DeepCausality: A Hypergeometric Framework for Context-Aware Causal Reasoning in Rust

Marvin Hansen
Emet-Labs.com
`marvin.hansen@emet-labs.com`

Abstract

Rapid advancement of artificial intelligence, particularly large language models (LLMs), has revealed powerful capabilities in pattern recognition, information synthesis, and even apparent reasoning. However, these systems frequently exhibit limitations in genuine explainability, struggle with dynamic real-time contextual relevance, and operate primarily on correlational understanding rather than explicit causal mechanisms. This often leads to issues like "hallucinations" and unreliability when faced with out-of-distribution data or tasks requiring deep, verifiable understanding.

This paper introduces DeepCausality, a omputational causality framework implemented in Rust, designed to address these fundamental limitations. DeepCausality is available as open source, hosted at the Linux Foundation, and available online at:

`https://deepcausality.com`

DeepCausality offers a robust system for constructing and reasoning over explicit causal models deeply embedded within rich, multidimensional, and dynamic contexts. Unique to the framework is its hypergeometric representation of both causality (via CausaloidGraphs of composable Causaloids) and context (via hypergraphs of Contextoids representing data, time, space, and spacetime).

This structural approach supports transparent composability and the integration of both Euclidean and non-Euclidean relational information. DeepCausality facilitates the creation of static and dynamic causal models, supports deterministic evaluation (with pathways for probabilistic extension), and allows for complex interactions where multiple causal graphs can link to one or more shared or distinct contexts. A key feature is the Causal State Machine (CSM), enabling direct linkage of causal inferences to deterministic actions for building dynamic control systems. We detail DeepCausality's architecture, its theoretical foundations rooted in operational causality and a layered reasoning model, and its implementation leveraging Rust's type system for safety, performance, and expressiveness. We discuss current limitations, compare DeepCausality with established causal paradigms, and argue for its significance as a foundational causal engine.

***K*eywords** Computational Causality · Context-Aware Reasoning · Causaloid · Quantum Gravity · Causal Models · Explainable AI · Hypergeometric Models · Causal State Machines · Grounded Intelligent Agency

Contents

1 Introduction

The enduring pursuit of artificial intelligence (AI) has been to create systems that not only emulate human cognitive functions but also comprehend and interact with the world in a meaningful, reliable, and ultimately trustworthy manner. Recent advancements, particularly in deep learning (DL) and large language models (LLMs), have demonstrated remarkable capabilities in processing vast data volumes, identifying complex patterns, and generating human-like responses, leading to their widespread adoption across diverse fields [1].

Despite these significant strides, contemporary AI, especially large-scale DL models, frequently encounters fundamental limitations when applied to domains demanding high levels of trust, verifiability, and a genuine understanding of underlying mechanisms. Such systems often operate as opaque "black boxes," making the rationale behind their decisions difficult to scrutinize [2]. Trained primarily on static datasets, they exhibit a "knowledge cut-off," limiting their adaptability to dynamic, real-time environments without specialized augmentations. Perhaps most critically, their foundation in statistical correlation, rather than causal understanding, means they can identify co-occurrences but not necessarily the "why" behind them, nor predict the outcomes of interventions [3]. This can lead to unreliable outputs or "hallucinations," especially when faced with out-of-distribution data or tasks requiring deep, verifiable insight.

To address these challenges, this preprint introduces DeepCausality, a computational causality framework engineered from first principles in Rust. DeepCausality is designed to enable the construction and execution of explicit, context-aware, and explainable causal models. It shifts the focus from purely statistical pattern matching to the modeling of generative mechanisms, providing a pathway for AI systems to reason about cause and effect within dynamic and multifaceted environments. DeepCausality's core contributions:

- A cornerstone of DeepCausality is its novel hypergeometric conceptualization of context, which moves beyond simple contextual variables to enable the creation of intricate context hypergraphs. These graphs are populated by Contextoids – specialized nodes representing rich, multi-dimensional information encompassing Data, Time, Space, and SpaceTime. This framework inherently supports dynamically adjustable contexts, allowing values and even structural relationships to evolve, and critically, it allows a single causal model to draw upon and integrate information from one or more such distinct context hypergraphs simultaneously. Furthermore, the system natively incorporates both Euclidean and non-Euclidean relational data, thereby grounding causal reasoning in a highly nuanced and comprehensive understanding of the operational environment.
- DeepCausality introduces a structurally composable approach to causal modeling , leveraging Causaloids – encapsulated, testable causal functions – as fundamental building blocks. These Causaloids are organized within CausaloidGraphs, which are hypergraphs explicitly representing intricate causal relationships. This architecture crucially employs recursive isomorphic causal data structures, meaning that elements within the causal graph (i.e., nodes representing causes) can themselves be entire sub-graphs or collections of other causes, thus enabling the intuitive construction of deeply complex, layered, and modular causal systems where macro-level causal phenomena can be decomposed into interacting micro-level causal mechanisms.
- The Causal State Machine (CSM) provides the crucial link between causal understanding and effective intervention within DeepCausality. It is architected to manage interactions between one or more causal models and their associated contexts, orchestrating a cycle of data extraction from these contexts to inform or update the models. Based on the collective causal inference derived—that is, the identification of specific active causes or system states—the CSM then deterministically initiates predefined actions. This capability is fundamental to facilitating the creation of complex, dynamic control and supervision systems that can respond to changes based on an explicit, causally reasoned understanding of the context in the operational environment.
- A reference implementation of DeepCausality is provided in Rust, a language selected specifically for its capacity to address the demanding requirements of a sophisticated causal reasoning engine. To handle potentially vast and rapidly changing contextual data alongside complex causal model evaluations, Rust's high-performance characteristics – stemming from efficient compilation to machine code and fine-grained control over system resources – are indispensable for systems designed for reliable operation.

We begin with the motivation followed by the necessary background. We then delve into the conceptual foundations of DeepCausality, drawing inspiration from operational views of causality. The core of the paper presents the architecture and detailed concepts of the DeepCausality framework, including its context engine, causal modeling engine, and Causal State Machines, and discusses its Rust implementation. We subsequently analyze current limitations, compare DeepCausality to alternative causal paradigms, and articulate its significance. Finally, we elaborate on the future direction of architecting Grounded Intelligent Agency by fusing DeepCausality with LLMs, and conclude with a summary of its potential impact.

2 Motivation

Recent advancements in artificial intelligence, particularly in deep learning (DL) and large language models (LLMs), have demonstrated remarkable proficiency in processing vast amounts of data, recognizing intricate patterns, and generating human-like text and responses [4, 5, 6]. These systems can summarize information, answer complex questions, and even generate creative content, leading to their rapid adoption across numerous fields. However, alongside these successes, a closer examination reveals inherent limitations stemming from their primarily correlation-based foundations.

2.1 Limitations of corelation based methods

The impressive achievements of deep learning models are undeniable, having revolutionized fields from natural language processing to computer vision [4, 5, 6, 7]. However, these successes are largely predicated on their strength as statistical techniques capable of sophisticated pattern learning or "curve fitting" over vast datasets [8, 7]. This foundation gives rise to several inherent limitations, hindering their ability to achieve genuine understanding and robust, trustworthy reasoning, particularly in complex, dynamic, and high-stakes environments [8, 7, 9]. These limitations include significant challenges in generalizing beyond the training data space and handling distribution shifts [7], vulnerability to subtle input perturbations [7], a lack of transparency and interpretability which is critical in high-stakes applications [7], and a fundamental inability to inherently reason about interventions or causality [8, 7, 9].

Correlation does not imply Causation: Deep learning models excel at identifying and exploiting statistical correlations within data [7, 9, 10, 11]. However, as famously articulated by Judea Pearl [12], these achievements are largely based on sophisticated "curve fitting" or pattern interpolation over vast datasets[12, 13, 7]. They do not inherently distinguish between correlation and causation [12, 10, 7, 9, 14, 15, 11, 16]. Machine learning applied to observational data [15], where predictive variables are not under the learner's control [10], can typically only learn correlations [10, 15]. Consequently, a model might learn that factor A is highly correlated with outcome B [10], but it cannot, without explicit causal information or intervention capabilities [12, 10, 7, 9, 15, 17, 18], determine if A causes B, B causes A, or if a hidden common cause C influences both [10, 15]. This limitation is critical for decision-making [10, 9, 19, 20, 21], as acting upon a spurious correlation [10, 15, 20] can lead to ineffective or even detrimental interventions [10, 20].

The Independent and Identically Distributed (IID) Data Assumption: Many foundational machine learning algorithms, and by extension some deep learning approaches, assume that data points are drawn independently from the same underlying distribution [22]. This assumption is frequently violated in real-world scenarios [7, 23]. Financial market data exhibits temporal dependencies and regime shifts [24]; social network data is characterized by complex interdependencies [25, 19]; industrial sensor data may be autocorrelated [26, 27, 28]. When the IID assumption breaks down, the performance and reliability of DL models can degrade significantly [7], as the patterns learned during training may no longer hold in production [7]. DARPA itself has noted the limitations of treating "each data set as an independent, uncorrelated input" and the need to model "underlying causal mechanism[s]" [29].

The "Black Box" Problem and Lack of Genuine Explainability: Deep neural networks, especially those with billions or trillions of parameters like modern LLMs, are often described as "black boxes". [7, 9]. While techniques from Explainable AI (XAI) can provide post-hoc rationalizations or highlight influential features, they often do not reveal the true internal computational path or the underlying reasoning process in a verifiable way [7, 9, 14]. Deep learning models learn statistical patterns and associations, but may not capture underlying causal mechanisms [7]. The opacity makes it difficult to debug models [30], and is a potential liability when using deep learning for problem domains medical diagnosis, where human users might like to understand how a system made a decision [30]. Such opacity can also lead to issues of bias [30]. Model explanations allow us to argue for model decisions and exhibit the situation when algorithmic decisions might be biased or discriminating, and precise explanations may facilitate model debugging and error analysis [30]. It is important to build trust, particularly in critical applications [30].

Knowledge Cut-off and Real-Time Relevance: LLMs are trained on data up to a specific point in time. Consequently, they lack knowledge of events, discoveries, or shifts in the world that occur after their training data concluded. While "search grounding" – augmenting LLMs with real-time internet search – provides access to current facts, it does not inherently equip the model with an understanding of evolving dynamics or the ability to integrate new information into a coherent, evolving causal model of a system's state [7]. The interpretation of new information is still performed through the lens of patterns learned from historical data [7].

Lack of Innate Conceptualization of Time, Space, and Causality: While deep learning models can learn patterns related to these concepts from data, they do not possess them as foundational, structured elements of their reasoning framework in the way humans do [31, 32, 12, 7]. Rather, true AI requires an innate grasp of fundamental concepts like time, space, and causality [12, 7, 9, 33].

Constraints of Euclidean Data Representation: Many deep techniques rely on embedding data into Euclidean vector spaces. However, as J.M. Bishop highlights, relational structures (e.g., social networks, molecular interactions, conceptual hierarchies) are often better represented using non-Euclidean geometries like graphs or hypergraphs. Forcing such data into Euclidean spaces can lead to a loss of information or the introduction of artificial relationships. [34, 33].

These limitations underscore the need for AI paradigms that go beyond statistical pattern matching to incorporate explicit causal understanding, robust contextual awareness, and transparent, verifiable reasoning. DeepCausality proposes as a framework designed to contribute to this endeavor.

2.2 Consequences of Over-Reliance on Correlational

The identified limitations of primarily correlation-based AI systems are not merely academic concerns; they precipitate significant and often detrimental consequences, both immediate and higher-order, when these systems are deployed in the real world. An over-reliance on statistical pattern matching without deeper causal understanding can lead to:

Failure in Novel or Dynamic Environments: Systems trained on specific data distributions (violating the IID assumption in practice) often fail catastrophically when encountering out-of-distribution data or shifts in the underlying generative process [7]. This brittleness makes them unreliable for applications in evolving settings, such as financial markets or autonomous navigation, where adaptability is paramount. The inability to distinguish causation from spurious correlation means models may latch onto ephemeral patterns that do not generalize.

Lack of Trust and Accountability: The "black box" nature of many deep learning models, where the reasoning pathway from input to output is inscrutable, severely erodes trust, especially in high-stakes decision-making like medical diagnosis or legal applications [30, 9]. Without genuine explainability, it is difficult to debug failures, assign responsibility for erroneous or biased outcomes, or assure users and regulators of the system's safety and fairness.

Ineffective or Harmful Interventions: If an AI system recommends an action based on a learned correlation that is not truly causal, the resulting intervention may be ineffective at best, or actively harmful at worst [10, 15]. For example, a system might correlate a symptom with a recovery outcome without understanding that both are caused by an underlying treatment, leading to potentially flawed recommendations if it tries to manipulate the symptom directly.

Stagnation in Scientific Discovery and Understanding: Progress in many scientific fields relies on uncovering underlying causal mechanisms, not just predicting outcomes. AI systems that cannot move beyond correlation to provide insights into these mechanisms offer limited value for advancing fundamental scientific understanding or generating truly novel, mechanistically sound hypotheses.

Ethical Concerns and Perpetuation of Bias: Models learning from biased historical data without an understanding of the causal factors generating those biases are prone to perpetuating or even amplifying societal inequities [30]. A causal perspective is often necessary to identify and mitigate such biases by understanding their origins.

Limited Adaptability and Continuous Learning: Systems lacking innate conceptualizations of fundamental constructs like time, space, and causality, or those constrained by fixed knowledge cut-offs, struggle with genuine continuous learning and adaptation. They cannot easily integrate new information into a coherent, evolving model of the world's underlying causal structure.

These consequences, ranging from practical failures in dynamic applications to fundamental limitations in achieving trustworthy and generalizable intelligence, collectively highlight a profound need for AI paradigms that can transcend the mere fitting of surface-level statistical associations. While pattern recognition has propelled AI to remarkable achievements, the increasing deployment of these systems in complex, high-stakes, and human-facing domains demands a move towards architectures built upon a more robust foundation of causal understanding and explicit mechanistic reasoning. Without this shift, AI systems will likely continue to struggle with out-of-distribution generalization, remain opaque in their decision-making processes, risk perpetuating harmful biases, and prove inadequate for tasks requiring true comprehension of how and why events unfold or how interventions will genuinely alter outcomes. Therefore, it is time for the development of new foundational frameworks capable of imbuing intelligent systems with a deeper, more principled, and ultimately more reliable grasp of the causal fabric of the world.

2.3 The Path Forward: Principles for Causally-Grounded Intelligence

These pervasive challenges and their significant consequences underscore the necessity for computational frameworks that move beyond purely statistical pattern matching and embrace principles of explicit causal modeling. We argue that a path forward towards more robust, trustworthy, and adaptable AI requires systems capable of adhering to several core philosophical tenets. Each of these tenets aims to directly mitigate one or more of the limitations and adverse effects identified above:

Explicitly representing causal mechanisms rather than just learning correlations: This directly counters the "correlation does not imply causation" problem, aiming to build models that reflect underlying generative processes. This provides a foundation for more reliable predictions, especially in novel situations, and enables more effective interventions by targeting true causes rather than mere symptoms or associated variables.

Deeply integrating rich, multi-dimensional, and dynamic context (including Euclidean and non-Euclidean relations) into the reasoning process: This addresses the failures arising from the IID assumption and the knowledge cut-off. By making models acutely aware of their operational environment and its evolution—across various data types, temporal scales, and spatial configurations—their adaptability and relevance in dynamic settings are significantly enhanced.

Making foundational assumptions transparent, verifiable, and integral to model validity and transfer: This tackles the "black box" problem and issues of trust. When assumptions are explicit and, where possible, testable (even if only against contextual consistency), the conditions under which a model's causal claims hold become clearer, facilitating safer deployment and more principled model transportability to new domains.

Employing structured, composable representations for causal knowledge that allow for transparency and scalability: This further aids explainability and helps manage the complexity of real-world causal systems. Modular, hierarchical models are easier to understand, debug, and extend, countering the opacity of monolithic neural networks.

Leveraging implementation choices that prioritize performance, safety, and reliability for practical application: This ensures that sophisticated causal reasoning capabilities can be deployed effectively in real-world systems, including those with demanding performance requirements or safety-critical functions.

It is crucial to recognize that the path forward is not necessarily a binary choice between correlation-based deep learning and explicit causal modeling. Rather, the most profound advancements in AI are likely to emerge from the synergy of both paradigms fusion together. Large-scale deep learning models excel at perceptual tasks and extracting complex patterns from vast, unstructured data. Structured causal frameworks can then leverage these extracted features and initial hypotheses, embedding them within a rigorous system of mechanistic reasoning, contextual understanding, and verifiable assumptions. This combination allows for the creation of intelligent systems that are both broadly aware and deeply analytical, capable of learning from data while understanding the causal fabric of the world they operate in. The remainder of this preprint introduces DeepCausality, a novel framework designed around these synergistic principles, aiming to provide a concrete pathway for developing such causally-grounded and ultimately more intelligent systems.

3 Related Work

The study of causality and causal inference aims to distinguish genuine cause-and-effect relationships from mere associations. Traditionally, establishing causality often relied on carefully controlled randomized controlled trials. However, significant theoretical advancements have shown that causal knowledge can be inferred from observational data by examining patterns of conditional independence among variables, given explicit assumptions [8].

A foundational framework for representing causal structures is based on graphical causal models, most notably Directed Acyclic Graphs (DAGs) [35, 36, 37, 38]. In these models, variables are typically represented by nodes, and directed edges indicate direct causal influences [36]. The impact of interventions, conceptualized by operators like the `do`-operator which sets a variable's value independently of its usual causes, can be analyzed within this framework to predict outcomes under hypothetical scenarios [36, 12]. The theoretical underpinnings of Structural Causal Models (SCMs), which are closely related to graphical models, have been extensively studied [39, 15, 40, 41, 42]. Methods exist for handling complex scenarios, including incorporating latent variables [43, 44, 45] and understanding the relationship between different causal models [46, 8]. Policy interventions in specific graphical structures, such as Lauritzen-Wermuth-Freydenburg (LWF) latent-variable chain graphs, have also been investigated [47]. This includes work providing a novel identification result for effects of policy interventions in these graphs [47].

3.1 Foundational Theories of Causal Inference

The endeavor to formalize and compute causal relationships draws upon several influential theoretical frameworks. Understanding these foundations is crucial for situating contemporary advancements and appreciating the nuances of different approaches to causal reasoning.

