per Timepoint (Final Epoch)")
    plt.xlabel("Timepoint")
    plt.ylabel("Loss")
    plt.grid(True)
    plt.tight_layout()
    plt.show()

def plot_ite_scatter(true_ite, pred_ite, model_name="Model"):
    """
```

```
    Plot predicted vs true ITE scatter plot.
    """
    assert true_ite.shape == pred_ite.shape, "Mismatched ITE shapes"

    true_flat = true_ite.flatten()
    pred_flat = pred_ite.flatten()

    plt.figure(figsize=(6, 6))
    sns.scatterplot(x=true_flat, y=pred_flat, alpha=0.3)
    max_val = max(np.max(true_flat), np.max(pred_flat))
    min_val = min(np.min(true_flat), np.min(pred_flat))
    plt.plot([min_val, max_val], [min_val, max_val], color="red", linestyle="--")
    plt.xlabel("True ITE")
    plt.ylabel("Predicted ITE")
    plt.title(f"ITE Scatter Plot: {model_name}")
    plt.grid(True)
    plt.tight_layout()
    plt.show()


def plot_loss_curves_all_models(loss_dict_all_models, split="Train"):
    """
    Plot loss per epoch and per timepoint for ALL MODELS.
    One row with 2 subplots: Loss per Epoch, Loss per Timepoint.
    """
    models = list(loss_dict_all_models.keys())

    fig, axes = plt.subplots(1, 2, figsize=(12, 5), sharey=False)

    # Plot Loss per Epoch
    for model in models:
        epoch_loss = loss_dict_all_models[model]["epoch"]
        axes[0].plot(epoch_loss, marker="o", label=model)
    axes[0].set_title(f"{split} Loss per Epoch")
    axes[0].set_xlabel("Epoch")
    axes[0].set_ylabel("Loss")
    axes[0].grid(True)

    # Plot Loss per Timepoint (last epoch)
    for model in models:
        per_tp = loss_dict_all_models[model]["per_timepoint"]
        last_tp_loss = per_tp[-1] if isinstance(per_tp[0], (list, np.ndarray)) else
            per_tp
        axes[1].plot(last_tp_loss, marker="o", label=model)
    axes[1].set_title(f"{split} Loss per Timepoint (Final Epoch)")
    axes[1].set_xlabel("Timepoint")
    axes[1].set_ylabel("Loss")
    axes[1].grid(True)

    # Common Legend
    handles, labels = axes[0].get_legend_handles_labels()
    fig.legend(handles, labels, title="Model", bbox_to_anchor=(1.05, 1), loc="upper
        left")

    plt.tight_layout()
    plt.show()
```

Listing 6: Loss functions

```python
def mse_loss(y_pred, y_true, mask):
    """
    Mean Squared Error loss with masking.
    Args:
        y_pred: predicted outcomes, shape [B, T]
        y_true: true outcomes, shape [B, T]
        mask: binary mask, shape [B, T], 1 if observed, 0 if censored
    Returns:
        scalar loss
    """
    loss = (y_pred - y_true) ** 2
    return (loss * mask).sum() / mask.sum()

def weighted_mse_loss(y_pred, y_true, mask, weights):
    """
    IPCW-adjusted Mean Squared Error loss with masking.
    Args:
        y_pred: predicted outcomes, shape [B, T]
        y_true: true outcomes, shape [B, T]
        mask: binary mask, shape [B, T]
        weights: Inverse Probability of Censoring Weights, shape [B, T]
    Returns:
        scalar loss
    """
    loss = (y_pred - y_true) ** 2
    return (loss * mask * weights).sum() / (mask * weights).sum()

def cdc_loss(latent_factual, latent_counterfactual):
    """
    Counterfactual Domain Confusion (CDC) loss.

    Encourages alignment between factual and counterfactual latent representations
    by minimizing cosine similarity distance.

