# Compositional Models for Estimating Causal Effects

**Purva Pruthi** PPRUTHI@CS.UMASS.EDU
**David Jensen** JENSEN@CS.UMASS.EDU
*College of Information and Computer Sciences*
*University of Massachusetts Amherst*
*Amherst, MA, USA*

**Editors:** Biwei Huang and Mathias Drton

## Abstract

Many real-world systems can be usefully represented as sets of interacting components. Examples include computational systems, such as query processors and compilers; natural systems, such as cells and ecosystems; and social systems, such as families and organizations. However, current approaches to estimating *potential outcomes* and *causal effects* typically treat such systems as single units, represent them with a fixed set of variables, and assume a homogeneous data-generating process. In this work, we study a *compositional* approach for estimating individual-level potential outcomes and causal effects in structured systems, where each unit is represented by an *instance-specific* composition of multiple heterogeneous components. The compositional approach decomposes unit-level causal queries into more fine-grained queries, explicitly modeling how unit-level interventions affect component-level outcomes to generate a unit's outcome. We demonstrate this approach using modular neural network architectures and show that it provides benefits for causal effect estimation from observational data, such as accurate causal effect estimation for structured units, increased sample efficiency, improved overlap between treatment and control groups, and compositional generalization to units with unseen combinations of components. Remarkably, our results show that compositional modeling can improve the accuracy of causal estimation even when component-level outcomes are unobserved. We also create and use a set of real-world evaluation environments for the empirical evaluation of compositional approaches for causal effect estimation and demonstrate the role of composition structure, varying amounts of component-level data access, and component heterogeneity in the performance of compositional models as compared to the non-compositional approaches.

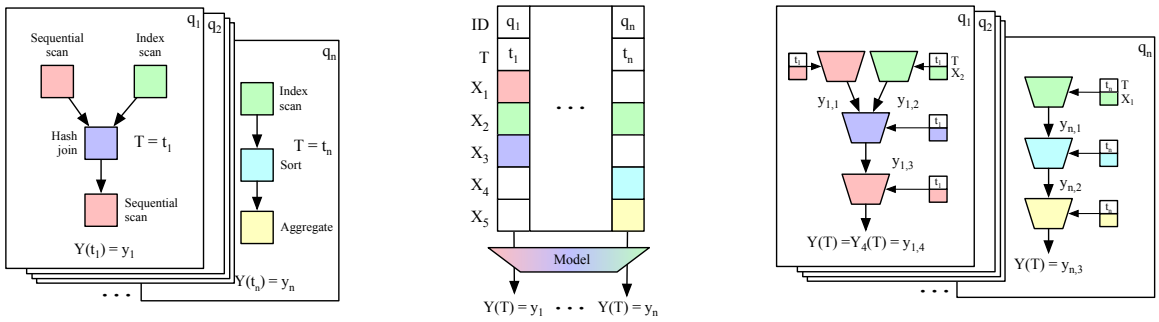
**Keywords:** Causal modeling, compositionality, systematic generalization

## 1. Introduction

Many applications require estimating individual-level potential outcomes and treatment effects, including personalized medicine (Curth et al., 2024), individualized instruction (Kochmar et al., 2022), and custom online advertising (Bottou et al., 2013). Standard approaches to heterogeneous treatment effect estimation (e.g., Hill, 2011; Athey and Imbens, 2016; Wager and Athey, 2018; Chernozhukov et al., 2018) typically assume that the units of analysis can be represented by a fixed set of random variables that are sampled from a fixed causal graph, following a homogeneous data generating process, known as unit homogeneity assumption (Holland, 1986).

However, many real-world systems are *heterogeneous* and *modular* — they decompose into heterogeneous functional *components* that interact in various ways to produce system behavior (Callebaut and Rasskin-Gutman, 2005; Johansson et al., 2022). Input data to such systems is often structured, variable-sized, and sampled from different causal graphs, making it challenging to reason

about the system's behavior. An alternative approach to analyzing such systems is to exploit their *compositionality*, assuming that the system behavior can be understood in terms of the behavior of familiar *re-usable* components and how they are composed. Estimating individual treatment effects for compositional systems is an important problem, particularly as the complexity of modern technological systems increases. Modern computational systems such as databases, compilers, and multi-agent systems can generate large amounts of experimental and observational data containing fine-grained information about the structure and behavior of modular systems, which often remains unused by the existing approaches for estimating causal effects.




(a) Example structured units (b) Unitary approach (c) Compositional approach

Figure 1: **Overview of key ideas:** (a) ***Structured units:*** Units are composed of multiple heterogeneous components. Each color represents a distinct component. Treatment $T$ is applied to the unit, and the compositional system processes the inputs under intervention, returning potential outcomes. (b) ***Unitary approach:*** Standard approaches to effect estimation flattens the underlying structure. They use a fixed-size representation for each unit, aggregating component-level information to estimate unit-level potential outcomes. (c) ***Compositional approach:*** The compositional approach models each unit with an instance-specific structure. Component-level covariates $X_j$ and outcomes $Y_j$ are used to train each component model, and component-level outcomes are hierarchically aggregated to estimate unit-level potential outcomes. Each color represents a distinct component model with different parameters.

Figure 1 provides a schematic overview of causal inference for compositional systems and how it is addressed by different approaches to causal estimation. Consider a relational database query execution system with component operations such as `scan`, `sort`, `aggregate`, and `join`. The system takes input including tables (e.g., A, B) and a query execution plan (e.g., `scan(join(scan(A),scan(B)))`) and returns a new table as output. Query executions represent the unit of analysis, and query plans explicitly describe the *compositional* and *hierarchical* structure of those units — query executions consist of heterogeneous component operations that can be combined in a vast number of different ways. In addition, the compositional structure of query execution is *instance-specific* — the number, kind, and structure among components may differ across each unit. These units can be represented as hierarchical graphs (e.g., parse-trees), where each node is a component operation and edges represent the information flow between the components.

Given such a system, consider modeling the causal effect of memory size on execution time for different query plans. This problem can be formulated as using observational data to estimate the

*individual-level* effects of interventions on structured units.[1] In real-world data on query execution, interventions such as memory size might be chosen based on the structure and features of the query, making the data *observational*. In the terminology of causal inference, each query execution is a *unit of analysis*, the features of the relations are *pre-treatment covariates*, memory size is the *intervention*, and execution time is the *potential outcome* (Rubin, 1974, 2005). We might also want to predict the effects for a population of arbitrary query executions that contain novel combinations of the component operations. In that case, it is desirable for the learned models to *compositionally generalize* to units with unseen combinations of the components. Other real-world use-cases of causal reasoning in structured data are discussed in the supplementary material (Section A).

Standard approaches to heterogeneous treatment effect estimation (e.g., Hill, 2011; Athey and Imbens, 2016) typically ignore the underlying structure and represent each compositional unit using a fixed-size feature representation, which poses several estimation and identifiability challenges. As the structure and complexity of each unit vary, estimating effects at the unit level requires reasoning about the similarity among the heterogeneous units in a high-dimensional space. Additionally, representing all the units with the same features leads to sparse feature representation and aggregation of the features of multiple instances of each component, causing identifiability challenges. We use the term *unitary models* to denote these approaches that exclusively model unit-level quantities.

In contrast, we study a *compositional* approach to causal effect estimation for structured units from observational data. This approach estimates the component-wise potential outcomes using the observational data available for that component, pooled across each instance of the component among units. It then forms an estimate of the unit-level potential outcomes and treatment effects by aggregating the component-level estimates according to the given compositional structure. This approach facilitates construction of *instance-specific* causal models (models whose structure changes based on the specific components present in the specific units being modeled) using *modular* neural network architectures to explicitly represent the components of each unit of analysis (Figure 1(c)).[2] **We formalize a novel *compositional* framework in the context of causal effects and potential outcomes** to facilitate the study of the compositional approach and provide a detailed analysis of the unique benefits and costs of such an approach for accurate unit-level CATE estimation. We focus on the case of hierarchically structured compositional systems without feedback and with simple interactions because this represents the minimal compositional system necessary to understand the key characteristics of the compositional approach and compare it to unitary modeling approaches.

We show that the compositional approach provides several novel benefits for causal inference from observational data. The instance-specific model allows *scalable* causal effect estimation for variable-size units by greatly reducing the inherent dimensionality of the task. Instance-specific modular architectures are widely used in associational machine learning for modeling large-scale natural language data, structured vision data, and sequential decision-making, providing sample efficiency and computation benefits (Shazeer et al., 2016; Pfeiffer et al., 2023). However, only a relatively sparse body of work in causal inference has focused on using instance-specific modular models using hierarchical and relational data (Maier et al., 2013; Lee and Honavar, 2016; Salimi et al., 2020; Ahsan et al., 2023), and even this work has been severely hampered by the lack of available data from compositional domains. To address this gap, **we introduce three novel and realistic**

1. Here, individual-level effect estimation refers to conditional average treatment effect (CATE) estimation and heterogeneous treatment effect estimation (Athey and Imbens, 2016; Shalit et al., 2017; Künzel et al., 2019).
2. Modular neural network architectures are chosen as an implementation choice to demonstrate a compositional approach, but component-wise causal estimation can be done using a variety of parametric and non-parametric model classes.

**evaluation environments to evaluate compositional approaches for causal effect estimation** — query execution in relational databases, matrix processing on different types of computer hardware, and simulated manufacturing assembly line data based on a realistic simulator.

The modular structure incorporated in the compositional approach facilitates effect estimation for units with unseen combinations of components, enabling *compositional generalization*. In various fields of machine learning — computer vision (Andreas et al., 2016), language (Hupkes et al., 2020), reinforcement learning (Peng et al., 2019), and program synthesis (Shi et al., 2024), researchers have studied the compositional generalization capabilities of the modular approaches as compared to non-modular approaches for prediction tasks (Bahdanau et al., 2019; Jarvis et al., 2023). However, a study of the benefits of the compositional approach compared to the standard approaches is missing in causal inference. **We study the relative *compositional generalization* capabilities of compositional and unitary approaches in estimating individual treatment effects for novel units.**

Individual effect estimation from observational data requires assumptions such as *ignorability* and *overlap* (Pearl, 2009; Rubin, 2005). Satisfying the overlap assumption becomes challenging as the dimensionality of covariates increases (D'Amour et al., 2021). Learning lower-dimensional representations of data that satisfies the ignorability and overlap assumption is desirable in such situations (Johansson et al., 2016, 2022). Exploiting the compositionality of the underlying data-generating process is one way to learn a lower dimensional representation, allowing better overlap between treatment groups. **We show that the compositional approach performs better than the unitary approach as the distributional mismatch between the treatment and control groups increases, especially in cases where treatment is assigned based on the unit's structure.**

Despite these potential benefits, learning compositional models for effect estimation has pitfalls, including larger numbers of parameters to estimate, sensitivity to individual components, and errors in modeling component interactions. In this paper, **we analyze the role of component-level data access, composition structure, and heterogeneity in component function complexity in the relative performance of the compositional approach**. For example, we observe that compositional and unitary approaches perform similarly when modeling systems with homogeneous component functions—such as matrix processing—where a single component (e.g., matrix multiplication) dominates the overall unit-level outcome and all component outcome functions belong to the same polynomial class. See Section 5 for additional pitfalls.

**Note:** We include a detailed discussion of the related work in the appendix (Section B).

## 2. Compositional Framework for Causal Effect Estimation

Below, we describe the compositional data-generating process in modular systems, provide the estimands of interest at the unit and component levels, and discuss identifiability assumptions.

**Preliminaries:** Assume that each unit $i$ has pre-treatment covariates $\mathbf{X}_i = \mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, a binary treatment $T_i \in \{0, 1\}$, and two potential outcomes $\{Y_i(0), Y_i(1)\} \in \mathcal{Y} \subset \mathbb{R}$ (Rubin, 1974, 2005). In observational data, we only observe one of the potential outcomes for each unit, $Y_i = Y_i(T_i)$, known as the *observed* or *factual* outcome. The missing outcomes $Y_{iCF} = Y_i(1 - T_i)$ are known as *unobserved* or *counterfactual* outcomes. Conditional average treatment effect (CATE) is defined as $\tau(x) : \mathbb{E}[Y_i(1) - Y_i(0)|\mathbf{X}_i = \mathbf{x}]$. Estimating CATE requires assumptions of ignorability, overlap, and consistency (Rosenbaum and Rubin, 1983). Under these assumptions, $\tau(x)$ is identifiable by $\tau(\mathbf{x}) = \mathbb{E}[Y_i|\mathbf{X}_i = \mathbf{x}, T = 1] - \mathbb{E}[Y_i|\mathbf{X}_i = \mathbf{x}, T = 0]$ (Pearl, 2009). CATE estimation typically uses direct outcome modeling with treatment as a feature, separate regression models (Künzel et al., 2019),

or propensity score methods (Kennedy, 2023). We illustrate the compositional approach by directly estimating the potential outcomes using shared treatment.

### 2.1. Compositional data generating process

Consider a compositional system with $k$ distinct and heterogeneous classes of components: $\mathcal{M} = \{M_1, M_2, \dots M_k\}$. All units share this set of reusable components. Each structured unit $Q_i : (G_i, \{\mathbf{X}_{ij}\}_{j=1:m_i})$ is described using an interaction graph $G_i$ and a set of component-specific covariates $\{\mathbf{X}_{ij}\}_{j=1:m_i}$, where $m_i$ denotes the number of components in unit $i$. Note that the number of components $m_i$ can be greater than the number of distinct components $k$ in the system, indicating the presence of multiple instances of each component class in some or all units.

The graph $G_i = (C_i, E_i)$ is a directed hierarchical tree representing component interactions (Figure 1(a)), with nodes $C_i$ and edges $E_i$. Each unit $i$ contains components $C_i = \{c_1, c_2, \dots c_{m_i}\}$, where each component belongs to a class $c \in M_o, o \in \{1, 2, \dots k\}$. $G_i$ defines the processing order of $m_i$ components for unit $i$, with instance-specific structure varying across units. Components process structured units from top-to-bottom, with final output from the bottom-most node. For an edge $c_j \to c_{j'}$, $c_j$ is the parent and $c_{j'}$ the child component. $Pa_{G_i}(c_{j'})$ denotes indices of components with direct edges to $c_{j'}$. Component $j$'s covariates are $\mathbf{X}_{ij} \in \mathbb{R}^{d_j}$, where subscript $i$ denotes the unit and $j$ denotes the component instance.

**Shared treatment and treatment assignment mechanism in structured units:** A unit-level treatment $T_i$ is selected for each unit, affecting the potential outcomes of some or all components through shared or distinct mechanisms. While the compositional framework allows component-specific treatment analysis, we focus on unit-level treatments to facilitate a direct comparison of the compositional and unitary approaches. The treatment assignment mechanism $P(T_i = 1|Q_i = q)$ depends on both the graph structure and joint covariate distribution, introducing two sources of observational bias: (1) distribution shift among the covariates; and (2) distribution shift in the structure and composition of the components between treatment groups.

