Large Causal Models for Temporal Causal Discovery

Nikolaos Kougioulis

Thesis submitted in partial fulfillment of the requirements for the

Master of Science Degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. Ioannis Tsamardinos

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH).

This page intentionally left blank

University of Crete
Computer Science Department

Large Causal Models for Temporal Causal Discovery

Thesis submitted by
Nikolaos Kougioulis
in partial fulfillment of the requirements for the
Master of Science Degree in Computer Science & Engineering

THESIS APPROVAL

Author: Nikolaos Kougioulis

Committee approvals:

Ioannis Tsamardinos
Professor, Thesis Supervisor

Sofia Triantafillou
Assistant Professor, Committee Member

Grigorios Tsagkatakis
Assistant Professor, Committee Member

Department approval:

Grigorios Tsagkatakis
Assistant Professor, Director of Graduate Studies

Heraklion, November 12, 2025

This page intentionally left blank

Large Causal Models for Temporal Causal Discovery

Abstract

Causal discovery for both cross-sectional and temporal data has traditionally followed a dataset-specific paradigm, where a new model is fitted for each individual dataset. Such an approach underutilizes the potential of multi-dataset and large-scale pretraining, especially given recent advances in foundation models. The concept of *Large Causal Models (LCMs)* envisions a class of pre-trained neural architectures specifically designed for temporal causal discovery. Existing approaches remain largely proofs of concept, typically constrained to small input sizes (e.g., five variables), with performance degrading rapidly to random guessing as the number of variables or model parameters increases. Moreover, current methods rely heavily on synthetic data, generated under arbitrary assumptions, which substantially limits their ability to generalize to realistic or out-of-distribution samples. This work addresses these challenges through novel methods for training on mixtures of synthetic and realistic data collections, enabling both higher input dimensionality and deeper architectures without loss of performance. Extensive experiments demonstrate that LCMs achieve competitive or superior performance compared to classical causal discovery algorithms, while maintaining robustness across diverse domains, especially on non-synthetic data cases. Our findings also highlight promising directions towards integrating interventional samples and domain knowledge, further advancing the development of foundation models for causal discovery.

This page intentionally left blank

Μεγάλα Αιτιακά Μοντέλα για Χρονική Αιτιακή Ανακάλυψη

Περίληψη

Η αιτιακή ανακάλυψη (Causal Discovery), τόσο για συγχρονικά (cross-sectional) όσο και για χρονικά (temporal) δεδομένα, ακολούθησε παραδοσιακά ένα πρότυπο ειδικό στο εκάστοτε σύνολο δεδομένων (dataset-specific paradigm), όπου ένα νέο μοντέλο εκτιμάται για κάθε μεμονωμένο, διαφορετικό σύνολο δεδομένων. Μια τέτοια προσέγγιση υποεκμεταλλεύεται τις δυνατότητες προεκπαίδευσης πολλαπλών συνόλων δεδομένων και μεγάλης κλίμακας, ειδικά λαμβάνοντας υπ'όψιν τις πρόσφατες εξελίξεις στα θεμελιώδη μοντέλα (Foundation Models). Η έννοια των *Μεγάλων Αιτιακών Μοντέλων (Large Causal Models)* οραματίζεται μια κατηγορία προ-εκπαιδευμένων νευρωνικών αρχιτεκτονικών ειδικά σχεδιασμένων για χρονική αιτιακή ανακάλυψη. Οι υπάρχουσες προσεγγίσεις παραμένουν σε μεγάλο βαθμό απλές επαληθεύσεις ιδεών, περιορισμένες σε μικρά μεγέθη εισόδου (π.χ., πέντε μεταβλητών), με την απόδοση να υποβαθμίζεται γρήγορα σε βαθμό τυχαίας εικασίας καθώς αυξάνεται ο αριθμός των μεταβλητών ή των παραμέτρων του μοντέλου. Επιπλέον, οι τρέχουσες μέθοδοι βασίζονται έντονα σε περιορισμένο πλήθος συνθετικών δεδομένων που έχουν παραχθεί υπό αυθαίρετες υποθέσεις, γεγονός που περιορίζει σημαντικά την ικανότητά τους να γενικεύονται σε ρεαλιστικά ή εκτός κατανομής δείγματα. Η παρούσα εργασία αντιμετωπίζει τις παραπάνω προκλήσεις μέσω νέων μεθόδων για εκπαίδευση σε μείξεις συνθετικών και ρεαλιστικών συλλογών δεδομένων, επιτρέποντας τόσο υψηλότερη διαστατιμότητα εισόδου όσο και βαθύτερες αρχιτεκτονικές χωρίς απώλεια απόδοσης. Εκτεταμένα πειράματα καταδεικνύουν πως τα μεγάλα αιτιακά μοντέλα επιτυγχάνουν ανταγωνιστική ή ανώτερη απόδοση σε σύγκριση με κλασσικούς αλγόριθμους αιτιακής ανακάλυψης, διατηρώντας παράλληλα στιβαρότητα, ειδικά σε περιπτώσεις μη συνθετικών δεδομένων. Τα ευρήματά μας υπογραμμίζουν επίσης υποσχόμενες κατευθύνσεις προς την ενσωμάτωση παρεμβατικών δειγμάτων και εκ των προτέρω γνώσης, προωθώντας περαιτέρω την ανάπτυξη θεμελιωδών μοντέλων για αιτιακή ανακάλυψη.

This page intentionally left blank

Contents

This page intentionally left blank

List of Tables

This page intentionally left blank

List of Figures

This page intentionally left blank

Preface

"A journey of a thousand miles begins with a single step."
– Tao Te Ching, *Lao Tzu*

This thesis marks the culmination of a long journey into the world of causality. Work presented here builds upon collaborative efforts between the Foundation for Research and Technology Hellas in Heraklion and Huawei Ireland Research Centre in Dublin, Ireland. It forms parts of a broader initiative that brought together academic research and industrial perspectives on developing robust causal discovery methods for real-world applications.

Writing these pages has proven to be both a technical challenge and a personal journey, including multiple setbacks, experimental loops as well as positive surprises and breakthroughs. I hope that this work will not only contribute to advancing robust causal discovery methods for decision making, but also inspire further research in building machine learning systems that can reason about the world, instead of merely recognizing patterns.

Acknowledgements

I would first like to thank my advisor for guidance and pointers towards this work, on a novel line of research that was not without its own challenges and struggles. His vision has shaped much of the presented material. A special thanks goes to Dr. Nikolaos Gkorgkolis, as both a collaborator and postdoctoral mentor, whose insight and directions are evident along these pages. I am grateful for our thoughtful discussions, creative ideas, countless meetings, software engineering, research writing and planning. I also want to thank all members of the AIOps team at Huawei Ireland Research Centre: Wang MingXue, who led this collaboration; Bora Caglayan, Andrea Tonon and Dario Simionato, for bridging the gap between the two teams with patience. Working with such talented people has been a rewarding aspect of this voyage. Finally, I want to express my deepest gratitude to my family for their unwavering love and encouragement. They have always been next to me along this journey. This includes my mother Sofia, my father Michalis and my brother Christos. Nothing in my life would have been possible without them.

Publications

This thesis is based upon the following papers:

1. *Temporal Causal-based Simulation for Realistic Time-Series Generation*
Nikolaos Gkorgkolis*, Nikolaos Kougioulis*, MingXue Wang, Bora Caglayan, Andrea Tonon, Dario Simionato & Ioannis Tsamardinos
(ArXiv Preprint: arXiv:2025.01.01)

2. *Large Causal Models for Temporal Causal Discovery*
Nikolaos Kougioulis, Nikolaos Gkorgkolis, MingXue Wang, Bora Caglayan, Andrea Tonon, Dario Simionato & Ioannis Tsamardinos
(in preparation)

*Authors contributed equally to this work.

1

Introduction

"Begin at the beginning, the King said, very gravely, and go on till you come to the end: then stop."

– Alice in Wonderland, *Lewis Carroll*

1.1 The Setting

Modern data engineering and analysis techniques, spanning fields from classic statistics to medical research, rely heavily on predictive modeling. They revolve around examining observed datasets and building predictive models on outcomes of interest, based on a set of predictive features. A fundamental limitation of predictive models is their inability to capture the underlying generative mechanisms driving the relationships between the observed variables, relying purely on correlation statistics from observational data.

The well-known adage, *Correlation does not imply causation*, exemplifies the fundamental challenge of causality, highlighting the need to move beyond predictive modeling and embrace causal reasoning. Importantly, the understanding of causal relationships is essential for answering "what-if" and "why" questions that go beyond mere prediction and association. In domains such as healthcare and climate change, one needs to know how an intervention (a change of the internal mechanisms of the examined system) will affect an outcome of interest, or even why a particular outcome occurred under certain observed conditions. In the end, for classical statistical and machine learning (ML) models, it all boils down to leveraging large portions of associational information and past experience from observational data, in order to construct, informally speaking, a pattern recognition system.

Although so far in history predictive models have proven more than adequately robust for a wide range of applications, distributions of samples are assumed to be independent and identically distributed (i.i.d.), thus drawn from a single, unchanged distribution (samples are obtained under the same, unaltered experimental scenarios). What's more, understanding the underlying mechanism between cause and effect enables causal models to cover a wide range of distributions: By altering experimental conditions with interventions, one is able to uncover the true propagation of causation and perform reliable

predictions under environmental changes. On the contrary, although statistical models have been proven robust for a variety of applications, they only account for modeling one general population distribution and are thus less interpretable.

As an example, consider predicting ice cream sales based on the number of sunburn cases in a country. There clearly exists a correlation between these two observed quantities, so one may construct a model to predict ice cream sales based on its linear or non-linear relationship with sunburn cases. However, we generally know (taking it as expert knowledge) that the number of sunburn cases is not a cause of ice cream sales, but rather a result of the sun's radiation in the summer period. Allowing some gentle introduction to the appropriate terminology and notation, we qualitatively represent causal relationships of an underlying system using *Directed Acyclic Graphs (DAGs)*[1], where observed variables are depicted as nodes and a directed edge $X \to Y$ is interpreted as "X is a direct cause of Y" , while "Y a direct effect of X". With this in mind, the causal DAG described before is Sun Radiation $\to IceCreamSales$ and Sun Radiation $\to$ Sunburn Cases.

Now imagine an alternate reality where the whole population is educated such that everyone applies adequate sunscreen before heading to the beach during summer. Using a model that performs its prediction trained on the false relationship discussed earlier in this modified population environment is guaranteed to produce highly inaccurate outcomes. This problematic case lies in the presence of the *unobserved (latent) confounding variable* Sun Radiation, which causally influences both Sunburn Cases and Ice Cream Sales (its direct children in the causal DAG). More often than not, creating a model that accounts for all possible eventualities is either impossible or intractable due to time or moral constraints. Of course, not all domains have been hindered by the observations we outlined, such as natural language processing (NLP), which has seen remarkable progress in recent years. Yet, our discussion so far further highlights the need for incorporating causality-based methods in machine learning algorithms to obtain a deeper understanding of examined systems. As Schölkopf et al. [2021] point out:

> *"If we wish to incorporate learning algorithms into human decision making, we need to trust that the predictions of the algorithm will remain valid if the experimental conditions are changed."*
>
> – Schölkopf et al. [2021]

As Reichenbach points out in the *common cause principle* [Reichenbach, 1956], if two variables X and Y are statistically dependent, then either (i) X causes Y, (ii) Y causes X or importantly (iii) *they share a common cause Z (a confounder $X \leftarrow Z \to Y$) that explains their association.* This principle lies at the heart of why observational data can be misleading: without accounting for all relevant variables, we risk attributing causal meaning to mere correlations. A classic illustration of this issue is *Simpson's Paradox* [Simpson, 1951], where trends apparent in aggregated data are reversed when disaggregated by a confounding variable. For instance, a treatment might appear beneficial in both men and women when analyzed separately, yet detrimental when the data is pooled together, due to the unequal distribution of underlying risk factors (e.g.

[1] Other graphical structures exist, but are outside the scope of this work.

a factor prominent in women).

Additionally, modern machine learning tools such as *feature importance scores* and *Shapley (SHAP) values* [Lundberg and Lee, 2017] often provide insights into which variables most strongly influence a prediction. Unfortunately, these measures are fundamentally *associational* and non-causal: they capture how much a feature contributes to predictive accuracy within the model, but not whether the feature exerts a causal effect on the outcome. Consequently, they remain vulnerable to the same confounding phenomena highlighted previously: A variable may appear to have high importance or a strong SHAP attribution simply because it is correlated with an unobserved cause, rather than because it directly drives the outcome.

Table 1.1: Comparison of different types of causal queries.

Type	**Notation**	**Interpretation**	**Data Source**
Observational	$\mathbb{P}(Y \mid X = x)$	How does Y vary when we observe $X = x$?	Observational
Interventional	$\mathbb{P}(Y \mid \text{do}(X = x))$	What happens to Y if we intervene and set $X = x$?	Interventional (RCTs, simulations)
Counterfactual	$\mathbb{P}(Y_{X=x'} \mid X = x, Y = y, Z = z)$	What would Y have been if $X = x'$, given we saw $X = x, Y = y, Z = z$?	Model-based (SCMs + Abduction)

Given our discussion so far, how would one proceed to measure causal effects correctly? The gold standard since its introduction by Ronald Fisher [Fisher, 1935], has been with randomized experiments and more specifically with *Randomized Control Trials (RCTs)*, where a subset of the examined population is randomly split into *control* and *treatment* groups, where one more more variables are deliberately manipulated (intervened) to isolate the causal effects being measured. As an illustrative example, consider a clinical trial where one needs to decide whether the newly introduced pill X prevents migraines. An RCT experiment would imply selecting a random patient sample, randomly splitting them into control (placebo pill) and treatment (pill X) groups and measure the causal effect of a quantity of interest (e.g. migraine pain level) on each group separately. The real magic of such controlled experiments lie in their ability to eliminate *confounding bias* through randomization. By balancing both observed and unobserved covariates across groups, RCTs ensure that differences in outcomes are attributed solely to the intervention and not to systematic distortions caused by pre-existing conditions. In digital marketing and advertising campaigns, RCTs take the form of *A/B tests*: users are randomly shown one of two versions of a webpage or a smartphone app (A or B) to determine which version has a greater causal effect on a desired outcome, such as *click-through rate* or *exposure rate* for a quantity of interest. Returning to our earlier ice cream and sunburn example, one would randomly assign a subset of the population to consume ice-cream (treatment group) and another to not (control group) and measure the causal effect of ice-cream consumption on sunburn cases. In formal terms, such a measurement can for example be the *Average Treatment Effect (ATE)*, which is viewed as the expected difference in outcomes between treated

and untreated populations: ATE $= \mathbb{E}[Y_1 - Y_0]$ where Y_1 and Y_0 represent the potential outcomes under treatment and control, respectively. If this difference is statistically distinguishable from zero, one concludes that the treatment has a causal effect on the outcome.