    Args:
        latent_factual: tensor of shape [B, T, D]
        latent_counterfactual: tensor of shape [B, T, D]

    Returns:
        scalar loss (1 - cosine similarity)
    """
    # Flatten batch and time for similarity
    lf = latent_factual.reshape(-1, latent_factual.size(-1))  # [B*T, D]
    lc = latent_counterfactual.reshape(-1, latent_counterfactual.size(-1))  # [B*T,
        D]
    similarity = F.cosine_similarity(lf, lc, dim=-1)
    return 1.0 - similarity.mean()
```

Listing 7: Metrics

```python
def weighted_mse(y_true, y_pred, weight=None):
    if weight is None:
        return np.mean((y_true - y_pred) ** 2)
    return np.average((y_true - y_pred) ** 2, weights=weight)

def weighted_rmse(y_true, y_pred, weight=None):
```

```
    """
    Root Mean Squared Error for factual outcomes.
    """
    return np.sqrt(weighted_mse(y_true, y_pred, weight))

def weighted_pehe(ite_true, ite_pred, weight=None):
    """
    Precision in Estimation of Heterogeneous Effects.
    """
    if weight is None:
        pehe = np.mean((ite_pred - ite_true) ** 2)
    else:
        pehe = np.average((ite_pred - ite_true) ** 2, weights=weight)
    return np.sqrt(pehe)

def weighted_ate(ite_true, ite_pred, weight=None):
    """
    Absolute difference in Average Treatment Effects.
    """
    if weight is None:
        return abs(np.mean(ite_pred) - np.mean(ite_true))
    return abs(np.average(ite_pred, weights=weight) - np.average(ite_true, weights=
        weight))

def weighted_policy_risk(ite_true, ite_pred, weight=None):
    """
    Policy Risk measures the proportion of incorrect treatment decisions.
    """
    treated_policy = (ite_pred > 0).astype(int)
    optimal_policy = (ite_true > 0).astype(int)
    incorrect = (treated_policy != optimal_policy).astype(int)
    if weight is None:
        return np.mean(incorrect)
    return np.average(incorrect, weights=weight)

def compute_all_metrics_full(
    ite_true, ite_pred,
    factual_true=None, factual_pred=None,
    weight=None,
    return_dict=True
):
    """
    Returns dictionary or list of metrics:
    - PEHE
    - ATE
    - RMSE (if factual outcomes are available)
    - Policy Risk
    """
    metrics = {
        "PEHE": weighted_pehe(ite_true, ite_pred, weight),
        "ATE": weighted_ate(ite_true, ite_pred, weight),
        "Policy␣Risk": weighted_policy_risk(ite_true, ite_pred, weight),
    }

    if factual_true is not None and factual_pred is not None:
        metrics["RMSE"] = weighted_rmse(factual_true, factual_pred, weight)
    else:
```