**Unit-level outcome and fine-grained outcomes:** Let $Y_i(t)$ denote the unit-level and $\{Y_{ij}(t)\}_{j=1:m_i}$ denote the component-level potential outcomes under treatment $t$ for unit $Q_i$. The interaction graph $G_i$ also defines the causal dependencies among potential outcomes. We make the causal Markov assumption for components in the graph $G_i$ that the potential outcome of a component $j$ directly depends on the component's covariates $\mathbf{X}_j$, shared treatment $T_i$ and the outcomes from the parent components. For component class $o \in \{1, 2, \dots k\}$, let $\mu_{ot}$ denote the ground-truth *expected* potential outcome function, shared across instances of the same component. If component-level noise is assumed to be zero-mean random variables $\epsilon_o(0), \epsilon_o(1)$ and $c_j \in M_o$,[3] the data-generating process for $Y_{ij}(t), j \in \{1, 2, \dots m_i\}$ and $t \in \{0, 1\}$:

$$Y_{ij}(t) = \mu_{ot}(\mathbf{X}_{ij}, \{Y_{il}(t)\}_{\{l \in Pa(c_j)\}}) + \epsilon_{io}(t). \tag{1}$$

The unit-level potential outcome is generated by aggregating component outcomes via an instance-specific function $g$: $Y_i(t) = g(Y_{i1}(t), Y_{i2}(t) \dots Y_{im_i}(t), G_i)$. For composition in the hierarchical graph, the data-generating process of each component's outcome already takes the input from the parent's outcome, following Markov dependence (Equation 1). We can define these outcomes cumulatively, meaning we aggregate them up as we go along, so that the final component's outcome

3. Additive noise simplifies modeling conditional distributions of component potential outcomes in Section 3.1, though the compositional approach also extends to non-additive noise.

represents the complete unit-level outcome, $Y_i(t) = Y_{im_i}(t)$, where $m_i$ indicates the last component in $G_i$. For example, consider query execution time: rather than measuring each component's time separately, we can measure the accumulated time of each component and all its parent components. In Figure 1(c), this cumulative approach means we can use the final sequential scan's outcome for $Q_1$ as the total execution time of the unit. This formulation allows us to learn the instance-specific aggregation functions and unit outcomes as part of learning the component potential outcomes.

**Note on a formal definition of causal compositionality:** A mathematically precise definition of compositionality is an active research area in machine learning (Ram et al., 2024; Elmoznino et al., 2025). Given the state of the current literature, our work adopts a data-generating process view of compositionality inspired by real-world computational systems with explicit causal mechanisms. More specifically, compositionality in our work is defined through: (1) structured units with instance-specific compositions; and (2) component interactions that are formalized through an interaction graph and the Markov assumption (Equation 1). Previous work in machine learning has used a similar data-generating process view to develop compositional approaches (Andreas et al., 2016; Wiedemer et al., 2024). We also provide a graphical plate notation-based representation of the compositional data generating process in Section C.

### 2.2. Unitary representation of compositional data

As already mentioned, an alternative to a compositional model is a unitary model in which units are represented by a single, fixed-size, high-dimensional feature vector, $\mathbf{X}_i \in \mathbb{R}^d$ that represents some aggregation of the component level input features $\{\mathbf{X}_{ij}\}_{j=1}^{m_i}$. For simplicity in our experiments, we assume the mean aggregation function (i.e., for $N_o$ occurrences of component class $o$ in unit $i$, $\mathbf{X}_{io} = \frac{1}{N_o}\sum_{j,c_j \in M_o} \mathbf{X}_{ij}$). Similarly, corresponding outcomes are also aggregated. To fairly compare the unitary and compositional approaches, we also incorporate structural information by including the number of instances of each component $N_{jl}$ present in each unit at each tree depth with maximum depth $L$: $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_k, N_{11}, N_{12} \ldots N_{kL}]$.

### 2.3. Causal Estimands

The CATE for a structured unit $q$ is conditional on both the structure $G_i$ and the set of component-level features $\{\mathbf{x}_{ij}\}_{j=1:m_i}$. Taking the conditional expectation with respect to the structure and variable-length representation of the units allows a more accurate definition of the CATE for compositional units than the standard unitary representation. For ease of notation to describe conditional distributions, we denote the combined inputs to a component $j$ as $\mathbf{Z}_j(t) = (\mathbf{X}_j, \{Y_l(t)\}_{l \in Pa(C_j)})$.

**Definition 1** *The conditional-average treatment effect (CATE) estimand for structured input $Q_i = q$ is defined as:* $\tau(q) = \mathbb{E}[Y_i(1) - Y_i(0)|Q_i = q] = \mathbb{E}[Y_i(1) - Y_i(0)|Q_i = (G_i, \{\mathbf{x}_{ij}\}_{j=1:m_i})]$

**Definition 2** *The conditional-average treatment effect (CATE) estimand for component $j$ with $\mathbf{X}_j = \mathbf{x}_j \in \mathbb{R}^{d_j}$ is defined as:* $\tau(\mathbf{z}_j) = \mathbb{E}[Y_j(1) - Y_j(0)|\mathbf{Z}_j = \mathbf{z}_j]$.

We define the component-wise distributions as $P(Y_j(t)|\mathbf{Z}_j(t) = \mathbf{z}_j)$. In hierarchical composition, the unit outcome equals the final component outcome: $\mathbb{E}[Y_i(t)|Q_i = q] = \mathbb{E}[Y_{im_i}(t)|Q_i = q]$. This conditional expectation can be expressed by marginalizing intermediate component outcomes using the Markov assumption (Equation 1).

$$\mathbb{E}[Y_i(t)|Q_i = q] = \int_{Y_{im_i-1}(t)} \int_{Y_{im_i-2}(t)} \cdots \int_{Y_{i1}(t)} \mathbb{E}[Y_{im_i}(t)|\mathbf{Z}_{im_i}(t)] \prod_{j=1}^{m_i-1} P(Y_{ij}(t)|\mathbf{Z}_{ij}(t))$$

We use the following nested expectation expression as shorthand for marginalization over intermediate component outcomes: $\mathbb{E}[Y_i(t)|Q_i = q] = \mathbb{E}_{Y_{i1:im_i-1}(t)}[\mathbb{E}[Y_{im_i}(t)|\mathbf{Z}_{im_i}(t)]]$.

### 2.4. Identifiability assumptions

The key identifiability assumptions for component-level causal estimands are similar to those for unit-level estimands — ignorability, consistency, and overlap. However, in structured units having multiple heterogeneous components and instance-specific composition, it is more plausible for these assumptions to hold for the component level rather than the unit level, particularly for ignorability and overlap.

**Ignorability:** Component-level ignorability assumes that component level potential and assigned treatment are independent conditioned on the components' covariates, i.e., $Y_j(1), Y_j(0) \perp T|\mathbf{X}_j$. Component-level ignorability is based on the intuition that components are distinct heterogeneous sub-systems that are specialized to process parts of the whole input. This suggests that a subset of the unit-level high-dimensional covariates is sufficient to predict a component's outcome, even when treatment might have been assigned based on the structure of the components or joint distribution of the component covariates. The component-level potential outcomes depend on both the component's pre-treatment covariates and the potential outcomes of the parent components. As the treatment is assigned before any component's potential outcomes are observed, we can assume that the component's covariates $\mathbf{X}_j$ are sufficient to satisfy ignorability assumptions.

**Overlap:** Component-level overlap assumes that overlap holds for the component level covariates $\mathbf{X}_j = \mathbf{x}_j$, i.e., $\forall \mathbf{x}_j \in \mathcal{X}_j, t \in \{0, 1\} : \ 0 < p(T = t|\mathbf{X}_j = \mathbf{x}_j) < 1$. When unit-level overlap holds, component-level overlap is implied automatically for the feature subset. In compositional data, there can be two sources for distribution mismatch. (1) *Structure-based treatment assignment:* Suppose treatment depends strongly on graph structure ($P(T = 1|G = g_1) = 1$, $P(T = 1|G = g_2) = 0$). For unitary representation including structural features $N_{jl}$, the overlap assumption is violated for units with the structures as $g_1$ and $g_2$, while overlap is maintained for compositional approaches as the structure is incorporated as part of the model rather than input representation. If we exclude structural information from unitary representation, then overlap is satisfied, but ignorability is violated because the structure is a confounder affecting both $T_i$ and $Y_i$. (2) *Covariate-based treatment assignment:* When treatment depends on the covariate distribution, both approaches' identifiability relies on overlap quality. Due to the compositional nature of units, violation of overlap for a component's covariates violates overlap for the unit-level and vice versa. However, due to the lower dimensionality of the component's covariates, the degree of distribution mismatch between the covariates is likely to be lower than the unit-level high-dimensional covariates.

**Identifiability for hierarchical composition:** The CATE estimand for a structured unit $Q_i = q$ is identified by the following: $\tau(q) = \mathbb{E}_{Y_{1:m_i-1}}[\mathbb{E}[Y_{m_i}|\mathbf{Z}_{m_i} = \mathbf{z}_{m_i}, T = 1]] - \mathbb{E}_{Y_{1:m_i-1}}[\mathbb{E}[Y_{m_i}|\mathbf{Z}_{m_i} = \mathbf{z}_{m_i}, T = 0]]$ if we assume Markov dependence assumption (Equation 1), component-wise ignorability, overlap, and consistency. The proof is provided in the supplementary material.

**Additive parallel composition (special case):** A special case of the interaction graph $G_i$ is when all the components are independent and parallely compute the potential outcomes. This condition can be expressed as when the potential outcomes of components are *conditionally independent* given the component's covariates, i.e., $Y_j(t) \perp Y_l(t)|\mathbf{X}_j \forall j \in \{1, \dots k\}, j \neq l$. Suppose the aggregation function is a linear combination of the component's outcomes, e.g., addition. The unit-level CATE can be expressed as the sum of the potential outcome estimands. $Y_{ij}(t) = \mu_{ot}(\mathbf{X}_{ij}) + \epsilon_{io}(t)$, $Y_i(t) =$

$\sum_{j=1}^{m_i} Y_{ij}(t)$. In that case, the CATE can be directly expressed as the linear combination of the component-level CATE and is identified as follows: $\tau(q) = \sum_{j=1}^{m_i} \tau(\mathbf{x}_{ij}) = \sum_{j=1}^{m_i} \mathbb{E}[Y_{ij}|\mathbf{x}_{ij}, T_i = 1] - \mathbb{E}[Y_{ij}|\mathbf{x}_{ij}, T_i = 0]$. The proof is provided in the supplementary material. Parallel composition appears across machine learning domains—from independent composition of latent object attributes in computer vision (Higgins et al., 2018; Wiedemer et al., 2024) to spatial skill composition in reinforcement learning (Van Niekerk et al., 2019). Similarly, sequential composition appears in the options framework in reinforcement learning (Sutton et al., 1999). Our work examines these varied composition structures to understand their impact on compositional generalization.

## 3. Learning compositional models from observational data

Below, we discuss the algorithm for learning the hierarchical composition model from observational data, given different amounts of information about the component's covariates and outcomes. We include algorithms for parallel structured composition models in Section E.

### 3.1. Hierarchical Composition Model

The below algorithm facilitates the modeling of noise variables $\epsilon_{ot}$ (assuming zero-mean additive noise variables in Equation 1) along with the *expected* potential outcome functions $\mu_{ot}$. This allows the marginalization of intermediate component-level potential outcomes to obtain unit-level CATE estimates for sequential and hierarchical compositions.

**Model Training:** The component models for estimating mean and variance of conditional distribution of the component-level potential outcomes for component class $o \in \{1, 2, \dots k\}$ are denoted by $(\hat{f}_{\theta_o}, \hat{\sigma}^2_{\psi_o}) : \mathbb{R}^{d_o} \times \mathbb{R}^D \times \{0, 1\} \to \mathbb{R} \times \mathbb{R}^+$, assuming maximum in-degree of the graph $G_i$ as $D$. Each model corresponding to component class $o$ is parameterized by separate and independent parameters $\theta_o$ for the mean and $\psi_o$ for the variance. For a given observational data set with $n$ samples, $\mathcal{D}_F = \{q_i, t_i, y_i\}_{i=1:n}$, we assume that we observe component-level features $\{\mathbf{x}_{ij}\}_{j=1:m_i}$, assigned treatment $t_i$ and fine-grained component-level potential outcomes $\{y_{ij}\}_{j=1:m_i}$ along with unit-level potential outcomes $y_i$. If component instance $c_j \in M_o$, training of each component model $o$ involves the independent learning of the parameters by minimizing the *negative log-likelihood loss*:

$$(\theta_o^*, \psi_o^*) := \arg\min \theta_o, \psi_o \frac{1}{N_o} \sum_{m=1}^{N_o} \left[ \frac{1}{2} \log(2\pi \hat{\sigma}^2_{\psi_o}(\mathbf{z}_m, t_m)) + \frac{(y_{ij} - \hat{f}_{\theta_o}(\mathbf{z}_m, t_m))^2}{2\hat{\sigma}^2_{\psi_o}(\mathbf{z}_m, t_m)} \right] \tag{2}$$

,where $N_o$ denotes the total number of component instances of component class $o$ across all the $N$ samples, and $m$ denotes the index of the component instance in class $o$.

**Model Inference:** To estimate the CATE for a unit $i$, a modular architecture consisting of $m_i$ component models is instantiated with the same input and output structure as $G_i$. During inference for treatment $T = t$, the predictions of potential outcomes of the parent's components $\{\hat{y}_{lt}\}_{l \in Pa(c_j)}$, $l \in M_o$ are sampled from a normal distribution $\hat{y}_{lt} \sim N(\hat{\mu}_{ot}, \hat{\sigma}^2_{ot})$. We slightly abuse the notation to denote the inferred variance and variance model using $\sigma^2$. The mean and variance of the $j^{th}$ component outcome are obtained using these samples: $\hat{\mu}_{jt} = \hat{f}_{\theta_o^*}(\mathbf{x}_j, \{\hat{y}_{lt}\}_{l \in Pa(c_j)}, t) = \hat{f}_{\theta_o^*}(\hat{z}_{jt}, t)$, $\hat{\sigma}^2_{jt} = \hat{\sigma}^2_{\psi_o^*}(\mathbf{x}_j, \{\hat{y}_{lt}\}_{l \in Pa(c_j)}, t) = \hat{\sigma}^2_{\psi_o^*}(\hat{z}_{jt}, t)$. The estimate $\mathbb{E}[Y_i(1) - Y_i(0)|Q_i = q]$ of CATE is obtained through Monte Carlo integration by averaging over $S$ samples: $\hat{\tau}(q) \approx \frac{1}{S} \sum_{s=1}^{S} [\hat{f}_{\theta_o^*}(\hat{z}^{(s)}_{m_i t}, 1) - \hat{f}_{\theta_o^*}(\hat{z}^{(s)}_{m_i t}, 0)]$, where each sample path $\hat{z}^{(s)}_{m_i t}$ is generated by hierarchically sampling from the component distributions: $\hat{y}^{(s)}_{jt} \sim \mathcal{N}(\hat{\mu}_{jt}, \hat{\sigma}^2_{jt})$ for $j = 1$ to $m_i - 1$.

### 3.2. Relaxing assumptions about component-level data access

The model description above assumes observed component-level covariates and outcomes. This assumption is often reasonable, given the wide availability of fine-grained data for many structured domains. However, other cases might exist when only the unit-level covariates $\mathbf{X}$ and outcomes are observed, and the component-level covariates $\mathbf{X}_j$ and outcomes $Y_j$ are unobserved. Below, we discuss hierarchical composition models for these cases.

**Case 1: *Unobserved* $\mathbf{X}_j$, *observed* $Y_j$:** We jointly learn the lower-dimensional component-level representations $\phi_o : \mathbb{R}^d \to \mathbb{R}^{d'_o}$, as well as the parameters of outcome functions $(\theta_o, \psi_o)$ by assuming $z_j = (\phi_o(x_j), \{y_{lt}\}_{l \in Pa(c_j)})$ in Equation 2.