However, controlled experiments like RCTs are often impractical due to cost, ethical constraints or logistical challenges, with many corporations running hundreds of them each day on their products. For instance, it would be prohibitive to force a population sample (including healthy individuals) to smoke for investigating whether smoking causes lung cancer. What is more, the space of possible A/B tests is combinatorial when one is considering combinations of several design choices or user behaviors, thus being unfeasible in practice.

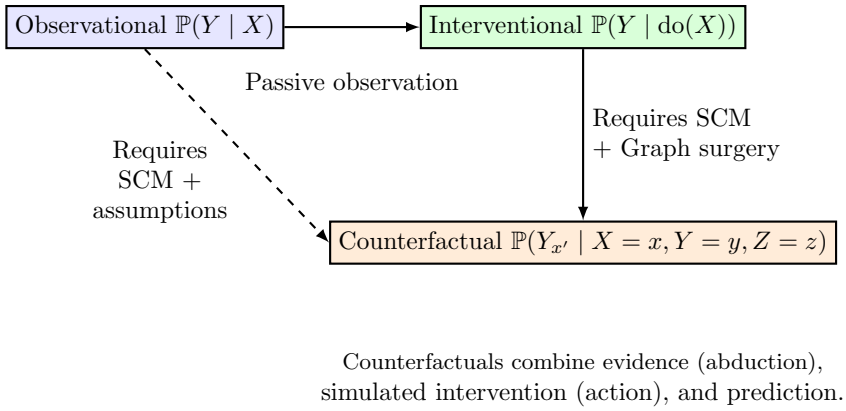


Figure 1.1: Illustration of the relationships and requirements between observational, interventional, and counterfactual causal queries.

This is where the formal framework of *structural causal models (SCMs)* becomes essential. SCMs, introduced by Judea Pearl [Pearl, 2009] who pioneered the development of the theory of Causality (Turing Award 2011), offer a mathematical representation of the causal mechanisms that generate a dataset. In brief, an SCM consists of (i) a graph (in this text, a DAG) $\mathcal{G}$ structure encoding the underlying causal relationships (ii) a set of structural equations (functional relationships) $X_i := f_i(\text{Pa}_i, \epsilon_i)$ where Pa_i are the parents of X_i in the DAG and (iii) independent noise terms ϵ_i representing unobserved randomness and possible confounders. In a nutshell, these structural equations *encapsulate how each variable in a system is generated from its direct causes (parents in the DAG).* The strength of such modeling lies in the ability to rigorously represent the causal mechanisms that generate a dataset, enabling the analysis of causal effects and interventions, and the estimation of causal quantities of interest, such as the ATE mentioned previously.

Furthermore, the concept of interventions is a fundamental operation in causal inference, formalized through *do-calculus* by Pearl [Pearl, 1995]. The notation $\mathbb{P}(Y|\text{do}(X = x))$ denotes the distribution of Y after an intervention on X, directly setting its value to x, irrespective of its natural causes. Such generated datasamples are called *interventional data* in contrast to *observational data* that are obtained from a purely observational perspective of the system. In terms of the SCM, this corresponds to *graph surgery*: removing all incoming

edges into X and replacing its value with a constant. This operation gives rise to a new graph, called the *mutilated graph*, which simulates a world where we have forced X to be x. This distinction is crucial because in observational data, one only has access to the conditional observational distribution $\mathbb{P}(Y|X = x)$ which may be influenced by confounding or selection bias. On the other hand, the interventional distribution $\mathbb{P}(Y|\text{do}(X = x))$ truly captures the causal effect of manipulating X on Y. So generally, the interested reader may notice that $\mathbb{P}(Y|X = x)$ (which may represent the outcome of a predictive model) is generally not equal to $\mathbb{P}(Y|\text{do}(X = x))$[2]. Finally, a third important pillar in Causality is the concept of *counterfactuals*, where one can quantify causal queries of the form *"Given that I have observed* $X = 0, Y = 0$ *and* $Z = 1$*, what would* Y *be if* Z *had been* 0*?"*. That is, a counterfactual probability of the form $\mathbb{P}(Y_{Z=0} = y|X = 0, Y = 0, Z = 1)$. Computation of such queries is evaluated with a three-step procedure known as *abduction*, *action*, and *prediction* using the known SCM [Pearl, 2009]. Here, $Y_{Z=0}$ denotes the potential outcome representing the value that Y would have taken, in a reality where $Z = 1, X = 0$ and $Y = 0$ have been observed. One concrete example of counterfactuals is: "Knowing that Hillary Clinton gave a speech in California and did not win the 2016 presidential election, would she have won had she not given that speech?". Unlike predictive explanations, which are conditional on inputs, counterfactuals model alternative realities rooted deeply in the structural model of the world, essential for decision-making under uncertainty and policy evaluation. Figure 1.1 and Table 1.1 provides an overview of the causal queries discussed so far.

In ML literature, one also encounters *counterfactual explanations (CFEs)* [Guidotti, 2024] which are commonly used as interpretability tools. These CFEs typically answer queries of the form "What minimal change in the input features would have altered the model's prediction?". One must note that they are not counterfactuals in the causal sense: they operate within the predictive model's input-output mapping, without assuming any underlying structural causal model. As a result, suggested changes may be infeasible, non-actionable, or causally meaningless. Table 1.2 provides an overview of the differences discussed between causal and predictive modeling.

All the above positive baggage of the theory of Causality is described by Judea Pearl in the *3-step ladder of Causation* [Pearl and Mackenzie, 2018]. Any inference merely by observing associations lies in *level 0* (as in statistical predictive models). The true power of causality belongs to interventions (*level 1*) and counterfactuals (*level 2*). From everyday choices to scientific inquiry, our actions are guided by the belief that certain causes will lead to specific effects. However, causality in real-world scenarios is often probabilistic rather than deterministic. As we have mentioned before, while smoking is a known cause of lung cancer, not all smokers develop the disease. This uncertainty is also where probability theory intersects with causal modeling, allowing us to quantify the likelihood of outcomes given specific causes.

Overall, Causality can be split into two main research areas: *causal discovery (CD)*, which aims to identify the true underlying causal relationships from

[2]They would be equal in the case of no backdoor-confounding path from X to Y, but this is outside the scope of this thesis.

Query	**Description**	**Causal**	**Predictive**
Prediction	Predict/Diagnose Y given X	Correct predictions	Correct predictions
What-if	What if I set $X_1 = 5$?	Correct prediction on Y	Possibly wrong prediction on Y
Decision Making	Optimal X_1 to increase Y given $X_2 = 6$	Correct decisions	Possibly wrong decisions
Interpretation	Importance of X_1 in affecting Y; features influencing Y	Correct estimate	SHAP/feature importance may fail
Counterfactual	Example: $Y = 3$ when $X_3 =$ "yellow". What if $X_3 =$ "green"?	Correct estimate	Not possible
Root Cause	What was the initial root cause of a failure?	Correct estimate	Possibly wrong estimates

Table 1.2: Comparison of causal vs. predictive modeling across different types of queries.

data under certain assumptions and *causal inference (CI)*, which concerns estimation and quantification of treatment effects without exhaustive experimentation. Formally, given a dataset $\mathcal{D} \sim \mathcal{P}_{\mathcal{D}}$, the goal of causal discovery is to infer the underlying causal model $\mathcal{G} \sim \mathcal{D}$. Methods developed for causal inference assume the ground truth causal structure as known. In practice, one starts by estimating the ground truth causal model using a causal discovery algorithm. Depending on the method, the discovered causal model may be paired with the estimated functional dependencies of the underlying SCM (e.g. some algorithms assume that data follow a linear generative process, so causal discovery is reduced to estimating the coefficients of the assumed linear model) or simply output the estimated DAG. In either case, one can use standard predictive models to estimate the functional relationships alongside valid adjustment sets to perform causal inference. This bridges causal discovery with interventional effect estimation: identifying optimal intervention strategies, computing ATEs, or performing counterfactual reasoning on outcomes of interest, as described previously.

In recent years, significant attention has shifted on *causal discovery for time-series (temporal) data* [Runge, 2018, Runge et al., 2019, 2023]. For the unfamiliar reader, such samples are no longer considered i.i.d. as the time dimension is also present. Time-series are widely observed across a domain of environments, as long as data is collected across time intervals and exhibit a temporal order. For example, consider a time series of climate measurements (temperature, humidity, atmospheric pressure) over a day in a city, sampled every fifteen minutes. The resulting data is a sequence of three-dimensional 96 observations, and the corresponding (multivariate) time series is a time-series of length 96. Another concrete example would be a time series of stock prices, where data is collected every day for a year, resulting in a time series of length 365. For the most part, investigating causality for time-series consists as an extension of the theory developed for the static, i.i.d. counterparts, with some additional assumptions and complexities that arise from temporal dependencies, possible

feedback loops, non-stationarity of the observed time-series, and unobserved confounding variables. Variables evolve over time, and causal influences can occur both contemporaneously (within the same time-index) and with various time lags (e.g. one day before). Without elaborating extensively in this part of the text, a *contemporaneous* (or *instantaneous*) relationship may occur if causal effects exist within an hourly period in an observed daily time-series.

Concurrently and much to our benefit, there has been no previous period for ML and artificial intelligence (AI) with as much bloom and research opportunities. The current wave of interest, initiated with ML more than two decades ago and continued with *deep learning (DL)* (the use of large neural network architectures for solving highly-complex, intractable tasks) has provided several radical solutions to numerous research questions, such as image recognition, that no previous AI period ever did. The witnessing of problems that have puzzled the scientific community for decades being deemed solved, like text-to-image generation, is an ongoing process. It is possible that since this ongoing take-off, the AI winter may not be coming anytime soon.

Even more remarkably, causality for time-series data has only recently found large-scale applications in industrial domains, many years after Pearl's seminal contributions. Terms such as *Causal AIOps* and *Causal AI* correspond to complementing traditional machine learning with causal discovery & causal inference methods, often based around modern neural architectures for industrial applications. Much to our excitement, *Causal Digital Twins* [Jakovljevic et al., 2021] are also a promising direction, where a causal model is constructed from data to model a real-world system, to perform real-time causal inference and decision making. This enables tasks such as the identification of optimal interventional policies, which are crucial for understanding and correctly interpreting data-driven decisions. This paves exciting research paths for developing new state-of-the-art (SOTA) methods for causality on time-series for solving real-world problems.

1.2 Motivation

The field of causal discovery has undergone an important paradigm shift with the introduction of deep learning-based approaches. A landmark contribution is the NOTEARS paper by Zheng et al. [2018], which recast causal discovery as a continuous optimization scheme with acyclicity constraint, by relaxing the combinatorial nature of the problem. This opened the door to a whole class of differentiable causal discovery methods. In the i.i.d. setting, this led to the introduction of works such as DECI [Geffner et al., 2024], AVICI [Lorch et al., 2022] and CSIvA [Ke et al., 2023], which leverage modern deep neural architectures and approaches to infer causal structure from data. Various works for temporal (time-series) data soon followed, with methods such as TCDF [Nauta et al., 2019], ACD [Löwe et al., 2022] and Rhino [Gong et al., 2022] among others. Assaad et al. [2022] and Deng et al. [2022] provide a self-contained review on works for temporal causal discovery, for the interested reader.

Despite their promise, existing approaches face a number of limitations. Classical causal discovery methods for time-series based on conditional independence testing, such as PCMCI [Runge, 2018], suffer from scalability issues due to the exponential number of independence tests required. While these methods

are provably sound if infinite sample size is available [Spirtes et al., 2001], they often fail to generalize in reduced, finite-sample regimes. Score-based methods, which search over graph structures using data likelihood or other criteria, such as DYNOTEARS [Pamfil et al., 2020], can underperform when functional dependencies are complex or nonlinear, on the advantage of faster convergence (as linearity of causal relationships is assumed). Importantly, all of these methods operate on a single input-output pair at a time, which limits their ability to learn from multiple samples and causal graphs of varying sizes, complexities and dynamics. As a result, the challenge of building causal discovery methods that generalize across data distributions and promise high scalability to realistic time-series scenarios remains unresolved.

Meanwhile, the rise of *foundation models* has revolutionized several domains of machine learning. In a nutshell, a foundation model (FM) refers to a pre-trained neural model, trained on large corpora of data [Bommasani et al., 2021]. In natural language processing, the introduction of BERT [Devlin et al., 2019] marked a turning point by demonstrating the power of pretraining Transformer-based [Vaswani et al., 2017] models on massive data corpora, in the context of text translation. Previous works, being reliant on recurrent neural networks (RNNs), have been proven limited by design, in contrast to the Transformer architecture which has efficiently transformed the field of NLP. This was followed by the GPT family, culminating in models like GPT-4 [Achiam et al., 2023], which use hundreds of billions of parameters to support robust zero-shot[3] generalization [Radford et al., 2019]. Foundation models have also emerged in vision and multimodal learning, e.g. CLIP [Radford et al., 2021] and DALL-E [Ramesh et al., 2021], exhibiting remarkable generalization across diverse downstream tasks. A unifying principle behind these models is that *massive pretraining on varied data can lead to robust representations and inference capabilities without any task-specific supervision.* Recent work has extended this idea to time-series forecasting. Models such as TiMeR and TiMeR-XL [Liu et al., 2024] adopt decoder-only Transformer architectures to autoregressively model temporal sequences. MOIRAI [Woo et al., 2024] uses a masked encoder and patch-based input for forecasting, with model sizes ranging from 14M to 311M parameters. Google's TimesFM [Das et al., 2024] follows a similar patch-based decoder design, with a 200M parameter footprint. These models demonstrate strong *out-of-distribution (OOD)*[4] forecasting capabilities, suggesting that general-purpose foundation models for temporal data are viable. An overview of these foundation models, along with their sizes, is shown in Table 1.3.