```
        metrics["RMSE"] = np.nan

    return metrics if return_dict else list(metrics.values())
```

Listing 8: Causal Transformer Model

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, max_len=500):
        super().__init__()
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2) * (-torch.log(torch.tensor
            (10000.0)) / d_model))
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        self.pe = pe.unsqueeze(0)  # shape: [1, max_len, d_model]

    def forward(self, x):
        return x + self.pe[:, :x.size(1)].to(x.device)


class CausalTransformer(nn.Module):
    def __init__(
        self, input_dim, embed_dim=128, num_layers=2, n_heads=4, dropout=0.1,
        use_ipcw=False, use_cdc=False, num_classes=3
    ):
        super().__init__()
        self.use_ipcw = use_ipcw
        self.use_cdc = use_cdc
        self.num_classes = num_classes

        # Tell the model at training time which head to use (MSE or CE)
        self._loss_type = "mse"  # default

        self.input_proj = nn.Linear(input_dim, embed_dim)
        self.pos_encoder = PositionalEncoding(embed_dim)

        encoder_layer = nn.TransformerEncoderLayer(
            d_model=embed_dim,
            nhead=n_heads,
            dim_feedforward=4 * embed_dim,
            dropout=dropout,
            batch_first=True
        )
        self.transformer_encoder = nn.TransformerEncoder(encoder_layer, num_layers=
            num_layers)

        # Regression heads (for MSE)
        self.head_y0 = nn.Sequential(
            nn.Linear(embed_dim, embed_dim // 2), nn.ReLU(), nn.Linear(embed_dim //
                2, 1)
        )
        self.head_y1 = nn.Sequential(
            nn.Linear(embed_dim, embed_dim // 2), nn.ReLU(), nn.Linear(embed_dim //
                2, 1)
        )

        # Classification heads (for CE)
```

```
        self.head_y0_logits = nn.Sequential(
            nn.Linear(embed_dim, embed_dim // 2), nn.ReLU(), nn.Linear(embed_dim //
                2, num_classes)
        )
        self.head_y1_logits = nn.Sequential(
            nn.Linear(embed_dim, embed_dim // 2), nn.ReLU(), nn.Linear(embed_dim //
                2, num_classes)
        )

    def forward(self, x, t, mask=None):
        """
        Args:
            x:     [B, T, F]   - input covariates
            t:     [B, T]      - binary treatment indicators (0 or 1)
            mask:  [B, T]      - 1 if observed, 0 if censored (optional)

        Returns:
            Dictionary with:
                - factual_pred
                - counterfactual_pred
                - latent_factual, latent_counterfactual (if use_cdc=True)
        """
        x = self.input_proj(x)
        x = self.pos_encoder(x)

        pad_mask = ~mask.bool() if mask is not None else None
        h = self.transformer_encoder(x, src_key_padding_mask=pad_mask)

        # CASE 1: Classification (CE)
        if self.training and hasattr(self, "_loss_type") and self._loss_type == "ce"
            :
            y0_logits = self.head_y0_logits(h)  # [B, T, C]
            y1_logits = self.head_y1_logits(h)  # [B, T, C]

            # Expand t for selecting logits
            t_exp = t.unsqueeze(-1).repeat(1, 1, y0_logits.size(-1))  # [B, T, C]

            y_factual_logits = torch.where(t_exp == 1, y1_logits, y0_logits)
            y_counterfactual_logits = torch.where(t_exp == 1, y0_logits, y1_logits)

            output = {
                "factual_pred": y_factual_logits,            # logits
                "counterfactual_pred": y_counterfactual_logits,
            }

        # CASE 2: Regression (MSE)
        else:
            y0_hat = self.head_y0(h).squeeze(-1)  # [B, T]
            y1_hat = self.head_y1(h).squeeze(-1)  # [B, T]

            y_factual = torch.where(t == 1, y1_hat, y0_hat)
            y_cf_pred = torch.where(t == 1, y0_hat, y1_hat)

            output = {
                "factual_pred": y_factual,
                "counterfactual_pred": y_cf_pred,
            }
```

```
103
104         # Add latent if using CDC
105         if self.use_cdc:
106             output["latent_factual"] = h
107             output["latent_counterfactual"] = h
108
109         return output
```

Listing 9: Recurrent Marginal Structural Network Model

```
1  import torch
2  import torch.nn as nn
3
4  class RMSN(nn.Module):
5      def __init__(self, input_dim, hidden_dim=128, num_layers=1, dropout=0.1,
           use_ipcw=False, use_cdc=False):
6          super().__init__()
7          self.use_ipcw = use_ipcw
8          self.use_cdc = use_cdc
9
10         self.gru = nn.GRU(
11             input_dim, hidden_dim, num_layers=num_layers,
12             batch_first=True, dropout=dropout if num_layers > 1 else 0.0
13         )
14
15         self.head_y0 = nn.Sequential(
16             nn.Linear(hidden_dim, hidden_dim // 2), nn.ReLU(), nn.Linear(hidden_dim
                   // 2, 1)
17         )
18         self.head_y1 = nn.Sequential(
19             nn.Linear(hidden_dim, hidden_dim // 2), nn.ReLU(), nn.Linear(hidden_dim
                   // 2, 1)
20         )
21
22     def forward(self, x, t, mask=None):
23         h_out, h_n = self.gru(x)  # [B, T, H]
24
25         y0_hat = self.head_y0(h_out).squeeze(-1)  # [B, T]
26         y1_hat = self.head_y1(h_out).squeeze(-1)  # [B, T]
27
28         y_factual = torch.where(t == 1, y1_hat, y0_hat)
29         y_cf_pred = torch.where(t == 1, y0_hat, y1_hat)
30
31         output = {
32             "factual_pred": y_factual,
33             "counterfactual_pred": y_cf_pred,
34         }
35
36         if self.use_cdc:
37             output["latent_factual"] = h_out
38             output["latent_counterfactual"] = h_out
39
40         return output
```