**Case 2: *Observed* $\mathbf{X}_j$, *unobserved* $Y_j$:** The model architecture remains the same as before, but we do not have individual component-level loss functions and only know the loss function for unit-level outcomes. Due to this, the parameters of the components are jointly learned to optimize the loss of estimating unit-level outcomes. If $\circ$ denotes the functional composition, and functions are composed in the same hierarchical order as $G_i$, then the joint loss function is given by: $[\theta_1, \theta_2 \dots \theta_k] := \arg\min_{\mathbf{\Theta}} \frac{1}{N} \sum_{i=1}^{N} (\hat{f}_{\theta_{m_i}} \circ \hat{f}_{\theta_{m_i-1}} \circ \cdots \circ \hat{f}_{\theta_1}(\mathbf{x}_{i1}, t_i) - y_i)^2$

**Case 3: *Unobserved* $\mathbf{X}_j$ and *unobserved* $Y_j$:** In this case, we only assume the knowledge of $G_i$. $[\theta_1, \theta_2 \dots \theta_k] := \arg\min_{\mathbf{\Theta}} \frac{1}{N} \sum_{i=1}^{N} (\hat{f}_{\theta_{m_i}} \circ \hat{f}_{\theta_{m_i-1}} \circ \cdots \circ \hat{f}_{\theta_1}(\phi_1(\mathbf{x}_i), t_i) - y_i)^2$

**Identifiability under non-observability:** Under the assumption of unobserved component-level data (e.g., case 3), the compositional model has unit-level information and structural knowledge, the same information as unitary models. The only difference between the two approaches is that structure is incorporated in the compositional model, while in the unitary model, it is passed in the form of high-level structural features. Thus, identification holds under the non-observability of component data access as long as the identifiability conditions (ignorability and overlap) hold for the unit-level covariates. As the unit-level features are aggregated, the ignorability might be affected due to the approximate representation of the units.

## 4. Experimental Infrastructure

We describe the experimental setup to evaluate the models across various *in-distribution* and *out-of-distribution* settings. We provide a detailed description of the causal effect estimation task, unit-level and component-level covariates, treatment, and outcomes for each domain in Section F [4].

### 4.1. Datasets

Research on modeling causal effects for compositional data has been hampered by the lack of real-world benchmarks. To facilitate effective empirical evaluation of the utility of compositional modeling of causal effect estimation, we introduce two benchmarks based on real-world computational systems and one benchmark based on a realistic simulation.

**Query execution in relational databases:** We collected real-world query execution plans data by running 10,000 publicly available SQL queries against the Stack Overflow database under different configurations (memory size, indexing, page cost), treating configuration parameters as interventions and execution time as the potential outcome, assuming additive composition.

4. Data and code for creating benchmarks and reproducing experiments is available at https://github.com/KDL-umass/compositional_models_cate.

**Manufacturing plant data:** We use a discrete-event simulation framework, Simpy, to generate realistic manufacturing plant data. The simulation includes four manufacturing processes (material processing, joining, electronics processing, and assembly) combined across 50 hierarchical manufacturing line layouts, assuming hierarchical composition. Each layout consists of various product demand (5-1,000) with different raw material inventories as covariates. Treatment compares five versus fifteen skilled workers, measuring total parts produced as outcomes.

**Matrix operations processing:** The dataset consists of 25,000 samples for 25 matrix expression structures (units) evaluated on two different computer hardware (treatment), with matrix dimensions ranging from 2 to 1,000. Component operations encompass 12 component operations (e.g., multiplication, inverse, singular value decomposition, etc.). We ensure each operation is executed individually, ensuring parallel composition with additive aggregation function. Matrix size is used as a biasing covariate to create a distribution mismatch between treatment groups.

**Synthetic data:** In addition to these real-world benchmarks, we also generate simulated data to systematically understand the role of component-level data access, composition structure, and heterogeneity in components' response surfaces. Composition structures include sequential and additive parallel composition. Structured units are generated by sampling binary trees (max depth=10) with $k = 10$ heterogeneous modules, each having $d_j = 1$ feature ($d = 10$ features total).

### 4.2. Models and Evaluation Criteria

**Compositional Models:** We implement four versions of the compositional models (hierarchical and parallel, depending on the domain) with varying amount of component-level data access as discussed in Sections 3.1 and 3.2: (1) *Compositional (observed $\mathbf{X}_j$, observed $Y_j$)*; (2) *Compositional (unobserved $\mathbf{X}_j$, observed $Y_j$)*; (3) *Compositional (observed $\mathbf{X}_j$, unobserved $Y_j$)*; and (4) *Compositional (unobserved $\mathbf{X}_j$, unobserved $Y_j$)*. The component-level models are implemented as neural networks, independently trained in case of access to fine-grained outcomes and jointly end-to-end trained otherwise. **Note:** Unless stated explicitly, the legend "compositional" in experimental results implies a model with observed component-level covariates $\mathbf{X}_j$ and outcomes $Y_j$.

**Baselines:** We compare the performance of the compositional models with three types of existing approaches for CATE estimation: (1) *TNet*, a neural network-based CATE estimator (Curth and Van der Schaar, 2021); (2) *X-learner*, a meta learner that uses plug-in estimators to compute CATE, with random forest as the base model class (Künzel et al., 2019); (3) Non-parametric *Double ML* (Chernozhukov et al., 2018); and (4) Vanilla *neural network* and *random forest*-based outcome regression models.

**Creation of observational data sets:** Real-world computational systems provide experimental datasets with observed outcomes for both treatments, from which we create observational datasets by introducing structure and covariate-based confounding bias (Gentzel et al., 2021). Bias strength ranges from 0 (experimental) to 10 (observational), with treatment probabilities varying between 0.01-0.99, creating treatment group distribution shifts. Unconfoundedness holds as biasing information is observed in both approaches. Higher "bias strength" indicates higher treatment probability for specific biasing covariate values and structure.

**Evaluating compositional generalization:** We evaluate compositional generalization by training on structures with 2 to $K$ module combinations and testing on combinations containing all $K$ module combinations. For example, when training on 2-module combinations, we use all possible pairs (e.g., $(C_1, C_2)$, $(C_1, C_3)$, $(C_2, C_3)$), including varied orders and repeated modules (e.g., $(C_1, C_2, C_1)$),

and test on larger combinations like $(C_1, C_2, C_3)$. For real-world data, we train on structures with smaller tree depths and test on larger ones.

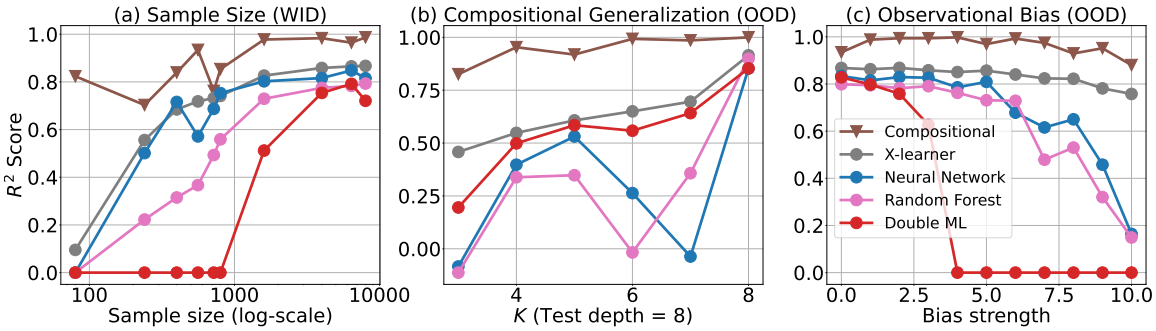



Figure 2: **Results on manufacturing domain** $(10,000$ samples). We report $R^2$ between CATE estimates and ground-truth effects (higher is better). (a) *Sample-size efficiency (WID):* Compositional models are more accurate and sample-efficient in CATE estimation for within-distribution settings. (b) *Compositional generalization (OOD):* Models are trained on units with tree depth $\leq K$ and evaluated on a test-set with depth $8$. Compositional models generalize to the unseen combinations due to compositional structure whereas non-compositional baselines perform comparably only after training on units similar to test data. (c) *Effect of increasing observational bias (OOD):* Models are trained and tested on data with increasing observational bias strength between assigned treatment and tree depth. Compositional models and X-learner are less affected by increased observational bias.

**Performance and Evaluation Metrics:** Performance is evaluated in two settings: *WID* (same structure/covariate distribution in train/test) and *OOD*, which includes (1) *compositional generalization* (testing on unseen component combinations) and (2) *observational bias* (structure/covariate-dependent treatment assignment). We measure PEHE loss (Hill, 2011): $\epsilon_{PEHE}(\hat{f}) = \mathbb{E}[(\hat{\tau}_{\hat{f}}(q) - \tau(q))^2]$, and $R^2$ score for datasets with large outcome values or high performance variability.

## 5. Findings

In this section, we provide the key findings from our experiments and discuss the mechanisms responsible for compositional models' performance compared to the baselines.

**Compositional models can provide substantially more accurate CATE estimation for structured units:** Figure 2(a) shows results from the manufacturing domain in which compositional and unitary models were learned from experimental data and evaluated in terms of their in-distribution performance. The compositional model has substantially lower error than the baselines, particularly for small sample sizes. More detailed analysis showed that the performance advantage of compositional models in this setting is primarily due to two factors: (1) units in this domain have heterogeneous hierarchical structure and instance-specific models more accurately represent this structure; and (2) compositional models were learned with higher sample efficiency due to the existence of multiple samples of each component per unit (see Figure 7(a) in the supplementary materials).

**Incorporating modular structure enables compositional generalization:** Figures 2(b) and 3 report the compositional generalization performance of the models in the manufacturing and synthetic domains, respectively. Compositional models were able to successfully generalize to high-complexity units (large depth or number of module combinations, respectively) even though they had only been

trained on low-complexity units. In contrast, unitary models perform worse (often *far* worse) until they are trained on units that equal or nearly equal the compositional complexity of the test units.

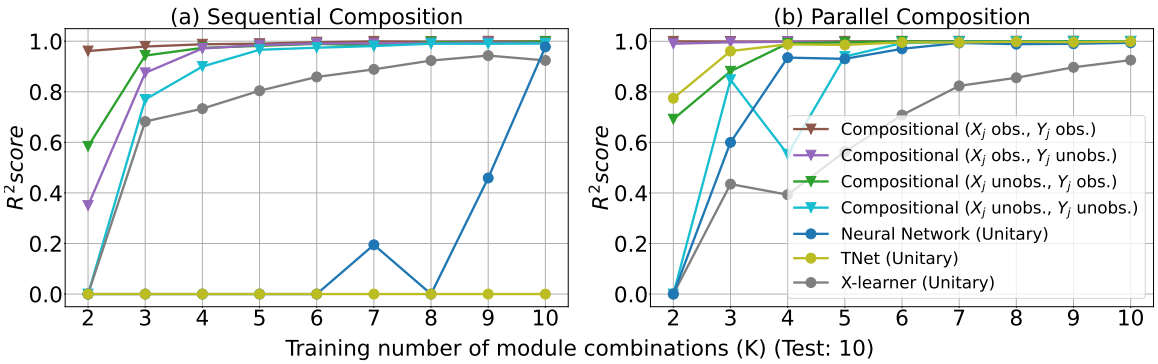



Figure 3: **Role of component-level data access and composition structure in the performance of compositional models:** $R^2$ score for models evaluated on compositional generalization task with varying degrees of component-level data access. (Higher is better; PEHE errors are reported in Figure 14 of the supplementary material). We observe that end-to-end trained models incorporating just modular structure compositionally generalize as trained on more module combinations. Unitary models show compositional generalization for additive parallel composition but perform comparably only for in-distribution combinations ($K$=10) for sequential composition, except X-learner. Note that the number of training samples increases as training depth increases.

**Compositional models are less affected by observational bias in treatment assignment:** Figure 2(c) reports the effects of differing degrees of observational bias based on the instance-specific structure in the manufacturing domain. Compositional models (and the unitary X-learner) are least affected by this form of observational bias. Other unitary models (Neural network and Random forest) are strongly affected, and another unitary model (double ML) is the most strongly affected due to its use of propensity score weighting. Figure 4(a) reports results for the query execution domain, where bias was introduced based on the covariate distribution. The error of all models increases because covariate-based bias affects both unit-level and component-level balance, but the error of the compositional model is lower and increases more slowly than the unitary baselines.

**When trained end-to-end, compositional models perform remarkably well even without component-level data access:** Figure 3 reports the performance of models trained on synthetic data with varying degrees of data access. Figure 3(a) and (b) report results on a synthetic domain in which units have sequential and parallel structures, respectively. Compositional models outperform unitary models regardless of their access to component-level covariates and outcomes (more for sequential case than parallel case as discussed below), demonstrating that compositional models can effectively estimate CATE for novel units in scenarios where component-level data is unavailable, and training uses only the compositional structure and unit-level data.

**Unitary approaches compositionally generalize more easily for additive composition structures than sequential structures:** The results in Figure 3 also show that unitary models can generalize for some composition types better than others. Figure 3(a) and (b) show results from units simulated with strictly sequential and (additive) parallel composition, respectively, with exactly the same component functions and covariate distributions. While most unitary models perform very poorly in the case of sequential composition, nearly all perform fairly well in the case of additive

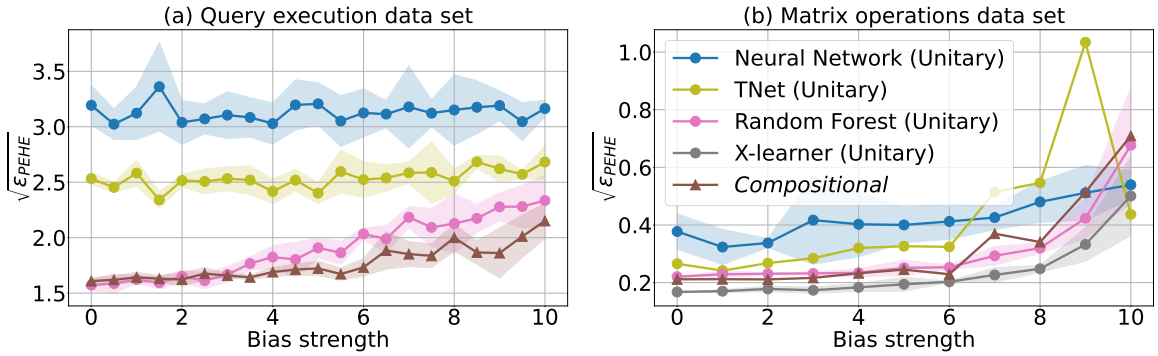



Figure 4: **Results for real-world data sets:** (a) *Query execution data set:* Compositional model estimates the effect more accurately as observational bias increases. (b) *Matrix operations data set:* All baselines perform similarly for this data set due to a single shared covariate, homogenous component outcome functions, and dominant contribution of matrix multiplication.

parallel composition, particularly as module combinations become more similar to the test data. The role of structure type in compositional generalization is often overlooked in relevant work in machine learning, where most work assumes either additive or sequential compositions.

**Some factors can eliminate the advantages of compositional models for causal estimation:** Figure 4(b) reports results for one realistic domain we studied — matrix operations — in which compositional models provide no substantial advantage over unitary baselines in both experimental (bias-strength = 0) and observational (bias-strength > 0) settings. This contrasts sharply with the superior performance of compositional models in the manufacturing (Figures 2, 3) and query execution (Figure 4(a)) domains. Further investigation identified three factors that explain this result: (1) the *dominance* of one component—matrix multiplication—in determining overall runtime (Figure 12); (2) *homogeneous* functional forms of the component outcome functions, such that additive composition leads to similar unit-level outcome functions (Figure 13); and (3) a *single shared covariate*—matrix size—that affects both unit-level and component-level outcomes, creating similar distribution imbalance issues at both the unit and component levels. In contrast, the query execution and manufacturing domains were more heterogeneous—different covariates affect component-level outcomes, no single component dominated in producing overall effects, component outcome functions belonged to different function classes, and composition structures were more complex. Thus, compositional models had higher relative performance in those domains.