Our work aims to extend this line of research towards causal discovery for time-series data. The concept of *Large Causal Models (LCMs)* envisions a class of deep, pre-trained neural architectures specifically designed for temporal causal discovery. These models are trained on a diverse corpus of multivariate time-series sequences, paired with their corresponding ground-truth structural causal models, with a single aim: *learning robust universal representations capable of performing zero-shot temporal causal discovery with refined lag estimation from arbitrary time-series samples* (Figure 1.2).

Unlike existing temporal state-of-the-art methods, which are often constrained

[3]The ability of a model to generalize to data that has not been included in training, i.e. does not follow the distribution of the training data.

[4]Performance on datasets that do not follow the distribution of the training data, also known as *zero-shot* in the context of foundation models.

Table 1.3: Comparison of foundation model sizes across NLP and time-series forecasting domains.

Model	**Domain**	**Parameters**
BERT [Devlin et al., 2019] (Base)	NLP	108M
GPT-1	NLP	117M
Timer [Liu et al., 2024]	Time-series Forecasting	84M[5]
MOIRAI (Small / Base / Large) [Woo et al., 2024]	Time-series Forecasting	14M / 91M / 311M
TimesFM [Das et al., 2024]	Time-series Forecasting	200M[6]

to small numbers of variables and suffer from significant performance degradation as input dimensionality and model size increases, we showcase robust generalization of our LCMs across varying numbers of variables, time-series lengths, and temporal dynamics. By leveraging both synthetic and *simulated* (realistically generated) datasets, our models aim to capture universal causal structures that are robust across domains, time scales, and data distributions. Moreover, causal discovery using LCMs is performed via a single forward pass over input data, making it significantly faster and scalable than traditional approaches, opening the door for real-time applications such as causal digital twins for decision-making and control.

Constructing foundation models for causal discovery raises several central challenges. First, input dimensionality (as in all pre-trained models) must be carefully considered: the maximum number of variables and timesteps directly impacts both feasibility and generalization. Inputs that are too small limit applicability, while overly large inputs may hurt performance. Second, training data quality and diversity are crucial. While existing approaches rely solely on synthetic data generated from arbitrary SCMs [Stein et al., 2024], purely synthetic datasets fail to capture the full complexity of real-world processes. To make matters worse, the availability of ground-truth causal structures from the real-world is scarce to non-existent. *Semi-synthetic data*, based on existing knowledge of real scenarios (such as physical models or approximations of them by stochastic dynamical systems) can offer a pragmatic compromise, but a principled methodology for generating high-quality, diverse, and realistic SCM time-series pairs at scale remains a key open challenge; one which we address in Chapter 3.

Figure 1.2: A pre-trained Large Causal Model (LCM) infers a causal graph from an unknown multivariate time-series sequence. (Training uses thousands of pairs of time-series and their corresponding ground-truth causal graphs under a supervised learning scheme.)

1.3 Related Work

This section provides an overview of recent work connecting causality with deep learning, focusing particularly on attention-based and foundation-model approaches. We highlight methods not only on causal discovery but also on causal inference, as they provide valuable insights for our own work.

1.3.1 Causal Inference with Attention

The integration of causal inference into deep learning models remains relatively nascent. One notable effort is *Causal Inference with Attention (CIvA)* by Zhang et al. [2023], which introduces a self-supervised transformer-based architecture for estimating Average Treatment Effects (ATEs) from observational data. Their approach leverages a primal-dual link between covariate balancing and self-attention, enabling the model to generalize under moderate shifts in causal mechanisms. While effective for ATE estimation, CInA is not designed for causal graph discovery and does not address temporal data.

1.3.2 Do-PFN

A very recent advancement in causal inference is *Do–PFN* by Robertson et al. [2025], which adapts Prior-data Fitted Networks (PFNs) for causal effect estimation. Rather than discovering the causal structure, Do–PFN focuses on learning to predict interventional outcomes (i.e., conditional interventional distributions, or CIDs) directly from observational data. The model is pre-trained on millions of synthetic datasets derived from randomly sampled SCMs, learning a meta-model capable of in-context causal inference without requiring knowledge of the ground-truth causal graph. While Do-PFN demonstrates impressive performance in both synthetic and semi-real evaluations, including frontdoor and backdoor adjustment cases, it is limited to point estimation tasks such as CID or CATE prediction. Importantly, Do–PFN does not attempt to recover or represent the underlying causal graph or temporal dynamics, which makes it orthogonal to our objective of structure discovery. Moreover, Do-PFN operates on i.i.d. tabular data and does not consider multivariate time-series or lagged causal dependencies. It also scales to graphs up to 10 nodes and degrades with noise, yet improves with dataset size. Do–PFN uses an explicit prior over SCMs and amortizes inference across them. It also captures uncertainty due to unidentifiability, i.e., integrating over model priors.

1.3.3 Sample, Estimate, Aggregate

In the causal discovery domain, Wu et al. [2024b] propose a method that aggregates subgraph-level causal estimates using axial attention alongside global statistical features such as inverse covariance. Although innovative, their method is limited to static (i.i.d.) data and does not extend to time-series or interventional settings.

1.3.4 CSIvA

Similarly, Ke et al. [2023] (with *Causal Structure Induction via Attention - CSIvA*), treat causal discovery as a supervised learning task using an autoregressive Transformer decoder with alternating attention, but their framework also targets non-temporal data only and is trained on synthetic SCMs.

1.3.5 Causal Pretraining

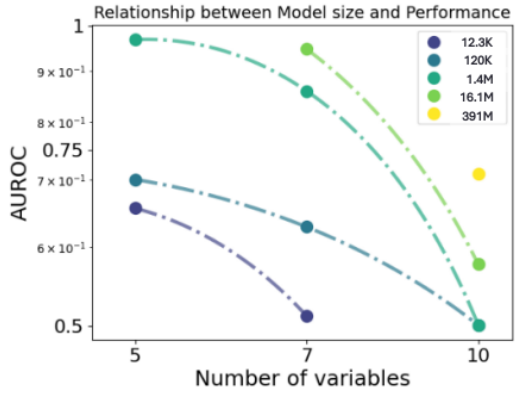


Figure 1.3: Illustration of the performance of Stein et al. [2024] models on the AUC-ROC metric, as a function of input dimensionality and model size. As input dimensionality and model size increase, introduced models show diminishing performance. Model sizes in the legend are defined as in Stein et al. [2024], i.e. from `12.3K` to over `391M` parameters, corresponding to observational input dimensions of 5 variables and 3 lags. Figure adapted from Stein et al. [2024, Figure 3].

Closest to our objectives is the recent work of Stein et al. [2024], who propose a collection of causal discovery models which they call *Causal Pretrained Neural Networks (CPNNs)*. They introduce several architectures, from RNNs to Transformers [Vaswani et al., 2017], and provide pre-trained weights of what they call `deep` model size. The transformer model (which expresses the current state-of-the-art), based on the Informer [Zhou et al., 2021] architecture, utilizes full attention (instead of sparse attention) and predicts temporal causal graphs from observational time-series data.

Although it serves as the first work on a foundation model-based approach for temporal causal discovery, it faces several limitations: models are trained solely on synthetic data, constrained to low input dimensionalities (up to 5 variables), serving more as a proof-of-concept rather than a complete generalizable model. This is reflected in the diminishing performance of their introduced models as input both input dimension and model parameters increase, to a point where they perform close to a random guessing[7] baseline (Figure 1.3 and Stein et al. [2024, Figure 3]). Additionally, provided models do not showcase strong

[7]The existence or not of an edge between two variables can be viewed as a coin flip with a 50% chance of success.

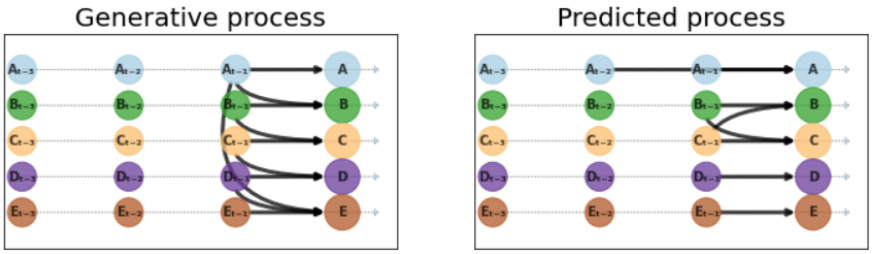


Figure 1.4: Out-of-distribution performance on fMRI semi-synthetic data of the provided transformer-based CPNN. Average True Positive Rate is 0.68. Figure generated using the relevant plotting and inference code provided by the authors.

generalization abilities to out-of-distribution (OOD) data (Figure 1.4), which is a key feature of our proposed models. Importantly, the above observations *do not represent what we envision as a large causal model* and pave the way to many research gaps in the field which we address in this text.

1.4 Positioning of Our Work

Our proposed large causal models (LCMs) extend the foundation model paradigm for temporal causal discovery along three main axes: scalability, data realism, and pretraining strategy. Building on Stein et al. [2024], we develop robust, deep models capable of handling higher-dimensional and complex datasets, evaluating generalization in out-of-distribution scenarios.

1. Scalability and Generalization: We demonstrate that temporal causal discovery can be effectively scaled beyond prior limits (up to twelve (12) variables and three (3) lags) without any loss of performance. LCMs maintain robustness under out-of-distribution conditions and support deeper architectures trained on large data corpora.

2. Realistic Training: We introduce methods for curating high quality training samples, combining synthetic SCMs with realistic causal model generation using the Temporal Causal-based Simulation (TCS) process. This approach bridges the gap between synthetic pretraining and real-world deployment.

3. Foundation Model-Style Training: We adopt a robust pretraining regime inspired by foundation models, enabling LCMs to learn general causal representations from heterogeneous datasets and adapt zero-shot to new domains.

To our knowledge, LCMs are the first models to combine temporal causal discovery, scalable foundation-style training, and robust out-of-distribution generalization by training on mixtures of synthetic and realistic data.

Furthermore, we present promising investigations into not only a novel architecture but also on the use of interventional data and incorporation of prior knowledge. While these directions are motivated by theoretical considerations

and promising early evidence, they are presented more as ongoing work rather than established results. Their inclusion reflects our broader goal of developing robust and generalizable causal foundation models for temporal causal discovery.

1.5 Guide to this Thesis

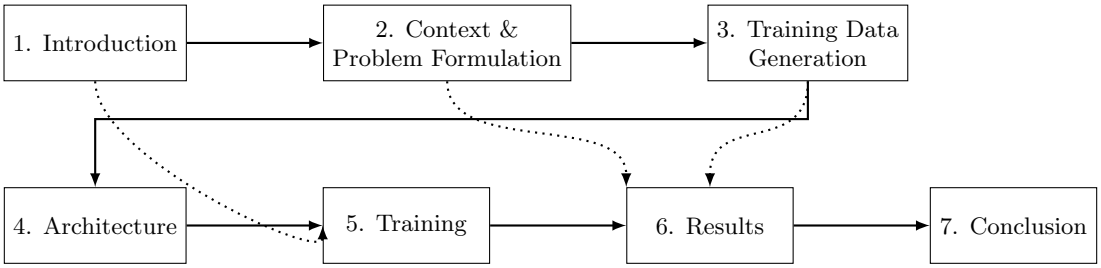


Figure 1.5: Roadmap of this thesis. Solid arrows denote the main sequential flow of chapters. Dotted arrows indicate supporting or cross-referenced dependencies between chapters.

This section provided a high-level overview of our text. Chapter 2 serves as an introduction to needed background, mainly in Causality and briefly to Deep Learning. It introduces Structural Causal Models (SCMs), their temporal extensions and known assumptions. Concerning Deep Learning, we provide a short overview on Deep Learning terminology for the uninitiated reader and elaborate on the Transformer architecture, which forms the backbone of our LCM, and on building blocks of neural nets that we utilize later on. In section 2.6, we formally define the main objective of our work. Chapter 3 details our synthetic data pipeline and the Temporal Causal Simulation (TCS) & Adversarial Causal Tuning (ACT) methods for generating realistic data pairs for training. Chapter 4 is dedicated to the architecture of our LCMs and input handling. Chapter 5 discusses training objectives, optimizers, and regularization methods that are essential for training of LCMs. Chapter 6 evaluates trained LCMs against baselines, with ablation and generalization studies. Chapter 7 concludes the text with contributions, limitations, and future directions. We illustrate a roadmap of this thesis in Figure 1.5.

This page intentionally left blank

2

Context & Problem Formulation

"As X-rays are to the surgeon, graphs are for causation."
– Judea Pearl

The main goal of this chapter is to underpin the essential theoretical background and terminology needed to formally define our work: Causal discovery from time-series data using a foundational neural approach (our LCMs). The purpose of this part is to serve as a conceptual and notational bridge between the high-level motivation discussed in Chapter 1 and the following chapters. We focus on Temporal Structural Causal Models (TSCMs) and key assumptions needed for principled causal discovery. Rather than providing a complete treatment of the theory of Causality, our aim is to introduce the necessary formalism tailored to our own setting. For readers seeking a broader perspective on causality, we refer to the seminal works by Pearl [2009], Pearl and Mackenzie [2018], Spirtes et al. [2001] and Peters et al. [2017]. For the case of causality and causal discovery in time-series, we point the reader to the works of Runge [2018], Runge et al. [2023], Assaad et al. [2022], Gong et al. [2024]. We also elaborate on some important aspects of deep learning, as it also serves as an integral part of our work. For foundational reading, we refer to the textbook by Goodfellow et al. [2016]. Section 2.1 introduces Structural Causal Models, which are then extended to Temporal SCMs (TSCMs) in Section 2.2, while Section 2.4 outlines the assumptions necessary for causal identifiability and model interpretation. Section 2.5 provides a short overview of the necessary deep learning terminology for the unfamiliar reader. Finally, Section 2.6 defines the main objective of our work.