The dominant paradigm in modern computational causality is arguably the **Structural Causal Model (SCM)** framework, extensively developed by Judea Pearl and his colleagues [12]. An SCM consists of a set of variables, some of which are designated as exogenous (external, uncaused within the model) and others as endogenous (their values are determined by other variables within the model). The relationships between these variables are represented by a set of structural equations, typically of the form $X_i = f_i(\mathbf{PA}_i, U_i)$, where $\mathbf{PA}_i$ are the direct causal parents of X_i in the associated causal graph, and U_i are exogenous error terms representing unmodeled influences or inherent stochasticity. These structural equations are considered to represent autonomous, invariant causal mechanisms. Graphically, SCMs are typically depicted using **Directed Acyclic Graphs (DAGs)**, where nodes represent variables and directed edges $X_j \to X_i$ indicate that X_j is a direct cause of X_i (i.e., $X_j \in \mathbf{PA}_i$). This graphical representation provides an intuitive way to encode causal assumptions and to determine statistical independencies via the criterion of *d-separation*.

A cornerstone of Pearl's framework is the ***do*-calculus**, a set of three axiomatic rules that allows for the inference of the effects of interventions from a combination of observational data and the causal graph structure, even when direct experimentation is not possible [12]. An intervention, denoted $do(X_j = x'_j)$, represents an external manipulation that sets the variable X_j to a specific value x'_j, thereby severing the links from its original parents $\mathbf{PA}_j$ and altering the system's natural dynamics. The ability to calculate post-intervention distributions, $P(Y|do(X = x))$, is central to predicting the consequences of actions and policies. **Bayesian Networks**, which are DAGs coupled with conditional probability distributions $P(X_i|\mathbf{PA}_i)$, are closely related to SCMs and are often used to represent the observational probability distribution $P(\mathbf{X})$ entailed by an SCM under specific assumptions about the error terms U_i. They provide a powerful tool for probabilistic inference under passive observation, but require the *do*-calculus or similar interventional logic to reason about causal effects.

Alongside SCMs, the **Potential Outcomes Framework**, also known as the Rubin Causal Model (RCM) [48], offers another rigorous foundation for causal inference, with early conceptualizations by Neyman [49] and formally developed for observational studies by Rubin [50]. It has been particularly influential in statistics, econometrics, and social sciences. This framework defines the causal effect of a treatment (or exposure) on an individual unit by considering the potential outcomes that unit would exhibit under different treatment assignments. For a binary treatment $T \in \{0, 1\}$, each unit i is conceptualized as having two potential outcomes: $Y_i(1)$, the outcome if unit i receives the treatment, and $Y_i(0)$, the outcome if unit i receives the control. The individual treatment effect (ITE) is then $Y_i(1) - Y_i(0)$. A fundamental challenge, often termed the "fundamental problem of causal inference," is that only one of these potential outcomes can be observed for any given unit [48]. Causal inference in this framework often relies on assumptions such as the Stable Unit Treatment Value Assumption (SUTVA), which posits no interference between units and well-defined treatment versions, and ignorability (or unconfoundedness), which states that treatment assignment is independent of potential outcomes, conditional on observed covariates [51]. This framework excels in clarifying the conditions needed for estimating average treatment effects (ATE) from observational data, often employing methods like matching, stratification, or inverse probability weighting based on propensity scores [51].

While SCMs and the Potential Outcomes framework have different notational and conceptual starting points, they have been shown to be largely consistent and can often address the same set of causal questions, with SCMs providing a more explicit language for encoding causal mechanisms and deriving identifiability conditions through graphical criteria [12]. These foundational theories provide the bedrock upon which most modern computational causality techniques, including those involving deep learning, are built or against which they are compared.

3.2 Counterfactuals: Reasoning About What Might Have Been

While discovering causal structures and predicting the effects of interventions (“What if we do $X = x$?”) are fundamental tasks in causal inference, the ability to compute **counterfactual queries** represents a deeper and often more insightful level of causal reasoning [12]. Counterfactuals address questions about alternative realities or "what might have been" (“What if X had been x', given that we observed $X = x$ and $Y = y$?”). This form of reasoning is crucial for tasks such as understanding individual responsibility, learning from past mistakes, diagnosing failures, and fine-tuning policies. It requires moving beyond population-level effects of interventions to consider specific individuals or units in specific factual circumstances [12, 52].

Within Pearl’s Structural Causal Model (SCM) framework, computing a counterfactual, denoted as $Y_x(u)$ (the value Y would have taken in unit u had X been x), involves a three-step algorithmic process [12]:

1. **Abduction:** Use the available factual evidence (e.g., observed values of some variables) to update the probability distribution over the exogenous variables U. This step accounts for the specific unit or situation under consideration by inferring the background conditions consistent with the observed facts.
2. **Action:** Modify the original SCM by replacing the structural equation for the counterfactual antecedent X with $X = x'$ (the hypothetical condition), effectively performing a "mini-surgery" on the model as in the *do*-calculus. The equations for other variables remain unchanged, reflecting the principle that interventions only alter the targeted mechanism directly.
3. **Prediction:** Compute the probability of the counterfactual consequent Y using the modified model and the updated distribution of U (from the abduction step). This yields the probability $P(Y_{x'} = y' | \text{evidence})$.

This process allows for a principled way to reason about hypothetical scenarios that differ from what was actually observed, effectively comparing parallel possible worlds [12, 53]. Foundational work also explored the bounding and identification of specific types of counterfactual queries related to probabilities of causation [54]. Formal systems like Pearl’s *do*-calculus provide tools for determining if causal effects under intervention are identifiable from observational data [55], and algorithms exist to automate this process [56], which are often prerequisite steps before full counterfactual queries can be comprehensively addressed.

The extension of these concepts to practical applications and more complex settings remains an active area of research. For instance, model-agnostic approaches aim to enable counterfactual reasoning without full specification of the SCM, particularly in dynamic environments where systems evolve over time. Furthermore, the domain of causal bandits, which focuses on online decision-making and learning under uncertainty, increasingly incorporates causal background knowledge and aspects of counterfactual reasoning to optimize sequences of actions and learn policies more efficiently than purely correlational reinforcement learning approaches [17, 18, 57, 58]. The capacity for counterfactual reasoning thus forms a critical component of advanced intelligent systems that can not only predict and act, but also reflect, learn, and adapt based on a deep understanding of cause and effect in alternative scenarios.

More recent work explores model-agnostic approaches to counterfactual reasoning, particularly in dynamic environments [59], and investigates optimizing treatment effects in such settings [60]. Causal bandits also incorporate causal background knowledge into online decision-making problems [17, 18, 57, 58].

3.3 Causal Discovery

A central and challenging task in the field is causal discovery, which focuses on learning the causal structure, represented by the graph, from observed data alone. A comprehensive survey categorizes existing methods for causal discovery on both independent and identically distributed (I.I.D.) data and time series data, including approaches for both types of data. According to this survey, categories include Constraint-based, Score-based, FCM-based, Hybrid-based, Continuous-Optimization-based, or Prior-Knowledge-based. Constraint-based methods infer relationships by testing for conditional independencies in the data [37, 61, 62]. Score-based methods search over potential graph structures and evaluate them based on how well they fit the data, often including a penalty for complexity [63]. The KGS method [64], for example, leverages prior causal information such as the presence or absence of a causal edge to guide a greedy score-based causal discovery process towards a more restricted and accurate search space. It demonstrates how incorporating different types of edge constraints can enhance both accuracy and runtime for graph discovery and candidate scoring, concluding that any type of edge information is useful. This method relates to the KCRL framework [64]. Continuous optimization techniques formulate causal discovery as an optimization problem, potentially involving differentiable approaches that can handle constraints like acyclicity. The NOTEARS framework is one such example [65], and studies have analyzed its performance and proposed post-processing algorithms to enhance its precision and efficiency. A study provides an in-depth analysis of the NOTEARS framework for causal structure learning, proposing a local search post-processing algorithm that significantly increased the precision of NOTEARS and other algorithms [66]. This work also deduced Karush-Kuhn-Tucker (KKT) optimality conditions for an equivalent reformulation of the NOTEARS problem [66]. Comparisons showed that a method called Abs-KKTS performed better than NOTEARS in terms of accuracy and computational efficiency on various graph types [66]. GraN-DAG is another approach that uses neural networks to handle non-linear causal relationships and utilizes a novel mechanism that uses interventional data to infer causal structures [67].

3.4 Causal Inference for Time Series

Causal inference for time series data introduces a unique set of challenges and opportunities compared to static, cross-sectional settings. The inherent temporal ordering of observations provides strong, intuitive information about potential causal directionality—causes generally precede their effects—but also necessitates methods that can handle auto-correlation, non-stationarity, feedback loops, and varying time lags in causal influences.

A foundational concept in this domain is **Granger Causality**, originally developed by Clive Granger for economic time series [68, 69]. A time series X_t is said to Granger-cause another time series Y_t if past values of X_t contain information that helps predict future values of Y_t beyond the information already contained in past values of Y_t itself. This is typically tested using vector autoregression (VAR) models and statistical tests on the coefficients of lagged variables [70, 71]. While widely applied, standard Granger causality is primarily about predictive improvement and may not always align with true mechanistic causation, especially in the presence of unobserved confounders, instantaneous effects, or non-linear relationships. Extensions and refinements have been developed to address some of these limitations, including non-linear Granger causality tests and methods incorporating multivariate information criteria.

To explicitly model evolving causal relationships and dependencies over time, **dynamic graphical models** have been developed. The Dynamic Uncertain Causality Graph (DUCG) [72] is one such framework, specifically designed to represent and reason about causal relationships that themselves change as a system evolves. DUCGs find applications in complex dynamic systems, such as fault diagnosis in nuclear power plants where understanding the temporal progression of component failures is critical [73, 74]. These models often aim to unify diagnostic reasoning (what caused an observed state?) with treatment or control strategies (what intervention will lead to a desired future state?) [75].

More recently, deep learning techniques have been increasingly applied to causal discovery and inference in time series. For example, the Time-series Causal Discovery Framework (TCDF) utilizes attention-based convolutional neural networks to learn causal relationships, explicitly trying to identify relevant time lags and dependencies [76]. Research in this direction often focuses on challenges such as optimizing hyperparameters for these complex models, ensuring robustness to varying noise levels and non-stationarities in the data, improving the interpretability of attention mechanisms to understand which past events are deemed causally salient, and developing robust causal validation methods beyond simple predictive accuracy.

Beyond these, other important research avenues in time series causality include:

- **Handling Unobserved Confounders:** Just as in static settings, unobserved common causes can induce spurious relationships between time series. Methods that attempt to detect or adjust for such confounding, perhaps using instrumental variable approaches adapted for time series or by searching for specific types of conditional independencies, are crucial.
- **State-Space Models and Causal Inference:** Integrating causal concepts with state-space models (e.g., Kalman filters and their non-linear extensions) allows for reasoning about causality between latent (unobserved) states as well as observed variables.
- **Interventional Time Series Analysis:** Developing methods to estimate the effect of specific interventions applied at certain points in time on the future trajectory of one or more time series. This is vital for policy evaluation and system control.
- **Causal Discovery from Irregularly Sampled or High-Dimensional Time Series:** Many real-world time series (e.g., medical patient data, sensor networks) are not regularly sampled or involve a very large number of variables, posing challenges for traditional methods.
- **Information-Theoretic Approaches:** Methods like Transfer Entropy [Schreiber, 2000, *Measuring information transfer*] provide a non-parametric way to quantify directed information flow between time series, offering an alternative perspective to Granger causality, especially for detecting non-linear interactions.

The temporal dimension thus adds significant complexity but also provides a powerful constraint (time ordering) that can be leveraged for causal reasoning, making this a vibrant and critical area of ongoing research.

3.5 The Role of Context and Temporality in Causal Inference

While foundational causal frameworks like SCMs implicitly allow for conditioning variables, the explicit, structured, and dynamic modeling of *context* as a multi-faceted entity is a growing area of focus, crucial for applying causal inference to complex, real-world systems. The Jiao et al. survey [77] highlights numerous deep learning applications where contextual understanding is paramount, from visual commonsense reasoning to multimodal interactions, and notes the challenges posed by contextual shifts and confounders.

Berrevoets et al. [78, 79] propose a conceptual "map of causal deep learning" (CDL) that explicitly incorporates dimensions for structural knowledge, parametric assumptions, and significantly, a **temporal dimension**. They argue that time is not merely another variable but introduces unique considerations in causal settings, such as the fundamental principle that causes precede effects, and the potential for feedback loops or evolving relationships in dynamic systems. Their framework aims to help researchers and practitioners categorize CDL methods based on how they handle these dimensions, including whether they operate on static data or explicitly model temporal dynamics. For instance, they differentiate models based on whether they assume "no structure," "plausible causal structures" (often derived from statistical independencies), or a "full causal structure" as input, and similarly categorize parametric assumptions from non-parametric to fully known factors. The temporal axis distinguishes between static models and those designed for time-series data where variables are observed repeatedly. While this work by Berrevoets et al. primarily offers a *taxonomy and conceptual guide* for the emerging field of CDL rather than a specific implemented reasoning engine, it underscores the increasing recognition of structured context, and especially temporality, as a first-class concern in bridging deep learning with robust causal inference.

This emphasis on temporal context aligns with established work in time series causality, such as Granger causality [68] and dynamic graphical models like DUCGs [72], which inherently focus on how relationships evolve over time. However, modern approaches, including those at the intersection of deep learning and causality, seek richer representations of temporal context beyond simple lagged variables. For example, methods like TCDF [76] attempt to learn relevant temporal dependencies and attention patterns. The challenge remains to develop frameworks that can uniformly reason over diverse types of contextual information—static attributes, explicit temporal sequences, spatial relationships (both Euclidean and non-Euclidean), and even abstract conceptual states—and integrate this rich contextual understanding directly into the causal reasoning process. The ability to model multiple, potentially interacting contexts, and to allow these contexts to be dynamically updated, is key to building causal AI systems that can adapt to real-world complexities.

3.6 Causal Inference and Discovery on Graph and Hypergraph Structures

The representation of causal relationships via graphical models, predominantly Directed Acyclic Graphs (DAGs) as foundational to Structural Causal Models (SCMs) [12], is a cornerstone of computational causality. Much research has focused on discovering these graph structures from observational or interventional data (causal discovery) and subsequently estimating causal effects based on the identified graph (causal inference). While traditional methods often assume simpler pairwise relationships, the inherent complexity of many real-world systems necessitates considering more intricate relational structures.

Recent work has begun to explicitly tackle causal inference in settings involving multi-way interactions best represented by hypergraphs. Ma et al. [80] directly address the problem of estimating Individual Treatment Effects (ITE) on hypergraphs, specifically accounting for high-order interference where group interactions (modeled by hyperedges) influence individual outcomes. Their proposed HyperSCI framework leverages hypergraph neural networks to model these spillover effects and uses representation learning to control for confounders, demonstrating the utility of explicitly considering hypergraph topology for ITE estimation from observational data. This represents a significant step beyond assuming only pairwise interference, which is common in ordinary graph-based causal inference. While the work by Ma et al. focuses on statistical ITE estimation on a *given* hypergraph, it highlights the increasing recognition of hypergraph structures as vital for certain causal problems.

The broader field of graph mining and network science also provides a rich backdrop, with techniques for link prediction and understanding influence spread, though these often operate at a correlational level rather than a strictly causal one. The challenge remains to bridge network science concepts with formal causal reasoning in these complex relational systems.

3.7 (Geometric) Deep Learning for Causal Inference and Representation

The intersection of deep learning with causal inference is a rapidly expanding research area, aiming to leverage the expressive power of neural networks to address challenges in causal representation learning, discovery, and effect estimation [81, 77]. Many approaches focus on adapting deep learning architectures to better estimate treatment effects from observational data, often by learning balanced representations of covariates to mitigate confounding bias or by modeling complex response surfaces.

Ramachandra [82] proposes the use of deep autoencoders for generalized neighbor matching to estimate ITE, focusing on dimensionality reduction while preserving local neighborhood structure, and also suggests using Deep Neural Networks (DNNs) for improved propensity score estimation (PropensityNet). These methods exemplify the application of standard deep learning architectures to enhance specific statistical tasks within the potential outcomes framework, typically under assumptions such as the Stable Unit Treatment Value Assumption (SUTVA), which precludes interference.

More fundamentally, researchers are exploring how geometric deep learning principles can inform the design of causal models capable of handling complex data structures and respecting informational constraints. Acciaio et al. [83] introduce a "universal causal geometric DL framework," featuring the Geometric Hypertransformer (GHT). Their work is concerned with the universal approximation of causal maps between discrete-time path spaces, which may be non-Euclidean metric spaces such as Wasserstein spaces, while strictly respecting the forward flow of information inherent in causal processes. The GHT employs hypernetworks to adapt its parameters over time and aims to provide theoretical guarantees for approximating Hölder continuous functions between these complex spaces. Although highly theoretical and with a focus on applications in stochastic analysis and mathematical finance, this line of work signifies a deep engagement with geometric structures, non-Euclidean spaces, and transformer-like attention mechanisms within a causal learning context. Their "geometric attention mechanism" operating on Quantizable and Approximately Simplicial (QAS) spaces represents a sophisticated approach to handling non-Euclidean output geometries.

The comprehensive survey by Jiao et al. [77] also details various methods where deep learning is applied to causal discovery (e.g., leveraging neural networks for GraN-DAG or extensions of NOTEARS) or to augment specific causal inference tasks within existing deep learning modalities (e.g., developing causal attention mechanisms in computer vision, or applying causal methods to Graph Neural Networks). This body of work collectively seeks to imbue deep learning models with a degree of causal awareness or to use their representational power to overcome limitations in traditional causal inference techniques. The ongoing challenge in this domain is to move beyond enhancing specific sub-tasks towards building more integrated and principled frameworks for comprehensive causal reasoning using deep learning.

3.8 The Quantum Gravity Causaloid Framework

A significant effort towards establishing a framework for probabilistic theories with dynamic causal structure was presented by Hardy [84]. Hardy proposes a framework aimed at unifying quantum theory (QT) and general relativity (GR) as a step towards quantum gravity (QG). The core of this unification lies in a generalized theory of causality, capable of describing both the probabilistic nature and fixed causal structure of QT, as well as the deterministic nature and dynamic causal structure of GR, within a single formalism. The core of this new framework of unified causality is the "causaloid," a mathematical object designed to encapsulate all information about the causal relationships within a physical system. The framework begins from an operational standpoint, focusing on "recorded data" which consists of "actions" and "observations" associated with "elementary regions" of spacetime. The causaloid itself is a theory-specific mathematical entity, primarily represented by a collection of "lambda matrices" (Λ). These matrices quantify how the complexity of describing a composite region (specifically, the number of fiducial measurements needed to determine its state) is reduced due to causal connections between its component elementary regions. Associated with any region R and an experimental procedure F_R resulting in outcome X_R are "r-vectors," denoted $r(X_R, F_R)(R)$, which are analogous to operators in QT [84]..