## 6. Conclusion

The compositional models for causal effect estimation show promise in complex, modular, and heterogeneous systems. Compositional modeling provides a scalable and practical perspective for instance-specific causal reasoning in modern technological systems. This work focuses on compositional models for shared treatment and individual effect estimation. Compositional causal reasoning about component-specific treatments, such as selecting the optimal configuration parameters for each component, and reasoning about the wide array of realistic interventions, such as adding, replacing, or removing components from the system, are useful directions for future work.

## Acknowledgments

The authors thank Pracheta Amaranath, Andy Zane, Kate Avery, Sankaran Vaidyanathan, Justin Clarke, the anonymous reviewers and the area chair for helpful comments and suggestions. Thanks to Anirban Ghosh for helping design the manufacturing assembly simulator. This research was supported by DARPA and the United States Air Force under the SAIL-ON (Contract No. w911NF-20-2-0005) program. Purva also received the Dissertation Writing Fellowship from the Manning College of Information and Computer Sciences in the Spring of 2024. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of DARPA, ARO, the United States Air Force, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

## References

Ragib Ahsan, David Arbour, and Elena Zheleva. Learning relational causal models with cycles through relational acyclification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12164–12171, 2023.

John Aldrich. Autonomy. *Oxford Economic Papers*, 41(1):15–34, 1989.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*, 2019.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14 (11), 2013.

Werner Callebaut and Diego Rasskin-Gutman. *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. MIT press, 2005.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018.

Alicia Curth and Mihaela Van der Schaar. Nonparametric estimation of heterogeneous treatment effects: From theory to learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1810–1818. PMLR, 2021.

Alicia Curth, Richard W Peck, Eoin McKinney, James Weatherall, and Mihaela van Der Schaar. Using machine learning to individualize treatment effect estimation: Challenges and opportunities. *Clinical Pharmacology & Therapeutics*, 2024.

Alexander D'Amour, Peng Ding, Avi Feller, Lihua Lei, and Jasjeet Sekhon. Overlap in observational studies with high-dimensional covariates. *Journal of Econometrics*, 221(2):644–654, 2021.

Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. Towards a formal theory of compositionality, 2025. URL https://openreview.net/forum?id=hKMPz3wkPV.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, volume 99, pages 1300–1309, 1999.

Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2006.

Amanda M Gentzel, Purva Pruthi, and David Jensen. How and why to use experimental data to evaluate methods for observational causal inference. In *International Conference on Machine Learning*, pages 3660–3671. PMLR, 2021.

Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT press, 2007.

Trygve Haavelmo. The probability approach in econometrics. *Econometrica: Journal of the Econometric Society*, pages iii–115, 1944.

Shonosuke Harada and Hisashi Kashima. Graphite: Estimating individual effects of graph-structured treatments. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 659–668, 2021.

David Heckerman and Michael P Wellman. Bayesian networks. *Communications of the ACM*, 38(3): 27–31, 1995.

I Higgins, N Sonnerat, L Matthey, A Pal, CP Burgess, M Bošnjak, M Shanahan, M Botvinick, D Hassabis, and A Lerchner. SCAN: Learning hierarchical compositional visual concepts. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Learning Representations (ICLR), 2018.

Jennifer L Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.

Paul W Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81 (396):945–960, 1986.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Devon Jarvis, Richard Klein, Benjamin Rosman, and Andrew M Saxe. On the specialization of neural modules. In *The Eleventh International Conference on Learning Representations*, 2023.

Connor Thomas Jerzak, Fredrik Daniel Johansson, and Adel Daoud. Image-based treatment effect heterogeneity. In *Conference on Causal Learning and Reasoning*, pages 531–552. PMLR, 2023.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International Conference on Machine Learning*, pages 3020–3029. PMLR, 2016.

Fredrik D Johansson, Uri Shalit, Nathan Kallus, and David Sontag. Generalization bounds and representation learning for estimation of potential outcomes and causal effects. *Journal of Machine Learning Research*, 23(166):1–50, 2022.

Jean Kaddour, Yuchen Zhu, Qi Liu, Matt J Kusner, and Ricardo Silva. Causal effect inference for structured treatments. *Advances in Neural Information Processing Systems*, 34:24841–24854, 2021.

Edward H Kennedy. Towards optimal doubly robust estimation of heterogeneous causal effects. *Electronic Journal of Statistics*, 17(2):3008–3049, 2023.

Seyedeh Baharan Khatami, Harsh Parikh, Haowei Chen, Sudeepa Roy, and Babak Salimi. Graph neural network based double machine learning estimator of network causal effects. *arXiv preprint arXiv:2403.11332*, 2024.

Ekaterina Kochmar, Dung Do Vu, Robert Belfer, Varun Gupta, Iulian Vlad Serban, and Joelle Pineau. Automated data-driven generation of personalized pedagogical interventions in intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, 32(2):323–349, 2022.

Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 302–313, 1997.

Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR, 2018.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Kathryn Blackmond Laskey. MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172(2-3):140–178, 2008.

Sanghack Lee and Vasant Honavar. On learning causal models from relational data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Samuel Lippl and Kim Stachenfeld. The impact of task structure, representational geometry, and learning mechanism on compositional generalization. In *ICLR 2024 Workshop on Representational Alignment*, 2024.

Marc Maier, Katerina Marazopoulou, David Arbour, and David Jensen. A sound and complete algorithm for learning causal models from relational data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 371–380, 2013.

Ryan Marcus and Olga Papaemmanouil. Plan-structured deep neural network models for query performance prediction. *Proceedings of the VLDB Endowment*, 12(11):1733–1746, 2019.

Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? *Advances in Neural Information Processing Systems*, 35:28747–28760, 2022.

Judea Pearl. *Causality*. Cambridge University Press, 2009.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32, 2019.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *arXiv preprint arXiv:2302.11529*, 2023.

Parikshit Ram, Tim Klinger, and Alexander G Gray. What makes models compositional? a theoretical view. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 4824–4832, 2024.

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.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688, 1974.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Babak Salimi, Harsh Parikh, Moe Kayali, Lise Getoor, Sudeepa Roy, and Dan Suciu. Causal relational learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 241–256, 2020.

Simon Schug, Seijin Kobayashi, Yassir Akram, Maciej Wolczyk, Alexandra Maria Proca, Johannes Von Oswald, Razvan Pascanu, Joao Sacramento, and Angelika Steger. Discovering modular solutions that generalize compositionally. In *The Twelfth International Conference on Learning Representations*, 2024.

Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pages 3076–3085. PMLR, 2017.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2016.

Claudia Shi, Dhanya Sridhar, Vishal Misra, and David Blei. On the assumptions of synthetic control methods. In *International Conference on Artificial Intelligence and Statistics*, pages 7163–7175. PMLR, 2022.

Kensen Shi, Joey Hong, Yinlin Deng, Pengcheng Yin, Manzil Zaheer, and Charles Sutton. Exedec: Execution decomposition for compositional generalization in neural program synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oTRwljRgiv.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136, 2011.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903, 2005.

Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, pages 6401–6409. PMLR, 2019.

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

Eli N Weinstein and David M Blei. Hierarchical causal models. *arXiv preprint arXiv:2401.05330*, 2024.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles. *Advances in Neural Information Processing Systems*, 36, 2024.

Sam Witty and David Jensen. Causal graphs vs. causal programs: The case of conditional branching. In *First Conference on Probabilistic Programming (ProbProg)*, 2018.

## Appendix A. Other examples of structured systems with compositional data

The causal questions of interest in the compositional domain are: How do the unit-level interventions impact the component-level outcomes to produce the overall unit's outcome? Many real-world phenomena require answering such causal questions about the effect of shared interventions on different components. We provide several real-world use cases where the compositional approach can be useful to reason about the effects of the interventions and make informed and personalized decisions.

- *Compiler optimization:* How do different hardware architectures *(intervention)* affect the compile time *(potential outcome)* of different source codes *(unit)* ? In this case, source code consists of multiple program modules; hardware architecture is the unit-level intervention that can affect the compiling of different source codes differently, and compile time is the outcome of interest.

- *Energy efficiency optimization:* How does a state-wide mandate *(intervention)* of shifting to more efficient electric appliances affect the monthly bill *(potential outcome)* of each building *(unit)* in the state? Each building can be assumed to consist of various compositions of electric appliances. The intervention might affect the bill of each kind of appliance differently, affecting the overall utility bill.

- *Supply chain optimization:* How is the processing time *(potential outcome)* of an order *(unit)* affected when a supply chain company shifts to a different supplier for various parts *(intervention)*? In this case, each order execution plan is the unit of analysis that consists of routing the information from different parties, suppliers, manufacturers, and distributors specific to each order; intervention can impact the processing time of different parties depending on the affected parts and order details.

- *Causal reasoning in multi-agent systems:* How do the model hyperparameters (architecture, agent implementation types) affect the accuracy (potential outcomes) of multi-agent systems (LLM agents) for different task instances (unit)? As multi-agent LLM-based systems are becoming an integral part of daily workflows, developing instance-specific causal reasoning models for such systems using compositional approaches is helpful.

## Appendix B. Related Work

In this section, we discuss the connections of the compositional approach with the existing work in causal inference and associational machine learning in greater detail.

**Causal effect estimation in structured domains:** In causal inference, a relatively sparse body of work has focused on treatment effect estimation on structured data in modular domains (Gelman and Hill, 2006; Salimi et al., 2020; Kaddour et al., 2021). For example, existing work in multi-level modeling and hierarchical causal models (Gelman and Hill, 2006; Witty and Jensen, 2018; Weinstein and Blei, 2024) leverages hierarchical data structure to improve effect estimation under unobserved confounders. There is also growing interest in heterogeneous effect estimation for complex data, such as images (Jerzak et al., 2023), structured treatments (e.g., graphs, images, text, drugs) (Harada and Kashima, 2021; Kaddour et al., 2021), and relational data (Salimi et al., 2020; Khatami et al., 2024). The compositional approach complements this line of research by focusing on the units

composed of multiple heterogeneous components. On the other hand, hierarchical causal models focus on units with hierarchical structures but homogeneous sub-units in a unit. Relational causal models primarily employ relational semantics to describe instance-specific structures and interactions among entities. The compositional approach uses simpler compositional semantics and focuses on the generic component-wise behavior of a system and interactions among them, where components can be objects, processes, and distinct heterogeneous modules in a system. Our focus also lies in the structured and compositional representation of the units rather than only treatments, which helps better estimate causal effects in the case of high-dimensional observational data. Another piece of related work is the fine-grained analysis of the potential outcomes to study the validity of synthetic control methods with panel data (Shi et al., 2022). This work focuses on reasoning about potential outcomes of homogeneous sub-units (individuals) to establish identifiability of unit-level (state) potential outcomes. The compositional approach employs similar fine-grained reasoning, but explicitly uses the fine-grained data for compositional causal modeling.

**Modularity and compositionality in SCMs:** The vast body of work under the structural causal model (SCM) framework (Pearl, 2009) typically summarizes a system's behavior with a fixed set of variables and assumes fixed causal structure among those variables (unless modified under explicit intervention via the do-operator). The causal model, a directed acyclic graph, represents the causal interactions among all variables. In the compositional approach, we assume an instance-specific structure among the components for a given unit. Thus, the compositional data-generating process could not be represented by a single SCM. Instead, a unique SCM corresponding to each composition structure would be required. This is beyond the scope of nearly all current work in SCMs, with a very few exceptions (e.g., (Laskey, 2008)).

In the specific context of structural causal models, the term *modularity* is sometimes used to refer to a model property in which the structural function of a given variable can be intervened upon without influencing the structural function of any other variable. This property is also known as *autonomy* (Haavelmo, 1944), *structural invariance* (Aldrich, 1989), and *independence of causal mechanism* (Peters et al., 2017). Note that modularity, in this sense, is a property of the model— It is true by definition (variables in an SCM are assumed to be modular), and it is absolute. In contrast, the modular structure that we reference in this paper is a property of both the system being modeled and (perhaps) the structure of a given model of that system. To enable compositional generalization, the compositional approach assumes that the data-generating distribution of outcomes of a component given component-specific inputs remains stable and invariant across the units *irrespective* of where the component appears in the structure of a unit. This assumption differs from the modularity assumption made in SCM, which assumes the stability for the conditional distribution of *each* random variable given its parents in the direct graph with respect to the interventions on the other conditional distributions.

**Compositional models in associational machine learning:** Our work is inspired by research on compositional models in machine learning that exploit the structure of underlying domains and explicitly represent it in the model structure (Heckerman and Wellman, 1995; Koller and Pfeffer, 1997; Friedman et al., 1999; Getoor and Taskar, 2007; Taskar et al., 2005; Laskey, 2008). For example, research in object-oriented and relational models has produced directed graphical models that explicitly reproduce known modular structure in the systems being analyzed (Koller and Pfeffer, 1997; Friedman et al., 1999; Laskey, 2008). In a similar way, modular architectures in deep neural networks have been designed to replicate the assumed modular structure of the systems that they attempt to model (Jacobs et al., 1991). Some work in probabilistic programming has a similar

flavor, in that the structure of the probabilistic program reflects known modular structure in the real-world system being analyzed (Lake et al., 2015). The *instance-specific* modular architectures are widely used in machine learning to model data in natural language, program synthesis, reinforcement learning, and combined vision and language problems, providing sample efficiency, systematic generalization, and computation benefits (Andreas et al., 2016; Shazeer et al., 2016; Pfeiffer et al., 2023). The closest work to the compositional models for causal effect estimation is using a mixture of expert (MoE) architecture (Jacobs et al., 1991; Shazeer et al., 2016) and modular neural networks (Socher et al., 2011; Andreas et al., 2016; Marcus and Papaemmanouil, 2019) in vision and language domains. However, most of the work in machine learning focuses on understanding the systematic generalization and sample efficiency benefits of compositional models for prediction tasks. At the same time, their role in reasoning about intervention effects is unexplored (Lake and Baroni, 2018; Hupkes et al., 2020; Mittal et al., 2022; Jarvis et al., 2023; Wiedemer et al., 2024; Schug et al., 2024; Lippl and Stachenfeld, 2024). We inspire the formalization of the compositional data-generating process using a systems perspective (e.g., query execution system) with explicit causal mechanisms and interventions. In contrast, vision/language domains often lack well-defined components and causal interactions due to the perceptual nature of the data. Additionally, compositionality is usually a property of the causal data-generating process, and viewing compositionality from the lens of causality, interventions, and components in the modular systems helps understand the compositional generalization characteristics of the models. This work takes the first step in that direction.

## Appendix C. Graphical representation of compositional data-generating process

In section 2, we use the potential outcomes (PO) framework (Rubin, 1974, 2005) to describe the compositional data generating process for causal inference because the PO framework simplifies the representation of the key elements of the compositional approach: (1) *small number of causal dependencies:* our focus is on estimating the causal effect of treatment (T) on component outcomes ($Y_j$) given component covariates ($\mathbf{X}_j$) without needing to explicitly model causal relationships among the covariates (and assuming that causal mechanisms of all random variables are modular), as required by structural causal models; (2) *component interactions:* we represent component interactions and composition structure through the interaction graph (Figure 1(a),(c)). While $G_i$ constrains the possible causal dependencies among random variables that could be represented as instance-specific causal graphs, $G_i$ also highlights the most important aspect of compositional models: the manner in which components interact to bring about system behavior. These elements are formalized through Equation 1.

Compositional data generating process described in Section 2 can also be represented using a plate-based notation, commonly used in the probabilistic graphical models to represent various model classes such as hierarchical causal models (Gelman and Hill, 2006; Witty and Jensen, 2018; Weinstein and Blei, 2024) and relational causal models (Maier et al., 2013; Lee and Honavar, 2016; Salimi et al., 2020; Ahsan et al., 2023). In Figure 5, we represent plate models for three different composition structures in which component-level variables lie inside an inner plate and the unit-level variables lie in the outer plate, with edges denoting the causal dependencies among component-specific covariates, potential outcomes, and unit-level treatment.