2.1 Structural Causal Models

The framework of Structural Causal Models is fundamental to Causality, as it represents the underlying causal mechanisms that govern an examined system. In order to start formally expressing X *causes* Y, one needs to define the observed variable Y as a function of X. The notation $:=$ is introduced to

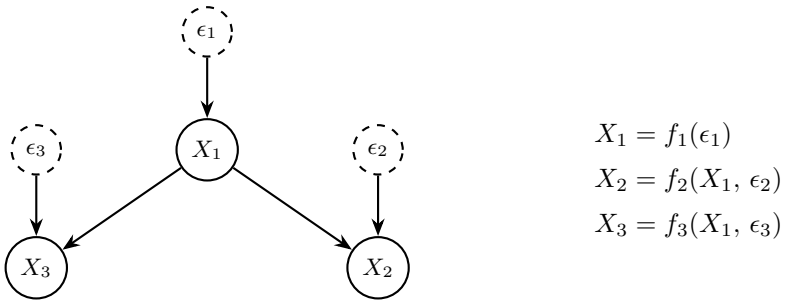


Figure 2.1: Illustration of a structural equation model for an inverse collider structure $X \leftarrow Z \rightarrow Y$. Each variable is a function of its direct causes and a noise term.

emphasize the asymmetric nature of this relationship, rather than merely representing an algebraic equality. That is, X causes Y, but not vice-versa. In the SCM framework, a variable Y is modeled as $Y := f(X, \epsilon)$, where f is a deterministic (parametric or non-parametric) function, representing the causal mechanism and ϵ an exogenous noise term which captures, among others, latent, unobserved factors and inherent randomness [Peters et al., 2017]. These noise variables are assumed to be mutually independent across variables and introduce the necessary stochasticity needed to model the causal mechanisms. Assuming no self-causation (a variable causing itself), the relationships between observed variables are qualitatively represented by a *Directed Acyclic Graph (DAG)* where the direct causes of a variable X are the parents of X in the DAG. As such, an edge $X \rightarrow Y$ in the DAG encodes the causal information that X *directly causes* Y. For example, consider the *collider* structure $X \rightarrow Z \leftarrow Y$, where X and Y share the same direct effect. This can be described by the equations $Z := f(X, Y, \epsilon)$, $X \sim \epsilon_X$, $Y \sim \epsilon_Y$ where ϵ_X, ϵ_Y are independent noise terms (e.g. sampled from a normal distribution) of X and Y respectively. An illustration is shown in Figure 2.1. If one ignores any causal interpretation of the functional mechanisms, this representation turns into a *Structural Equation Model (SEM)*. Formally, we define an SCM as a tuple of (i) a set of *endogenous* variables $\mathcal{V}$, a set of *exogenous* variables $\mathcal{U}$ and the set $\mathcal{F}$ of functions of the form $f \equiv f(\text{Pa}(V_i), \epsilon_i)$, where $\text{Pa}(V_i)$ denotes the direct causes of V_i (parents of V_i in the DAG), to generate each endogenous variable as a function of other variables. In practice, the functions f_i are often assumed to follow the form of an *additive noise model (ANM)*. In this case, the functional relationships are of the form

$$V_i := f(\text{Pa}(V_i), \epsilon_i) = f(\text{Pa}(V_i)) + \epsilon_i \tag{2.1}$$

which simplifies estimation and data generation, while still allowing for representing a rich class of linear and non-linear causal mechanisms. In a nutshell, this serves as just another assumption to facilitate causal identifiability and inference.

2.2 The Temporal Setup

The case for time-series data can be viewed as a natural generalization of the above, but taking into account the additional temporal dimension of data. Let $\{\mathbf{V}_t\}_{t\in\mathbb{Z}}$ denote a multivariate stochastic process, with $V_t = (V_t^1, \dots, V_t^N)$ representing the state of the system at time t. A *Temporal Structural Causal Model (TSCM)* [Runge, 2018] consists of the structural assignments

$$V_t^j \;:=\; f_j\big(\mathrm{Pa}(V_t^j),\, \epsilon_t^j\big), \qquad j = 1, \dots, N, \;\; t \in \mathbb{Z}, \tag{2.2}$$

where for each variable index j, ϵ_t^j is an independent exogenous noise term, $\mathrm{Pa}(V_t^j) \subseteq \{V_{t-\tau}^i : i = 1, 2, \dots, N,\ \tau = 0, \dots, , \ell_{\max}\} \setminus \{V_t^j\}$ denotes the *causal parents* of V_t^j occurring at the same (if the existence of contemporaneous effects is assumed) or earlier time steps $t-1, t-2, \dots$, up to a finite maximal lag $\ell_{\max} > 0$. Analogously to the atemporal case, each function f^j is the functional deterministic causal mechanism that determines the value of V_t^j given the direct causes $\mathrm{Pa}(V_t^j)$ and the corresponding noise term ϵ_t^j. The TSCM can then be written as a tuple $(\mathcal{G}, \mathcal{F}, \mathcal{E})$ where $\mathcal{G}$ is the causal graph, $\mathcal{F}$ is the set of functional dependencies f^j and $\mathcal{E}$ is the set of noise terms ϵ_t^j.

The causal parents $\mathrm{Pa}(V_t^j)$, also called *direct causes*, are selected from the set $\{V_t^j, \dots, V_{t-\ell_{\max}}^j\}$. This formalism encodes both lagged and contemporaneous causal effects ($\ell_{\max} = 0$) to be represented, making it suitable for modeling a wide range of dynamical systems. To ensure well-posedness, we restrict all parent sets to occur at most $\ell_{\max}$ time-steps in the past, thus disallowing backward in-time causation. Recall that contemporaneous edges $V_t^i \to V_t^j$ may occur if causal effects exist beyond the presumptive granularity (e.g. hourly causal mechanisms against a daily assumed lag). Furthermore, it is assumed that the temporal process is governed by a finite horizon of causal interactions and that each variable has a bounded number of causal inputs. Again, the functional relationships between the observed variables are represented by a DAG where edges move forward in time. As a concrete example, consider an observed process with the time-series V^1, V^2 and V^3 with the following causal relationships: Variable V^1 directly causes variable V^2 with lag 1 and V^2 directly causes V^3 with lag 2. As the process extends through time, so do the edges of the causal graph. The process (and as such the causal edges) may evolve, which makes the causal graph time-dependent. That is, different causal graphs may correspond to different time horizons and time windows. This pivots towards our first definition of the most important causal assumption for time-series data: *Causal Stationarity.*

Definition 1 (Causal Stationarity, Runge [2018]) *Consider an SCM described as in Equation 2.2. If the causal relationships between variables $(V_{t-\tau}^i, V_t^j)$ for lag $\tau > 0$ also hold for all time-shifted versions $(V_{t'-\tau}^i, V_{t'}^j)$, the described process is causally stationary. Informally, the graph structure and noise distribution of the SCM are time-invariant.*

Using the previous example, that means that, for the current timepoint t, there exist causal edges such as $V_{t-1}^1 \to V_t^2$ and $V_{t-2}^2 \to V_t^3$. By causal stationarity, these relations also hold for their time-shifted versions: $V_{t'-1}^1 \to V_{t'}^2$ and $V_{t'-2}^2 \to V_{t'}^3$ for all $t' \leq t$. Like in the atemporal case, indirect causation is

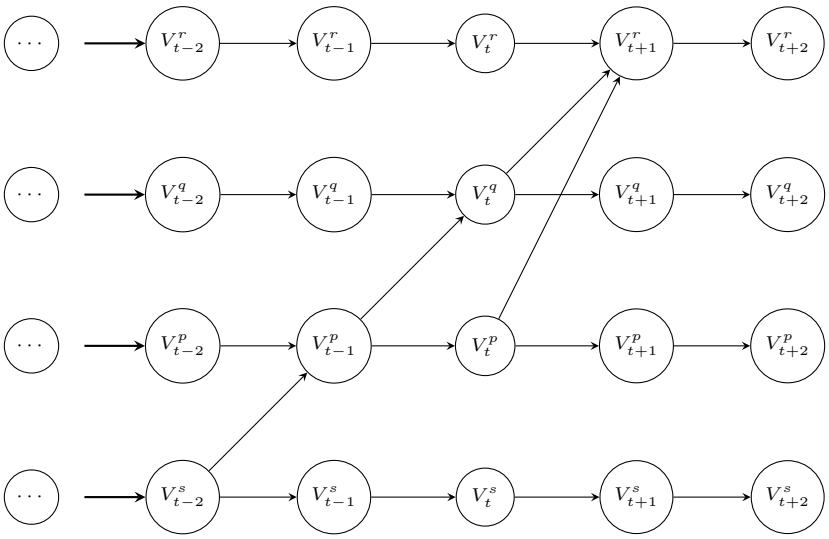


Figure 2.2: Full time causal graph: The temporal SCM extends infinitely into the past (indicated by $\cdots$ and incoming arrows) and shows both auto- and cross-variable lagged dependencies. Notice that in this example, no causal stationarity is assumed.

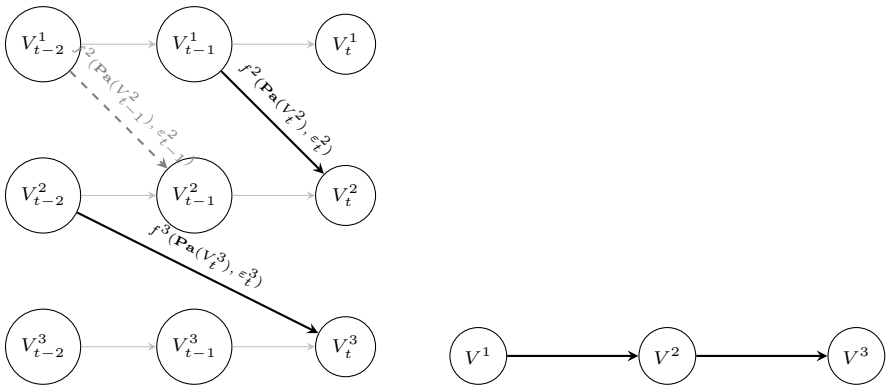
implied through directed paths. In literature, this invariance of causal effects in time is also referred to as time homogeneity [Gong et al., 2024].

The causal dependencies implied by a TSCM can be visualized as a directed graph unrolled over time. The cautious reader may have observed that due to the temporal axis, multiple graph abstractions are possible depending on the intended level of granularity. We first define the most complete representation, the *full-time causal graph*:

Definition 2 (Full-Time Causal Graph) *Let $\{\mathbf{V}_t\}_{t\in\mathbb{Z}}$ be a multivariate stochastic process governed by a Temporal SCM. The full-time causal graph is the infinite directed acyclic graph (DAG) whose nodes are all variables V_t^i for every $i \in \{1, \dots, N\}$ and every time step $t \in \mathbb{Z}$, and whose edges correspond to direct causal relationships $V_{t-\tau}^i \to V_t^j$ as defined by the TSCM. This graph includes both within-variable (autoregressive) and cross-variable temporal dependencies across all lags up to $\ell_{\max}$.*

Figure 2.2 illustrates such a structure, where both temporal self-dependencies and cross-variable influences are made explicit. This infinite graph represents the most faithful unfolding of the SCM in time. If causal stationarity is not assumed (as illustrated), then different causal graphs correspond to different time horizons, further complicating the causal discovery task. Secondly, the full-time causal graph can be truncated to a specific time-horizon, resulting in the *time-lagged (also called windowed) causal graph*, with causal edges up to $\ell_{\max}$.

Definition 3 (Lagged Causal Graph - Window Causal Graph) *A lagged causal graph is a directed acyclic graph (DAG) that represents causal relationships between time-shifted variables in a multivariate time-series. Formally, let*



(a) Lagged causal graph (b) Summary causal graph

Figure 2.3: Illustration of a temporal SCM with max lag $\ell_{\max} = 2$. (a) The lagged causal graph shows variable V^1 causing V^2 with lag 1, and V^2 causing V^3 with lag 2. Dashed edges encode the assumption of causal stationarity, meaning that the same causal dependencies recur over time. Functional dependencies between lagged variables are shown directly on edges. (b) The summary causal graph discards time lag information, representing only the aggregated causal influences.

$\mathbf{V}_t = \{V_t^1, V_t^2, \ldots, V_t^k\}$ *denote the set of observed variables at time step* t*. A lagged causal graph* $\mathcal{G}$ *contains directed edges of the form* $V^i_{t-\tau} \to V^j_t$*, where* $\tau \in \{1, 2, \ldots, \ell_{\max}\}$ *is a positive time lag. An edge* $V^i_{t-\tau} \to V^j_t$ *indicates that* V^i *is a direct cause of* V^j *with a time lag of* τ*.*

An example of a lagged causal graph is illustrated in Figure 2.3 (a). Finally, the lagged causal graph can be further simplified to a more concise representation, the *summary causal graph*.

Definition 4 (Summary Causal Graph) *Given a lagged causal graph* $\mathcal{G}$ *with edges of the form* $V^i_{t-\tau} \to V^j_t$ *for various lags* τ*, the corresponding summary graph* $\mathcal{S}$ *contains an edge* $V^i \to V^j$ *if there exists at least one lag* τ *such that* $V^i_{t-\tau}$ *is a direct cause of* V^j_t *in* $\mathcal{G}$*.*

Essentially, a summary graph is a simplified representation of the lagged causal structure in a time-series, where the temporal information about lags is abstracted away, as shown in Figure 2.3 (b). The summary graph collapses temporal information and only displays whether a causal relationship exists between two variables, but it does not specify the lag or delay at which the causal influence occurs. Note that from a lagged causal graph one can obtain the summary graph (e.g. from Figure 2.3 (a) to Figure 2.3 (b)), but not vice-versa, as the lag of causal relationships is not encoded in the summary structure. Summary graphs that retain temporal information can of course be obtained from the lagged causal graph, but remain outside the scope of our text.

2.3 Probabilistic Quantifications

To introduce our next causal assumptions, one needs to understand how causal structures relate to probabilistic reasoning, by the formalism of *Bayesian networks (BNs)* [Pearl et al., 1988]. BNs provide a compact representation of a joint probability distribution using a directed acyclic graph. Each node corresponds to a random variable, and each directed edge represents a statistical dependence.