A key innovation is the "causaloid product", which is governed by the causaloid (via the lambda matrices). This product combines r-vectors of sub-regions to form the r-vector for a composite region, e.g., $r(R_1 \cup R_2) = r(R_1)\hat{}r(R_2)$. This product aims to unify the different ways systems are composed in QT, such as tensor products for spacelike separated systems and sequential (matrix) products for timelike evolutions. Probabilities for joint outcomes, conditioned on experimental settings, are then derived from these r-vectors. A crucial feature of the causaloid formalism is that it does not impose a fixed causal structure or a background time a priori. Instead, the causal relations are implicitly defined by the causaloid itself. Hardy demonstrated how both classical probability theory and quantum theory can be cast within this framework, with the differences between theories being encoded entirely in the specification of their respective causaloids. The ultimate aim is to provide a structure wherein the dynamic causal aspects of GR can be consistently combined with the probabilistic nature of QT. Hardy also introduces "causaloid diagrams" as a visual tool to represent and compute the causaloid based on local lambda matrices for nodes (elementary regions) and links (pairwise connections), particularly under simplifying assumptions met by QT and classical probability [84]..

3.9 Deep Learning

Deep learning has achieved remarkable success in various tasks, including representation learning and prediction [7]. Related areas include neural architecture search [85, 86] and techniques for handling complex relationships in data, such as those explored using hypergraphs [34, 87]. Hypergraphs, introduced by Berge in 1973 [87], can model multi-way relationships and have found applications in areas like visualization [88, 89], partitioning [90, 91, 92], and recommender systems [93, 94, 95, 96, 97, 98]. Link prediction, particularly in multiplex networks, is another active area where deep learning is applied [99, 100].

More recently, researchers have explored the intersection of causal inference and deep learning, with the aim of leveraging the strengths of both fields [7]. A notable direction involves incorporating prior causal knowledge into deep generative models, enabling the generation of data that respects a given causal graph. While combining causal discovery with generative modeling is a goal, these methods are often constrained by the fundamental limitations of causal discovery [37] [2]. Specific efforts include incorporating causal graphical prior knowledge into predictive modeling [101] and matching learned causal effects with domain priors in neural networks [102]. Applications in finance have also utilized informed machine learning frameworks based on a priori causal graphs for prediction tasks. The area of causal reinforcement learning and causal bandits also represents significant related work in combining causality with learning agents that interact with environments [17, 18, 57, 58, 20, 103].

3.10 Computational Causality Libraries

A vibrant ecosystem of Python libraries has emerged over time, providing tools for various aspects of causal inference, discovery, and analysis. These libraries typically build upon foundational causal theories and aim to make causal methods accessible to data scientists, researchers, and engineers. This report summarizes several key libraries shaping the Python landscape for computational causality.

3.10.1 DoWhy (Microsoft)

Developed by Microsoft Research, DoWhy[1] [104] is perhaps one of the most well-known libraries aiming to provide an end-to-end workflow for causal inference. Its philosophy centers on explicitly separating the causal modeling assumptions from the statistical estimation steps, adhering to a four-stage process: 1) Modeling the causal assumptions (often using graphical models), 2) Identifying the target causal estimand based on the model, 3) Estimating the causal effect using appropriate statistical methods (like propensity scores, regression, instrumental variables), and 4) Refuting the obtained estimate through robustness checks. DoWhy aims to unify concepts from both Pearl's Structural Causal Models (SCMs) and the Potential Outcomes framework. It integrates with other libraries like EconML and CausalML for specific estimation tasks and is designed to be a general-purpose tool for applied causal analysis in data science projects.

3.10.2 EconML (Microsoft)

EconML[2] [105] focuses specifically on estimating heterogeneous treatment effects (HTE) – understanding how the effect of an intervention or treatment varies across different individuals or subgroups. It heavily leverages machine learning techniques to model complex conditional outcome expectations and propensity scores while incorporating causal identification strategies to ensure the validity of the effect estimates. Key methodologies implemented include Double Machine Learning (DML), Orthogonal Random Forests, Deep Instrumental Variables (DeepIV), and various "meta-learners" (S-learner, T-learner, X-learner) that adapt standard ML models for causal effect estimation. EconML is particularly powerful for applications in economics, personalized medicine, and targeted marketing where understanding individualized causal responses is critical.

3.10.3 CausalML (Uber)

Developed initially at Uber, CausalML[3] [106] is another library primarily focused on treatment effect estimation and, notably, uplift modeling. Uplift modeling specifically aims to estimate the incremental impact of an intervention on an individual's behavior – identifying who would be positively influenced by an action (e.g., receiving a promotion) compared to doing nothing. CausalML provides implementations of various uplift algorithms, including tree-based methods (causal trees/forests) and meta-learners similar to those in EconML. It's geared towards practical industry applications, especially in customer relationship management (CRM) and marketing, where optimizing interventions based on predicted individual uplift is a key objective

3.10.4 CausalNex (McKinsey)

CausalNex[4] takes a different approach, focusing more strongly on causal discovery and the use of Bayesian Networks for causal reasoning. It provides tools to learn causal graph structures from data, potentially incorporating domain knowledge to constrain the search space. It implements structure learning algorithms (like NOTEARS) and allows users to fit Bayesian Networks to the data based on the learned (or provided) graph structure. Once the network is built, users can perform queries (e.g., conditional probability queries, interventions via the do-calculus if the graph assumptions hold) to understand relationships and simulate scenarios within the modeled system. CausalNex is particularly useful for exploring and visualizing complex systems where understanding the network of causal influences is a primary goal.

[1]https://github.com/py-why/dowhy
[2]https://github.com/py-why/EconML
[3]https://github.com/uber/causalml
[4]https://github.com/mckinsey/causalnex

4 DeepCausality: A Framework for Hypergeometric Contextual Causal Reasoning

DeepCausality is engineered as a comprehensive, open-source computational causality framework, implemented in Rust, designed to enable the construction, execution, and management of explicit, context-aware causal models. It moves beyond purely statistical correlations by focusing on the representation of underlying causal mechanisms, deeply embedded within rich, multi-dimensional, and dynamic contextual environments. The framework is built upon a philosophy that emphasizes not only the modeling of causal links, but also the explicit articulation and verification of the Observations that inform these links and the Assumptions upon which their validity rests. This principled approach achieves a synthesis of causal understanding through three primary architectural pillars: a sophisticated Context built upon hypergeometric principles to model diverse environmental factors including non-Euclidean relationships; a versatile Causal Modeling Engine that employs structurally composable, recursive CausaloidGraphs to represent complex causal logic derived from rigorous inference; and integrated Causal State Machines that provide a deterministic linkage between identified causal states and actionable interventions. By providing these tools, DeepCausality facilitates the development of more transparent, reliable, and adaptive intelligent systems capable of nuanced reasoning about cause and effect.

4.1 Design Philosophy of DeepCausality

The architecture and implementation of DeepCausality are the result of a deliberate design philosophy refined through practical implementation, aiming to create a system that is expressive, performant, robust, and adaptable for complex causal reasoning tasks. Several core principles underpin the framework:

DeepCausality prioritizes the explicit encoding of known or hypothesized causal mechanisms within Causaloids. Crucially, it also provides first-class support for defining and managing observations from the empirical data and assumptions that encode the conditions under which causal claims are believed to hold true. This emphasis on explicit representation of mechanisms, the data that informs them, and the assumptions that bound them, is fundamental to achieving transparency, explainability, and a systematic approach to model validation and transfer. The operational nature of Causaloids, where each causal link is a testable function, ensures that the logic and its foundational assumptions are inspectable and verifiable through a defined Inference process.

A second core principle is the deep primacy of context. Recognizing that real-world causality is rarely context-free, DeepCausality elevates context from a set of conditioning variables to a rich, structured, and dynamic Context Hypergraph[5]. This allows causal models to be deeply embedded within, and intricately responsive to, their operational environment, supporting multi-dimensional data (Data, Time, Space, SpaceTime), multiple distinct contexts, and both Euclidean and non-Euclidean relational information. This comprehensive contextualization aims for a faithful representation of reality.

The separation of causal logic from dynamic state emerged as a critical architectural decision. Distinguishing between the relatively stable causal mechanisms encoded in the CausaloidGraph[6] and the evolving environmental state (managed by the Context Hypergraph) allows the CausaloidGraph to maintain efficient reasoning performance while reacting to real-time contextual changes. This design facilitates time-synchronous systems capable of operating on multi-scale temporal data.

Modularity and composability are also central. The recursive isomorphic nature of CausaloidGraphs enables hierarchical construction of complex models from simpler, reusable components, supporting an engineering approach to building large-scale causal systems. The trait-based design in the Rust implementation further enhances this, allowing for extension with custom data types and behaviors.

Finally, the choice of Rust as the implementation language reflects a commitment to performance, memory safety, and reliability—guarantees paramount for critical decision-making systems or high-throughput causal analytics. Rust's type system and trait mechanism are extensively leveraged to build an expressive yet efficient engine.

These guiding principles, including the foundational role of explicit observations and verifiable assumptions, collectively aim to provide a framework capable of representing complex causal knowledge in an understandable, maintainable, efficient, and robust manner, particularly for tackling the challenges of real-world dynamic systems and ensuring trustworthy causal inference.

[5] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/context_types/context_graph/mod.rs`

[6] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/causaloid_graph/mod.rs`

4.2 Concepts of DeepCausality

DeepCausality's approach to computational causality is built upon several differentiating conceptual pillars, designed to offer enhanced expressiveness, flexibility, and operational capability compared to traditional methods. These core concepts underpin its architecture and enable its unique approach to modeling complex systems. At the forefront is a hypergeometric conceptualization of both context and causality, where relationships are not limited to pairwise interactions but can involve N-ary connections, represented via hypergraph structures. This allows for a more faithful modeling of group interactions and complex dependencies. Central to this is the Contextoid, an atomic unit capable of representing multidimensional contextual information, Data, Time, Space, and SpaceTime, including explicit support for both Euclidean and non-Euclidean geometries, and the Causaloid, an encapsulated, testable causal function forming the building block of causal models. These Contextoid are organized into recursive isomorphic CausaloidGraphs, permitting complex, layered causal models to be built modularly. The framework also emphasizes operational causality, where causal links are defined by executable functions, and direct intervention capabilities through Causal State Machines. Furthermore, the principle of multiple, adjustable contexts allows models to draw from and adapt to diverse, evolving information sources simultaneously, providing a level of contextual richness crucial for real-world applications. These concepts collectively enable DeepCausality to support the construction of deterministic, explainable, and deeply contextualized causal reasoning systems.

4.2.1 Operational Causality

While DeepCausality is compatible with insights from various causal inference schools, its core "Causaloid" structure is aligned with an operational view of causality, inspired by concepts such as Lucien Hardy's work on operational causality as foundation for quantum gravity [84]. Instead of defining causality purely through statistical dependencies (e.g., conditional independence relations as in Bayesian Networks) or counterfactual contrasts alone, DeepCausality emphasizes the notion of a cause as an identifiable condition or process that, when active or present, leads to a specific effect or triggers a specific output through a defined function.

A causal link in DeepCausality is not merely an observed association but is encoded as a testable, executable function. This causal function takes input(observations, data from context) and evaluates whether a specific, predefined causal relationship holds true. If the conditions for the relationship are met, the function yields a positive result (e.g., true, or an "effect" being present); otherwise, it yields a negative result. This operational definition aligns with the scientific method in which hypotheses (causal links) are tested against evidence.

This perspective moves beyond simply asking "Are A and B related?" to asking "Does the presence of A, under context C, through a defined mechanism F, lead to B?". The mechanism F is explicitly encoded in the causal function. This is crucial because it makes the causal claim transparent, inspectable, and subject to refinement. While Hardy's work applies this to fundamental physics to understand the structure of spacetime events, DeepCausality adapts this operational spirit to model causal relationships in arbitrary complex systems. The "Causaloid" is the fundamental data structure that embodies this operational view of a single causal link.

4.2.2 Four Layers of Causal Reasoning

DeepCausality's architecture conceptually maps to four interconnected layers of reasoning, providing a structured approach from raw data to causal understanding:

1. **Observations**: This layer represents the raw data received from the environment – sensor readings, market prices, user inputs, etc. By themselves, observations are uninterpreted facts that lack intrinsic meaning. The corresponding observation[7] data type in Rust is the starting point for any reasoning process.
2. **Assumptions**: Causal reasoning, unlike some purely data-driven methods, relies on explicit and, importantly, testable assumptions. These assumptions provide the frame of reference for interpreting observations. They might concern the reliability of data sources, the expected range of values, or preconditions that must hold for a causal model to be applicable. Deep causality provides a designated Assumption[8] type to encode and test explicitly stated assumptions.
3. **Inferences**: This layer involves deriving insights or conclusions from observations under the stated assumptions. It's about interpreting the raw data in light of what is assumed to be true. This is where patterns are identified and initial judgments are made. DeepCausality's documentation highlights that even at this stage, one can estimate the ratio of inferable, inverse-inferable (absence of cause associated with absence of effect), and non-inferable observations, which provides early clues about the strength of potential causal links. DeepCausality encapsulates inference in a dedicated Inference[9] type to streamline inferences from observations.
4. Causality: This is the layer in which established causal relations govern subsequent inferences. A causal relation, often expressed as (*IF A THEN E*, *AND IF NOT A THEN NOT E*), is typically established after rigorous testing at the inference layer (e.g., observing a high co-occurrence of A with E, and NOT A with NOT E). Once a causal relationship is considered to hold (e.g., IF pressure $>$ threshold THEN valve_failure_imminent), it becomes a rule that dictates the results for future matching observations.

DeepCausality's core types (Observation, Assumption, Inference, Causaloid) and their associated reasoning traits directly map to these conceptual layers, providing a structured way to build causal models.

4.2.3 Hypergeometric Representation

The term "hypergeometric" in DeepCausality refers to its reliance on graph-theoretic structures, specifically hypergraphs, to represent both context and causal relationships. This is a deliberate design choice that offers several advantages over purely algebraic or flat relational models.

Simplifying Complexity: Modeling complex causality can lead to either complex algebra with simple structures or simple arithmetic with complex structures. DeepCausality opts for the latter, believing that structural complexity is more manageable and scalable through computational graph techniques than highly intricate algebraic formulations, especially when dealing with hundreds or thousands of interacting causal factors.

Explicit Relationships: Hypergraphs allow for the explicit representation of N-ary relationships, where an edge (hyperedge) can connect any number of nodes. This is more expressive than simple graphs for modeling complex dependencies, such as multiple conditions leading to a single effect, or a single cause influencing multiple effects through different pathways.

Recursive Composability: A key feature is that nodes within these hypergraphs can themselves be entire sub-graphs. A CausaloidGraph can have nodes that are individual Causaloids or other CausaloidGraphs. This recursive isomorphism allows for hierarchical and modular construction of highly complex causal models, breaking them down into manageable, understandable, and reusable components.

Identifiable Reasoning Paths: Reasoning over a graph structure allows for the tracing of "explanation paths" – the sequence of activated causaloids and contextual data points that led to a particular conclusion. This is fundamental to DeepCausality's end-to-end explainability.

By representing both context and causal models as hypergraphs, DeepCausality creates a unified structural approach for describing and reasoning about complex, interacting systems.

[7] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/observation/mod.rs`

[8] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/assumption/mod.rs`

[9] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/inference/mod.rs`

4.3 Context

The context is a foundational element that enables causal models to be grounded in the specifics of a given situation and to handle diverse, dynamic, and multi-faceted contextual information. It allows DeepCausality to move beyond simplistic causal models towards systems that can understand and react to the nuanced, multi-dimensional, and dynamic nature of the environments in which they operate. It is the foundation upon which deeply contextualized and therefore more accurate and relevant causal inferences can be built. The core elements that comprise context in DeepCausality:

Contextoids:

The primary building block for context is the Contextoid. Each Contextoid is a node in a context hypergraph and encapsulates a single unit of contextual information. DeepCausality defines four primary types of contextoids:

1. **Data Contextoid**: Stores arbitrary data of a generic type T. This can be a raw sensor reading, a calculated metric, a textual feature, or any other piece of information relevant to the causal model.
2. **Time Contextoid**: Represents a unit of time, specified by a TimeScale (e.g., Year, Month, Day, Nanosecond) and a time value of type T (typically an unsigned integer).
3. **Space Contextoid**: Represents a unit of space, potentially with up to three coordinates (X, Y, Z) of type T. This does not inherently carry unit information (e.g., meters vs. feet), which must be managed by the application logic.
4. **SpaceTime Contextoid:** Combines spatial and temporal information, representing a unit of space at a specific unit of time. For example, in a drone application, a SpaceTime contextoid could represent the drone's GPS coordinates (lat, lon, alt) at timestamp in nanosecond resolution. Multiple sensor readings (Data contextoids) could then be linked to this single SpaceTime contextoid, indicating they were all taken at the same place and time

The Context Hypergraph:

Individual contextoids are organized into a Context object, which is internally represented as a hypergraph. Nodes are contextoids, and hyperedges represent relationships between them. This structure allows for flexible and complex contextual models because real-world causality rarely occurs in a vacuum; it is almost always conditioned by, and interacts with, a complex web of surrounding circumstances, historical states, and environmental factors. The Context Hypergraph provides the means to explicitly model this intricate web, making it an active and integral part of the causal reasoning process itself.

The choice of a hypergraph over a simple graph or other data structures is deliberate and offers significant advantages for representing complex contextual relationships:

1. **Representing N-ary Relationships:** Unlike simple graphs where an edge connects only two nodes, a hyperedge in a hypergraph can connect an arbitrary number of nodes. This is crucial for context because many contextual situations involve relationships between more than two entities.
2. **Flexibility and Expressiveness:** Hypergraphs are highly flexible and can represent a wide variety of relational structures, including simple graphs, hierarchical structures, and overlapping sets, all within a unified framework. This allows the Context Hypergraph to adapt to the diverse ways contextual information is structured in different domains.
3. **Modularity and Composition:** Portions of a context hypergraph can represent distinct but related contextual facets (e.g., a sub-hypergraph for weather conditions, another for economic indicators). These can be reasoned over separately or their interrelations can be explicitly modeled with hyperedges connecting nodes across these sub-graphs.

4.3.1 Adjustable Protocol:

A core challenge in modeling real-world systems with causal inference is the inherent dynamism of their operational environments; contextual factors are rarely static, as sensor readings drift, market conditions fluctuate, and environmental parameters change. To ensure that causal models remain relevant and accurate over time, the contextual information they rely upon must possess a degree of plasticity. DeepCausality addresses this necessity through its Adjustable protocol, as detailed in its conceptual[10] and architectural[11] documentation. This protocol is a mechanism designed to provide a uniform and controlled way of modifying the internal state of Contextoids within the Context Hypergraph, even within structurally static contexts.