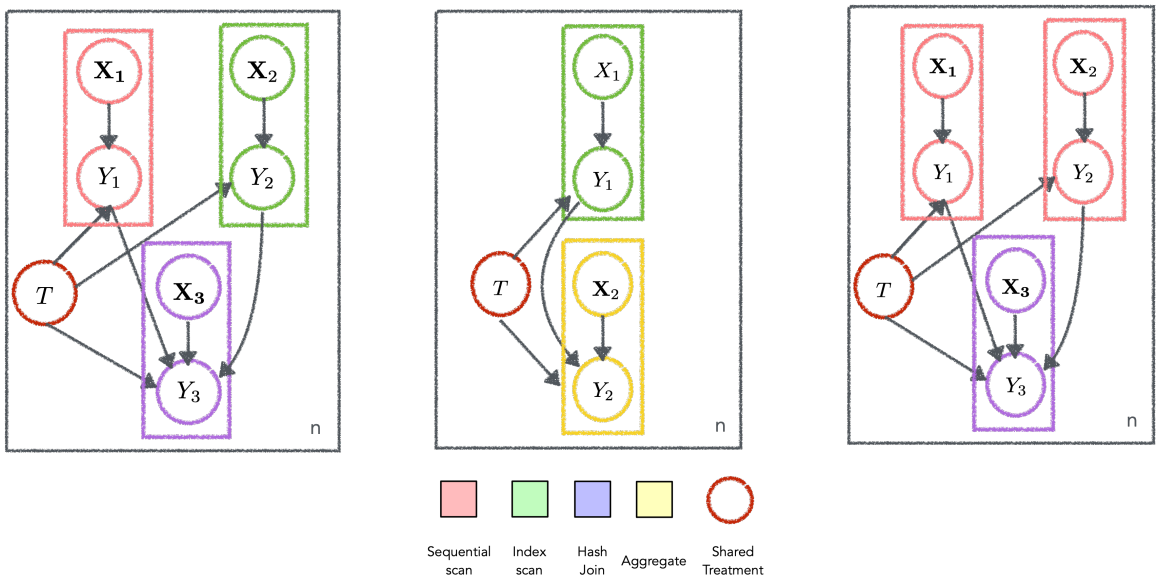



Figure 5: **Graphical representation of compositional causal models:** Each plate model represents the data-generating process of unit-level and component-level variables for a given instance-specific composition structure ($G_i$), shown for three different structures here. $X_j$ denotes the component-specific covariates, $Y_j$ denotes the component-specific outcomes, and $T$ denotes the unit-level shared treatment. Each distinct color represents the fixed data generating process for a specific component, that might appear in multiple units.

## Appendix D. Identifiability Results

### D.1. Definition, assumptions and auxiliary lemmas

We first define the necessary distributions and provide some simple results. Assume that each unit $i$ has pre-treatment covariates $\mathbf{X}_i = \mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, a binary treatment $T_i \in \{0, 1\}$, and two potential outcomes $\{Y_i(0), Y_i(1)\} \in \mathcal{Y} \subset \mathbb{R}$ (Rubin, 1974, 2005). In the observational data, we only observe one of the potential outcomes for each unit, $Y_i = Y_i(T_i)$ known as the *observed* or *factual* outcome and missing outcomes $Y_{iCF} = Y_i(1 - T_i)$ are known as *observed* or *counterfactual* outcome.

**Definition 3** *The conditional average treatment effect (CATE) is defined as*

$$\tau(x) : \mathbb{E}[Y_i(1) - Y_i(0)|\mathbf{X}_i = \mathbf{x}]$$

We first show that under the assumptions of ignorability and consistency, the CATE function $\tau(x)$ is identifiable by $\tau(\mathbf{x}) = \mathbb{E}[Y_i|\mathbf{X}_i = \mathbf{x}, T = 1] - \mathbb{E}[Y_i|\mathbf{X}_i = \mathbf{x}, T = 0]$ (Pearl, 2009) (Rosenbaum and Rubin, 1983). We assume a joint distribution function $p(\mathbf{X}_i, T_i, Y_i(1), Y_i(0))$, such that $(Y_i(1), Y_i(0) \perp T_i)|\mathbf{X}_i$ and $0 < P(T = 1|\mathbf{X}_i = \mathbf{x}) < 1$, for all $x$. We also assume consistency; that is, we assume that we observe $y_i = Y_i(1)|T_i = 1$ and $y_i = Y_i(0)|T_i = 0$.

**Lemma 4**

$$\begin{aligned} &\mathbb{E}[Y_i(1) - Y_i(0)|\mathbf{X}_i = \mathbf{x}] \\ &= \mathbb{E}[Y_i(1)|\mathbf{X}_i = \mathbf{x}] - \mathbb{E}[Y_i(0)|\mathbf{X}_i = \mathbf{x}] \end{aligned} \tag{3}$$

$$= \mathbb{E}[Y_i(1)|\mathbf{X}_i = \mathbf{x}, T_i = 1] - \mathbb{E}[Y_(0)|\mathbf{X}_i = \mathbf{x}, T_i = 0] \tag{4}$$

$$= \mathbb{E}[Y_i|\mathbf{X}_i = \mathbf{x}, T_i = 1] - \mathbb{E}[Y_i|\mathbf{X}_i = \mathbf{x}, T_i = 0] \tag{5}$$

Equality (3) is due to the linearity of the expectation. Equality (4) follows from the ignorability assumption in which we assume that $Y(T)$ is independent of $T$ conditioned on $\mathbf{X}$. Equality (5) follows from the consistency assumption. The last equation consists of only observable quantities and can be estimated from the data if we assume overlap, $0 < P(T_i = 1|\mathbf{X}_i = \mathbf{x}) < 1$, for all $\mathbf{x}$.

**Definition 5** *The conditional-average treatment effect (CATE) estimand for structured input $Q_i = q$ is defined as:* $\tau(q) = \mathbb{E}[Y_i(1) - Y_i(0)|Q_i = q] = \mathbb{E}[Y_i(1) - Y_i(0)|Q_i = (G_i, \{\mathbf{x}_{ij}\}_{j=1:m_i})]$

For ease of notation to describe conditional distributions, we denote the combined inputs to a component $j$ as $\mathbf{Z}_j(t) = (\mathbf{X}_j, \{Y_l(t)\}_{l \in Pa(c_j)})$.

**Definition 6** *The conditional-average treatment effect (CATE) estimand for component $j$ with $\mathbf{X}_j = \mathbf{x}_j \in \mathbb{R}^{d_j}$ is defined as:* $\tau(\mathbf{z}_j) = \mathbb{E}[Y_i(1) - Y_i(0)|\mathbf{Z}_j = \mathbf{z}_j]$.

### D.2. Identifiability for hierarchical composition

We define the component-wise distributions as $P(Y(t)|\mathbf{Z}_j(t) = \mathbf{z}_j)$. In hierarchical composition, the unit outcome equals the final component outcome: $\mathbb{E}[Y_i(t)|Q_i = q] = \mathbb{E}[Y_{im_i}(t)|Q_i = q]$. This conditional expectation can be expressed by marginalizing intermediate component outcomes using the Markov assumption (Equation 1).

$$\mathbb{E}[Y_i(t)|Q_i = q] = \int_{Y_{im_i-1}(t)} \int_{Y_{im_i-2}(t)} \cdots \int_{Y_{i1}(t)} \mathbb{E}[Y_{im_i}(t)|\mathbf{Z}_{im_i}(t)] \prod_{j=1}^{m_i-1} P(Y_{ij}(t)|\mathbf{Z}_{ij}(t)) \tag{6}$$

We use the following nested expectation expression as shorthand for marginalization over intermediate component outcomes:

$$\mathbb{E}[Y_i(t)|Q_i = q] = \mathbb{E}_{Y_{i1:im_i-1}(t)}[\mathbb{E}[Y_{im_i}(t)|\mathbf{Z}_{im_i}(t)]] \tag{7}$$

**Assumption 1 (Component-level ignorability)** *The component level potential outcomes and assigned treatment are independent conditioned on the component level covariates, i.e., $Y_j(1), Y_j(0) \perp T|\mathbf{X}_j$.*

**Assumption 2 (Component-level overlap)** *The overlap holds for the component level covariates $X_j = x_j$, i.e., $\forall x_j \in \mathcal{X}_j, t \in \{0, 1\} : \ 0 < p(T = t|\mathbf{X}_j = \mathbf{X}_j) < 1$*

**Assumption 3 (Component-level consistency)** *The consistency holds for the component level covariates, i.e., $y_j = Y_j(0)|t = 0$ and $y_j = Y_j(1)|t = 1$.*

As the treatment is assigned before any component's potential outcomes are observed, we can assume that the component's covariates $\mathbf{X}_j$ are sufficient to satisfy ignorability assumptions, i.e., $Y_j(t) \perp T|X_j$ Consider the conditional distribution $P(Y_j(t)|\mathbf{Z}_j(t))$. Assuming component-level ignorability for component $j$, we can write

$$P(Y_{ij}(t)|\mathbf{Z}_{ij}(t)) = P(Y_{ij}(t)|\mathbf{X}_j, \{Y_l(t)\}_{l\in Pa(c_j)})$$
$$= P(Y_{ij}|\mathbf{X}_j, \{Y_l(t)\}_{l\in Pa(c_j)}, T=t)$$

Now, assuming consistency for components $c_l$, where $l \in Pa(c_j)$, we can write $Y_l(t)|T=t = Y_l$.

$$P(Y_{ij}(t)|\mathbf{Z}_{ij}(t)) = P(Y_{ij}|\mathbf{X}_j, \{Y_l\}_{l\in Pa(c_j)}, T=t) = P(Y_{ij}|\mathbf{Z}_{ij}, T=t)$$

Similarly, $\mathbb{E}[Y_{im_i}(t)|\mathbf{Z}_{im_i}(t)]$ can be written as $\mathbb{E}[Y_{im_i}|\mathbf{Z}_{im_i}, T=t]$, assuming component-level ignorability and consistency. Substituting these quantities in Equation 6, we get the below result.

$$\tau(q) = \int_{Y_{im_i-1}}\int_{Y_{im_i-2}}\cdots\int_{Y_{i1}} \mathbb{E}[Y_{im_i}|\mathbf{Z}_{im_i}=\mathbf{z}_{im_i}, T=1]\prod_{j=1}^{im_i-1} P(Y_{ij}|\mathbf{Z}_{ij}=\mathbf{z}_{iz}, T=1) -$$

$$\int_{Y_{im_i-1}}\int_{Y_{im_i-2}}\cdots\int_{Y_{i1}} \mathbb{E}[Y_{im_i}|\mathbf{Z}_{im_i}=\mathbf{z}_{im_i}, T=0]\prod_{j=1}^{im_i-1} P(Y_{ij}|\mathbf{Z}_{ij}=\mathbf{z}_{iz}, T=0)$$

Using shorthand notation, we get the desired result.

$$\tau(q) = \mathbb{E}_{Y_{1:m_i-1}}[\mathbb{E}[Y_{m_i}|\mathbf{Z}_{m_i}=\mathbf{z}_{m_i}, T=1]] - \mathbb{E}_{Y_{1:m_i-1}}[\mathbb{E}[Y_{m_i}|\mathbf{Z}_{m_i}=\mathbf{z}_{m_i}, T=0]]$$

Component-level overlap assumption ensures that the estimand is identified using observational data.

### D.3. Identifiability for additive parallel composition

In this section, we first describe the assumptions that the data-generating process follows for additive parallel composition. Then, we prove the identifiability results for additive parallel composition.

**Assumption 4** ***Additivity*** *assumes that the ground-truth component-level potential outcomes add up to generate the ground-truth unit-level potential outcome, i.e.,* $Y_i(1) = \sum_{j=1}^{m_i} Y_{ij}(1)$, $Y_i(0) = \sum_{j=1}^{m_i} Y_{ij}(0)$.

**Assumption 5** ***Conditional independence*** *assumption among potential outcomes implies that the ground-truth potential outcomes of a component $j$ are conditionally independent of outcomes of other components $l$ ($l \neq j$) given the component's covariates $\mathbf{X}_j$:* $Y_j(T) \perp Y_l(T)|\mathbf{X}_j$

Assuming conditional independence assumption and Markov assumption, the data-generating process for additive parallel composition can be written as: $Y_{ij}(t) = \mu_{ot}(\mathbf{X}_{ij}) + \epsilon_{io}(t)$

Assuming additivity assumption 4, we get

$$\tau(q) = \mathbb{E}[\sum_{j=1}^{m_i} Y_{ij}(1) - \sum_{j=1}^{m_i} Y_{ij}(0)|Q_i = (G_i, \{\mathbf{x}_{ij}\}_{j=1:m_i})]$$

Due to the linearity of the expectation, we get the following:

$$\tau(q) = \mathbb{E}[\sum_{j}^{m_i} Y_{ij}(1)|Q_i = (G_i, \{\mathbf{x}_{ij}\}_{j=1:m_i})] - \mathbb{E}[\sum_{j=1}^{m_i} Y_{ij}(0)|Q_i = (G_i, \{\mathbf{x}_{ij}\}_{j=1:m_i})]$$

Assuming conditional independence assumption among the component-level potential outcomes and Markov assumption, we get

$$\tau(q) = \sum_{j=1}^{m_i} \mathbb{E}[Y_{ij}(1)|\mathbf{x}_{ij}] - \mathbb{E}[Y_{ij}(0)|\mathbf{x}_{ij}] = \sum_{j=1}^{m_i} \mathbb{E}[Y_{ij}(1) - Y_{ij}(0)|\mathbf{x}_{ij}] = \sum_{j=1}^{m_i} \tau(\mathbf{x}_{ij})$$

Now, we prove the identifiability result for additive parallel composition.

$$\tau(q) = \sum_{j=1}^{m_i} \mathbb{E}[Y_{ij}(1)|\mathbf{x}_{ij}] - \mathbb{E}[Y_{ij}(0)|\mathbf{x}_{ij}]$$

Assuming component-level ignorability, we get

$$\tau(q) = \sum_{j=1}^{m_i} \mathbb{E}[Y_{ij}(1)|\mathbf{x}_{ij}, T=1] - \mathbb{E}[Y_{ij}(0)|\mathbf{x}_{ij}, T=0]$$

Assuming component-level consistency, we get

$$\tau(q) = \sum_{j}^{m_i} \mathbb{E}[Y_{ij}|\mathbf{x}_{ij}, T=1] - \mathbb{E}[Y_{ij}|\mathbf{x}_{ij}, T=0]$$

Component-level overlap assumption ensures the estimand is identified using observational data.

## Appendix E. Learning compositional models from observational data

In this section, we discuss the algorithm for the additive parallel composition model discussed in Section 2.4.

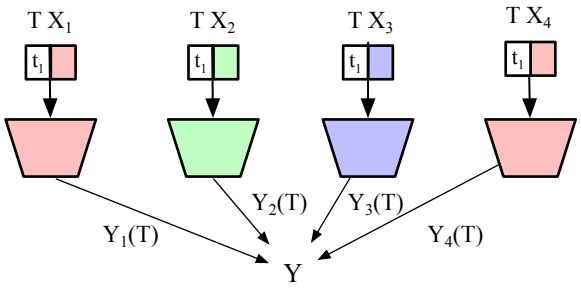



Figure 6: Model architecture for parallel composition model

The key idea is that the component-level models for effect estimation are instantiated specific to each unit, and outcomes of one component are not shared with other components as we assume conditional independence among the potential outcomes given component-level features and shared treatment. In addition, linearity of expectation applies to the additive composition model, so we can directly compute the CATE estimates by just estimating the expected component-level potential outcomes. See Figure 6 for model architecture for parallel composition.