Briefly, a Bayesian network is defined as a pair $(\mathcal{G}, \mathbb{P})$, where $\mathcal{G} = (\mathbf{V}, \mathbb{P})$ is a DAG with nodes $\mathbf{V} = \{X_1, \ldots, X_n\}$, and $\mathbb{P}$ is a joint distribution over $\mathbf{V}$, which factorizes, according to the chain rule, based on the structure of $\mathcal{G}$. That is, each variable X_i is conditionally independent of its non-descendants given its parents $\mathrm{Pa}(X_i)$ in the graph:

$$\mathbb{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid \mathrm{Pa}(X_i)). \tag{2.3}$$

This factorization, which Bengio et al. [2019] call *disentangled factorization*, provides both computational efficiency and semantic clarity: each conditional distribution models a local dependency, and the global distribution emerges from their composition. Compared to SCMs which explicitly model causal mechanisms through assignments $X_i := f_i(\mathrm{Pa}(X_i), \epsilon_i)$, BNs represent probabilistic dependencies only. While every SCM induces a Bayesian Network (through the distribution entailed by its noise and mechanisms), the converse does not hold; Bayesian Networks may not capture the directionality or manipulability implied by causality, so interventions and counterfactuals are not applicable. If one assumes causal interpretation on the edges, the resulting BN can be treated as a *causal BN*. In a nutshell, the key advantage of Bayesian Networks lies in their ability to encode and reason about conditional independencies, which are central to many causal discovery approaches.

2.4 Causal Assumptions

We now elaborate on causal assumptions needed for performing identifiability of the true causal graph and consequently, sound causal inference. We have already elaborated on the assumption of causal stationarity in Section 2.2. All causal discovery algorithms are governed by a set of assumptions, regarding the statistical properties of the data and the underlying causal structure. It is thus evident that making and understanding causal assumptions is also vital for correct interpretation not only of the discovered causal structure. We define the following assumptions in a way to be applied to either a (standard) causal graph or analogously to a temporal causal graph.

Definition 5 (Causal Markov Condition, [Spirtes et al., 2001]) *Let $\mathcal{G}$ be a causal graph with vertex set $\mathbf{V}$ and $\mathbb{P}$ a probability distribution over $\mathbf{V}$, generated by the causal structure induced by $\mathcal{G}$. Then $\mathcal{G}$ and $\mathbb{P}$ satisfy the Markov Condition if and only if $\forall\ W \in \mathbf{V}$, W is independent of its non-descendants (non causal effects) given its parents (direct causes) on $\mathcal{G}$.*

Essentially, the Causal Markov Condition assures that conditional independences between variables are exactly those entailed by the causal graph (i.e. by d-separation).

Definition 6 (Faithfulness, [Spirtes et al., 2001]) *Let $\mathcal{G}$ be a causal graph and $\mathbb{P}$ a probability distribution over $\mathbf{V}$. We say that $\mathcal{G}$ and $\mathbb{P}$ are faithful to each other if and only if (iff) the all and only the independence relations of $\mathbb{P}$ are entailed by the Causal Markov condition of $\mathcal{G}$. Specifically, $\mathcal{G}$ and $\mathbb{P}$ satisfy the Faithfulness Condition if-f every conditional independence relation true in $\mathbb{P}$ is entailed by the Causal Markov Condition applied to $\mathcal{G}$.*

In other words, given a causally sufficient set of variables $\mathcal{U}$ in a population N, *every* conditional independence relation that holds in the density over $\mathcal{U}$ *is entailed by the local directed Markov condition for the causal DAG of N*. The key argument behind assuming faithfulness of the true causal graph and the corresponding distribution induced by observed data is that non-faithfulness would lead to determinism in the observed system. As everything would cause everything and no stochasticity is involved, performing causal queries would prove invalid.

Definition 7 (Causal Sufficiency, [Spirtes et al., 2001]) *A set $\mathbf{V}$ of variables is causally sufficient for a population iff in the population every common cause of any two or more variables in $\mathbf{V}$ is in V, or has the same value for all units in the population. The common cause Z of two or more variables in a DAG $X \leftarrow Z \rightarrow Y$ is called a confounder of X and Y. Hence causal sufficiency implies no unobserved confounders. The notion of causal sufficiency is being used without explicitly mentioning the population.*

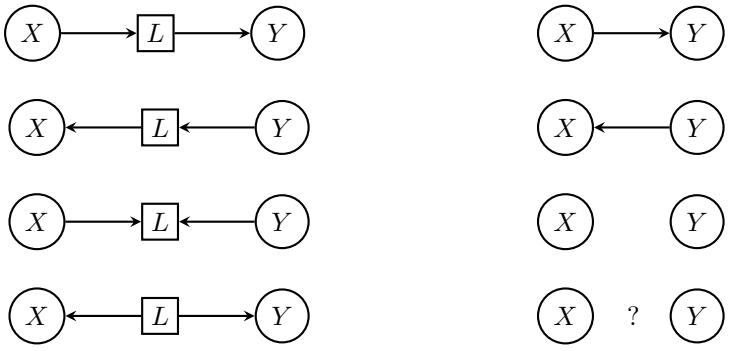


Figure 2.4: Illustrations for the presence of a hidden confounder L for two variables X, Y, in the case of atemporal data. In the first case, where X indirectly causes Y with L acting as a mediator, we observe that $Dep(X, Y)$ so we infer $X \rightarrow Y$, and similarly $X \leftarrow Y$ in the second case. In the third case where X and Y share a hidden common effect, we infer $Indep(X, Y)$ and thus discover no edge between X and Y. However, in the case of L being a hidden common cause of X and Y, there exists no DAG that captures the dependencies.

To put it simply, causal sufficiency is the *assumption of no unobserved variables*. Unobserved variables are called *latent* or *hidden*. The fact that DAGs are

unable to adequately capture latent confounders becomes apparent in the example at Figure 2.4, for the case of i.i.d. data. The assumption can be adapted similarly for the case of time-series, where one can assume latent confounders may exist at a predefined time-lag. For example V_t^1 and V_t^2 may be confounded by the latent variable V_{t-1}^3. SCMs without unobserved confounders are also called *Markovian models* as the noise terms of the variables are independent. The observed variables in SCMs with unobserved confounding can have dependent noise terms, called *semi-Markovian models*.

It must be noted that in practice, not all variables in the underlying SCM are measured. A projected causal graph on the observed subset is represented as an *Acyclic Directed Mixed Graph* (ADMG), in which a bidirected arrow $V_{t-\tau}^i \leftrightarrow V_t^j$ indicates the presence of an unobserved common cause with lag τ. For atemporal data, causal structures that enable marginalization of an SCM (a model with a subset of variables) exist [Richardson and Spirtes, 2002], but remain completely outside our scope. This formulation also underlies algorithms for causal discovery under latent confounding, or when inferring contemporaneous effects where directionality is unknown.

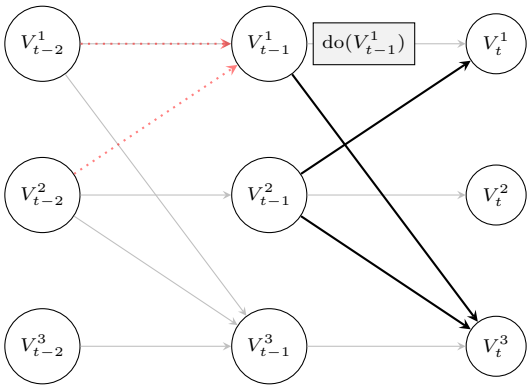


Figure 2.5: A dense lagged causal graph with an intervention on V_{t-1}^1, where V^2 directly causes V^1 & V^3 with lag 1 and V^1 directly causes V^3 with lag 1. The intervention $\mathrm{do}(V_{t-1}^1)$ removes all incoming edges to V_{t-1}^1, shown as red dotted lines, while preserving its downstream influence on V_t^3. The interventional value replaces the natural dynamics of V_{t-1}^1, breaking any backdoor paths through its original causes.

As we have seen in Chapter 1, the first rung that differentiates a probabilistic model with a causal model is the ability to perform interventions, that is, modifying the internal causal mechanisms of the examined system. We define interventions for the temporal setting, which only differs in the temporal dimension of the variables. TSCMs allow interventions to be performed in a simple manner: Modify the structural assignment in the SCM of a variable X^k to x. This corresponds to a *hard intervention* at a specific timestep t. The resulting interventional distribution $\mathbb{P}(Y_t|\mathrm{do}(X_t^k = x))$ captures the causal effect of setting X to x. The reader should recall that the observational and interventional distributions may differ because non-causal associations (e.g., due to confounders) are blocked under interventions. In the causal graph, this corresponds to performing *graph surgery*, by removing all incoming edges to X_t^k and

replacing the structural equation with $X_t^k := x$, due to what is known as the *modularity assumption*. An example is illustrated in Figure 2.5. One may also apply different interventions to different variables X_t^i, X_t^j, (*multiple interventional targets*) at the same timestep, resulting in an interventional distribution $\mathbb{P}(Y_t|\text{do}(X_t^i = x_t^i, X_t^j = x_t^j))$. For soft interventions, as the name may suggest, instead of explicitly modifying the structural assignment, one adds a noise term (either sampled from a known parametric distribution like a Gaussian, or sampled from a prior distribution) to the structural assignment. Consequently, interventions allow us to define causality in a much more rigid manner: A variable X^i causes X^j if an intervention on X^i leads to a direct change in X^j. Conclusively, do-calculus provides a set of transformation rules to express interventional (and counterfactual, although outside of our scope) distributions in terms of observable quantities, given a true or discovered causal graph. It underpins identification results, determining whether a causal effect can in principle be recovered from observational data alone or whether additional assumptions or interventional data are required. For causal discovery algorithms, handling of interventional data leads further into correct identifiability of the true causal graph, which may not be possible with observational data alone.

2.5 Brief Deep Learning Overview

This section aims to provide a high-level overview of deep learning for the unfamiliar reader, which is the second important pillar our work is based on.

The backbone of deep learning is the concept of *artificial neural network (ANN)*, abbreviated as neural network (NN). Such a network consists of a collection of connected units named neurons. The name stems from the introduction of the first model, Rosenblatt's Multilayer Perceptron (MLP) [Rosenblatt, 1958], which loosely mimics the connection of biological neurons in the brain. Each neuron is connected to other neurons like a synapse in the brain, with signal being propagated through the network using weights associated with the connections, which are real numbers. The larger the number, the stronger the connection.

Similarly to supervised learning algorithms, a neural network assumes a data input and constructs an output, based on a collection of data instances, in order to find a suitable representation between them, via a process called *training*. By processing such ground truth input and output pairs, training the model employs a *supervised learning scheme* (see Figure 2.6). In practice, training the model refers to adjusting its set of trainable parameters (such as the weight between neurons as illustrated before) in order to find the relationship between the input and target data pairs (assuming it exists). Training such a model consists of numerous important steps. As training consists of a mathematical optimization approach (since the problem is intractable), the goal is to minimize an objective function (loss function) that measures the error in performance, often as the absolute difference between the model's estimated outputs and the ground truth targets. Minimization of such a function is attained by the adjustment of the trainable parameters, which occurs iteratively as dictated by the training *optimizer*, based on gradients computed through *backpropagation* [LeCun et al., 1988, Werbos, 2002]. Backpropagation applies the chain rule of calculus in reverse (illustrated in Figure 2.7), computing how much each

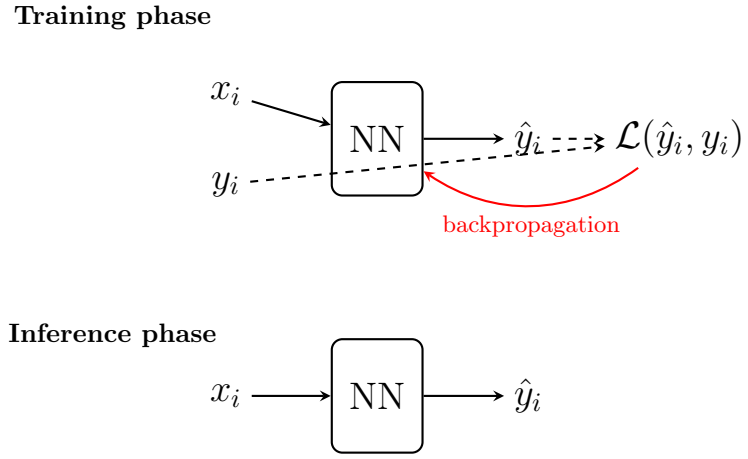


Figure 2.6: Illustration of neural network training and inference. During training, both input-output pairs (x_i, y_i) are provided to compute the loss $\mathcal{L}(\hat{y}_i, y_i)$ and update model parameters via backpropagation. During inference, only the input x_i is given, and the trained network produces an estimated output $\hat{y}_i$.

parameter contributed to the error signal. This allows the optimizer to make informed adjustments. Common optimizers include Stochastic Gradient Descent (SGD) [Ruder, 2016] and Adam [Kingma, 2015], both of which aim to efficiently navigate the high-dimensional loss landscape.

Once trained, the model can generalize to unseen data in the *inference phase*, where it takes only input x_i and outputs a prediction $\hat{y}_i$. Neural network models are known as universal approximators [Hornik et al., 1989]. This illustrates that in cases where statistical, machine learning, or even very simple methods fail due to the intractable nature of the function f between input and output, it becomes significant to consider fitting a neural network model.

Progress in neural networks and deep learning has often been driven by the development of general-purpose architectures that outperform prior methods across a wide range of tasks. Some advancements stem from novel ideas that reshape the field, while others result from the resurgence of older techniques enabled by increased data availability and computational power. Early models such as perceptrons [Rosenblatt, 1958] laid the foundation for modern neural networks, but their inability to represent non-linearly separable functions, such as XOR [Minsky and Papert, 1969], led to skepticism and a temporary stagnation in research. During the 1980s and 1990s, more expressive models were proposed, including self-organizing maps (SOMs) [Kohonen, 1982], Hopfield networks [Hopfield, 1982] and Boltzmann machines [Ackley et al., 1985], all of which introduced new paradigms in unsupervised learning, associative memory and energy-based modeling. With the resurgence of deep learning, driven by larger datasets and modern hardware, architectures like feedforward networks, convolutional neural networks (CNNs) [LeCun et al., 1989], recurrent neural networks (RNNs) [Rumelhart et al., 1986] and LSTMs [Hochreiter and Schmidhuber, 1997] became dominant in practical applications. Meanwhile, methods such as support vector machines (SVMs) [Cortes and Vapnik, 1995] and conditional random fields (CRFs) [Lafferty et al., 2001] contributed significantly to

structured prediction tasks before deep learning approaches took center stage. A key turning point came with the introduction of Transformers [Vaswani et al., 2017], which moved beyond the limitations of recurrence-based models by relying entirely on attention mechanisms. This shift enabled greater parallelism and improved the ability to capture long-range dependencies, particularly in sequential and structured data. Transformers have since become the backbone of modern foundation models, which aim to generalize across diverse tasks by leveraging vast amounts of data, compute, and expressive representations, an idea that also underpins the work proposed in this thesis.