The Adjustable protocol is conceptualized as an optional extension to the fundamental Contextoid types, namely Data, Time, Space, and SpaceTime. By default, Contextoids are immutable post-instantiation, a design choice that promotes data integrity and predictability, particularly in certain audited or foundational contextual layers "Non-adjustable means that, after instantiation, the structure is immutable..."[12]. However, when contextual elements must reflect ongoing changes, their corresponding Contextoid types can implement the Adjustable trait. This trait defines a specific interface for sanctioned mutations, thereby allowing the values encapsulated by a Contextoid to be altered without changing the Contextoid's identity—its unique ID within the hypergraph—or its established relational linkages, which are the hyperedges.

The Adjustable protocol primarily exposes two optional methods that implementing types can choose to override. The first method, $update(new_value)$, is conceptualized as the mechanism for replacing the existing value within a Contextoid with entirely new data. This is typically invoked when fresh information becomes available from an external source or as the result of an internal calculation. For instance, a Data Contextoid representing the "latest stock price" would have its update() method called each time a new price tick is received from a market data feed. Similarly, a SpaceTime Contextoid tracking a moving object would be updated with new coordinates and a new timestamp as the object's position changes. The crucial aspect is that the update() method signifies a complete replacement, reflecting the most current state of that particular contextual element.

In contrast, the second method, $adjust(adjustment_factor)$, is designed for corrections, relative modifications, or incremental changes to the existing value within a Contextoid. This is particularly useful when the current value needs to be refined based on new information, a known bias, or an influencing factor, rather than being wholly superseded. For example, a sensor reading encapsulated in a Data Contextoid might be subject to a known calibration drift over time; an adjust() method could therefore apply a correction factor to mitigate this drift. In more complex scenarios, such as the spatio-temporal adjustments required for gravitational effects mentioned in the documentation[13], the adjust() method might take a more complex $adjustment_factor$, for instance, an ArrayGrid[14] representing a transformation matrix, to modify the internal coordinates or temporal values of AdjustableSpaceTime Contextoids.

The protocol's design intentionally makes these methods optional by providing empty default implementations. This allows implementers to provide only the functionality that is strictly required by their specific Contextoid type. If a Contextoid only ever needs to be fully updated with new values, only the update() method needs to be implemented, thus promoting minimalism and avoiding unnecessary code.

The significance of the Adjustable protocol for handling evolving environments is multi-fold. Firstly, it provides a uniform interface for mutation across different types of Contextoids—be they standard (Data, Time, Space, SpaceTime) or custom user-defined contextoid types—so long as they implement the protocol. This uniformity simplifies the design of higher-level context management systems that need to interact with a heterogeneous collection of contextual elements. Secondly, it facilitates a decoupling of the update logic from the core context structure. DeepCausality deliberately does not provide a global context update mechanism by default. Instead, it provides the means for updates via the Adjustable protocol and wisely leaves the strategy of when and how to apply these updates to the application developer. This is a critical design decision, recognizing that the correct order and timing of updates can be highly domain-specific and are crucial for maintaining consistency, especially when dealing with multiple interacting data streams or inter-dependent contextual variables.

Thirdly, the Adjustable protocol is essential for facilitating real-time responsiveness. By allowing Contextoids to be updated or adjusted efficiently, causal models that query the Context Hypergraph can access the most current state of the environment. This is fundamental for real-time decision-making systems that must react promptly to emergent changes.

[10]https://deepcausality.com/docs/concepts/
[11]https://deepcausality.com/docs/architecture/
[12]https://deepcausality.com/docs/concepts/
[13]https://deepcausality.com/docs/concepts/
[14]https://deepcausality.com/blog/the-grid-type/

Finally, the protocol-based approach is inherently future-proof and extensible. If new types of mutations or contextual operations become necessary, such as a hypothetical delete() operation for contextual data, the Adjustable protocol could be extended with new method signatures and empty default implementations. Custom Contextoid types implementing these new methods would then remain compatible with the overall context management framework, ensuring long-term adaptability of the system.

The Adjustable protocol endows DeepCausality's Context Hypergraph with the necessary plasticity to accurately mirror evolving realities. It allows contextual information to be a living representation of the environment, continuously refined and updated, thereby ensuring that the causal reasoning performed upon it remains grounded, relevant, and capable of driving adaptive behavior in dynamic systems. This capacity for controlled mutability within the contextual layer is a key enabler for building causal models that are not just static representations of knowledge but active participants in understanding and responding to a continuously changing world.

4.3.2 Multiple Contexts

A significant advancement in DeepCausality, introduced from version 0.6 onwards is its architectural capacity to allow causal models to utilize multiple distinct contexts simultaneously. This capability transcends the already powerful notion of a single, rich Context Hypergraph by enabling a system to maintain and draw upon several independent or interrelated contextual realms. This feature is a significant enhancement to the framework's representational power, crucial for modeling systems where causality is influenced by diverse, potentially disjoint, yet concurrently relevant sets of information.

Prior to this enhancement, while a single Context hypergraph could be internally complex and multi-faceted, all contextual information accessible to a given causal model had to reside within that unitary structure. The introduction of multiple contexts allows a causal model, specifically the causal function within a Causaloid, to be endowed with the ability to reference and query data from several different contexts, each with its own unique ID, structure, and set of Contextoids. This fundamentally alters the landscape for modeling intricate real-world scenarios where different types of contextual information might be best organized, updated, or conceptualized separately, yet all contribute to a holistic causal understanding.

The practical implementation[15] of this feature involves an Application Programming Interface (API) where additional contexts can be created and assigned unique identifiers. A causal model can then be designed to specifically target these different context IDs when it needs to fetch particular pieces of contextual data. This means a single `causal_function` might, for instance, retrieve temporal information from `Context_ID_1` (perhaps a master timeline), spatial data from `Context_ID_2` (a geographical information system context), and specific sensor readings from `Context_ID_3` (a real-time data stream context), all within the course of evaluating a single causal hypothesis.

The importance of managing multiple distinct contexts is best understood through concrete application domains where such separation and selective integration are natural and powerful. The documentation[16] allude to several such scenarios. Consider the financial industry, particularly in the modeling of synthetic instruments like future spreads. A spread's behavior is intrinsically tied to the behavior of its constituent legs (e.g., a long-term future contract and a short-term future contract) as well as the characteristics of the spread itself. With multiple contexts, a DeepCausality system can maintain `Context_LongLeg` containing all relevant data for the long-term contract (its price series, volatility, specific news events impacting it), a `Context_ShortLeg` for the short-term contract, and a `Context_Spread` containing derived data like the price differential, historical spread volatility, and perhaps arbitrage opportunity indicators. A causal model designed to predict spread behavior or identify trading opportunities can then intelligently query all three contexts, drawing precisely the information needed from each distinct source to make a robust inference. Attempting to meld these fundamentally different, though related, information sets into a single, monolithic context graph could lead to an overly complex and less maintainable structure.

Similarly, in the Internet of Things (IoT) domain, an advanced industrial monitoring system might involve sensor networks deployed across different physical plants or monitoring entirely different types of processes. One Context could represent the environmental sensor data (temperature, humidity) from Plant A, while another Context represents the operational parameters (machine RPM, energy consumption) from specialized machinery in Plant B, and yet another Context might track overall supply chain logistics data relevant to both. A higher-level causal model assessing overall production risk or efficiency could then tap into these distinct contextual sources. Each context can be updated independently based on its own data streams and lifecycle, yet their information can be synergistically combined at the point of causal reasoning.

[15] https://deepcausality.com/blog/announcement-multiple-contexts/

[16] https://deepcausality.com/blog/announcement-multiple-contexts/

This multi-context architecture also enhances modularity and scalability. Different teams or modules within a larger system could be responsible for maintaining and updating specific contexts relevant to their domain of expertise. For example, a meteorological team might maintain a detailed `WeatherContext`, while an economic analysis team maintains an `EconomicIndicatorContext`. A causal model predicting agricultural output could then be configured to utilize both, without either team needing to understand the full intricacies of the other's contextual data, as long as clear interfaces for querying relevant `Contextoids` are defined.

The crucial design element enabling this is that the causal function within a Causaloid receives an immutable reference not just to *a* context, but to an overarching context management system that can resolve queries to *specific, identified* contexts. The API allows for setting a "current" additional context for operations, and for checking the existence and retrieving data from any named or ID-referenced context[17]. This ensures that while the causal logic can be complex and draw from diverse sources, the process remains explicit and manageable.

The support for multiple distinct contexts elevates DeepCausality from a framework for reasoning within a single rich world-model to one capable of reasoning across a federation of specialized world-models. This mirrors how complex human reasoning often draws upon distinct but relevant bodies of knowledge. It is a sophisticated feature that significantly extends the framework's capacity to model the intricate, multi-faceted causal tapestries found in complex systems, providing a more organized, maintainable, and expressive foundation for advanced causal inference.

4.3.3 Euclidean and Non-Euclidean Contexts:

A particular aspect of DeepCausality's architecture is its sophisticated handling of the geometric nature of contextual information, explicitly accommodating both Euclidean and Non-Euclidean Contexts. This dual capability significantly broadens the framework's applicability. It allows DeepCausality to model causality in domains where relationships are defined by abstract connectivity or conceptual similarity rather than solely by physical proximity in a metric space.

Traditionally, spatial context in computational models often defaults to a Euclidean interpretation, wherein entities are located by coordinates in a one, two, or three-dimensional space, and relationships such as distance are defined by standard metrics. DeepCausality provides robust support for such conventional representations. Its default `Space` Contextoid, for instance, can store up to three coordinates (X, Y, Z) of a generic type `T`, and the `SpaceTime` Contextoid naturally extends this to incorporate a temporal dimension. This approach is perfectly suitable for a vast range of applications, such as modeling the physical layout of a sensor network, the trajectory of an autonomous drone, or the geographical distribution of environmental factors. In these scenarios, causal influences might indeed propagate based on physical distance or containment within defined spatial regions.

However, the profound innovation within DeepCausality lies in its explicit recognition that many crucial contextual relationships are not inherently Euclidean. Consider social networks, where influence propagates along ties of friendship or collaboration, not necessarily physical nearness. Think of biological systems, where proteins interact based on specific binding sites and pathway memberships, forming a complex interaction network. Forcing such inherently non-Euclidean relationships into an ill-fitting Euclidean vector space can lead to a significant loss of fidelity, the introduction of artificial constraints, or an obscuring of the true underlying causal pathways, a concern echoed by J.M. Bishop [33] regarding the limitations of Euclidean representations in Artificial Intelligence.

DeepCausality's architecture addresses this challenge through its flexible, trait-based design, particularly the `Spatial<V>` trait. This trait defines a protocol—a set of methods that a type must implement to be considered "spatial" within the framework—but it crucially *does not mandate a Euclidean interpretation* for how those methods are implemented or what constitutes a "spatial" relationship. While the default `Space` structure provides a convenient Euclidean implementation, users are empowered to create custom types that implement the `Spatial<V>` protocol to embody entirely non-Euclidean or abstract notions of "space" and "relationship."

For instance, a custom type named `SocialGraphNodeContext` could implement `Spatial<UserID>`, where methods like `get_neighbors()` would return directly connected users in a social network graph, and a "distance" metric, if defined, might represent degrees of separation. Similarly, a `CorrelationClusterContext` could implement `Spatial<AssetID>`, where its methods define adjacency or proximity based on assets exceeding a certain correlation threshold or belonging to the same statistically derived cluster. Another example might involve a `SectorGraphContext` implementing `Spatial<NodeId>` for an industry classification graph, where relationships are defined by graph connectivity. The core DeepCausality reasoning engine, when interacting with context via the CausaloidGraph, remains agnostic to the specific underlying geometry. It requests spatial information or evaluates relationships through the standardized `Spatial<V>` trait methods; the concrete implementation provided by the custom Contextoid type—whether it is based on Cartesian coordinates, graph adjacencies or conceptual hierarchy links—determines the nature of the "space" through which causal effects are considered to propagate.

[17]https://deepcausality.com/blog/announcement-multiple-contexts/

The significance of this design lies in its ability to model causality directly within the most relevant relational structure of the domain under study. It obviates the need to artificially project complex, abstract dependencies onto a potentially unsuitable geometric substrate. Instead, the domain's natural topology, be it a network, a feature space manifold, a categorical hierarchy, or a conceptual map, can be directly represented and reasoned over. This allows the causal functions within Contextoid to operate on concepts like "influence," "similarity," "membership," or "connectivity" as primary contextual factors, leading to more faithful, insightful, and potent causal models.

Furthermore, the uniformity provided by the trait system means that these diverse spatial and relational conceptualizations can coexist and interact within a single, comprehensive DeepCausality application. A causal model might simultaneously draw upon a Euclidean `SpaceTime` context representing the physical location of an event and a non-Euclidean `SocialInfluenceContext` representing the network of actors involved. This capacity to fluidly integrate and reason across different conceptualizations of space and relationship, all grounded within a rigorous causal framework, unlocks a profound potential for understanding and modeling systems based on new and unique contextual capabilities.

4.3.4 Contextual capabilities:

In DeepCausality, context intrinsically becomes an active and integral part of the causal reasoning process itself and enables variety of capabilities.

Enables Deep Contextualization: The primary importance is that it allows causal models (Causaloids) to be deeply and specifically contextualized. Instead of a causal rule operating on global variables or a simple set of parameters, it can query a rich, structured context for precisely the information it needs, relevant to the current observation's time, location, or other defining characteristics. For example, a causal rule for assessing crop yield might perform very differently in "Drought" conditions versus "AdequateRainfall" conditions. The Context Hypergraph can explicitly represent these conditions (perhaps as Data or Symbolic Contextoids), and the Causaloid can query the current weather context before making its inference.

Grounds Abstract Causal Rules in Concrete Reality: Abstract causal relationships ("IF A THEN B") gain practical meaning and applicability only when A and B can be mapped to specific, observable, and contextually relevant phenomena. The Context Hypergraph serves as this mapping layer, providing the concrete instantiations of variables and conditions that the causal rules operate upon.

Supports Reasoning Across Multiple Dimensions and Resolutions: Because Contextoids can represent Data, Time, Space, and SpaceTime, the Context Hypergraph allows causal models to reason about phenomena that unfold across these dimensions. For example, a model for urban traffic flow might consider current traffic density (Data), time of day (Time), specific road segments (Space), and sensor readings from particular intersections at particular times (SpaceTime).

Facilitates Dynamic and Adaptive Reasoning: With Adjustable Contextoids, the Context Hypergraph is not static; it can evolve in real-time as new information arrives or as the environment changes. Causal models querying this dynamic graph will therefore adapt their reasoning to the most current state of the world.

Manages Complexity in Multi-Factor Systems: Many real-world systems are influenced by a multitude of interacting factors. The Context Hypergraph provides a way to organize this complexity, making it explicit and accessible to the causal model. This prevents causal rules from becoming overly complex by having to manage all contextual variations internally; instead, they can delegate contextual queries to the specialized context.

Enables Non-Euclidean Relational Context: As stressed previously, the ability of the context (and its underlying Spatial<V> trait) to represent non-Euclidean relationships is a profound advantage. This means the "space" in which causes operate can be a social network, a conceptual hierarchy, a correlation matrix, etc. The Context Hypergraph provides the structure to define these abstract "contextual landscapes."

4.4 Causality

While the Context Engine provides the rich, structured environmental grounding, the core causal reasoning within DeepCausality is performed by its versatile Causal Modeling Engine. This engine is designed to move beyond purely statistical or algebraic formulations of causality, opting instead for a structural and operational conceptualization. It facilitates the construction of explicit causal models through composable units of causal logic that operate within the rich environments provided by the Context Engine. Key to this engine are the concepts of the Causaloid, the mechanism of contextual causal reasoning, and the powerful paradigm of recursive isomorphic causal data structures embodied in the CausaloidGraph.

4.4.1 The Causaloid: A Unit of Causation

The fundamental building block for expressing causal relationships within DeepCausality is the Causaloid, a concept inspired by Lucien Hardy's work in operational physics aimed at formalizing cause-and-effect in physical systems [84, 107]. In DeepCausality, a Causaloid is adapted to serve as an encapsulated, operational unit that encodes a single, testable causal hypothesis or a specific causal mechanism. As described in section 4.2, each Causaloid is defined by a **causal function** (f_χ in our formalism, Section 4.6). This function takes an input observation (e.g., data representing a potential cause or a system state) and, critically, can receive an immutable reference to one or more Context hypergraphs. The causal function then executes specific logic to determine whether, given the observation and the relevant contextual information, the posited causal relationship holds or the cause is considered active. Typically, this evaluation results in a Boolean outcome (true/active or false/inactive), thereby providing a deterministic assessment of the individual causal link.

This operational definition is pivotal. Instead of representing a cause as a mere statistical dependency or an abstract node in a graph whose meaning is externally imposed, the Causaloid internalizes the logic of the causal link itself. This makes the causal assertion explicit, inspectable, and directly testable. For instance, a Causaloid designed to detect a 'golden cross' in financial markets would encapsulate a function that accesses specific moving average values (from `Data Contextoids`) linked to relevant time points (from `Time Contextoids`) within a financial market Context, and then applies the mathematical criteria for a golden cross. This explicit encoding of the mechanism within the Causaloid's function is a core tenet that supports the framework's emphasis on explainability.

4.4.2 Contextual Causal Reasoning

A defining feature of DeepCausality, setting it apart from many traditional causal modeling approaches, is its deep and intrinsic integration of context into the reasoning process [107]. Contemporary computational causality often assumes that the modeled causal relations constitute the entirety of the model, or treats context as simple conditioning variables. DeepCausality, however, elevates context from a passive backdrop to an active, structured entity—the Context Hypergraph as detailed in Section 4.3—that actively participates in the evaluation of causal relationships.

As outlined in section 4.2 is achieved by providing the causal function within each Causaloid with an immutable reference to the relevant Context instance(s) (see C_{refs} in the formalism, Section 4.6). This allows the causal logic to be highly adaptive and specific to the circumstances. The causal function can dynamically query the Context Hypergraph for any Contextoid (Data, Time, Space, SpaceTime) or relational information it requires to make its determination. For example, a causal model for a physical system might have Causaloids whose functions query SpaceTime Contextoids for location and time, Data Contextoids for ambient temperature or pressure from the same spatio-temporal vicinity, and then evaluate a physical law based on these contextualized inputs.