**Model Training:** The component models for estimating component-level potential outcomes are denoted by $\hat{f}_{\theta_o} : \mathbb{R}^{d_o} \times \{0, 1\} \to \mathbb{R}$. Each model corresponding to component class $o \in \{1, 2, \ldots k\}$

is parameterized by separate and independent parameters $\theta_o$. For a given observational data set with $n$ samples, $\mathcal{D}_F = \{q_i, t_i, y_i\}_{i=1:n}$, we assume that we observe component-level features $\{\mathbf{x}_{ij}\}_{j=1:m_i}$, assigned treatment $t_i$ and fine-grained component-level potential outcomes $\{y_{ij}\}_{j=1:m_i}$ along with unit-level potential outcomes $y_i$. If component instance $c_j \in M_o$, training of each component model $o$ involves the independent learning of the parameters by minimizing empirical risk: $\theta_o^* := \arg\min_{\theta_o} \frac{1}{N_o} \sum_{m=1}^{N_o} (\hat{f}_{\theta_o}(\mathbf{x}_m, t_m) - y_m)^2$, where $N_o$ denotes the total number of component instances of component class $o$ across all the $N$ samples, and $m$ denotes the index of the component instance belonging to class $o$. To clarify, $t_m$ denotes the assigned treatment $T_i$ for the unit $i$ from which component instance $m$ data sample is obtained, and $y_m$ is the corresponding component-level outcome. The potential outcome of each component is computed using input features of that component, shared unit-level treatment, and *observed* potential outcomes of the parent's component.

**Model Inference:** To estimate CATE for a unit $i$, a modular architecture consisting of $m_i$ component models is instantiated with the same number of components as in the unit $i$. During inference for treatment $T = t$, due to conditional independence assumption, $\hat{y}_{ijt} = \hat{f}_{\theta_o^*}(\mathbf{x}_{ij}, t)$. The estimate of CATE is obtained by taking the sum of the potential outcome estimates of all component instances $\hat{\tau}(q) = \sum_{j=1, j\in M_o}^{m_i} \hat{f}_{\theta_o^*}(\mathbf{x}_{ij}, 1) - \hat{f}_{\theta_o^*}(\mathbf{x}_{ij}, 0)$.

### E.1. Relaxing assumptions about component-level data access for additive parallel composition

The model description above assumes observed component-level covariates and outcomes. This assumption is often reasonable, given the wide availability of fine-grained data for many structured domains. However, other cases exist when only the unit-level covariates $\mathbf{X}$ and outcomes are observed, and the component-level covariates $\mathbf{X}_j$ and outcomes $Y_j$ are unobserved. Below, we discuss hierarchical composition models for these cases.

**Case 1: *Unobserved* $\mathbf{X}_j$, *observed* $Y_j$:** We jointly learn the lower-dimensional component-level representations $\phi_o : \mathbb{R}^d \times \mathbb{R}^{d'_o}$, as well as the parameters of outcome functions. If we assume component instance $c_j \in M_o$, then $\hat{f}_{\theta_o}$: $\theta_o := \arg\min_{\theta} \frac{1}{N_o} \sum_{i=1}^{N_o} (\hat{f}_{\theta_o}(\phi_o(\mathbf{x}_i), t_i) - y_{ij})^2$. $\phi_o$ is jointly trained with the parameters $\theta_o$, so that the relevant variables for predicting $y_{ij}$ are selected.

**Case 2: *Observed* $\mathbf{X}_j$, *unobserved* $Y_j$:** The model architecture remains the same as before, but we do not have individual component-level loss functions and only know the loss function for unit-level outcomes. Due to this, the parameters of the components are jointly learned to optimize the loss of estimating unit-level outcomes. Due to additive composition, the joint loss function is given by: $[\theta_1, \theta_2 \dots \theta_k] := \arg\min_{\mathbf{\Theta}} \frac{1}{N} \sum_{i=1}^{N} (\hat{f}_{\theta_{m_i}} + \hat{f}_{\theta_{m_i-1}} + \cdots + \hat{f}_{\theta_1}(\mathbf{x}_{i1}, t) - y_i)^2$

**Case 3: *Unobserved* $\mathbf{X}_j$ and *unobserved* $Y_j$:** In this case, we only assume the knowledge of $G_i$. In this case, the model is equivalent to a mixture of experts (MoE) (Jacobs et al., 1991) architecture with addition as gating function. $[\theta_1, \theta_2 \dots \theta_k] := \arg\min_{\mathbf{\Theta}} \frac{1}{N} \sum_{i=1}^{N} (\hat{f}_{\theta_{m_i}} + \hat{f}_{\theta_{m_i-1}} + \cdots + \hat{f}_{\theta_1}(\phi_1(\mathbf{x}_i), t) - y_i)^2$

## Appendix F. Experimental Infrastructure

### F.1. Data sets

In this section, we describe the details of the two benchmarks based on real-world computational systems — query execution, matrix operations and one benchmark based on a realistic simulation — manufacturing plant data.

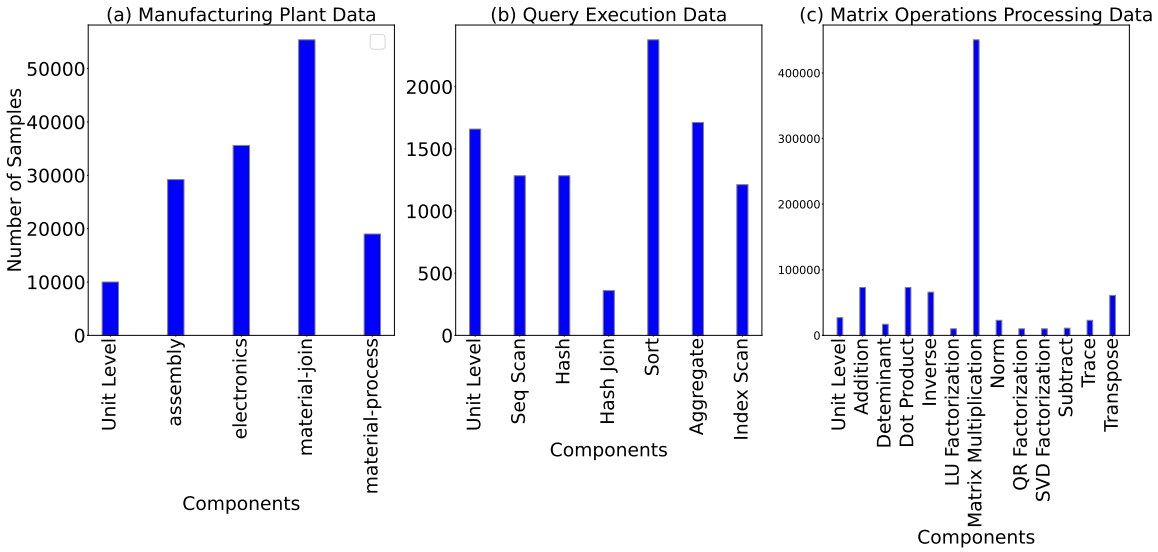



Figure 7: **Number of samples for units and component instances for different domains:** (a) Manufacturing data set showing component re-use across layouts ($10,000$ units). (b) Query execution data set ($1,500$ units). (c) Matrix operations processing data set ($25,000$ units)

### F.1.1. QUERY EXECUTION SYSTEM

We first collect 10000 most popular user-defined Math Stack Overflow queries. We install a PostgreSQL 14 database server and load a 50 GB version of the publicly available Stack Overflow Database. We then run these queries with different combinations of the configuration parameters listed in Table 1. In all our experiments, our queries were executed with PostgreSQL 14 database on a single node with an Intel 2.3 GHz 8-Core Intel Core i9 processor, 32GB of RAM, and a solid-state drive. PostgreSQL was configured to use a maximum of 0 parallel workers to ensure non-parallelized executions so that additive assumption about operations is satisfied (`max_parallel_workers_per_gather = 0`). Before each run of the query, we begin from the cold cache by restarting the server to reduce caching effects among queries. Many database management systems provide information about the query plans as well as actual execution information through convenient APIs, such as `EXPLAIN ANALYZE` queries. Usually, Postgres reports the total run-time of each operation, along with children's operations. We mainly model the query plans with the following operations — Sequential Scan, Index Scan, Sort, Aggregate, Hash, Hash Join as the occurrence of these operations in collected query plans was good, providing a large number of samples to learn the models from data. For CATE estimation experiments, we select 1500 query plans in which effect sizes were significant and were actually a result of the intervention rather than random variation in the run-time due to the stochastic nature of the database execution system. Each SQL query is run 5 times, and the median execution time is taken as the outcome. We evaluate the combined treatment effect of increasing working memory size and adding indices on structured query execution plans (Table 1). Increasing working memory affects the run-time of sorting, hash join, and aggregation operations, as can be seen in Figure 8. Adding additional indices modifies the structure of the execution plan by switching from a sequential scan component to an index scan component, as discussed below.

| Treatment | Working Memory | Temp Buffers | Indices |
|---|---|---|---|
| T=0 | 64 KB | 800 KB | Primary key indexing |
| T=1 | 50 MB | 100 MB | Secondary key indexing |

Table 1: Treatment details for query execution data set

**Change in the structure of query execution plans as a result of interventions on configuration parameters:** For some interventions on the configuration parameters and for some queries, the query planner doesn't return the same query plan. It returns the query plan with a changed structure as well as modified features of the components. This makes sense as that is the goal of query optimizers to compare different plans as resources change and find the most efficient plan. For example, increasing the working memory often causes query planners to change the ordering of Sort and aggregate operations, changing the structure as well as inputs to each component. These interventions are different from standard interventions in causal inference in which we assume that the covariates of the unit remain the same (as they are assumed to be pre-treatment) and treatment only modifies the outcome. In this case, a few features of the query plan are modified as a result of the intervention (and thus are post-treatment), while other features remain the same. Prediction of which features would change is part of learning the behavior of the query planner under interventions. In this work, we have mostly focused on learning the behavior of the query execution engine and assumed that the query planner is accessible to us. For simplicity, we assume that we know of the change in structure as a result of the intervention for both models. We leave the learning of the behavior of query optimizers under interventions for future work. This case provides another challenge for the task of causal effect estimation, even in the case of randomized treatments (bias strength = 0); due to the modified features of the query plans, the distribution of features in control and treatment populations might differ, providing an inherent observational bias in the dataset coming from the query optimizer. As long as we provide the information about modified query plans for both models, we believe that our comparisons are fair. For changed query structure, CATE estimand can be thought of as conditional on the same query but two different query plans.

$$\tau(Q_i) = \mathbb{E}[Y_i(1) - Y_i(0)|Q_i]$$

$$\tau(Q_i) = \mathbb{E}[\mathbb{E}[Y_i(1)|Q_{p_i}(1)] - \mathbb{E}[Y_i(0)|Q_{p_i}(0)]]$$

**Covariates used for query execution data for model training:** See Table 2 for information about the high-dimensional covariates and component-specific covariates used for training models on query execution plans data set.

### F.1.2. MANUFACTURING PLANT DATA

We use a process-based discrete-event simulation framework, Simpy, to generate realistic manufacturing plant data. The plant aims to produce the final product by processing raw materials and assembling intermediate parts. The simulation comprises ***four*** distinct manufacturing processes: Material Processing, Material Joining, Electronics Processing, and Assembly combined and re-used across 50 manufacturing line layouts with varying hierarchical structures. Each scenario consists of

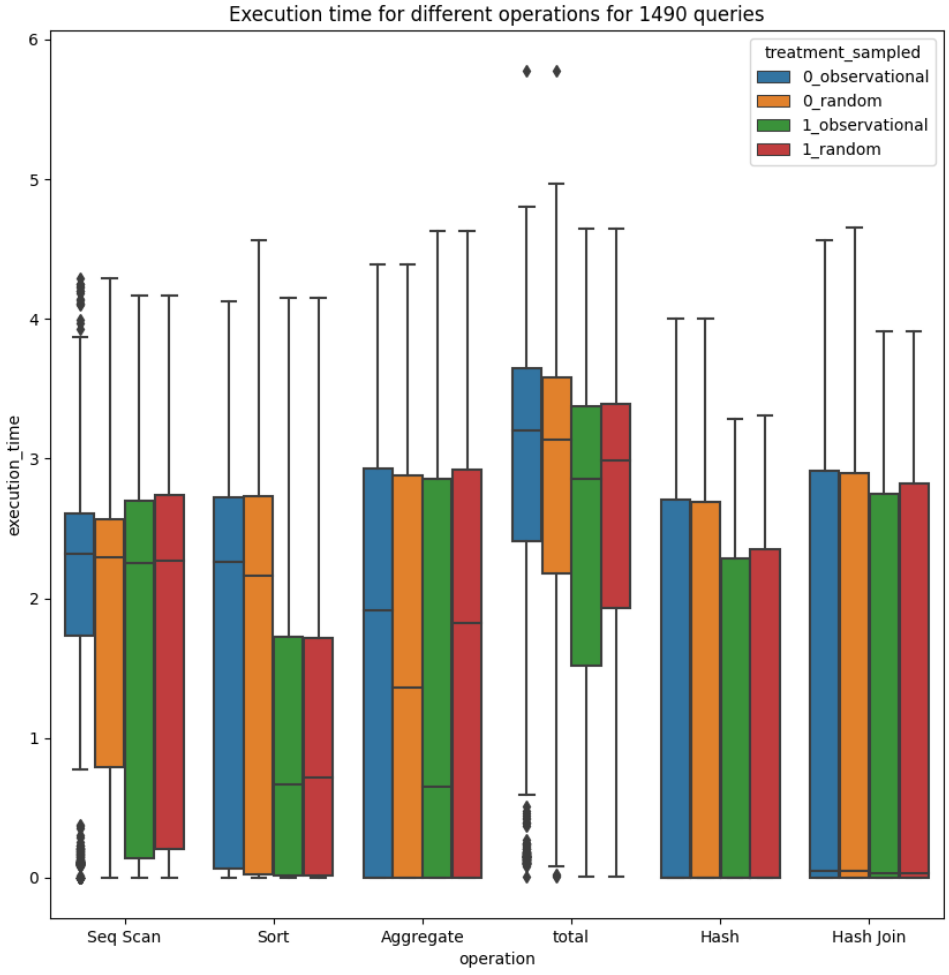



Figure 8: **Treatment effect on query executions for 1500 queries:** Ground-truth causal effect estimate of increasing memory for experimental data (random) and observational data created with bias strength = 1. 0 means low memory, and 1 means high memory. We can see that increasing memory has the most effect on sort and aggregate operation and the least effect on the sequential scan.

| Model | Component | Training features | Outcome |
|---|---|---|---|
| Unitary | | num_Sort, num_Hash_Join, num_Seq_Scan, num_Hash, num_Index_Scan, num_Aggregate, num_complex_ops, Sort_input_rows, Hash Join_input_rows, Hash Join_left_plan_rows, Hash Join_right_plan_rows, Seq Scan_input_rows, Hash_input_rows, Index Scan_input_rows, Aggregate_input_rows | total_time |
| Compositional | Sequential Scan | Seq_Scan_input_rows, Seq_Scan_plan_rows | seq_scan_time |
| Compositional | Index Scan | Index_Scan_input_rows, Index_Scan_plan_rows | index_scan_time |
| Compositional | Hash | Hash_input_rows, Hash_plan_rows | hash_time |
| Compositional | Hash Join | Hash_Join_left_input_rows, Hash_Join_right_input_rows, Hash_Join_plan_rows | hash_join_time |
| Compositional | Sort | Sort_input_rows, Sort_plan_rows | sort_time |
| Compositional | Aggregate | Aggregate_input_rows, Aggregate_plan_rows | aggregate_time |

Table 2: Covariates used by unitary and compositional models for query execution plans data set

simulations with product demand varying from 5 to 1000, with different raw material inventories (pre-treatment covariates) available for each demand. The intervention consists of the availability of multiple workers with two different skill levels – (1) 5 workers with a higher mean of skill distribution and (2) 15 workers with a lower mean of skill distribution (Figure 10). The goal is to estimate the effect of workers' (with different skill levels) availability on the number of parts produced. Figure 11 shows the effect of treatment on total output (total number of parts produced), total time (total processing time of each scenario), and output per time (total parts produced/time) distributions. This data satisfies causal Markov dependence among the potential outcomes as the quality of the part processed by the component is explained by the raw-material inputs, intervention, and the quality of the directly connected parent components. There are a total of 39 unit-level covariates and around 10 covariates per component.