Listing 10: Counterfactual Recurrent Network

```
1  class CRN(nn.Module):
2      def __init__(self, input_dim, hidden_dim=128, num_layers=1, dropout=0.1,
```

```
        use_ipcw=False, use_cdc=True):
        super().__init__()
        self.use_ipcw = use_ipcw
        self.use_cdc = use_cdc

        self.gru = nn.GRU(
            input_dim, hidden_dim, num_layers=num_layers,
            batch_first=True, dropout=dropout if num_layers > 1 else 0.0
        )

        self.head_y0 = nn.Sequential(
            nn.Linear(hidden_dim, hidden_dim // 2), nn.ReLU(), nn.Linear(hidden_dim
                // 2, 1)
        )
        self.head_y1 = nn.Sequential(
            nn.Linear(hidden_dim, hidden_dim // 2), nn.ReLU(), nn.Linear(hidden_dim
                // 2, 1)
        )

    def forward(self, x, t, mask=None):
        h_out, h_n = self.gru(x)  # [B, T, H]

        y0_hat = self.head_y0(h_out).squeeze(-1)  # [B, T]
        y1_hat = self.head_y1(h_out).squeeze(-1)  # [B, T]

        y_factual = torch.where(t == 1, y1_hat, y0_hat)
        y_cf_pred = torch.where(t == 1, y0_hat, y1_hat)

        output = {
            "factual_pred": y_factual,
            "counterfactual_pred": y_cf_pred,
        }

        if self.use_cdc:
            output["latent_factual"] = h_out
            output["latent_counterfactual"] = h_out

        return output
```

Listing 11: Marginal Structural Model

```
class MSM(nn.Module):
    def __init__(self, input_dim, hidden_dim=128, dropout=0.1, use_ipcw=False,
        use_cdc=False):
        super().__init__()
        self.use_ipcw = use_ipcw
        self.use_cdc = use_cdc

        # Process each time step independently: linear model
        self.head_y0 = nn.Sequential(
            nn.Linear(input_dim, hidden_dim), nn.ReLU(), nn.Linear(hidden_dim, 1)
        )
        self.head_y1 = nn.Sequential(
            nn.Linear(input_dim, hidden_dim), nn.ReLU(), nn.Linear(hidden_dim, 1)
        )

    def forward(self, x, t, mask=None):
        # x - [B, T, F]
```

```
        B, T, F = x.shape

        x_flat = x.view(B * T, F)

        y0_hat = self.head_y0(x_flat).view(B, T)
        y1_hat = self.head_y1(x_flat).view(B, T)

        y_factual = torch.where(t == 1, y1_hat, y0_hat)
        y_cf_pred = torch.where(t == 1, y0_hat, y1_hat)

        output = {
            "factual_pred": y_factual,
            "counterfactual_pred": y_cf_pred,
        }

        return output
```

Listing 12: Python libraries used

```
Python version: 3.12.9
Anaconda Navigator version: 2.6.6
pandas==2.2.3
numpy==1.26.4
matplotlib==3.7.5
seaborn==0.13.2
statsmodels==0.14.2
scikit-learn==1.4.2
missingno==0.5.2
scipy==1.11.4
torch==2.6.0+cpu
tqdm==4.67.1
hydra-core==1.3.2
omegaconf==2.3.0
optuna==2.10.1
```