The framework's support for multiple contexts, as discussed in Section 4.3, further enriches this capability. A single Causaloid can be designed to draw information from several distinct Context Hypergraphs, each representing different facets of the environment or different sources of information (e.g., one context for market data, another for news sentiment, a third for macroeconomic indicators). This allows for the construction of models that reason holistically across diverse, yet relevant, information domains. Diagram 2 labeled "Contextual Causal Model" illustrates how specific Causaloids within a larger model can be contextualized by accessing data from a shared Context, while others might remain context-free if their logic is purely observational or self-contained. This selective contextualization provides fine-grained control over how environmental factors influence specific causal links within a broader model.

4.4.3 Recursive Isomorphic Causal Data Structures: The CausaloidGraph

To scale from simple causal links to arbitrarily complex causal systems, DeepCausality employs a powerful structural paradigm: recursive isomorphic causal data structures, primarily realized through the CausaloidGraph[18]. While individual Causaloids represent atomic causal hypotheses, the CausaloidGraph allows these units to be organized into intricate networks that model complex causal interdependencies (formalized in Section 4.6.3).

A CausaloidGraph is itself a hypergraph where each node represents a causal element. The "recursive isomorphic" nature, a key innovation visually depicted in Figure 1, means that a node (v_g) within a CausaloidGraph (G) is not restricted to being only a simple, elementary Causaloid. Instead, the payload of such a node (payload_g) can encapsulate:

1. A single, elementary `Causaloid` (χ).
2. A collection of Causaloids (e.g., implemented as a ‘CausalArray‘, ‘CausalHashMap‘, ‘CausalVector‘, or ‘CausalVecDeque‘ where the collection itself might have an aggregate evaluation logic (e.g., "activate if any causaloid in this collection is active").
3. Another entire CausaloidGraph (G'), effectively creating a sub-model or a nested causal structure.

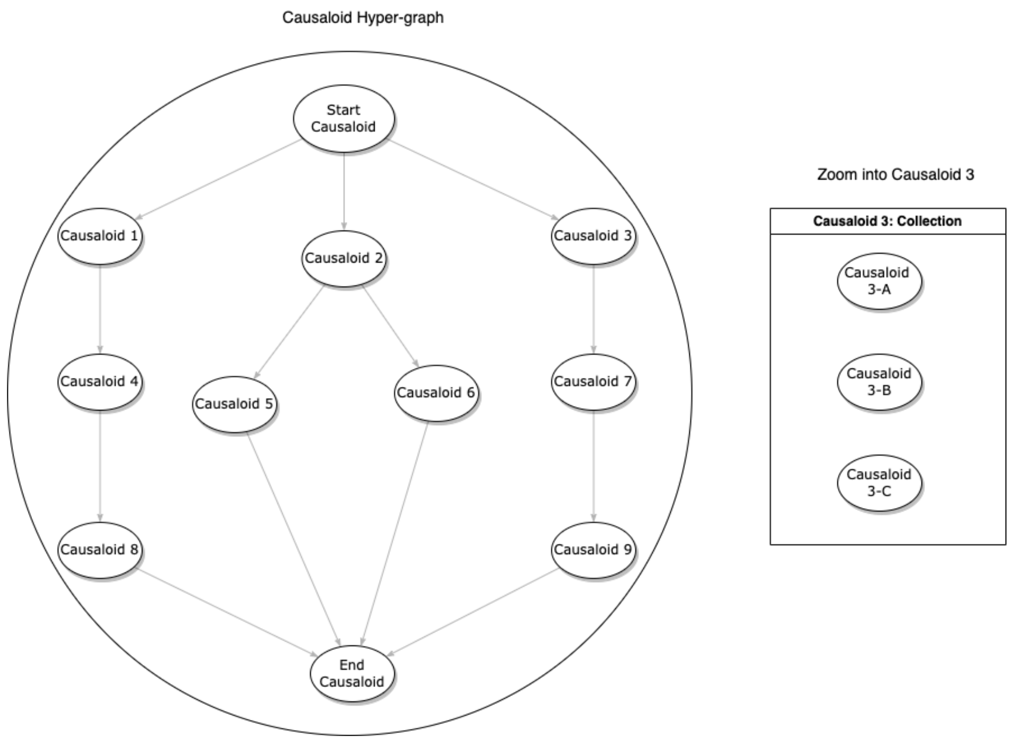

Figure 1: The Causaloid Diagram illustrating the recursive isomorphic structure of a CausaloidGraph. Nodes within the graph can be elementary Causaloids, collections of Causaloids, or even encapsulate entire sub-`CausaloidGraphs`, enabling hierarchical and modular causal model construction.

This recursive definition, as illustrated in Figure 1, allows for a highly modular and hierarchical approach to building complex causal models. Macro-level causal phenomena can be represented by high-level nodes in a graph, and these nodes can then be decomposed into more detailed sub-graphs representing the interacting micro-level causal mechanisms that constitute the higher-level effect. This "transparent composability" [19] means that a complex system can be broken down into understandable, manageable, and potentially reusable causal modules.

[18]https://deepcausality.com/docs/concepts
[19]https://deepcausality.com/docs/concepts

Reasoning over a CausaloidGraph involves evaluating its constituent causal elements and propagating their activation states according to the (hyper)edges (E_G) that define their logical interdependencies (logic_e). DeepCausality supports various reasoning modes over the graph, such as reasoning over the entire graph, a specific Causaloid node, a defined sub-graph between start and stop nodes, or along the shortest path between two Causaloids. The activation state of a Causaloid (once active, it typically remains so until negatively re-evaluated based on new inputs) allows the graph to maintain a memory of which causal conditions have been met, facilitating reasoning over evolving states based on changing input data and context.

Contextualized causal reasoning is implemented by passing an immutable reference implicitly into the causal function. This mechanism also allows us to contextualize certain Causaloids while leaving others context-free if this is a desirable option. The diagram below, Figure 2, shows this scenario in which specific Causaloids use data from a Context while other Causaloids remain context-free.

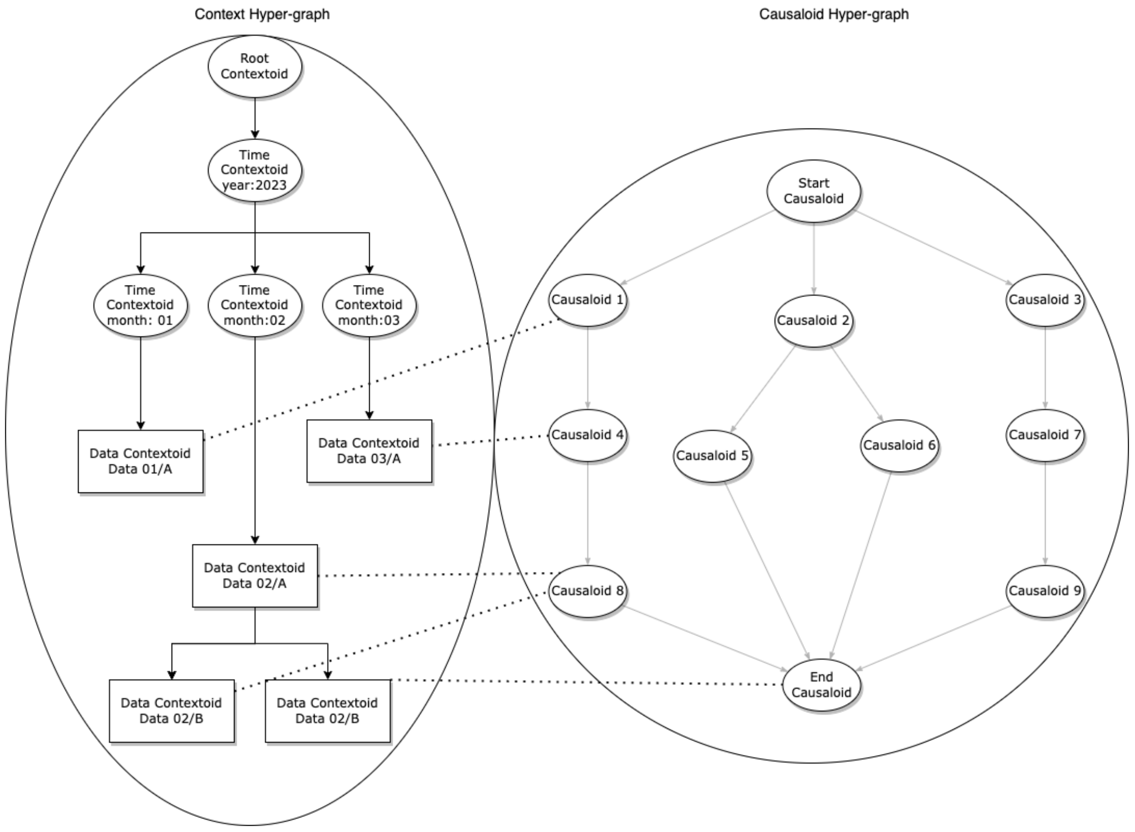

Figure 2: Illustration of a Contextual Causal Model in DeepCausality. Specific `Causaloids` are shown interacting with an external `Context` drawing contextual data to inform their evaluation. Other `Causaloids` operate without direct contextual input, demonstrating the flexibility of selective contextualization.

This structural approach, combining operational Causaloids with deeply contextualized reasoning and recursively composable CausaloidGraphs, provides DeepCausality with a robust and expressive engine for modeling and executing complex causal knowledge.

4.5 Causal State Machine

A crucial further step for many practical causality based applications is the translation of these causal insights into tangible actions or interventions. The **Causal State Machine (CSM)** is the component within DeepCausality specifically engineered to bridge this gap, enabling a direct and deterministic linkage from identified causal conditions to predefined operational responses. This facilitates the creation of dynamic control, supervision, and autonomous decision-making systems grounded in explicit causal logic.

The conventional view of causal models often separates the modeling process from subsequent intervention, primarily for reasons of flexibility and analytical clarity. While this separation remains valid and useful for many analytical or exploratory use cases, it can be suboptimal for systems requiring tight, deterministic coupling between causal understanding and immediate action, such as in dynamic control systems or automated supervision. Traditional finite state machines (FSMs), often employed in control systems, typically require all possible system states to be known upfront. However, with the advent of cloud-native applications and other dynamically evolving systems, such as those involving programmatically managed software services or parametric causal models that are brought online and offline dynamically, the complete set of system states may not be known at design time. This limitation restricts the applicability of conventional FSMs.

DeepCausality's CSM offers a more flexible and powerful paradigm. It generalizes the concept of a state machine by defining states not merely as arbitrary system configurations, but specifically as *identified causes* or *combinations of causes* becoming active within one or more supervised CausaloidGraphs. When a particular causal state is recognized through the evaluation of the associated causal model(s) against current observations and context, the CSM is designed to trigger a specific, predefined action or intervention that leads to a desired effect or system adjustment[20]. The core architecture of a CSM in DeepCausality involves this explicit mapping:

- **Causal States**: A state within the CSM corresponds to a specific pattern of activation across one or more Causaloids in the supervised CausaloidGraph(s). This could be a single Causaloid becoming active (e.g., "Sensor_Pressure_Exceeds_Threshold") or a more complex logical combination of multiple Causaloid activations (e.g., "(Smoke_Detected AND Temperature_Rising) OR Emergency_Override_Signal_Active").
- **Deterministic Actions**: Each defined causal state is linked to a deterministic action. These actions are typically implemented as regular Rust functions, which allows for arbitrary complexity. An action could range from simple logging or alerting, to modifying the system's operational parameters, adjusting context variables for downstream models, triggering external API calls, or initiating complex control sequences.

A key characteristic of the CSM is its potential for dynamic configuration[21]. Unlike traditional FSMs, a DeepCausality CSM does not necessarily require all possible causal states and their corresponding actions to be enumerated at the initial design phase. Instead, as new components, systems, or causal hypotheses (represented by new Causaloids or CausaloidGraphs) are introduced or come online, their relevant causal states and associated actions can be dynamically registered with the supervising CSM. For instance, if a system being monitored (e.g., an industrial plant) has new sensors added, the causal rules pertaining to these sensors and the actions to be taken upon their triggering specific conditions can be seamlessly integrated into the existing CSM structure. This allows the CSM to adapt and extend its supervisory capabilities as the system it monitors evolves.

The CSM, therefore, acts as an orchestrator, as formalized in Section 4.6.4. It can manage the flow of information by triggering data extraction from relevant Contextoids, ensuring these updates inform the supervised causal models, and then, based on the collective inference drawn (the active causal states), it initiates precise actions. This operational loop facilitates the development of systems that are not only reactive to their environment but can also be proactive if the causal models have predictive capabilities regarding future states. The explicit link between a well-understood causal condition and a defined action enhances the overall system's explainability and trustworthiness; one can clearly trace why a particular action was taken by examining the active causal state that triggered it and the underlying causal logic and contextual data that led to that state's activation. This contrasts significantly with control systems based on opaque machine learning models where the rationale for an action might be obscure.

In summary, the Causal State Machine is a critical component that elevates DeepCausality from a purely analytical framework to one capable of driving direct, reasoned intervention. It provides the necessary mechanisms for building dynamic control and supervision systems that operate on a foundation of explicit, context-aware causal understanding, enabling more robust, adaptable, and explainable automation.

[20] https://deepcausality.com/docs/csm
[21] https://deepcausality.com/docs/intro

4.6 Formal Specification of DeepCausality

To provide a rigorous underpinning for the conceptual framework described, this section introduces a formal mathematical specification for the core components of DeepCausality, including foundational elements for empirical grounding and assumption management, followed by Context, then Causality types, and the Causal State Machine.

4.6.1 Foundational Elements: Observation, Assumption, and Inference

At the base of empirical causal reasoning within DeepCausality are Observations, the Assumptions under which they are interpreted, and the Inference process that links them to causal claims.

An **Observation Instance**, denoted o_{data}, represents a single empirical data point or a collection of related measurements. It is defined primarily by its content:

$$o_{data} = (id_{obs}, \text{val}_{obs}, \text{eff}_{obs})$$

where:

- $id_{obs} \in \mathbb{I}$ is a unique identifier for the observation, where $\mathbb{I}$ is a suitable set of identifiers.
- $\text{val}_{obs} \in$ NumericalValue (or a more general data type $\mathcal{V}_{obs}$) is the primary measured value associated with a potential causal factor.
- $\text{eff}_{obs} \in$ NumericalValue (or a Boolean/categorical type $\mathcal{V}_{eff}$) is the observed outcome or effect associated with this observation.

A dataset typically consists of a set of such observations $D_{obs} = \{o_{data,1}, \ldots, o_{data,N}\}$. The Observable trait ensures access to these components.

An **Assumption Instance**, denoted A_{smp}, articulates a condition believed to hold for a given analysis or context, under which causal interpretations are made. It is defined as:

$$A_{smp} = (id_{asmp}, \text{desc}_{asmp}, f_{asmp}, \text{status}_{asmp})$$

where:

- $id_{asmp} \in \mathbb{I}$ is a unique identifier for the assumption.
- desc_{asmp} is a textual description (e.g., String).
- f_{asmp} is the **assumption evaluation function** (EvalFn):

  $$f_{asmp} : \mathcal{P}(D_{obs}) \times \text{ContextAccessor}(\mathcal{C}_{\text{relevant}}) \rightarrow \{\text{true}, \text{false}\}$$

  This function takes relevant observational data (e.g., a subset of D_{obs}, denoted $\mathcal{P}(D_{obs})$ as the power set or a specific subset type) and potentially contextual information from relevant contexts $\mathcal{C}_{\text{relevant}} \subseteq \mathcal{C}_{sys}$, and returns whether the assumption is considered to hold.
- $\text{status}_{asmp} = (\text{is\_tested}, \text{is\_valid})$ are Boolean flags indicating the verification status of the assumption.

The Assumable trait mandates methods to access these components and to verify the assumption (which executes f_{asmp} and updates status_{asmp}).

An **Inference Instance**, denoted I_{inf}, represents a tested hypothesis about a potential causal link, derived from observations under specified assumptions. It is defined as:

$$I_{inf} = (id_{inf}, \text{question}_{inf}, \text{strength}_{obs}, \text{threshold}_{inf}, \text{effect\_val}_{inf}, \text{target\_val}_{inf}, \text{status}_{inf})$$

where:

- $id_{inf} \in \mathbb{I}$ is a unique identifier.
- question_{inf} is a textual description of the inferential question being posed.
- $\text{strength}_{obs} \in [0, 1]$ is a numerical value representing the observed strength or probability of the hypothesized relationship, computed from D_{obs} (e.g., derived from $P(E|A)$ and $P(\neg E|\neg A)$ after applying ObservableReasoning).
- $\text{threshold}_{inf} \in [0, 1]$ is a predefined threshold against which strength_{obs} is compared.
- effect_val_{inf} and target_val_{inf} are values (of type NumericalValue or similar) used to define the conditions for a positive inference.
- $\text{status}_{inf} = (\text{is\_inferable}, \text{is\_inverse\_inferable})$ are Boolean flags. The Conjoint Delta, $\Delta_{CJ}(I_{inf}) = |1 - \text{strength}_{obs}|$, is implicitly defined.

The Inferable trait provides methods to access these components and determine status_{inf}. These Inference objects, when validated against Assumptions, form the basis for encoding Causaloids.

4.6.2 Context Formalism

A **System Context Capability** is defined as a finite set of potentially distinct Context Hypergraphs:

$$\mathcal{C}_{sys} = \{C_1, C_2, \ldots, C_k\}$$

where each C_i is an individual **Context Hypergraph**.

An individual **Context Hypergraph** C is defined as a tuple:

$$C = (V_C, E_C, ID_C, \text{Name}_C)$$

where:

- $ID_C \in \mathbb{N}$ is a unique identifier for this context within $\mathcal{C}_{sys}$.
- Name_C is a descriptive name (e.g., String).
- V_C is a finite set of **Contextoid** nodes.
- E_C is a finite set of **Hyperedges**.

A **Contextoid** $v \in V_C$ is defined as a tuple:

$$v = (id_v, \text{payload}_v, \text{adj}_v)$$

where:

- $id_v \in \mathbb{I}$ is a unique identifier for the contextoid within C.
- payload_v is a tagged union:

  $$\text{payload}_v \in \{\text{Data}(d) \mid d \in \mathcal{D}_T\} \cup \{\text{Time}(t) \mid t \in \mathcal{T}\} \cup \{\text{Space}(s) \mid s \in \mathcal{S}\} \cup \{\text{SpaceTime}(st) \mid st \in \mathcal{ST}\}$$

  where $\mathcal{D}_T, \mathcal{T}, \mathcal{S}, \mathcal{ST}$ are sets of possible data, temporal, spatial, and spacetime values respectively. Temporal values t are typically $(\text{scale}, \text{unit})$ with scale $\in \mathcal{T}_{\text{scale}}$ and unit $\in \mathcal{T}_{\text{unit}}$. Spatial values s are (x, y, z) with coordinates in T_{coord}. The interpretation (Euclidean or Non-Euclidean) depends on T_{coord} and the functions defined via the Spatial<V> trait.
- $\text{adj}_v = (\text{update}_v, \text{adjust}_v)$ are optional functions implementing the Adjustable protocol, where $\text{update}_v : \mathcal{V}_{\text{payload}} \to \text{void}$ and $\text{adjust}_v : \mathcal{V}_{\text{adj\_factor}} \to \text{void}$.