A factory has the following features:

1. **Inventory of raw items** (Covariates): This includes parts and items which come into the factory to be processed into a final product. For this example, we use the following:
   (a) *Fastener:* This includes parts such as screws, nuts and bolts.
   (b) *Electronic component:* electronic components are parts used in the assembly of electronic assemblies such as printed circuit boards.
   (c) *Raw material:* These are parts such as metal blanks, plastic sheets, sheet-metal etc.

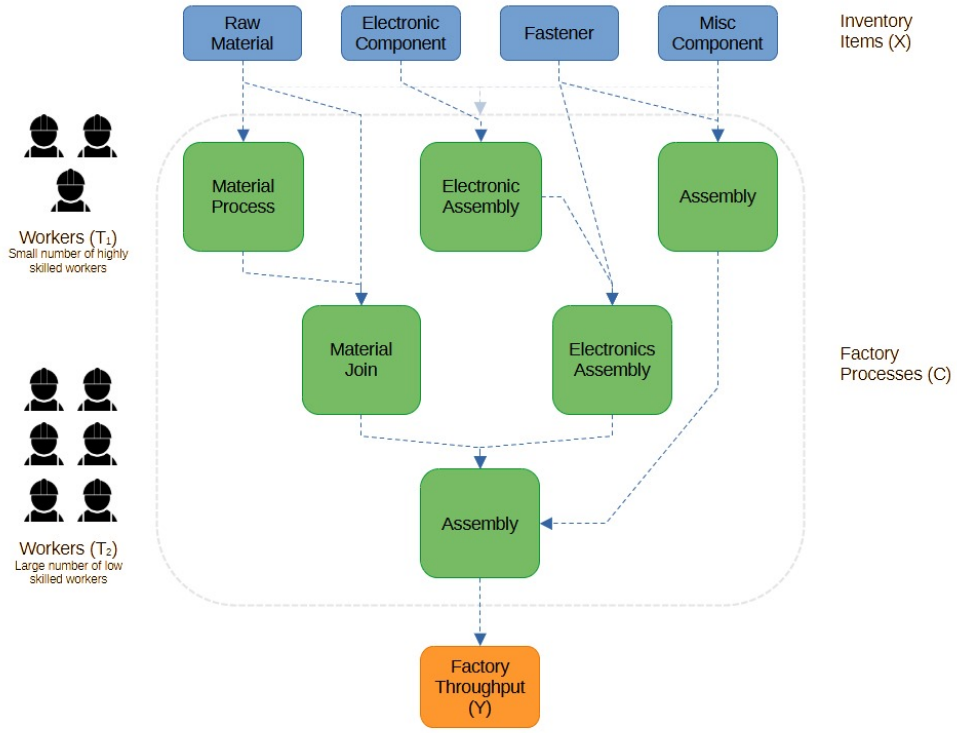



Figure 9: Illustrative figure explaining manufacturing assembly system

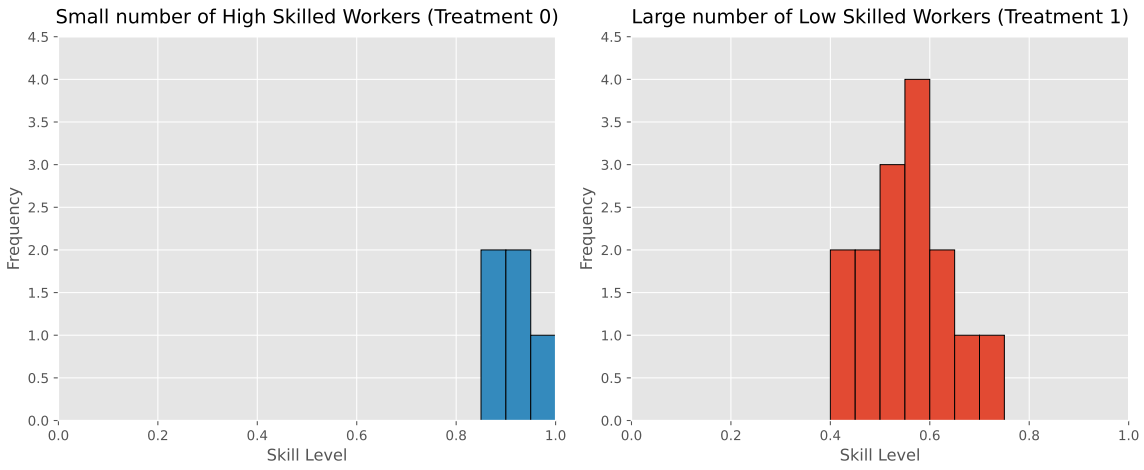



Figure 10: **Treatment for manufacturing plant data:** skill distribution for five (5) vs. fifteen (15) skilled workers.

(d) *Misc Component:* Parts such as belts, pulleys, gears etc.

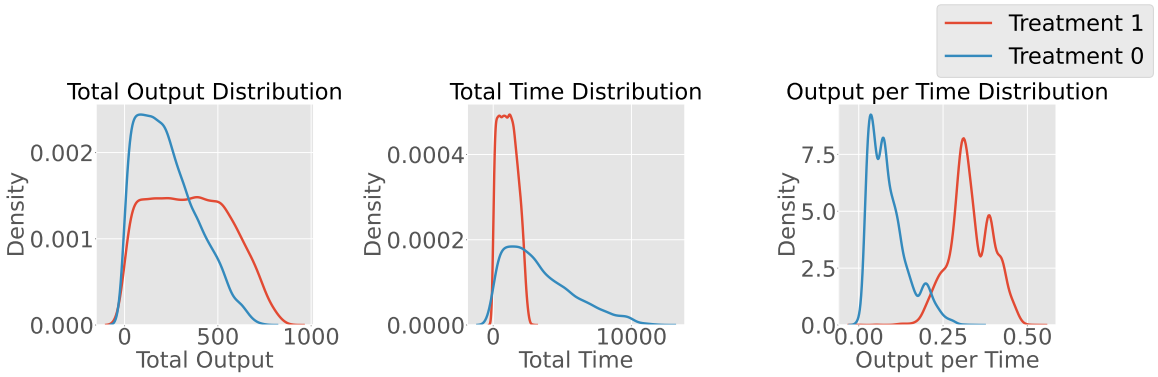



Figure 11: **Distribution of different potential outcomes for manufacturing data :** This figure shows the distribution of various potential outcomes for treatment and control groups for manufacturing plant data — (a) total output (total number of parts produced), (b) total time (total processing time of each scenario), and (c) output per time (total parts produced/time) distributions. We consider total output (total number of parts produced (a)) in our experiments.

2. **Process archetypes** (Components): These are processing stations which consume inventory of raw items and parts from other processes to produce a part. A process is defined by its processing time, i.e. how long does it ideally take to produce the part, and complexity, i.e how complex the given task is. Archetypes used in this simulation are:

   (a) *Material Processing:* takes 1 part, does some processing and outputs the processed part (eg: painting, coating, heat treatment etc.)

   (b) *Material Joining:* takes two parts and joins them together to output another part (eg: welding, riveting, fasteners to put two parts together)

   (c) *Electronics Processing:* takes multiple components and electronics to produce a electronic part (eg: soldering, PCB manufacturing)

   (d) *Assembly:* takes components from other stations and puts them together to form an assembly (eg: final product assembly)

3. **Workers** (Treatment): workers are a common resource pool of people working on a process. At a given time one process can have only one worker. A worker is defined by their skill which directly impacts how long a process will take over its base time and how much scrap (the final part is unusable and cannot proceed to the next step) and rework (redo aspects of the process increasing the amount of time the process takes) the process may produce.

#### F.1.3. MATRIX OPERATIONS PROCESSING

We generate a matrix operations data set by evaluating complex 25 complex matrix expressions on two computer hardware with different processors and RAM (treatment) and evaluate the execution time for each treatment (potential outcomes). The matrix size of matrices varies from 2 to 1000, resulting in total 25000 unit samples. The expressions contain 12 component operations, e.g., matrix multiplication, inverse, singular value decomposition, etc. We ensure each operation is executed individually, ensuring parallel composition with additive aggregation function. Matrix size is used

as a biasing covariate to create a distribution mismatch between treatment groups. Treatment 0 means operations are processed on computer hardware with an 8-core, 32GB of RAM, and treatment 1 means operations are processed on 1-core, 4 GB RAM. Figure 13 shows the potential outcome functions for each treatment for unit-level and component data. Figure 12 shows that the matrix multiplication operation is the dominant operation, taking the most of the execution time (50%) across all matrix expressions

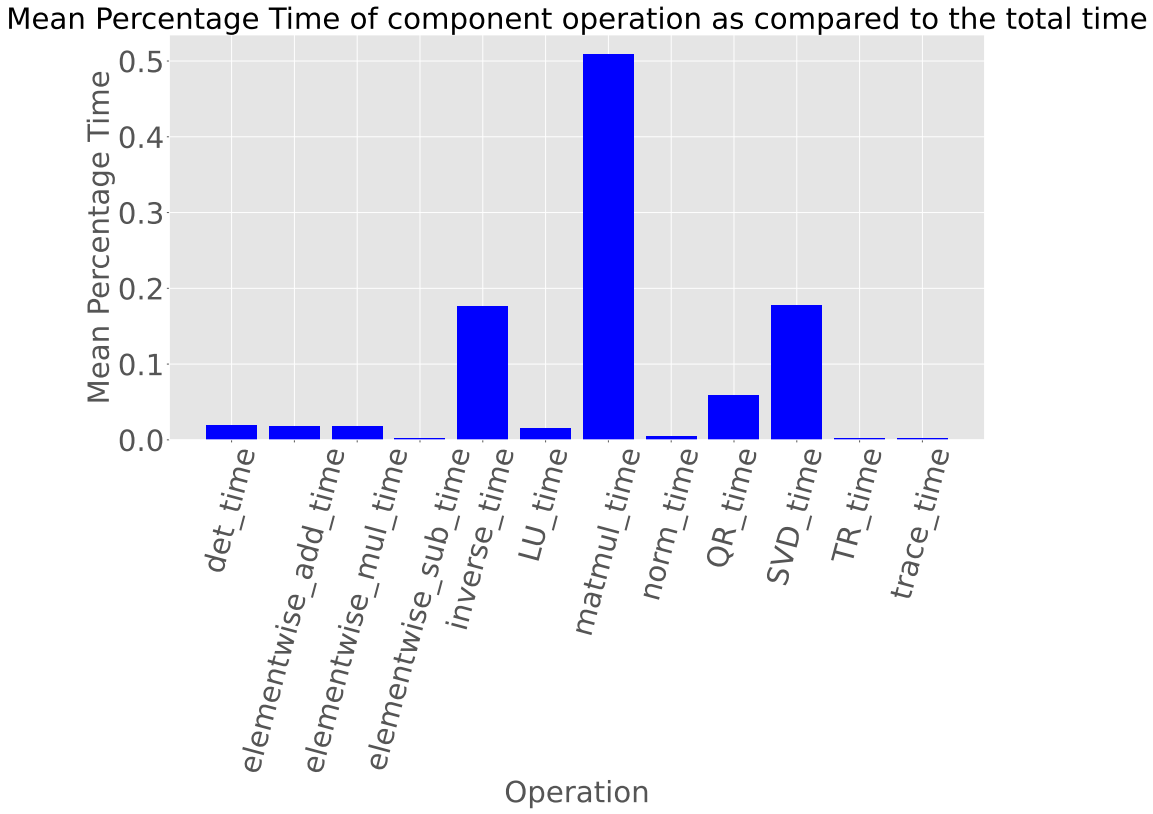



Figure 12: **Dominant contribution of component operation in total execution time:** This figure shows the dominant contribution of matrix multiplication's execution time in total execution time. Percentage contribution of each component is calculated for each unit. Mean is taken over 25000 unit instances corresponding to 25 expression structrues.

### F.1.4. SYNTHETIC DATA GENERATION:

We generate data sets with varying characteristics to test model performance for units with different structures and composition functions. Structured units are generated by sampling binary trees (max depth=10) with $k$=10 heterogeneous modules, each having $d_j$=1 feature ($d$=10 (covariates) + 10 (structural information) total). All components' total sum of features is used as a biasing covariate to create a distribution mismatch based on joint covariates distribution. For observational bias based on the structure, the depth of the trees is used as a proxy for structural information. The covariate distribution for each component is sampled from a multivariate Gaussian and uniform distribution with a mean ranging between 0 and 3 and covariance ranging between 0 and 3. The potential outcome

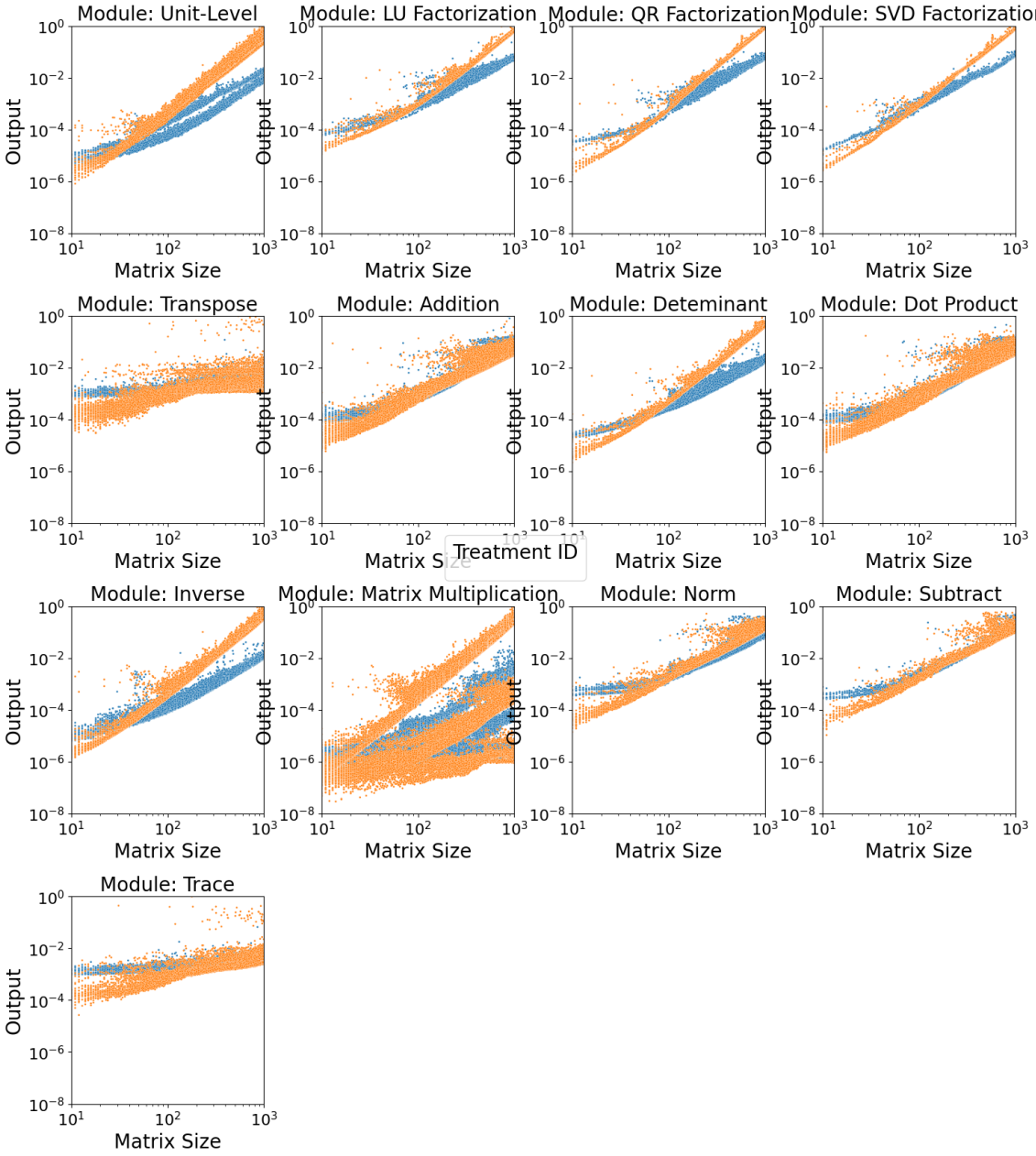



Figure 13: **Homogeneous functional forms of the component outcome function for matrix operations data set:** Ground-truth outcome functions for components for matrix operation data set. Blue and orange colors correspond to binary treatments 0 and 1. Intervention corresponds to two different compute hardware on which matrix expressions are processed.

for each treatment is a polynomial smooth function with different parameters for each treatment to generate heterogeneous treatment effects. For fixed structure data generation, the depth of the tree is fixed to 10 so that every unit has the same number and kind of components. For the variable structure setting, the depth of the tree randomly varies between 4 and 10, and components are sampled with

replacement. For compositional generalization, an equal number of trees are generated for every module combination from 2 to 10. For parallel composition, the potential outcome is simulated for each component for each treatment as a function of input features and treatment. For sequential and hierarchical composition, the potential outcome is a function of input features, treatment, and potential outcomes of the parent components.