A neural network consists of multiple *layers*, which may be of the same kind or of different kind, the combination of which can be selected in almost any desired order. A *sequential* architecture consists of layers that are linearly stacked, that is, each layer receives input from only its previous layers and only outputs to its forward layers. The first layer is usually referred to as the *input* layer and the last one as the *output* layer, with the in-between layers referred as *hidden layers* (unscreen blocks). In a way, such a neural architecture can be thought as a stack of building blocks, each of which is responsible for a specific task, from bottom to top (Figure 2.7). All layers consist of different number of parameters (real numbers, vectors or matrices) based on their design. The parameters that are tunable during training are called *trainable* or *learnable parameters*. The collection θ of trainable parameters represent the neural network f_θ. Typically those are the weights (noted by W, w, or U, u) and biases (notated b). Depending on the layer's defined operation, these weights are applied to its input x, and by adding the biases b, one obtains its resulting output y. This output will then be fed as input into the next layer and so forth, until the output layer delivers the final estimation of the model. Layers that do not introduce any trainable parameters exist, such as ones that apply operations (e.g. averaging, pooling etc.). They are usually added in order to prepare the data from the previous layer to be fed to the upcoming layers. They may also be applied by taking the network's final output layer and formulating the final predictions of the overall model.

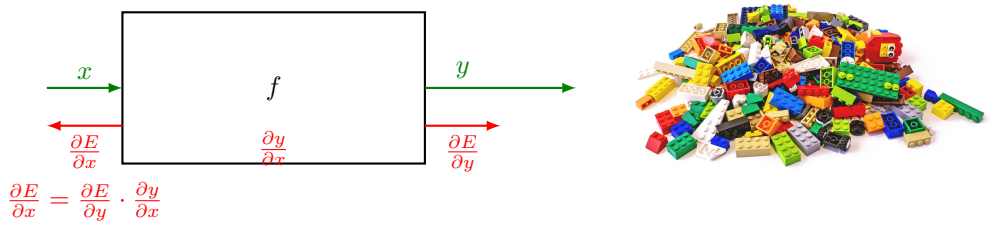
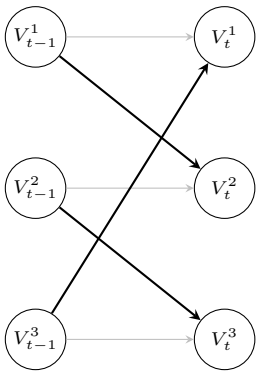


Figure 2.7: Training deep neural networks can be thought as building blocks, leaving architecture and capabilities to the imagination, with gradients being computed during backpropagation.

Apart from selecting the architecture of the model, consisting of the type, amount and order of layers, there are other types of parameters to choose from within each specific layer. Such are called *hyperparameters* such as weight initializers, stride, dilation rate, padding etc. In order to successfully train a network, optimal selection of layers, parameters and hyperparameters, as well



(a) Lagged causal graph ($\ell_{\max} = 1$)

(b) Lagged adjacency tensor slice at lag 1

$\mathbb{A}^{(0)}$

	V^1_{t-1}	V^2_{t-1}	V^3_{t-1}
V^1_t	0	0	**1**
V^2_t	**1**	0	0
V^3_t	0	**1**	0

Interpretation: $\mathbb{A}^{(0)}_{ji} = 1$ indicates a directed edge $V^i_{t-1} \to V^j_t$. For $\ell_{\max} = 1$, the slice index is 0.

Figure 2.8: Illustrative example of a three-variable lagged causal graph with $\ell_{\max} = 1$ (left) and its corresponding lagged adjacency tensor (right) of shape $[3, 3, 1]$. The tensor entry $\mathbb{A}_{j,i,\ell_{\max}-\ell}$ indicates whether a directed causal edge from $V^i_{t-\ell}$ to V^j_t exists.

as their initialization, are all crucially important. Overall, assembling a neural network model after defining the problem at hand is a non-trivial and demanding task. It requires familiarity with the specific problem, creativity, competence and knowledge within the area of neural networks, plus being informed about the latest advancements and state-of-the-art approaches. It should be noted that selection of the ideal architecture and parameters is a different field at first place, as inevitably to this day it still requires careful fiddling, guessing, experimental trial-and-error procedures and at many times ad-hoc approaches that are empirically shown to improve the state-of-the-art. Finally, although simple in theory, training deep models in practice presents challenges such as vanishing gradients, overfitting, and instability. Techniques like normalization layers, residual connections, and regularization help mitigate these issues. A discussion on how some of the above issues are mitigated is presented in Chapter 5.

2.6 Problem Formulation

Formally, we seek to construct a *pre-trained neural architecture capable of learning causal graphs from time-series data.* Recall from Section 1.1 that the objective of causal discovery is to infer the underlying causal graph $\mathcal{G} \sim \mathcal{D}$ of a dataset $\mathcal{D} \sim \mathcal{P}_{\mathcal{D}}$. Our task is to build a *large causal model* (LCM), a foundation model capable of performing causal discovery directly from time-series data, outputting a temporal causal DAG from varying input lengths, domains, and underlying causal mechanisms. Let $\mathbf{X} = \{\mathbf{X}_t\}_{t=1}^n \in \mathbb{R}^{n \times k}, \quad \mathbf{X}_t = (X_t^1, \ldots, X_t^k) \in \mathbb{R}^k$ denote a multivariate time series with n timesteps and k variables. The goal of an LCM is to infer a time-lagged causal graph that captures the directed temporal dependencies among the variables. Specifically:

A *large causal model (LCM)* is a parametric function f_θ mapping a multivariate time series $\mathbf{X}$ to a *lagged causal adjacency tensor*

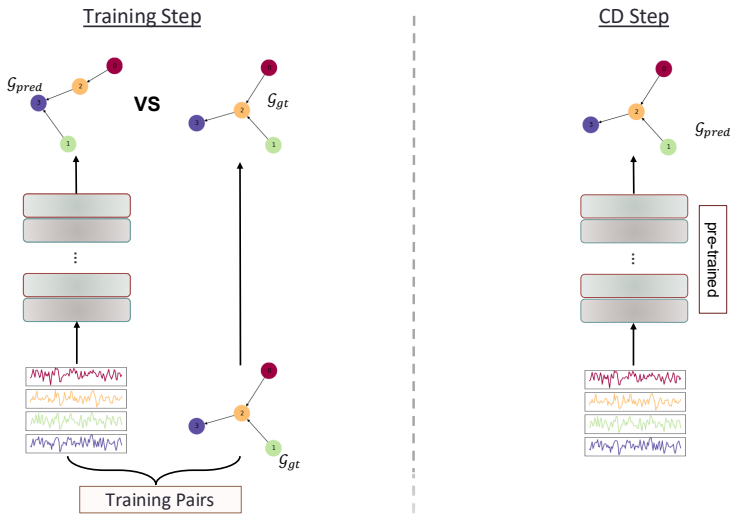


Figure 2.9: Training step (left) and inference (causal discovery) step (right) of a large causal model f_θ, represented as a sequence of blocks. During training, the model learns to map input time-series $\mathbf{X}$ to causal graphs by minimizing a supervised loss between the predicted graph $\mathcal{G}_{\text{pred}}$ and ground-truth $\mathcal{G}_{\text{gt}}$. During inference, the pre-trained model directly outputs $\mathcal{G}_{\text{pred}}$ for unseen inputs.

$$\mathbf{X} \xrightarrow{f_\theta} \hat{\mathbb{A}}, \tag{2.4}$$

where $\hat{\mathbb{A}} \in [0,1]^{k \times k \times \ell_{\max}}$. Each entry $\hat{\mathbb{A}}_{j,i,\ell_{\max}-\ell}$ represents the model's confidence in the directed causal relationship $X^i_{t-\ell} \to X^j_t$. The maximum lag $\ell_{\max}$ is a fixed hyperparameter bounding temporal dependencies, analogous to assumptions in classical methods such as PCMCI [Runge, 2018]. The notation $\hat{\mathbb{A}}_{ji\ell}$ implies j is the row/effect and i is the column/cause. The same notation convention applies to the ground truth adjacency tensor $\mathbb{A}$.

Unlike large language models or time-series forecasting foundation models that rely on self-supervision, causal discovery *lacks natural self-supervised signals*. The ground-truth causal structure cannot be inferred from data alone, as it is fundamentally unidentifiable without external assumptions or supervision. Therefore, training follows a *supervised learning scheme*, where pairs of ground truth lagged causal graphs and corresponding time-series samples are provided, generated from a known TSCM. This is illustrated in Figure 2.8.

As in neural networks (Figure 2.6), two distinct phases are involved: (i) *training*, where the model learns to map time-series data to causal graphs by provided with pairs of time-series data and their corresponding causal graphs in order to minimize its loss function; and (ii) *inference (causal discovery)*, where the model (in constant time), once trained, directly outputs the predicted causal graph for unseen time-series data in the form of a lagged adjacency tensor $\hat{\mathbb{A}}$, that best approximates the ground truth $\mathbb{A}$. Figure 2.9 conceptually illustrates the training and inference phase.

To make the problem well-posed under a supervised learning setting, we assume (i) sufficient overlap between training and test distributions [Ke et al., 2023, Stein et al., 2024, Wu et al., 2024b], (ii) causal assumptions (described in Table 4.3), and (iii) bounded temporal dependencies via fixed $\ell_{\max}$. This formulation highlights the need for large and diverse training collections of temporal data and their corresponding causal graphs. In practice, $\hat{\mathbb{A}}$ corresponds to a *soft prediction tensor*: it assigns a confidence score to each potential lagged edge rather than a hard decision. To evaluate performance against the ground truth, these scores are binarized via a thresholding operator $\hat{\mathcal{G}} = 1\{\hat{\mathbb{A}} \geq \tau\}$ (elaborated in Chapter 5). Different values of τ yield different precision-recall trade-offs, summarized using ROC/AUC and related metrics in Chapter 6. Additionally, in order to handle samples and TSCMs of varying number of samples, variables and lags, a padding mechanism must be implemented as in conventional foundation models. A thorough discussion of the above is provided in Chapter 4.

3

Data Generation

"If I had six hours to chop down a tree, I'd spend the first four sharpening the axe."

– Abraham Lincoln

A central limitation of current causal discovery methods, either in the temporal or i.i.d. setting, is the exclusive reliance on synthetic datasets for their evaluation, benchmarking and recently, for foundational-inspired methods (Section 1.3), training. This chapter outlines our methodologies for generating vast data instances[1] to support robust training of our large causal models. Synthetic data generation pipelines are typically constructed using predefined causal structures and known functional dependencies from randomly generated SCMs. While such approaches offer precise control and clarity over the data generation process, they often fail to capture the complexity and statistical idiosyncrasies of real-world time-series. To address this, we propose generation methods of both *synthetic* and *simulated* (realistically generated) data. Briefly, we opt for training on a mixture of both synthetic and simulated datasets.

We begin with an introduction to the context of data generation (Section 3.1) and proceed by outlining the main challenges for causal discovery (Section 3.2). The remainder is structured as follows: Section 3.3 outlines the generation process of synthetic data pairs, which serve as a learning foundation for our Large Causal Models. Section 3.4 introduces the *Temporal Causal-based Simulation (TCS)* framework, a modular and model-agnostic generative method for causal models and corresponding time-series samples, as well as the *Adversarial Causal Tuning (ACT)* module, which enables fine-tuning of TCS based on a Min-max optimization approach. Unlike traditional synthetic data generators in literature, our methodology learns the structural and functional dependencies, as well as the noise distribution, directly from time-series samples, and by allowing fine-tuning of the process further enhances the realism of generated causal models and bridges the domain gap between synthetic training and applicability to real-world scenarios. This part serves as a crucial component for enhancing the generalization performance of our LCMs, allowing us to leverage

[1] We refer to a pair of a multivariate time-series and its corresponding causal graph as a *data instance* or simply dataset, when no ambiguity exists regarding the form of data. Multiple pairs of data instances are referred to as a *data collection.*

data of higher quality for not only training, but also evaluation and benchmarking. Section 3.6 covers the curation of training and validation pairs used for model training, as well as the construction of our benchmark datasets. Finally, Section 3.6.4 provides an overview of our approach for partitioning data into shards for efficient training of LCMs.

3.1 Introduction

In the foundation model paradigm, model performance and generalization is closely tied to training data quality and diversity [Bommasani et al., 2021]. In the same way foundation models for natural language processing (e.g. Large Language Models-LLMs like the family of GPTs [Radford et al., 2019, Brown et al., 2020, Achiam et al., 2023]) and computer vision (e.g. CLIP [Radford et al., 2021], DALL-E [Ramesh et al., 2021]) rely on vast and heterogeneous data collections to fully capture the complexities of the real world in order to generalize effectively, a foundation model for causal discovery would require training data that reflect the complexities and properties of real-world causal systems.

For the most part, obtaining real data-cases is a straightforward task for domains unrelated to causality. Consider for example the case of constructing a neural model for recognizing images (such as dogs, cats, a glass bottle etc.). Real-world images are widely available online and can even be collected easily nowadays. For a supervised learning task like image classification, it is then a matter of *annotating* each image with its ground truth label. Such labeled image collections have already been curated, such as Imagenet [Deng et al., 2009]. A similar scenario exists for time-series forecasting models as well [Hahn et al., 2023, Kim et al., 2025]. Foundation models for time-series forecasting have been developed and trained on such data collections, such as the Transformer XL [Dai et al., 2019], the Informer [Zhou et al., 2021] etc.