A **Hyperedge** $e \in E_C$ is a tuple: $e = (V_e, \text{kind}_e, \text{label}_e)$ where $V_e \subseteq V_C$ ($|V_e| \geq 1$), $\text{kind}_e \in \mathcal{K}_{\text{relation}}$, and label_e is optional.

4.6.3 Causaloid and CausaloidGraph Formalism

A **Causaloid** χ is a tuple: $\chi = (id_\chi, f_\chi, \mathcal{C}_{refs}, \text{desc}_\chi, \mathcal{A}_{linked}, I_{linked})$ where:

- $id_\chi \in \mathbb{I}$.
- $f_\chi : \mathcal{O}_{\text{type}} \times \text{ContextAccessor}(\mathcal{C}_{refs}) \to \{\text{true}, \text{false}\}$ is the causal function. $\mathcal{O}_{\text{type}}$ is the observation type.
- $\mathcal{C}_{refs} \subseteq \mathcal{C}_{sys}$ are referenced contexts.
- desc_χ is a description.
- $\mathcal{A}_{linked}$ (optional) is a set of id_{asmp} for linked Assumptions.
- I_{linked} (optional) is an id_{inf} for the founding Inference.

A **CausaloidGraph** G is a hypergraph: $G = (V_G, E_G, ID_G, \text{Name}_G)$ where:

- V_G is a set of causal nodes $v_g = (id_g, \text{payload}_g)$. The payload payload_g can be a Causaloid χ, a collection $\{\chi_i\}$, or another CausaloidGraph G', reflecting recursive isomorphism.
- E_G is a set of causal hyperedges $e_g = (V_{\text{source}}, V_{\text{target}}, \text{logic}_e)$, where logic_e defines the functional relationship.

The **state** of G is $S_G : V_G \to \{\text{active}, \text{inactive}\}$.

4.6.4 Causal State Machine (CSM) Formalism

A **Causal State Machine** M is a tuple: $M = (\mathcal{G}_M, \mathcal{C}_M, Q, \mathcal{P}_{state}, A, \delta)$ where:

- $\mathcal{G}_M = \{G_1, \ldots, G_m\}$ are supervised CausaloidGraphs.
- $\mathcal{C}_M \subseteq \mathcal{C}_{sys}$ are relevant Context Hypergraphs.
- Q is a finite set of causal states.
- $\mathcal{P}_{state}$ is a set of state activation predicates $P_q : \prod_{i=1}^{m} \text{StateSpace}(G_i) \to \{\text{true}, \text{false}\}$ for each $q \in Q$.
- A is a finite set of deterministic actions $a : \mathcal{W}_{\text{state}} \to \mathcal{W}'_{\text{state}}$.
- $\delta : Q_{\text{active}} \to A$ is the action triggering function, where $Q_{\text{active}} = \{q \in Q \mid P_q(\ldots) = \text{true}\}$.

A **Causal State Machine** M orchestrates actions based on the state of one or more causal models within their contexts. It is defined as a tuple: $[M = (\mathcal{G}M, \mathcal{C}M, Q, P, A, \delta)]$ where:

- $\mathcal{G}M = G_1, G_2, \ldots, G_m$ is the set of CausaloidGraphs supervised by this CSM.
- $\mathcal{C}M \subseteq \mathcal{C}sys$ is the set of Context Hypergraphs relevant to the supervised models and potentially the CSM's state logic.
- Q is a finite set of **causal states**. Each state represents a meaningful condition derived from the activation states of the supervised CausaloidGraphs.
- P is a set of **state activation predicates**. For each state $q \in Q$, there is a predicate P_q that maps the combined activation states of the supervised graphs to a Boolean value: $[P_q : \text{StateSpace}(G_1) \times \cdots \times \text{StateSpace}(G_m) \to \text{true}, \text{false}]$ where $\text{StateSpace}(G_i)$ represents the set of all possible activation mappings SG_i. A state q is active if P_q evaluates to true.
- A is a finite set of deterministic **actions**. Each action $a \in A$ is a function representing an intervention or operation, potentially modifying the environment, the context, or the system state: $a : \text{WorldState} \to \text{WorldState}'$.
- δ is the **action triggering function**. It maps active causal states to specific actions: $[\delta : Q_{\text{active}} \to A] where Q_{\text{active}} = q \in Q \mid P_q(\ldots) = \text{true}$. Note that multiple states might be active simultaneously, potentially triggering multiple actions based on the CSM's execution semantics (e.g., parallel execution, prioritized execution).

4.7 Implementation in Rust

The conceptual and formal architecture of DeepCausality is concretely realized through a reference implementation in the Rust programming language. This choice was driven by the need for a high-performance, memory-safe, and expressive environment capable of managing the complexities inherent in advanced causal modeling. The Rust implementation translates the abstract concepts of `Contextoids`, `Causaloids`, `CausaloidGraphs`, and Causal State Machines into robust and efficient software components, with a particular emphasis on leveraging Rust's powerful trait system and generics to achieve modularity, extensibility, and type safety. We will highlight key aspects of this implementation, focusing on the `Causaloid` as the elemental unit of causation and the `CausaloidGraph` as the structure for composing complex causal models.

4.7.1 The Causaloid Implementation: Encapsulating Causal Logic

The `Causaloid`, representing an individual causal link or hypothesis, is a central struct in the Rust implementation[22]. It encapsulates an identifier, a description, its current activation state (managed with thread-safe primitives like `Arc<RwLock<bool»` for potential concurrent access), and crucially, the causal logic itself.

A key design feature is the `Causaloid`'s internal `CausalType` enum[23], which allows a single `Causaloid` struct to represent one of three distinct structural forms:

1. A `CausalType::Singleton`: This represents an elementary cause, directly holding an optional causal function. This function can be context-free (type `CausalFn`) or context-aware (type `ContextualCausalDataFn`), with the latter taking an immutable reference to a `Context` object.
2. A `CausalType::Collection`: This allows a `Causaloid` to encapsulate an entire collection (e.g., a `Vec`) of other `Causaloids`. The evaluation of such a "collection" `Causaloid` typically involves applying reasoning logic over its constituent members, facilitated by the `CausableReasoning` trait.
3. A `CausalType::Graph`: This enables a `Causaloid` to encapsulate an entire `CausaloidGraph` struct, thereby directly realizing the recursive isomorphic nature of the causal modeling engine.

This enum-based design, combined with generic type parameters for context-related data types (D, S, T, ST, V), makes the `Causaloid` struct remarkably versatile. Various constructors are provided to create `Causaloids` of each `CausalType`, optionally associating them with specific `Context` instances if their causal logic is context-dependent.

The behavior of a `Causaloid` is governed by its implementation[24] of the `Causable`[25] trait. This trait mandates methods like `explain()`, `is_active()`, `is_singleton()`, `verify_single_cause()`, and `verify_all_causes()`. The implementation of these methods for the `Causaloid` struct intelligently dispatches behavior based on its internal `CausalType`:

- For a `Singleton`, `verify_single_cause()` executes its stored causal function (either context-free or context-aware).
- For a `Collection`, `verify_all_causes()` delegates to the `reason_all_causes()` method of the encapsulated collection (which benefits from the `CausableReasoning` extension trait). `is_active()` might check if any member of the collection is active.
- For a `Graph`, `verify_all_causes()` delegates to the `reason_all_causes()` method of the encapsulated `CausaloidGraph` (which benefits from the `CausableGraphReasoning` trait).

This elegant delegation ensures that a `Causaloid`, regardless of its internal complexity (simple function, collection, or entire graph), presents a uniform interface to the rest of the system, particularly when it is itself a node within a higher-level `CausaloidGraph`.

[22]`https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/causaloid/mod.rs`
[23]`https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/causaloid/causal_type.rs`
[24]`https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/types/reasoning_types/causaloid/causable.rs`
[25]`https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/protocols/causable/mod.rs`

4.7.2 The CausaloidGraph Implementation: Composing and Reasoning Over Causal Networks

The `CausaloidGraph` is the primary structure for composing multiple `Causaloids` into complex causal networks. In the Rust implementation, this is realized as a generic struct, `CausaloidGraph<T> where T: Causable + PartialEq`, which internally uses a graph data structure (specifically, an `UltraGraph<T>`[26]) to store `Causable` nodes and their relationships. The power and ergonomics of the `CausaloidGraph` implementation are significantly enhanced by a layered trait design:

1. **Base Graph Operations via** `CausableGraph<T>`[27]: This trait provides the fundamental Application Programming Interface (API) for graph manipulation, such as adding or removing `Causaloid` nodes and edges, checking for their existence, managing a root node, and retrieving graph metrics like size and number of active nodes. A crucial method in this trait is `get_graph(&self) -> &CausalGraph<T>`, which returns a reference to the underlying graph data structure (the `UltraGraph` instance). This accessor is key to enabling default implementations in higher-level traits. The `impl CausableGraph<T> for CausaloidGraph<T>` block provides the concrete logic for these operations, typically by delegating to the methods of the underlying `UltraGraph`.
2. **Defaulted Reasoning Logic via** `CausableGraphReasoning<T>`[28]: This trait extends `CausableGraph<T>` by providing a suite of sophisticated reasoning methods, such as `reason_all_causes()`, `reason_subgraph_from_cause()`, `reason_shortest_path_between_causes()`, and `reason_single_cause()`. Critically, these methods are provided with *default implementations* directly within the trait definition (or are implemented in a blanket way for any type that implements `CausableGraph<T>`). These defaults leverage the `get_graph()` accessor to operate on the graph structure and call the `Causable` trait methods (like `verify_single_cause`) on the individual `Causaloid` nodes. For instance, `reason_from_to_cause()` implements a depth-first-search-like traversal, verifying each `Causaloid` along the path using observation data. Users of `CausaloidGraph` automatically receive this rich reasoning functionality simply because `CausaloidGraph<T>` implements trait `CausableGraph<T>`.
3. **Defaulted Explaining Logic via** `CausableGraphExplaining<T>`[29]: Similar to reasoning, this trait extends `CausableGraph<T>` and provides default implementations for methods like `explain_all_causes()` and `explain_shortest_path_between_causes()`. These methods traverse the graph (again, using `get_graph()`) and aggregate the string explanations obtained by calling the `explain()` method on each constituent `Causaloid`.

This strategic use of traits with default implementations, relying on a minimal set of required methods in a base trait (`CausableGraph<T>` requiring `get_graph()`), dramatically reduces code duplication and complexity. It means that the sophisticated logic for graph traversal, data mapping, and aggregation for both reasoning and explaining only needs to be written once, within the default trait methods. Any specific graph structure that correctly implements the basic `Causaloid` storage and access via `CausableGraph<T>` can instantly inherit this full suite of advanced capabilities. This is a powerful demonstration of Rust's ability to build highly abstract yet efficient and maintainable systems, significantly lowering the implementation burden for complex functionalities. The clear separation of concerns—basic graph structure, reasoning logic, and explaining logic, each in its own trait—further enhances modularity and testability.

The concrete realization of these causal graph structures within DeepCausality utilizes `UltraGraph`[30], a custom-developed Rust graph library designed to provide an ergonomic API layer over robust underlying graph representations. `UltraGraph` leverages the hypergraph capabilities of the `petgraph` crate, specifically employing an adjacency matrix-based graph structure from `petgraph` to represent the network of `Causaloids` and their potentially N-ary relationships (hyperedges). To facilitate efficient node management and direct access, `UltraGraph` typically combines this with a `HashMap` for storing the `Causaloid` nodes themselves, keyed by their identifiers. This approach allows DeepCausality to benefit from `petgraph`'s foundational graph algorithms and data structures while offering a simplified and more specific interface for causal modeling tasks, including methods for direct node retrieval, neighbor access, and shortest path calculations. The `impl CausableGraph<T> for CausaloidGraph<T>` block then implements DeepCausality's specific graph operations by interfacing with these `UltraGraph` capabilities.

[26] `https://github.com/deepcausality-rs/deep_causality/tree/main/ultragraph`

[27] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/protocols/causable_graph/graph.rs`

[28] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/protocols/causable_graph/graph_reasoning.rs`

[29] `https://github.com/deepcausality-rs/deep_causality/blob/main/deep_causality/src/protocols/causable_graph/graph_explaining.rs`

[30] `https://github.com/deepcausality-rs/deep_causality/tree/main/ultragraph`

4.7.3 The Context Implementation: Traits, Generics, and Hypergraphs

The Context architecture relies on the language's powerful trait system and generics to manage diverse contextual data types and their interactions within a hypergraph structure, while ensuring type safety and enabling extensibility. This design facilitates a clean and idiomatic separation between immutable and mutable contextual elements, a key factor in ensuring both data integrity and adaptability.

At the foundation are several elemental traits, primarily defined in the conceptual module for context nodes[31], which establish the contracts for different kinds of contextual information. The Datable trait marks identifiable data-holding entities. More specialized traits include Temporable<V>, for objects possessing a time scale and unit (with TimeScale being a dedicated enum[32]); Spatial<V>, for those with up to three spatial coordinates (X, Y, Z) of a generic numeric type V; and SpaceTemporal<V>, which combines both spatial and temporal properties, also providing access to a temporal coordinate t. The generic parameter V for these traits is constrained with standard Rust operational traits (e.g., Default, Copy, Clone, Hash, Eq, arithmetic operations) ensuring that dimensional values are workable and can represent a wide range of numeric types suitable for diverse geometric interpretations, including both Euclidean and non-Euclidean spaces.

Concrete instantiations of these concepts are provided as distinct Rust structs. For example, an immutable temporal node is represented by Time<T>[33], a spatial node by Space<T>[34], a combined spatio-temporal node by SpaceTime<T>[35], and a general data-holding node by Data<T>[36]. Each of these structs implements its corresponding elemental trait (e.g., impl Temporable<T> for Time<T>). These structs often utilize procedural macros like deep_causality_macros::Constructor and Getters for ergonomic instance creation and field access.

To handle evolving environments where contextual information must change, DeepCausality provides distinct Adjustable counterparts for these node types, such as AdjustableTime<T>[37] and AdjustableSpaceTime<T>[38]. These structs implement the Adjustable<T> trait[39], which defines optional update and adjust methods. These methods take an ArrayGrid as an argument, providing a structured way to pass new values or transformation parameters. For instance, the impl Adjustable<T> for AdjustableTime<T> demonstrates how an incoming 1D ArrayGrid value is used to either replace (update) or modify (adjust) the internal time_unit, along with necessary validation checks (e.g., ensuring time does not become negative). Similarly, the implementation for AdjustableSpaceTime<T> uses a 4D PointIndex to extract new x, y, z, and time values from the ArrayGrid for updates or adjustments. This explicit separation between immutable and adjustable types ensures that mutability is an opt-in feature, enhancing the predictability and safety of context management.

The Contextoid<D,S,T,ST,V> struct[40] serves as a generic wrapper, encapsulating a unique identifier and a ContextoidType<D,S,T,ST,V> enum[41]. This enum acts as a tagged union, holding one of the concrete dimensional struct instances (e.g., Datoid(Data<MyPayloadType>), Tempoid(AdjustableTime<MyTimeUnit>), Spaceoid(Space<MyCustomNonEuclideanCoord>)). The Contextoid itself then implements the overarching Contextuable<D,S,T,ST,V> trait[42], providing a uniform interface (e.g., via the vertex_type() method) regardless of the specific dimensional data it holds. This design allows a ContextHypergraph to store heterogeneous Contextoid types while still enabling type-safe access and efficient static dispatch through pattern matching on the ContextoidType enum.

The overall organization of Contextoids and their interrelationships is managed by the main Context<D,S,T,ST,V> struct[43]. This struct acts as the primary container, holding a base_context (typically an UltraGraph<Contextoid<...» instance for the default context) and, optionally, an extra_contexts field (e.g., a HashMap<u64, UltraGraph<Contextoid<...»>) to manage multiple, uniquely identifiable context hypergraphs.

[31] `contextuable.txt`
[32] `time_scale.txt`
[33] `time.txt`
[34] `space.txt`
[35] `sapce_time.txt`
[36] `data.txt`
[37] `adjutable_time.txt`
[38] `adjustable_space_time.txt`
[39] `adjustable.txt`
[40] `contextoid.txt`
[41] `contextoid_type.txt`
[42] The impl Contextuable for Contextoid is defined in `contextuable.txt` which appears to be part of the Contextoid module structure.
[43] `context_graph.txt`

The base_context and each of the extra_contexts within the main Context struct are implemented as instances of a hypergraph data structure. This is also built upon the UltraGraph library, which, by utilizing the underlying hypergraph support of the petgraph crate (specifically its adjacency matrix graph implementation), provides the necessary foundation for representing Contextoids as nodes and their N-ary contextual relationships as hyperedges. This choice ensures efficient storage and access mechanisms for Contextoids and their complex interdependencies. The methods defined in the ContextuableGraph and ExtendableContextuableGraph traits are then implemented by the Context struct, leveraging these UltraGraph functionalities to manipulate the respective context hypergraphs.

The Context struct then implements the ContextuableGraph<D,S,T,ST,V> trait for operations on the base context, and the ExtendableContextuableGraph<D,S,T,ST,V> trait (both defined in the contextuable_graph trait file[44] but likely with concrete implementations for the Context struct in files like `contextuable_graph.txt` and `extendable_contextuable_graph.txt` located within the Context struct's module) for managing and interacting with these additional contexts. These traits define a clean API for adding, removing, and querying nodes (Contextoids) and edges (which are typed with a RelationKind enum[45]) within the selected context hypergraph. The extensive use of generics (D, S, T, ST, V) throughout these definitions allows users to integrate their own specific data types for observations, time units, and spatial coordinates, facilitating the framework's adaptability to diverse domains and geometric representations, including sophisticated non-Euclidean models. This meticulous, trait-driven, and generically-typed implementation in Rust culminates in a uniquely expressive, performant, and robust system for deeply contextualized causal reasoning.