### F.2. Models and Baseline implementation

1. *Hierarchical composition models:* Each distinct component in a structured unit is implemented as a separate MLP module with a hidden dimension size = $2 * input\_size$ and batch size = $64$ for each component. Models were trained using Adam optimizer with a learning rate of $0.01$ (and adaptive cosine learning rate schedule with a starting learning rate of $0.001$ for certain data sets). The total mean squared loss for all the component outcomes was optimized for the observed fine-grained outcomes of the hierarchical model. In contrast, loss for only unit-level potential outcomes was optimized for the unobserved fine-grained outcomes model. For the unobserved fine-grained outcomes model, the outcomes are passed hierarchically through the component interaction graph $G_i$ as illustrated in Figure 1(c).

2. *Additive parallel composition models:* We implement an additive parallel model using two model classes: random_forest and neural_network. The differences between hierarchical and additive parallel composition models are as follows: (1) In parallel composition, the outcomes of each component are computed independently and not shared hierarchically. Second, an additive aggregation function is assumed, i.e., the unit-level outcome is the sum of the component's outcomes. A three-layer, fully connected MLP architecture is used for neural network models with hidden layer dimension = 64 and ReLU activations. Models were trained using Adam optimizer with a learning rate of $0.01$. For unobserved component-level outcomes, a mixture of experts (MoE) architecture (Jacobs et al., 1991; Shazeer et al., 2016) is used where each expert receives the high-dimensional covariates $\mathbf{X} \in \mathbb{R}^d$, initialized with the same number of experts as a number of distinct modules in the domain. We use a simple addition of the experts' outcomes for the gating mechanism, as our data sets satisfy additive composition. For unobserved component-level covariates, entire unit-level covariates are passed to each component model to learn component-specific representations as part of learning component outcome functions.

**Baselines:** X-learner and non-parametric double machine learning implementation is from Econml library and random forests were used as the base models. TNet (Curth and Van der Schaar, 2021) implementation is taken from the GitHub repository catenets.

## Appendix G. Additional Results

## Appendix H. Algorithms

#### H.0.1. HIERARCHICAL COMPOSITION MODEL

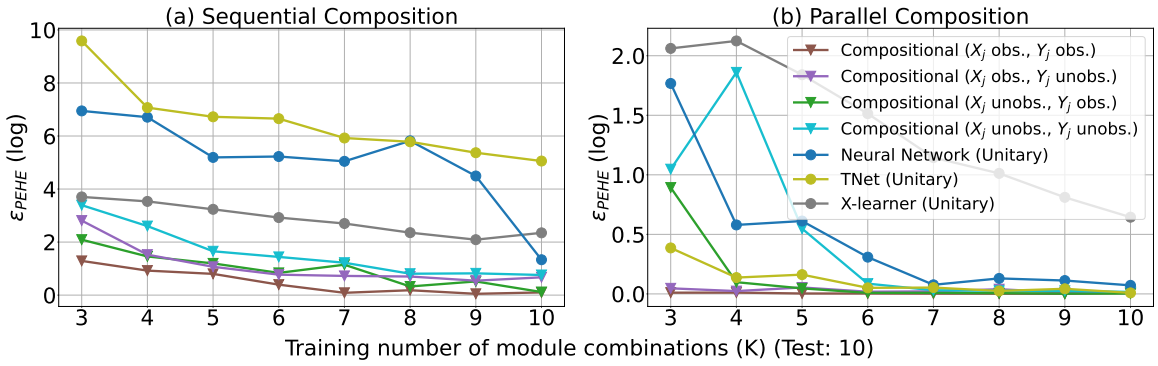



Figure 14: **Role of component-level data access and composition structure in the performance of compositional models:**PEHE error (in log) for models evaluated on compositional generalization task with varying degrees of component-level data access. (Lower is better). We observe that end-to-end trained models incorporating just modular structure compositionally generalize as trained on more module combinations. Unitary models show compositional generalization for additive parallel composition but perform comparably only for in-distribution combinations ($K$=10) for sequential composition, except X-learner. Note that the number of training samples increases as training depth increases.

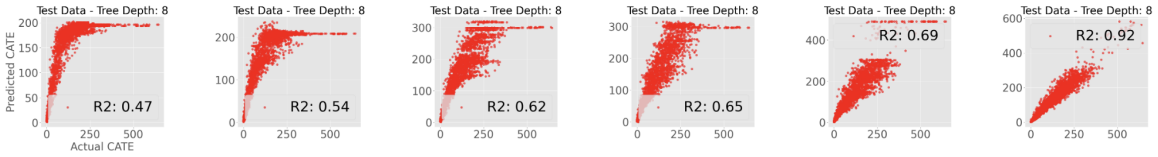



Figure 15: **Compositional generalization scatter plot for unitary model (X-learner):** This figure shows the scatter plot for test performance for the compositional generalization experiment. The training depths are varied from left to right $K$ = 3 to 8 (left to right)

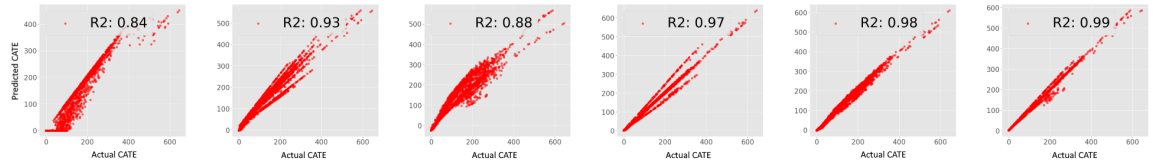



Figure 16: **Compositional generalization scatter plot for compositional model:** This figure shows the scatter plot for the predicted **test** performance of the compositional model on the test depth=8 when trained on increasing depths $K$ = 3 to 8 (left to right).

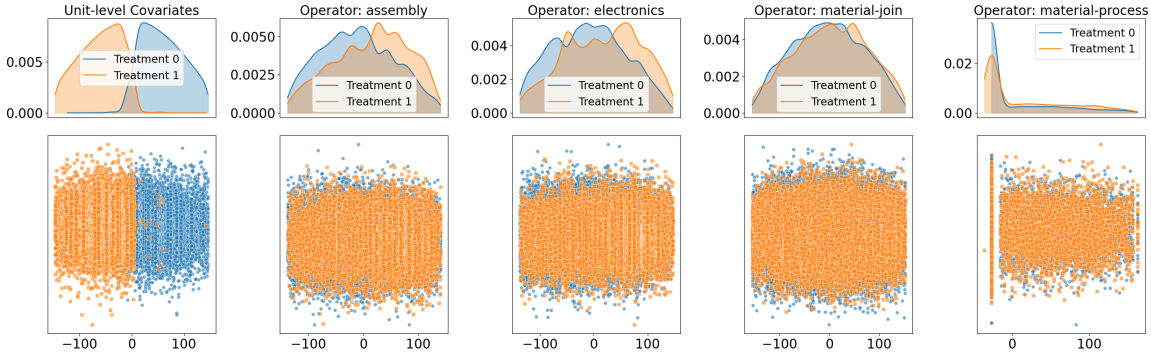



Figure 17: **Distribution mismatch due to structure-based treatment assignment for manufacturing data**. These density plots (top) and scatter plots (bottom) show the distribution mismatch between treatment and control groups at the unit level and component level. The plots are generated by first projecting the high-dimensional unit-level and component-level covariates to 1-dimensional using TSNE. The observational bias is created using tree_depth feature of the hierarchical structure.

**Algorithm 1** Hierarchical Composition Model: Training

**Input:** Factual data set: $\mathcal{D}_F = \{q_i : \{\mathbf{x}_{ij}\}_{j=1:m_i}, t_i, y_i, \{y_{ij}\}_{j=1:m_i}\}_{i=1:n}$, number of distinct components $k$.
**Result:** Learned potential outcome models for each component: $\{\hat{f}_{\theta_1}, \hat{f}_{\theta_2}, \hat{f}_{\theta_3} \dots \hat{f}_{\theta_k}\}$
**while** not converged **do**
  $loss\_1, loss\_2, loss\_3, loss\_k = 0$
  **for** $i = 1$ **to** $n$ **do**
    Get the order of the components in which input is processed by using post-order traversal of the tree $G_i$.
    $orderedList \leftarrow post\_order\_traversal(G_i)$
    **for** component $j$ in $orderedList$ **do**
      $o \leftarrow component\_class\_index(j)$
      // Potential outcome of a component depends on the potential outcome of the parent components according to graph $G_i$ (assuming binary tree)
      **if** component $j$ has parents in $G_i$ **then**
        $\hat{y}_{ij} = \hat{f}_{\theta_o}(\mathbf{x}_{ij}, \{y_{lt_i}\}_{l \in Pa(c_j)}, t_i)$
      **else if** component $l$ does not have parents **then**
        $\hat{y}_{ij} = \hat{f}_{\theta_o}(\mathbf{x}_{ij}, t_i)$
      **end if**
      $loss\_o = loss\_o + (\hat{y}_{ij} - y_{ij})^2$
    **end for**
  **end for**
  Calculate gradients for the parameters for each module
  **for** $o = 1$ **to** $k$ **do**
    $\delta_o \leftarrow \triangle_{\theta_o} \frac{1}{N_o} loss\_o$
    $\theta_o \leftarrow \theta_o - \alpha\delta_o$ independent training of all the component models.
  **end for**
  Check convergence criterion
**end while**

**Algorithm 2** Hierarchical Composition Model: Inference

**Input:** Test data set: $\mathcal{D}_{\mathcal{T}} = \{q_i : \{\mathbf{x}_{ij}\}_{j=1:m_i}\}_{i=1:n}$, learned potential outcome models for each component: $\{\hat{f}_{\theta_1}, \hat{f}_{\theta_2}, \hat{f}_{\theta_3} \dots \hat{f}_{\theta_k}\}$,
**Result:** CATESamples
**Procedure:**
$CATESamples \leftarrow \{\}$
**for** $i = 1$ **to** $n$ **do**
  Get the order of the operation in which input is processed by post-order traversal of the tree
  $orderedList \leftarrow post\_order\_traversal(G_i)$
  **for** component $j$ in $orderedList$ **do**
    $o \leftarrow component\_class\_index(j)$
    **if** component $j$ has parents in $G_i$ **then**
      $\hat{y}_{ijt_i} = \hat{f}_{\theta^*_o}(\mathbf{x}_{ij}, \{\hat{y}_{ilt_i}\}_{l \in Pa(c_j)}, t_i)$
    **else if** component $l$ does not have parents **then**
      $\hat{y}_{ijt_i} = \hat{f}_{\theta^*_o}(\mathbf{x}_{ij}, t_i)$
    **end if**
  **end for**
  $\hat{\tau}(q_i) = \hat{y}_{im_i}(1) - \hat{y}_{im_i}(0)$, get the difference between estimated potential outcomes of the last component in $G_i$
  $CATESamples \leftarrow CATESamples \cup \{(q_i, \hat{\tau}(q_i))\}$
**end for**

**Algorithm 3** Additive Parallel Composition Model: Training

**Input:** Factual data set: $\mathcal{D}_F = \{q_i : \{\mathbf{x}_{ij}\}_{j=1:m_i}, t_i, y_i, \{y_{ij}\}_{j=1:m_i}\}_{i=1:n}$, number of distinct components $k$.
**Result:** Learned potential outcome models for each component: $\{\hat{f}_{\theta_1}, \hat{f}_{\theta_2}, \hat{f}_{\theta_3} \dots \hat{f}_{\theta_k}\}$
**Procedure:**
$\mathcal{D}_1 \leftarrow \{\}, \mathcal{D}_2 \leftarrow \{\}, \mathcal{D}_3 \leftarrow \{\} \dots \mathcal{D}_k \leftarrow \{\}$
**for** $i = 1$ **to** $n$ **do**
  **for** $j = 1$ **to** $m_i$ **do**
    $o \leftarrow component\_class\_index(j)$ index of distinct component class for $j^{th}$ component instance.
    $\mathcal{D}_o \leftarrow \mathcal{D}_o \cup \{\mathbf{x}_{ij}, t_i, y_{ij}\}$
  **end for**
**end for**
**for** $o = 1$ **to** $k$ **do**
  $N_o \leftarrow len(\mathcal{D}_o)$
  $\theta_o := \arg\min_\theta \frac{1}{N_o} \sum_{m=1}^{N_o} (\hat{f}_{\theta_o}(\mathbf{x}_m, t_m) - y_m)^2$ independent training of all the component models.
**end for**

**Algorithm 4** Additive Parallel Composition Model: Inference

**Input:** Test data set: $\mathcal{D}_{\mathcal{T}} = \{q_i : \{\mathbf{x}_{ij}\}_{j=1:m_i}\}_{i=1:n}$ and potential outcome models for each component: $\{\hat{f}_{\theta_1}, \hat{f}_{\theta_2}, \hat{f}_{\theta_3} \dots \hat{f}_{\theta_k}\}$,
**Result:** CATESamples
**Procedure:**
$CATESamples \leftarrow \{\}$
**for** $i = 1$ **to** $n$ **do**
  **for** $j = 1$ **to** $m_i$ **do**
    $o \leftarrow component\_class\_index(j)$
    $\hat{y}_{ij1} \leftarrow \hat{f}_{\theta^*_o}(\mathbf{x}_{ij}, 1)$
    $\hat{y}_{ij0} \leftarrow \hat{f}_{\theta^*_o}(\mathbf{x}_{ij}, 0)$
  **end for**
  $\hat{y}_{i1} = \sum_{j=1}^{m_i} \hat{y}_{ij1}$
  $\hat{y}_{i0} = \sum_{j=1}^{m_i} \hat{y}_{ij0}$
  $\hat{\tau}(q_i) = \hat{y}_{i1} - \hat{y}_{i0}$
  $CATESamples \leftarrow CATESamples \cup \{(q_i, \hat{\tau}(q_i))\}$
**end for**