An ongoing challenge in causal discovery is obtaining real data-cases, as it represents a much more nuanced task than generating purely synthetic ones. In the absence of real-world data and causal graph pairs, existing works regarding causal discovery on both i.i.d. and temporal data tend to focus either on using either (i) *purely synthetic* samples (causal models with randomized Erdős-Renyi [Brouillard et al., 2020] or Barabási-Albert [Barabási and Albert, 1999] structures) or (ii) *semi-synthetic*, based on existing knowledge of real cases and scenarios (e.g., simulated Markov processes of physical systems). Examples of such datasets include the Kuramoto model of coupled oscillators [Kuramoto, 1975, Löwe et al., 2022] and brain connectivity models such as fMRI [Smith et al., 2011]. Observational data from synthetic causal models are then generated through *ancestral sampling* [Murphy, 2023, Section 4.2.5] from the specified temporal SCM, and interventional data, if needed, in a similar manner. While this approach provides full control over the form of the causal structure and resulting causations, it leads to inherent limitations: synthetic data generation operates under idealized assumptions, including fully specified models, simplified noise distributions, absence of latent confounding, noisy interactions etc. As a result, generated data often fail to capture the complexity, heterogeneity, and non-stationary characteristics of real-world time-series. Real time-series samples are vastly available, but each one must be accompanied with its ground

truth causal mechanism, from which data samples are generated. As many causal assumptions on which causal discovery and causal inference methods are built upon (as elaborated in Chapter 2) are unknown or unlikely to hold in such real-world systems, the need for realistic data cases is even more pressing.

Regarding the training of foundation models, the authors of TimesFM [Das et al., 2024] articulate this principle clearly in the context of time-series forecasting:

> *"Synthetic data helps with the basics... Real-world data adds real-world flavor... This helps TimesFM understand the bigger picture and generalize better when provided with domain-specific contexts not seen during training."*
>
> – *Rajat Sen and Yichen Zhou*, TimesFM authors[2]

This sharp observation holds equally true for foundation models for causal discovery: synthetic data encode fundamental structural and temporal dependencies, while realistic data capture the statistical irregularities and intractable functional complexities inherent in real-world systems.

Surprisingly, causal discovery methods, including recent foundation models presented in Section 1.3 such as CP [Stein et al., 2024], SEA [Wu et al., 2024b] and Do-PFN [Robertson et al., 2025] are trained exclusively on synthetic datasets. Such models are rarely exposed to the structural uncertainty, noise variability or irregular patterns present in real data. Furthermore, their evaluation is typically restricted to synthetic benchmarks, limiting their demonstrated generalization capacity in practical deployment scenarios.

To address the above concerns, our work adopts a dual-strategy approach for data generation. Alongside a conventional yet rigid synthetic data generation pipeline from temporal SCMs, we introduce realistically generated datasets constructed using Adversarial Causal Tuning - ACT, inspired by AutoML [Hutter et al., 2019]. In short, the method generates time-series and corresponding lagged causal graphs by learning directly from real data samples. It decomposes the data generation process into three modular phases: (i) discovery of the causal graph, (ii) approximation of the functional dependencies and (iii) noise distribution modeling, all under a Min-Max fine tuning scheme for optimal causal model selection. In this way, we allow generation of time-series that are both causally coherent and statistically realistic.

3.2 The Challenge

The task of generating synthetic instances is evidently a more straightforward problem than the one of generating realistic ones. Various works that address synthetic causal data generation have been introduced, in both temporal and atemporal settings [Cinquini et al., 2021, Wen et al., 2021], yet few assess the realism and quality of generated data. Concerning time-series, methods taking into account sophisticated and configurable generative frameworks for both data and the causal structure itself exist (such as Lawrence et al. [2020]). Methods

[2] `https://research.google/blog/a-decoder-only-foundation-model-for-time-series-forecasting` (Last Accessed: May 2025).

such as Causal Chambers by Gamella et al. [2024] for generating realistic time-series data from physical systems under controlled experimental conditions have also been recently introduced. A key difference between our approach and theirs is that our proposed pipeline remains fully algorithmic, enabling scalable and diverse data generation, in contrast to a closed and controlled experimental setup. The work of Stein et al. [2024] also includes a synthetic generator of time-series, as well as the Tigramite package by Runge et al. [2019]. A shortcoming of these works is that during ancestral sampling over the causal graph, cuasal effects may grow too large due to the functional relationships used, thus resulting in values outside floating point range. This issue is handled by re-instantiating the sampling process, which can prove time-consuming and cumbersome for generating thousands of instances. What's more, some chosen functional relationships or parametric assumptions about the noise distributions may result in causal graphs that are either too trivial for the model to generalize or too hard for the model to extract any meaningful context. Furthermore, the work of Stein et al. [2024], although it follows an additive time-invariant SCM formulation for samples, it directly proceeds by randomly populating the lagged adjacency tensor, which remains of questionable soundness, as no TSCM is directly introduced. Finally, graphs may prove to be too dense (causal graphs should be, in general, sparse), which may lead the model to either overfit or underfit the data. We take into consideration the above points for the introduction of our own synthetic generation pipeline to output coherent time-lagged causal graphs for training and evaluation of our LCMs.

In the case of real-world data, the challenge is much more complex. The main argument against the sole use of synthetic data is that reliance on data generated under idealized assumptions, e.g., additive noise models, pre-defined structural priors, or linear interactions, limits data quality and as such the generalization capabilities of our LCMs (i.e. Figure 1.4 in Chapter 1). As we will elaborate in detail in Section 3.4, common evaluation protocols for realistic data generators use single-metric comparisons [Cheng et al., 2024], which do not fully capture the generative realism or causal fidelity. What's more, assumptions of both purely synthetic and semi-synthetic causally generated data are unlikely to hold in practice, as real-world models are subject to inherent restrictions and do not fit under closed conditions assumed by such methods. Generating datasets that are both causally coherent and statistically realistic remains a core challenge addressed in this work. We call such datasets *simulated* data.

The assumption that *distributions of training and test data cases must highly overlap* (stemming from adopting a supervised learning approach as elaborated in Section 2.6), poses another fundamental challenge in data generation. As one may expect, a supervised classification model (not limited to a neural network) trained on linear instances is expected to perform poorly on non-linear cases. This simple observations further highlights the need for *curating a large data collection with various complexities, sizes and causal properties for training and evaluation.* For synthetic cases, this includes a wide family of graph sizes, node degrees, functional relationships and noise distributions, among others.

To extend the limited dimensionality from five (5) variables of Stein et al. [2024] to more than double (12), the need for constructing an optimal curation of both synthetic and simulated data remains even more significant. The challenge of constructing a vast collection of data (among other number of variables, lags of causal relationships, functional dependencies etc.) is evident in order to

avoid degradation of model performance as input dimensionality and model parameters increase (Figure 1.3 in Chapter 1).

Finally, one should be accounting for causal assumptions that are not only reflected in the training data but also in the assumptions of the model. For instance, consider generating observational samples from a causal structure $X_t \leftarrow Z_{t-1} \rightarrow Y_t$ where Z is assumed to be a latent confounder and then removed from the corresponding SCM. A model that assumes causal sufficiency would potentially infer (based on these observational samples) a directed edge between X_t and Y_t (of any lag), which clearly is not the case and results in false interpretation of discovered causal relationships.

3.3 Generating Synthetic Causal Models

Recall from Chapter 2 that we formulate causal mechanisms as a (temporal) SCM, represented by a tuple $(\mathcal{G}, \mathcal{F}, \mathcal{E})$, where $\mathcal{G}$ is the causal DAG, $\mathcal{F}$ the set of functional dependencies, and $\mathcal{E}$ the set of noise distributions. Loosely speaking, a generator of random TSCMs and observational samples thus consists of a two-step process. First, a causal DAG instance $\mathcal{G}$ is sampled from a predefined family of DAGs $\mathbb{G}$ with various random properties. These may include the number of nodes, edge density (e.g., in-degree and out-degree distributions), the maximum number of lags $\ell_{\max}$ in the case of temporal SCMs, and the overall graph structure (e.g., uniform distribution of degrees). With the causal structure known, functional dependencies $f_i = f(V_t^i|\mathrm{Pa}(V_t^i))$ are sampled for each $X_i \in \mathcal{G}$ given its parents in the DAG, from known functional families $\mathcal{F}_1, \ldots, \mathcal{F}_k$, along with noise distributions $\epsilon_i \sim \mathcal{E}_1, \ldots, \mathcal{E}_m$. For nodes V_t^i where $\mathrm{Pa}(V_t^i) = \emptyset$, values are sampled directly from the assumed noise distribution. With both the causal structure and quantitative dependencies defined, the second step consists of generating samples from the SCM.

To illustrate sample generation, consider a simple i.i.d. example with three variables X, Y, Z and the causal DAG described by the collider $X \rightarrow Z \leftarrow Y$. This causal graph is associated with the functional dependency $Z := f(X, Y, \epsilon_Z)$ and noise distributions $X \sim \epsilon_X, Y \sim \epsilon_Y$. Observational samples are obtained by respecting the topological order of the DAG and generating samples recursively over the descendants (similar to ancestral sampling in latent variable models [Murphy, 2023]). For instance, one sample may be $X = 0.02, Y = 0.12$ and $Z = f(0.02, 0.12, 0.03) = 1$, where values from the noise distributions are substituted into the functional dependencies.

The topology of each synthetic SCM depends on the random graph generation method. Two main methods exist for random DAG generation in the literature. The first follows the *Erdős-Renyi (ER)* paradigm [Fienberg, 2012], as in Hagberg et al. [2008], with variations also used by Brouillard et al. [2020] and Ke et al. [2022]. Generation begins with an empty graph for a given number of nodes, then randomly and equiprobably picks directed edges to add based on a probability p. Unlike Hagberg et al. [2008], acyclicity checks are applied to reject edges that would introduce cycles, as shown in Algorithm 1. This process produces approximately uniform node degree distributions.

The second approach, the *Barabási-Albert (BA)* algorithm [Hagberg et al.,

2008, Barabási and Albert, 1999], generates scale-free graphs whose degree distribution follows a power law. Such networks exhibit a "rich-get-richer" phenomenon where a few nodes accumulate disproportionately many connections, a property often seen in real-world networks like social graphs.

In our experiments, we adopt the *Erdős-Renyi* method for generating *random lagged temporal causal graphs* (Figure 2.3 (a)), as it allows fine-grained control over edge probabilities and density. In time-series settings, the additional temporal dimension must be considered. Since we are interested in lagged causal graphs, Algorithm 1 is adapted as follows: the number of nodes V is sampled from a discrete uniform distribution $\mathcal{U}(V_{\min}, V_{\max})$, the edge probability p from $\mathcal{U}(0,1)$, and the maximum causal effect lag $\ell_{\max}$ from $\mathcal{U}(1, \ell_{\max})$. The resulting time-lagged graph contains $V \cdot \ell_{\max}$ lagged nodes at timesteps $\tau = 1, \ldots, \ell_{\max}$ and V nodes at the current timestep $\tau = t$. Directed edges (heading forward through time) are added randomly according to the chosen p, ensuring acyclicity. Furthermore, *no contemporaneous edges are assumed*, i.e., edges of the form $V_l^i - V_l^j$ do not exist.

The *Barabási-Albert* model, while useful for studying scale-free networks, tends to create overly connected hubs that are less representative of causal graphs in scientific domains, and is therefore left unconsidered.

Algorithm 1 Erdős-Rényi DAG generator

Input: n, p
Output: a DAG $\mathcal{G}$

initialize $\mathcal{G}$, $\mathcal{G}^{'}$ as empty directed graphs with n nodes
initialize *edges* with all the possible directed arcs in $\mathcal{G}$, $m = n \cdot p$
while $|edges| > 0$ and (number of edges in $\mathcal{G} \leq m$) **do**
  $e \leftarrow$ pop(shuffle(edges))
  if $p \leq$ random uniform sample in $[0, 1]$ **then**
    $D \leftarrow \mathcal{G}^{'}$
    add e to D
    if V has a cycle **then**
      $D \leftarrow \mathcal{G}^{'}$
    else
      add e to $\mathcal{G}$
      add e to $\mathcal{G}^{'}$
    end if
  end if
end while
return $\mathcal{G}$

After generating the DAG structure with the desired properties, functional dependencies $f^j(\mathrm{Pa}(V_t^j), \epsilon_t^j)$ for each variable $V_t^j \in \mathcal{G}$ are sampled from known functional families $\mathcal{F}_1, \ldots, \mathcal{F}_k$, along with the corresponding noise distributions. This finalizes the SCM, from which observational data can be sampled.

Generation of samples is carried out using causal ancestral sampling, similar to ancestral sampling in latent variable models (LVMs) [Murphy, 2023]. In short, variables without parents are sampled from their noise distributions, and variables with parents are computed from their functional dependencies, respecting topological order. In the temporal case, parent variables are fetched

from lagged causal parents, and initial "warm-up" samples are discarded to ensure stability in the generated time-series, following Runge [2018]. Assuming a known time-lagged SCM $\mathcal{G}$ with maximum lag $\ell_{\max}$, the process is described in Algorithm 2.

Algorithm 2 Temporal SCM Ancestral Sampling (ANCESTRAL)

Input: Temporal Causal Graph $\mathcal{G}$, max lag $\ell_{\max}$, number of timesteps T, number of warmup steps W
Output: Generated time-series $\{\mathbf{X}_t\}_{t=1}^{T}$

Initialize $X_{t=0}^i$ for all $i \in \{1, \dots, N\}$ with random noise
for $t = W + \ell_{\max} + 1$ to T **do**
  for each variable X_t^i in topological order of $\mathcal{G}$ **do**
    Determine lagged parents $\mathrm{Pa}_{X_t^i} \leftarrow \{X_{t-k}^j \mid (X^j, X^i) \in \mathcal{G}, 1 \leq k \leq \ell_{\max}\}$
    Compute $X_t^i \leftarrow f_i(\mathrm{Pa}_{X_t^i}) + \epsilon_t^i$
  end for
end for
return $\{\mathbf{X}_t^i\}_{W+\ell_{\max}}^{T}$

We now elaborate on the generation hyperparameters used for synthetic sample creation. We select $V_{\min} = 3, V_{\max} = 12$ and $\ell_{\max} = 3$. Two-variable SCMs are trivial under our assumptions (edges move forward in time and no latent confounders exist). The choice $V_{\max} = 12$ more than doubles the dimensionality of current state-of-the-art datasets, while $\ell_{\max} = 3$ balances (i) realism in short-term dependencies and (ii) tractability in training. Sampling graphs up to these limits aligns with the design constraints of our LCMs (as discussed in Section 2.6 and Chapter 4). A summary of these parameters is shown in Table 3.1.