[44] `contextuable_graph.txt`
[45] `relation_kind.txt`

5 Advanced Modeling with DeepCausality

5.1 Context Modeling

5.1.1 Static vs. Dynamic Context

DeepCausality's context distinguishes between Static Contexts and Dynamic Contexts. This distinction is pivotal, carrying substantial implications for model architecture, system efficiency, and the class of causal problems the system can effectively address.

n a Static Context, the entire topology of the Context Hypergraph—encompassing the defined types of Contextoids, their specific interconnections via hyperedges, and the overall relational schema of the contextual landscape—is established a priori and remains invariant throughout the system's operational phase. While this structural immutability is a defining feature, it is crucial to understand that the values encapsulated within individual Contextoids are generally expected to change. For instance, a Data Contextoid representing a "current-position" in a drone model will be continually updated with new values representing the position, yet its predefined role and its connections to other temporal or instrument-specific Contextoids within the static graph structure will persist. The documentation [46] explicitly states, "The context structure is defined beforehand for a static context..." and that for such fixed structures, "...only values stored in it get updated". This paradigm is particularly advantageous in scenarios where the contextual variables impacting a causal system and their interdependencies are well-understood and exhibit long-term stability. The primary benefits are enhanced computational efficiency and predictability. Static structures permit optimized query paths and memory layouts, as noted by the assertion that this approach is more memory efficient. Such contexts are ideal for domains like established industrial process control, where sensor networks are fixed. Even within such structurally static environments, the Adjustable protocol[47] remains applicable to individual Contextoids, allowing their internal data to be modified through update() or adjust() operations without necessitating any alteration to the overarching graph topology.

Conversely, Dynamic Contexts offer a paradigm of structural plasticity, wherein the very architecture of the Context Hypergraph can be algorithmically generated, augmented, or pruned during runtime. This implies the capacity to introduce novel Contextoid types, forge new relational hyperedges, or modify existing structural components as the system interacts with and learns from its operational environment. The documentation[48] confirms this by stating, "...whereas for a dynamic context, the structure is generated dynamically at runtime," thereby enabling "more dynamic and self-adaptable designs". This capability is indispensable for systems deployed in open-world, non-stationary environments where not all contextual factors can be anticipated at design time. Consider applications such as autonomous robotic exploration of uncharted territories; as the robot encounters new environmental features or objects, it can dynamically instantiate new Contextoids (e.g., representing a newly mapped spatial region or a previously unidentified object type) and integrate them into its evolving contextual model. Similarly, modeling influence in evolving social networks, where participants and relationships frequently change, necessitates a dynamic contextual framework. The implicit potential for dynamic sensor integration in an IoT network, such as incorporating data from a temporary drone inspection service into an existing industrial monitoring system.

However, the power of dynamic contexts introduces certain engineering challenges. The documentation acknowledges that this approach is "less efficient and more complex because it requires regular context pruning (removing values no longer needed) to avoid excessive memory usage..."[49]. The effective management of a mutable graph structure demands sophisticated mechanisms for ensuring data consistency, controlling memory allocation to prevent unbounded growth or the accumulation of orphaned contextual elements, and designing robust query interfaces that can gracefully handle a potentially evolving schema. The task of "context pruning"—intelligently identifying and removing obsolete or low-relevance Contextoids and their associated hyperedges—becomes a critical, non-trivial component of system maintenance.

The provision for both static and dynamic contextual paradigms recognizes that different problem domains and operational requirements call for different trade-offs between structural stability (and its attendant efficiencies) and structural adaptability (and its capacity to handle novelty and environmental drift). A hybrid approach, where stable, well-understood contextual facets are modeled statically while more volatile or exploratory aspects leverage dynamic sub-contexts, is also implicitly supported by the framework's modular design. The underlying hypergraph representation serves as a versatile structural primitive capable of accommodating both modes of operation, allowing engineers to tailor the contextual grounding of their causal models precisely to the demands of the application at hand.

[46]https://deepcausality.com/docs/concepts/

[47]https://deepcausality.com/docs/concepts/

[48]https://deepcausality.com/docs/concepts/

[49]https://deepcausality.com/docs/intro/

5.2 Causality Modeling

5.2.1 Dynamic Temporal Causal Reasoning

A significant challenge in applying causal reasoning to dynamic systems is the effective and efficient handling of temporal information. Traditional time series analysis often employs sliding window approaches or auto-regressive models where the computational cost or model complexity can scale with the length of the historical data considered. DeepCausality offers an alternative perspective, particularly when the primary causal structure (*i.e.*, the set of causal mechanisms and their interdependencies represented by the CausaloidGraph) is assumed to be stable over time, even if the contextual data it operates upon is highly dynamic.

The core insight involves the interaction between a structurally stable CausaloidGraph and a dynamically updated Context Hypergraph. This Context Hypergraph is not merely a linear sequence of events; it explicitly represents temporal information at multiple, user-defined scales via Time and SpaceTime Contextoids, each associated with a specific TimeScale enum (e.g., Year, Month, Day, Hour, Minute, Second, Nanosecond)[50]. At each discrete data sampling interval, new observations update the relevant Contextoids at the appropriate scales within the Context Engine. For instance, a high-frequency trading system might update nanosecond-level price Contextoids, while also concurrently updating minute-level and hour-level aggregations or features (like moving averages or volatility measures) stored in separate Contextoids.

When the CausaloidGraph is subsequently evaluated, its constituent Causaloids can query the Context Engine for the *current* state of these Contextoids at any of the modeled time scales. A single causal function f_χ might simultaneously access “last_price_tick_at_nanosecond_resolution”, “average_price_last_minute”, and “volatility_last_hour” by querying distinct Contextoids, each representing a different temporal granularity or aggregation. The causal functions then operate on this rich, multi-scale, time-synchronous contextual data.

The crucial point is that if the topology of the CausaloidGraph itself—the set of Causaloids (V_G) and their relational (hyper)edges (E_G)—does not change from one time step to the next, then the computational complexity of traversing this graph to perform a reasoning task remains constant, denoted as $O(k)$ where k is a function of $|V_G|$ and $|E_G|$. Access to different time scales within the Context Engine is typically managed via efficient lookups (e.g., using the TimeIndexExt with HashMaps as described in the implementation of CustomContext[51] and Context[52]), allowing for near-instantaneous retrieval of contextual data at the required granularity. This direct, indexed access to variable time scales without re-computation or complex windowing logic across the entire history for each query is a distinctive feature.

This architecture offers constant-time graph traversal characteristics for the causal reasoning logic *per evaluation cycle*, irrespective of the raw history length, because the history and its multi-scale aggregations are encapsulated and managed within the Context Engine. The CausaloidGraph operates on the "present moment" as defined by the current state of its accessible, multi-scale contexts. This approach facilitates a non-linear conceptualization of time’s influence:

1. Time as Explicit, Multi-Scale Context: Time itself is not just a sequence but a structured entity within the Context Hypergraph, modeled via Time or SpaceTime Contextoids that are explicitly aware of their TimeScale. Causal functions can directly query Contextoids representing phenomena at, for example, the second, hour, and day scale simultaneously.
2. History Encapsulated in Contextual State at Relevant Scales: The influence of the past is summarized or reflected in the current state of other Data Contextoids, potentially maintained at various aggregations. The causal model reasons based on these relevant historical summaries, not by directly traversing an infinitely long chain of past raw events within its own structure. The Adjustable protocol allows these summary Contextoids to be updated efficiently.
3. Non-Linear Causal Functions: The causal functions within Causaloids (f_χ) can be arbitrarily complex non-linear functions, modeling how the *same* causal mechanism responds differently to varying contextual inputs, including information from different temporal scales provided by the Context Engine.

This architecture allows for a nuanced understanding of time’s influence. Instead of a purely linear, chain-like dependence on all past states, time’s influence is mediated through a structured, multi-scale Context Engine. The CausaloidGraph remains structurally lean and performant for reasoning, while the Context Engine manages the complexity of temporal data, its historical aggregation, and its presentation at variable scales. The separation of

[50]`time_scale.txt`
[51]`time_index.txt`
[52]`indexable.txt`

concerns—stable causal logic in the CausaloidGraph and dynamic, multi-scale environmental state in the Context Hypergraph—is key to this efficiency and flexibility.

6 Discussion

DeepCausality represents a novel approach to computational causality, offering a comprehensive framework implemented in Rust for constructing and reasoning with explicit, context-aware causal models. Its core contributions—the hypergeometric conceptualization of context and causality, the operational nature of composable Causaloids, the integrated Causal State Machines, and the leveraging of Rust's capabilities—aim directly at addressing significant limitations inherent in contemporary correlation-based AI paradigms, particularly concerning explainability, dynamic adaptation, and mechanistic understanding. The significance of this framework lies in its potential to enable the development of more reliable, transparent, and trustworthy intelligent systems capable of operating effectively within complex, evolving environments. Applications requiring deep contextual grounding and verifiable reasoning, such as dynamic risk management in finance, autonomous control systems in IoT, or modeling intricate socio-technical phenomena, stand to benefit substantially from this approach.

One valid consideration pertains to the framework's complexity and associated learning curve. DeepCausality introduces several powerful abstractions—hypergraphs for both context and causality, specialized Contextoid types, recursive Causaloid structures, non-Euclidean reasoning primitives, and the Adjustable protocol; all of it in the Rust ecosystem, which itself demands a degree of programming proficiency. This confluence of novel concepts may present a steeper initial learning curve compared to utilizing established, simpler DAG-based tools or Python libraries with extensive community support. We argue, however, that this complexity is not arbitrary but rather a reflection of the inherent intricacy required to faithfully model the multi-faceted, context-dependent nature of real-world causal systems, which simpler formalisms might unduly oversimplify. Furthermore, the framework's design actively seeks to manage this complexity through modularity—complex CausaloidGraphs can be built from simpler, reusable Causaloid components—and the structural clarity afforded by explicit graph representations. Rust's strong typing also aids correctness during development. Nonetheless, the development of user-friendly tooling, including visualization aids and model construction interfaces, as noted in our limitations, is recognized as crucial future work to significantly lower the barrier to entry and enhance usability.

Another important consideration involves the scalability of the hypergeometric representations. While Rust provides a high-performance foundation, algorithms operating on general hypergraphs can theoretically exhibit challenging computational complexity and memory consumption characteristics compared to operations on simpler graph structures, particularly as the number of nodes (Contextoids, Causaloids) and hyperedges scales into the millions. DeepCausality's initial design documentation noted promising conceptual performance, and the implementation strives for efficiency, for instance, through performant custom types like ArrayGrid[53]. Preliminary synthetic benchmarks, executed on standard Apple M3 Max hardware without specialized GPU acceleration (see Appendix A), provide encouraging early data on this front. These results indicate that reasoning over collections of Causaloids scales from approximately 100 nanoseconds for small collections (10 elements) to around 44-51 microseconds for large collections (10,000 elements). For CausaloidGraph operations on linear graph structures, reasoning over all causes in a small graph (10 nodes) takes approximately 2.7 microseconds, scaling to around 70 milliseconds for large graphs (10,000 nodes). Notably, reasoning over a single, directly accessed cause remains exceptionally fast, consistently in the 10 nanosecond range regardless of overall graph size, highlighting the efficiency of direct causaloid evaluation. Even for small multi-layer graphs, reasoning times remain in the low microsecond or even sub-microsecond range for specific path or subgraph evaluations. While these initial figures demonstrate strong CPU-based performance for the tested scales and graph types, the behavior under truly massive scale deployments (e.g., millions of nodes) or with more complex, densely interconnected real-world hypergraph topologies warrants further rigorous investigation. Comprehensive benchmarking across diverse graph sizes, densities, query patterns, and context interaction complexities, alongside exploration of graph optimizations, advanced parallelization strategies remain important avenues for future research.

Furthermore, while the per-core inference rate for the most complex graph operations benchmarked, such as the approximately 28 shortest-path inferences per second observed on a 10,000-node linear graph, might initially seem modest compared to GPU-centric metrics, this perspective overlooks a crucial strategic advantage inherent in DeepCausality's efficient CPU-bound design. Notably, performance for moderately complex tasks is significantly higher; the equivalent shortest-path reasoning on a 1,000-node graph achieves approximately 3,500 inferences per second on the same hardware. This demonstrates that substantial reasoning throughput is attainable even for non-trivial causal models on a single CPU core. The true power then lies in exploiting the high parallelizability of many causal reasoning workloads—including parallel evaluation of independent models, extensive backtesting, or agent-based simulations—through horizontal scaling. Consequently, achieving massive aggregate throughput becomes feasible by distributing DeepCausality instances across inexpensive, readily available commodity CPU clusters. A modest cluster could thus potentially execute hundreds of thousands of 1k-node causal reasoning operations per second.

[53]https://deepcausality.com/blog/the-grid-type/

This cost effective scaling contrasts sharply with the significant capital and operational expenditure required to scale large deep learning models, which often necessitates costly and specialized GPU infrastructure. This capability to achieve massive parallelism and high overall inference rates through cost-effective horizontal scaling on standard hardware represents a significant, practical advantage, democratizing access to sophisticated causal reasoning and offering a compelling alternative for deploying complex AI systems without prohibitive hardware investments.

The current reliance on manual construction of causal models represents a significant practical limitation acknowledged by the project. DeepCausality presently excels as an engine for encoding, executing, and reasoning with explicitly provided causal knowledge or hypotheses. It does not, in its current form, incorporate algorithms for automated causal structure discovery from raw data. This focus positions it strongly for domains where encoding expert knowledge, regulatory rules, or first-principles scientific models is paramount, or where the primary goal is the rigorous testing of specific, predefined causal hypotheses within a rich context. In these scenarios, the ability to explicitly define the model is a strength, ensuring transparency and alignment with domain understanding. However, it clearly differentiates DeepCausality from tools primarily focused on data-driven causal discovery. We view DeepCausality as complementary to such tools; discovery algorithms could generate candidate causal structures that are then implemented, refined, and executed within DeepCausality's robust contextual reasoning framework. Integrating structural learning capabilities directly, potentially by adapting techniques from reinforcement learning or neuro-evolution to operate on its hypergraph structures, remains a key and exciting direction for future development.

The framework's current emphasis is primarily deterministic, focusing on causal functions yielding boolean outcomes and CSMs triggering definite actions. While this design choice strongly favors explainability, verifiability, and the construction of reliable control systems, it might appear to simplify the inherent stochasticity present in many real-world systems. It is important to clarify that the architecture is not fundamentally restricted to determinism. The flexibility of using arbitrary Rust code within a causal_function allows for the encapsulation of probabilistic logic (e.g., returning true if an internal probabilistic calculation exceeds a threshold). Moreover, pathways for more explicit probabilistic extensions exist, such as modifying Causaloids to output probabilities or incorporating probabilistic Contextoids. While developing first-class support for sophisticated probabilistic reasoning paradigms (e.g., Bayesian inference directly integrated with the graph structures) is a potential future enhancement, the current deterministic core provides a solid, understandable foundation.

Regarding the scope of theoretical novelty, DeepCausality builds upon significant foundational ideas, particularly drawing inspiration from Hardy's operational perspective on causality. Its primary novelty lies not in proposing a fundamentally new abstract theory of causation itself, but rather in the specific, innovative synthesis, integration, and operationalization of these concepts into a coherent computational framework. The unique contributions reside in the concrete realization of the hypergeometric context engine (with its multi-dimensional, multi-context, adjustable, and non-Euclidean capabilities), the recursive and composable CausaloidGraph structure, the tight integration with the action-oriented Causal State Machines, and the embodiment of this entire system within the high-performance, memory-safe environment of Rust. The contribution is thus providing a powerful, practical, and performant engineering framework designed to make sophisticated, context-aware causal reasoning feasible for complex systems development.

Despite these considerations and limitations, DeepCausality offers substantial value in specific application areas where existing approaches may fall short. Its unparalleled capability for deep, multi-faceted contextualization allows models to capture environmental nuances—from multi-resolution temporal patterns in finance to interacting Euclidean and non-Euclidean spatial relationships in socio-technical systems—that are difficult to represent adequately in simpler frameworks. The direct, deterministic link between causal inference and action provided by Causal State Machines offers a unique advantage for building explainable and verifiable control systems, particularly crucial in safety-critical or regulated domains where opaque decision-making is unacceptable. The Rust implementation itself unlocks potential for high-performance, real-time causal analytics and deployment in resource-constrained or embedded environments. Lastly, its structured approach to reasoning and context management makes it an exceptionally strong candidate for serving as the rigorous causal engine within hybrid AI architectures, grounding the perceptual abilities of LLMs with mechanistic understanding.

DeepCausality presents a sophisticated framework for computational causality, consciously designed to address key weaknesses in contemporary AI. It offers a unique blend of structural expressiveness, deep contextual awareness, operational capability via CSMs, and implementational robustness through Rust. While acknowledging the current limitations regarding usability tooling, automated discovery, and extensive probabilistic support, the framework provides a solid and extensible foundation. DeepCausality, in its current form, is a powerful engine for executing explicit, context-dependent causal logic with high performance and explainability, and its most profound impact may lie in enabling a new generation of hybrid intelligent systems capable of combining broad pattern recognition with deep, verifiable, causal understanding.

A Appendix: Preliminary Performance Benchmarks

The following tables summarize preliminary performance benchmarks for key DeepCausality operations, executed on an Apple MacBook Pro with an M3 Max processor. These benchmarks were run using the standard Rust ‘criterion‘ library (‘cargo bench‘). The benchmark code is available on Github[54]. It is important to note that these are synthetic benchmarks, primarily utilizing linear or simple multi-layer graph structures, and do not involve complex context interactions or GPU acceleration. They provide an initial indication of the framework’s CPU-based performance characteristics at different scales.

The scales are defined as follows:

- Small: 10 Causaloids (for collections/maps/graphs)
- Medium: 1,000 Causaloids
- Large: 10,000 Causaloids

Table 1 presents results for reasoning over collections (Vector/Array) and Maps (HashMap/BTreeMap) containing Causaloids. Table 2 details performance for various reasoning tasks on linear ‘CausaloidGraph‘ structures. Table 3 shows results for reasoning on small multi-layer graphs (benchmarks for medium/large multi-layer graphs were not included in the provided results). All times reported are the typical execution times (median/mean) derived from multiple samples.

Table 1: Benchmark Results: Reasoning over Causaloid Collections and Maps.

Operation Description	Scale (Size)	Typical Time
Collection (‘Vec‘/‘Array‘) Reasoning	Small (10)	101.01 ns
	Medium (1k)	3.9123 µs
	Large (10k)	43.979 µs
Map (‘HashMap‘/‘BTreeMap‘) Reasoning	Small (10)	50.365 ns
	Medium (1k)	4.6652 µs
	Large (10k)	50.797 µs

Table 2: Benchmark Results: Reasoning over Linear Causaloid Graphs.

Reasoning Task on Graph	Scale (Nodes)	Typical Time
Reason over All Causes	Small (10)	2.7601 µs
	Medium (1k)	509.94 µs
	Large (10k)	70.221 ms
Reason over Subgraph from Cause	Small (10)	1.5070 µs
	Medium (1k)	245.25 µs
	Large (10k)	34.933 ms
Reason Shortest Path Between Causes	Small (10)	1.6900 µs
	Medium (1k)	286.08 µs
	Large (10k)	35.424 ms
Reason over Single Cause	Small (10)	10.349 ns
	Medium (1k)	9.9537 ns
	Large (10k)	10.010 ns

[54]https://github.com/deepcausality-rs/deep_causality/tree/main/deep_causality/benches

Table 3: Benchmark Results: Reasoning over Small Multi-Layer Causaloid Graphs.

Reasoning Task on Graph	Scale (Nodes)	Typical Time
Reason over All Causes	Small (10)	1.2483 µs
Reason over Subgraph from Cause	Small (10)	489.42 ns
Reason Shortest Path Between Causes	Small (10)	427.45 ns
Reason over Single Cause	Small (10)	10.252 ns

These preliminary results demonstrate efficient CPU-bound performance for reasoning tasks on the tested structures and scales, particularly highlighting the near-constant O(1) time for single cause lookups, irrespective of graph size. The scaling for full graph traversals appears roughly linear or slightly super-linear for the linear graphs tested up to 10,000 nodes. However, as noted in the Discussion, further benchmarking on more complex, densely connected hypergraphs and under various context interaction scenarios is necessary to fully characterize scalability limits. The raw benchmark output also indicated the presence of outliers in some measurements, suggesting potential sources of variability (e.g., cache effects, OS scheduling) that warrant deeper investigation in future performance analyses.

References

[1] Andrea Matarazzo and Riccardo Torlone. A survey on large language models with some insights on their capabilities and limitations, 2025.

[2] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. In *Natural language processing and Chinese computing: 8th cCF international conference, NLPCC 2019, dunhuang, China, October 9–14, 2019, proceedings, part II 8*, pages 563–574. Springer, 2019.

[3] Judea Pearl and Dana Mackenzie. *The Book of Why*. Basic Books, 2018.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[5] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[6] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[7] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021. Extract from Causal Deep Learning.pdf [38].

[8] Judea Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*, 2018.

[9] Jean Kaddour, Aengus Lynch, Qi Liu, Matt J Kusner, and Ricardo Silva. Causal machine learning: A survey and open problems. arXiv preprint arXiv:2206.15475, 2022. Extract from Causal Deep Learning.pdf [38].

[10] Elias Bareinboim and Judea Pearl. Causal inference by surrogate experiments: z-identifiability. *arXiv preprint arXiv:1210.4842*, pages 113–120, 2012.

[11] Ruocheng Guo, Lu Cheng, Jundong Li, P. Richard Hahn, and Huan Liu. A survey of learning causality with data: Problems and methods. arXiv preprint 1809.09337, 2018. Extract from Towards Causal Representation Learning.pdf [110].

[12] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, 2nd edition, 2009. Extract from Towards Causal Representation Learning.pdf [115].

[13] Elias Bareinboim, Juan D Correa, Duligur Ibeling, and Thomas Icard. On pearl's hierarchy and the foundations of causal inference. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 507–556, 2022. Extract from Causal Deep Learning.pdf [36].

[14] J Gerard Wolff. Solutions to problems with deep learning. *arXiv preprint arXiv:1801.05457*, 2018.

[15] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference - Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA, 2017. Extract from Towards Causal Representation Learning.pdf [116].

[16] I. Guyon, D. Janzing, and B. Schölkopf. Causality: Objectives and assessment. In I. Guyon, D. Janzing, and B. Schölkopf, editors, *JMLR Workshop and Conference Proceedings: Volume 6*, pages 1–42. MIT Press, Cambridge, MA, USA, 2010. Extract from Towards Causal Representation Learning.pdf [110].

[17] Finnian Lattimore, Tor Lattimore, and Mark D Reid. Causal bandits: Learning good interventions via causal inference. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. Extract from Causal Deep Learning.pdf [42].

[18] Sanghack Lee and Elias Bareinboim. Structural causal bandits: Where to intervene? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. Extract from Causal Deep Learning.pdf [50].

[19] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. Causal inference for recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 426–431, 2020. Extract from Causal Deep Learning.pdf [37].

[20] Yangyi Lu, Amirhossein Meisami, Ambuj Tewari, and William Yan. Regret analysis of bandit problems with causal background knowledge. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 141–150, Aug 2020. Extract from Causal Deep Learning.pdf [52].

[21] Virginia Aglietti, Xiaoyu Lu, Andrei Paleyes, and Javier González. Causal bayesian optimization. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Con-ference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3155–3164, Aug 2020. Extract from Causal Deep Learning.pdf [52].

[22] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[23] R. Kohavi, R. Longbotham, D. Sommerfield, and R. Henne. Controlled experiments on the Web: Survey and practical guide. *Data Mining and Knowledge Discovery*, 18:140–181, 2009. Extract from A Few Useful Things to Know about Machine Learning.pdf [26].

[24] M. Pena, A. Arratia, and L. Belanche. Multivariate dynamic kernels for forecasting financial time series. *ICANN*, 2016. Extract from Predicting SP500 Index direction with Transfer Learning and a Causal Graph as main Input.pdf [82].

[25] Cenk Baykal, Vamsi K Potluru, Sameena Shah, and Manuela M Veloso. Bandit sampling for multiplex networks. *arXiv preprint arXiv:2202.03621*, 2022.

[26] Mateusz Binkowski, Gautier Marti, and Philippe Donnat. Autoregressive convolutional neural networks for asynchronous time series. *arXiv preprint arXiv:1703.04122*, 2017.

[27] Dashe Li, Jiajun Sun, Huanhai Yang, and Xueying Wang. An enhanced naive bayes model for dissolved oxygen forecasting in shellfish aquaculture. *IEEE Access*, 8:217917–217927, 2020. Extract from Causal Deep Learning.pdf [50].

[28] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances*, 5(11):eaau4996, 2019.

[29] Defense Advanced Research Projects Agency (DARPA). Assured neuro symbolic learning and reasoning (ansr) summary. `https://www.darpa.mil/research/programs/assured-neuro-symbolic-learning-and-reasoning`. Accessed: May 12, 2025.

[30] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. Causal interpretability for machine learning-problems, methods and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(1):18–33, 2020.

[31] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

[32] Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

[33] John Mark Bishop. Artificial intelligence is stupid and causal reasoning won't fix it, 2020.

[34] Xavier Ouvrard. HYPERGRAPHS: AN INTRODUCTION AND REVIEW. *arXiv preprint arXiv:2002.05014*, March 2020. Extract from 2002.05014.pdf [1].

[35] Thomas S Verma. Causal networks: semantics and expressiveness. *Technical Report R-65, Cognitive Systems Laborator, University of California, Los Angeles*, 1986.

[36] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988. Extract from Theoretical Impediments to Machine Learning.pdf [102].

[37] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019. Extract from Causal Deep Learning.pdf [43].

[38] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[39] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[40] Juan Correa and Elias Bareinboim. A calculus for stochastic interventions: Causal effect identification and surrogate experiments. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 10093–10100, 2020.

[41] D. Janzing, R. Chaves, and B. Schölkopf. Algorithmic indepen-dence of initial condition and dynamical law in thermodynamics and causal inference. *New Journal of Physics*, 18(9), 2016. Extract from Towards Causal Representation Learning.pdf [88].

[42] Jonas Peters, Stefan Bauer, and Niklas Pfister. Causal models for dynamical systems. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 671–690. ACM Digital Library, 2022. Extract from Causal Deep Learning.pdf [39].

[43] Karthika Mohan and Judea Pearl. Graphical models for processing missing data. *Journal of the American Statistical Association*, 116(534):1023–1037, 2021. Extract from Causal Deep Learning.pdf [41].

[44] K. Mohan and J. Pearl. Graphical Models for Processing Missing Data. Technical Report R-473, Department of Computer Science, University of California, Los Angeles, CA, 2017. Submitted. Extract from Theoretical Impediments to Machine Learning.pdf [102].

[45] Thomas S Richardson and Peter Spirtes. Causal inference via ancestral graph models. *Oxford Statistical Science Series*, pages 83–105, 2003.

[46] Thomas S. Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 1990. Extract from Causal Deep Learning.pdf [40].

[47] Eli Sherman, David Arbour, and Ilya Shpitser. General identification of dynamic treatment regimes under interference. In *International Conference on Artificial Intelligence and Statistics*, pages 3917–3927. PMLR, 2020.

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

[49] Jerzy Splawa-Neyman, Dorota M Dabrowska, and Terrence P Speed. On the application of probability theory to agricultural experiments. essay on principles. section 9. *Statistical Science*, pages 465–472, 1990.

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

[51] 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.

[52] A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume I, pages 230–237, Menlo Park, CA, 1994. MIT Press. Extract from Theoretical Impediments to Machine Learning.pdf [101].

[53] S.L. Morgan and C. Winship. *Counterfactuals and Causal Inference: Methods and Principles for Social Research*. Analytical Methods for Social Research. Cambridge University Press, New York, NY, 2nd edition, 2015. Extract from Theoretical Impediments to Machine Learning.pdf [102].

[54] Jin Tian and Judea Pearl. Probabilities of causation: Bounds and identification. *Annals of Mathematics and Artificial Intelligence*, 28(1):287–313, 2000.

[55] Jin Tian and Judea Pearl. On the identification of causal effects. *Technical Report R-290-L, Department of Computer Science, University of California, Los Angeles*, 2002.

[56] Ilya Shpitser, Thomas S Richardson, and James M Robins. An efficient algorithm for computing interventional distributions in latent variable causal models. *arXiv preprint arXiv:1202.3763*, 2012.

[57] Jingwen Zhang, Yifang Chen, and Amandeep Singh. Causal bandits: Online decision-making in endogenous settings. arXiv preprint arXiv:2211.08649, 2022. Extract from Causal Deep Learning.pdf [50].

[58] Blair Bilodeau, Linbo Wang, and Daniel M. Roy. Adaptively exploiting d-separators with causal bandits, 2022. Extract from Causal Deep Learning.pdf [51].

[59] Jeroen Berrevoets, Sam Verboven, and Wouter Verbeke. Model-agnostic counterfactual reasoning for outcomes in dynamic environments. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 792–802, Jul 2021. Extract from Causal Deep Learning.pdf [41].

[60] Jeroen Berrevoets, Sam Verboven, and Wouter Verbeke. Treatment effect optimisation in dynamic environments. *Journal of Causal Inference*, 10(1):106–122, 2022. Extract from Causal Deep Learning.pdf [51].

[61] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[62] Frederick Eberhardt. Introduction to the foundations of causal discovery. *International Journal of Data Science and Analytics*, 3(2):81–91, 2017. Extract from Causal Deep Learning.pdf [43].

[63] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[64] Uzma Hasan and Md Osman Gani. Kcrl: A prior knowledge based causal discovery framework with reinforcement learning. In *Machine Learning for Healthcare Conference*, pages 691–714. PMLR, 2022.

[65] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. Extract from Causal Deep Learning.pdf [42].

[66] Uzma Hasan, Emam Hossain, and Md Osman Gani. A survey on causal discovery methods for i.i.d. and time series data, 2024.

[67] Yinghua Gao, Li Shen, and Shu-Tao Xia. Dag-gan: Causal structure learning with generative adversarial nets. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3320–3324. IEEE, 2021.

[68] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[69] Clive WJ Granger. Testing for causality: A personal viewpoint. *Journal of Economic Dynamics and control*, 2:329–352, 1980.

[70] John Geweke. Measurement of linear dependence and feedback between multiple time series. *Journal of the American statistical association*, 77(378):304–313, 1982.

[71] P Padav. Granger causality in time series–explained using chicken and egg problem, 2021.

[72] Qin Zhang. Dynamic uncertain causality graph for knowledge representation and reasoning: Discrete dag cases. *Journal of Computer Science and Technology*, 27(1):1–23, 2012.

[73] H. C. Deng and Q. Zhang. Cubic DUCG (dynamic uncertain causality graph) recursive algorithm and its implementation in nuclear power plant fault diagnoses. M.S. thesis, Dept. Eng. Phys., Tsinghua Univ., Beijing, China, 2018. Extract from TNNLS.2019.2953177.pdf [94].

[74] J. Hu, L. Zhang, A. Wang, and S. Li. Accident prevention by fault propagation analysis and causal fault diagnosis based on Granger causality test. In *Proc. 13th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, pages 1554–1558, Jul 2017. Extract from TNNLS.2019.2953177.pdf [93].

[75] N. Deng and Q. Zhang. Towards Dynamic Uncertain Causality Graphs for the Intelligent Diagnosis and Treatment of Hepatitis B. *Symmetry*, 12:1690, 2020. Extract from The Application of Dynamic Uncertain Causality Graph Based Diagnosis and Treatment.pdf [98].

[76] Meike Nauta, Doina Bucur, and Christin Seifert. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):19, 2019.

[77] Licheng Jiao, Yuhan Wang, Xu Liu, Lingling Li, Fang Liu, Wenping Ma, Yuwei Guo, Puhua Chen, Shuyuan Yang, and Biao Hou. Causal inference meets deep learning: A comprehensive survey. *Research*, 7:0467, 2024.

[78] Jeroen Berrevoets, Krzysztof Kacprzyk, Zhaozhi Qian, and Mihaela van der Schaar. Navigating causal deep learning, 2022.

[79] Jeroen Berrevoets, Krzysztof Kacprzyk, Zhaozhi Qian, and Mihaela van der Schaar. Causal deep learning, 2024.

[80] Jing Ma, Mengting Wan, Longqi Yang, Jundong Li, Brent Hecht, and Jaime Teevan. Learning causal effects on hypergraphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1202–1212, 2022.

[81] Zizhen Deng, Xiaolong Zheng, Hu Tian, and Daniel Dajun Zeng. Deep causal learning: representation, discovery and inference. *arXiv preprint arXiv:2211.03374*, 2022.

[82] Vikas Ramachandra. Deep learning for causal inference. *arXiv preprint arXiv:1803.00149*, 2018.

[83] Beatrice Acciaio, Anastasis Kratsios, and Gudmund Pammer. Designing universal causal deep learning models: The geometric (hyper) transformer. *Mathematical Finance*, 34(2):671–735, 2024.

[84] Lucien Hardy. Probability theories with dynamic causal structure: a new framework for quantum gravity. *arXiv preprint gr-qc/0509120*, 2005.

[85] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017. Extract from GRAPH HYPERNETWORKS.pdf [72].

[86] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. Extract from GRAPH HYPERNETWORKS.pdf [72].

[87] Claude Berge. *Graphs and hypergraphs*, volume 7. North-Holland publishing company Amsterdam, 1973. Extract from 2002.05014.pdf [3].

[88] Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, and Peter Rodgers. The State-of-the-Art of Set Visualization: The State-of-the-Art of Set Visualization. *Computer Graphics Forum*, 35(1):234–260, February 2016. Extract from 2002.05014.pdf [6].

[89] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLoS ONE*, 9(6):e98679, June 2014. Extract from 2002.05014.pdf [13].

[90] U.V. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, July 1999. Extract from 2002.05014.pdf [10].

[91] K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, and U.V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, page 10 pp., Rhodes Island, Greece, 2006. IEEE. Extract from 2002.05014.pdf [11].

[92] Wenyin Yang, Guojun Wang, Md Zakirul Alam Bhuiyan, and Kim-Kwang Raymond Choo. Hypergraph partitioning for social networks based on information entropy modularity. *Journal of Network and Computer Applications*, 86:59–71, May 2017. Extract from 2002.05014.pdf [20].

[93] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Xintao Ding, and Ji Zhang. A novel social network hybrid recommender system based on hypergraph topologic structure. *World Wide Web*, 21(4):985–1013, July 2018. Extract from 2002.05014.pdf [21].

[94] Denny Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pages 1601–1608, 2007. Extract from 2002.05014.pdf [21].

[95] Wenhui Wu, Sam Kwong, Yu Zhou, Yuheng Jia, and Wei Gao. Nonnegative matrix factorization with mixed hyper-graph regularization for community detection. *Information Sciences*, 435:263–281, April 2018. Extract from 2002.05014.pdf [19].

[96] Taisong Jin, Jun Yu, Jane You, Kun Zeng, Cuihua Li, and Zhengtao Yu. Low-rank matrix factorization with multiple Hypergraph regularizer. *Pattern Recognition*, 48(3):1011–1022, March 2015. Extract from 2002.05014.pdf [13].

[97] Lei Zhu, Jialie Shen, Hai Jin, Ran Zheng, and Liang Xie. Content-Based Visual Landmark Search via Multimodal Hypergraph Learning. *IEEE Transactions on Cybernetics*, 45(12):2756–2769, December 2015. Extract from 2002.05014.pdf [22].

[98] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing*, 216:150–162, December 2016. Extract from 2002.05014.pdf [22].

[99] Vamsi K Potluru, Robert E Tillman, Prashant Reddy, and Manuela Veloso. Deeplex: A gnn for link prediction in multiplex networks. In *SIAM Workshop on Network Science*, 2020. Extract from Bandit Sampling for Multiplex Networks.pdf [28].

[100] M. Zhang and Y. Chen. Link Prediction Based on Graph Neural Networks. In *Proc. Advances in Neural Information Process-ing Systems*, 2018. Extract from Generative Causal Explanations for Graph Neural Networks.pdf [77].

[101] Takeshi Teshima and Masashi Sugiyama. Incorporating causal graphical prior knowledge into predictive modeling via simple data augmentation. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 86–96, Jul 2021. Extract from Causal Deep Learning.pdf [44].

[102] Sai Srinivas Kancheti, Abbavaram Gowtham Reddy, Vineeth N Balasubramanian, and Amit Sharma. Matching learned causal effects of neural networks with domain priors. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10676–10696, Jul 2022. Extract from Causal Deep Learning.pdf [44].

[103] Guy Tennenholtz, Uri Shalit, Shie Mannor, and Yonathan Efroni. Bandits with partially observable confounded data. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 430–439, Jul 2021. Extract from Causal Deep Learning.pdf [53].

[104] Amit Sharma and Emre Kiciman. Dowhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020.

[105] Miruna Oprescu, Vasilis Syrgkanis, Keith Battocchi, Maggie Hei, and Greg Lewis. Econml: A machine learning library for estimating heterogeneous treatment effects. In *33rd Conference on Neural Information Processing Systems*, page 6, 2019.

[106] Yang Zhao and Qing Liu. Causal ml: Python package for causal inference machine learning. *SoftwareX*, 21:101294, 2023.

[107] The DeepCausality Project Contributors. Deepcausality: Hyper-geometric computational causality, 2023. Official website for the DeepCausality framework.