Table 3.1: Summary of synthetic SCM generation parameters.

Parameter	**Range / Values**	**Sampling**
# Variables N	3 to 12	Uniform discrete
# Lags $\ell_{\max}$	1 to 3	Uniform discrete
Functional dependencies	See Table 3.2	Uniform discrete
Noise distributions	See Table 3.3	Categorical
Edge probabilities p_{edge}	Dynamic	Categorical

To prevent pathological graph densities, p_{edge} is scaled adaptively to the number of possible lagged edges $E = N^2 \cdot \ell_{\max}$. Specifically, p_{edge} is sampled from a discrete set of values inversely proportional to E, with categorical weights favoring sparsity (Table 3.4). If no edges are sampled during graph generation, p_{edge} is iteratively increased by 1% until a valid graph is obtained. This dynamic adjustment ensures diversity across sparse, medium, and dense regimes while maintaining realism. Additionally, p_{edge} is upper bounded by 0.15 to avoid dense graphs.

Functional forms include linear, polynomial, multiplicative, bounded, and sinusoidal mappings, selected from Table 3.2. Each causal edge is assigned

Table 3.2: Functional dependencies used in synthetic SCM generation.

Family	Functional form	Description
Linear	$f(x) = ax + b$;	Additive linear effect
	$f(x_1, x_2) = a_1x_1 + a_2x_2 + b$	Linear combination of parents
Non-linear	$f(x) = \alpha_n x^n + \dots \alpha_0 x_0$;	Polynomial
	$f(x) = e^x$;	Exponential Transformation
	$f(x) = \sin(x)$;	Sinusoidal
	$f(x_1, x_2) = x_1x_2$;	Multiplicative interaction
	$f(x) = \log(1 + \lvert x\rvert)$;	Logarithmic transformation
Bounded	$f(x) = \tanh(x)$	Hyperbolic tangent (bounded in $[-1, 1]$)
	$f(x) = \sigma(x)$	Sigmoid $\left(\frac{1}{1+e^{-x}}\right)$

Table 3.3: Noise distributions used in synthetic SCMs.

Name	Distribution	Parameters
Gaussian	$\mathcal{N}(\mu, \sigma^2)$	$\mu = 0, \sigma = 0.25$
Uniform	$\mathcal{U}(a, b)$	$a = 0, b = 1$

a strength coefficient $\alpha_{ji\lambda} \sim \mathcal{U}(0.3, 0.5)$, controlling the contribution of each parent variable. These coefficients populate the nonzero entries of the lagged adjacency tensor $\mathbb{A} \in \mathbb{R}^{V \times V \times \ell_{\max}}$, interpreted as edge-level causal strengths rather than causal effect estimates. Noise distributions are sampled from those listed in Table 3.3.

Finally, each synthetic sample is padded or truncated to a fixed sequence length $L_{\max} = 500$ and variable dimension $V_{\max} = 12$ to ensure batch processing and model input compatibility (see Section 4.3).

Table 3.4: Edge probability values used for different synthetic causal graph densities. E denotes the number of possible lagged edges.

Edge Regime	Threshold	Values	Weights	Notes
Small graphs	$E < 100$	$\frac{3}{E}, \frac{5}{E}, \frac{7}{E}$	$0.6, 0.3, 0.1$	Promotes sparsity
Medium graphs	$100 \leq E < 200$	$\frac{5}{E}, \frac{7}{E}, \frac{9}{E}$	$0.6, 0.3, 0.1$	Moderate edge density
Large graphs	$E \geq 200$	$\frac{9}{E}, \frac{12}{E}, \frac{15}{E}$	$0.6, 0.3, 0.1$	Higher density permitted

3.4 Generating Simulated Pairs

In contrast to synthetic data, simulated (realistically generated) data aim to reflect the underlying causal dynamics of real-world systems by reverse-engineering the structural and functional dependencies from actual time-series. Essentially, we address the problem of generating time-series data from a causal model with the same distribution as a given real dataset, that is, constructing a probabilistic causal digital twin.

For this purpose, we introduce the *Temporal Causal-based Simulation (TCS)* framework, a modular and model-agnostic generative [3] method for generating time-series along with their temporal causal graphs, which incorporates a fine-tuning process for optimal model selection called *Adversarial Causal Tuning (ACT)*. Generated data is not only statistically coherent but also maintains interpretable causal consistency, making it suitable for training and benchmarking of our Large Causal Models.

Ideally, what one is looking for is the ability to generate data that is both realistic and comparable to real-world cases, and as such a way to discrimimate between simulated data of good and poor quality. In literature, there exist various methods for asserting whether two data samples resemble each other. One way to measure similarity between (observed) real and synthetic data is by the *Maximum Mean Discrepancy (MMD)* [Gretton et al., 2006] metric. MMD measures the distance between two probability distributions by embedding the data into a reproducing kernel inner-product space and calculating the difference in the mean embeddings. The MMD score is given by $\text{MMD}^2(X,Y) = \mathbb{E}_{X,X}[k(X,X)] + \mathbb{E}_{Y,Y}[k(Y,Y)] - 2\mathbb{E}_{X,Y}[k(X,Y)]$, where X and Y are the real and synthetic data samples respectively and the expectations are taken over all points in the selected data batch.

A more useful way to assess data similarity is using *Classifier 2-sample tests (C2STs)*. The main idea behind C2STs, introduced by Gretton et al. [2012], is to train a machine learning classifier f_θ to distinguish samples $X = \{x_1, \ldots, x_n\}$ sampled from distribution P against samples $Y = \{y_1, \ldots, y_m\}$ sampled from distribution Q. Intuitively, if the null hypothesis $H_0 : P = Q$ holds, then the classifier (trained on the labeled training set $\mathcal{D} = \{(x_i, 0)\} \cup \{(y_j, 1)\}$) should be unable to distinguish between samples and should demonstrate performance close to 1/2, or chance level when evaluated on a holdout test set. By performance, we refer to the AUC-ROC score of the classifier [Bradley, 1997], to which we refer to as *discrimimation score* AUC_D. As Lopez-Paz and Oquab [2017] demonstrated, classifier 2-sample tests offer several attractive properties for evaluating data similarity, such as assessing the quality of samples generated by models where the likelihood is intractable (cannot be computed analytically) but sampling from the generating process is tractable.

To highlight our contributions in realistic data generation, we present key arguments against the suitability of the work by Cheng et al. [2024] for training and evaluating our LCMs. It remains the closest to our work, in the sense that it is applicable to time-series and has been introduced as a method to benchmark causal discovery algorithms. However, it faces several shortcomings. It begins by fitting a residual-based neural network (*causally disentangled neural network - CDNN*), based on either an MLP or an LSTM architecture, to model a *vector autoRegressive (VAR)* process, using all historical variables for a specified maximum lag; it then extracts a *hypothetical causal graph (HCG)* based on computed feature importance values, using methods such as deep Shapley additive exPlanations (DeepSHAP) [Lundberg and Lee, 2017], while being able to generate data auto-regressively from the fitted model. As the authors themselves point out throughout their paper, the extracted HCG using feature importance does not constitute causal discovery and does not represent a causal model. As such,

[3] A generative model is a type of machine learning model designed to learn the underlying probability distribution of a dataset and is therefore capable of generating new, synthetic data samples that are statistically and structurally similar to the training data.

causal queries are not possible and the discovered graph does not represent a causal digital twin. As their extracted hypothetical causal graph represents a *summary graph* with no lag information and not a *lagged causal graph* (Figure 2.3), its scope remains limited. Additionally, although a generative mechanism for data exists by the fitted model, it does not reflect or approximate the underlying TSCM mechanism. Consequently, generation of interventional samples is unfeasible: the resulting process may generate temporally correlated samples, but not reason causally, e.g. to estimate $P(Y|\text{do}(X = x))$.

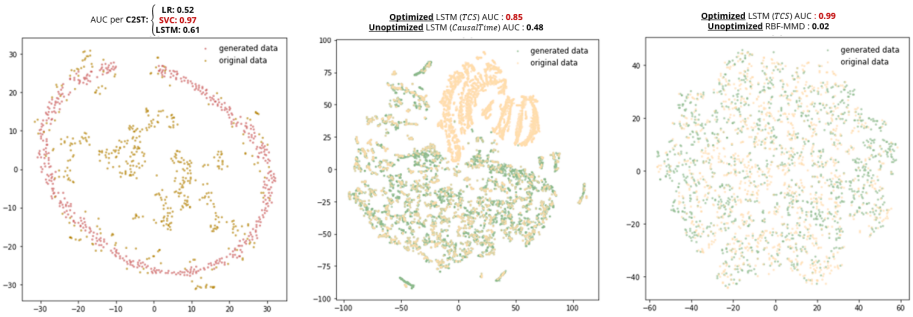


Figure 3.1: t-SNE [Maaten and Hinton, 2008] projections of original and simulated data across different datasets. Optimized C2ST variants outperform standard methods (e.g., MMD) in detecting distributional differences, even when visual separability is low.

Furthermore, CausalTime relies on a single discriminator to evaluate its simulation quality in addition to the MMD metric. As shown in Figure 3.1 leads to misleading conclusions about the quality of generated data. Data are projected in the plane using t-SNE [Maaten and Hinton, 2008], and the discriminator is trained to distinguish between real and generated data. In Figure 3.1 (a), only the support vector classifier (SVC)-based C2ST is able to distinguish simulated from original samples, which have been generated using trigonometric functions. The other two C2STs, based on logistic regression (LR) and long-short-term memory (LSTM) are unable to do so, with an AUC_D of 0.52 and 0.61 respectively. A similar experimental scenario is shown in Figure 3.1 (b) on the `AirQualityUCI` dataset where an optimized[4] LSTM classifier discriminates, while the unoptimized one fails. Finally, Figure 3.1 (c) reveals the inadequacy of the maximum mean discrepancy metric, as although it is close to zero and data seem indistinguishable with the naked eye, an optimized LSTM classifier is still able to separate them. Consequently, the need for introducing a causal model generator by utilizing optimized C2STs is clear.

[4] An optimized classifier refers to a model whose hyperparameters or configuration are tuned to maximize its performance (e.g., AUC_D), whereas an unoptimized classifier uses default or untuned parameters.

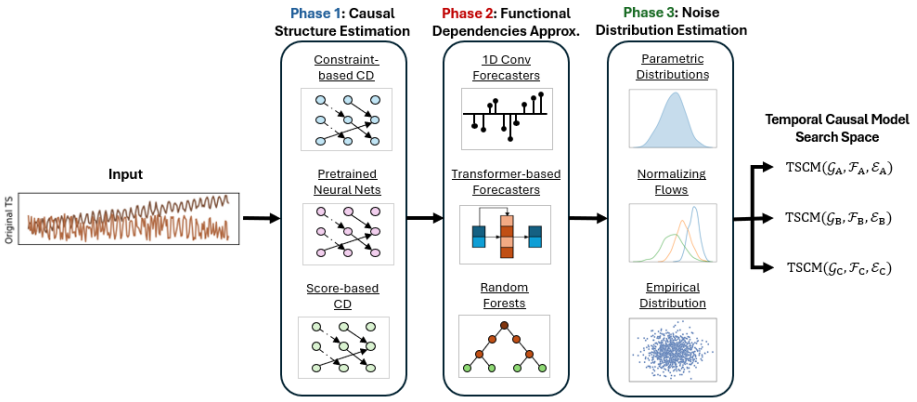


Figure 3.2: Given a multivariate time-series sample of real data, the method begins by estimating the lagged causal graph through temporal causal discovery (*Phase 1*). It then estimates the functional dependencies between variables using a wide range of forecasters (*Phase 2*). Based on the predictions of these forecasters, it learns the noise distribution through a variety of density estimation methods (*Phase 3*). Each combination of methods for Phases 1, 2 and 3 constitutes a configuration for TCS, that results in a unique *temporal causal model*. By considering different configurations, TCS creates a search space for the temporal causal model that best describes the real data.

3.4.1 Temporal Causal-based Simulation (TCS)

This subsection is dedicated to the main generative method for producing realistic time-series data and their respective time-lagged causal graph, called *Temporal Causal-based Simulation (TCS)*, depicted in Figure 3.2. Our method can be summarized briefly by the following statement: *Select the model that generates simulated data which, despite our best efforts, are indistinguishable from the real training data.* By best efforts, we refer to the fine-tuning phase. We summarize the formulation as follows: (i) a *Generation phase*, where a module performs a search in the space of causal models by generating TSCMs and (ii) a *Tuning phase*, where another module searches the space of discriminators to distinguish between real and simulated data, called *Adversarial Causal Tuning (ACT)*. The above procedure leads to a Min-max selection scheme, similarly to generative adversarial networks (GANs) [Goodfellow et al., 2014]. We now describe the main phases of the TSCM search space and ACT.

3.4.1.1 Generation of TSCMs

Phase 1: Causal Discovery Initially, TCS retrieves the data's causal structure. To achieve this, the method employs a variety of causal discovery methods for temporal data, either established as a gold standard or express the current state-of-the-art. Specifically, we consider the algorithms *PCMCI* [Runge, 2018] and *DYNOTEARS* [Pamfil et al., 2020]. PCMCI is a CD method based on conditional independence testing, serving as a temporal extension of PC [Spirtes et al., 2001] established through literature, while DYNOTEARS constitutes an extension of the NOTEARS [Zheng et al., 2018] CD algorithm on temporal data,

based on continuous optimization. These methods are also used as benchmarks for evaluating our trained LCMs and are discussed in detail at Section 6.3.

We now describe the generation of TSCMs with some mathematical rigor. Given a multivariate time-series dataset $\mathcal{D}$ over $\mathbf{V}$, the first phase performs temporal causal discovery using an CD algorithm dependent on a given parameter configuration $\mathbf{B}_{\text{CD}}$ (e.g. for DYNOTEARS, the regularization constants $\lambda_{\mathbf{W}}, \lambda_{\mathbf{A}} \in \mathbf{B}_{\text{CD}}$). That is, obtain:

$$\mathcal{G}, \{\mathbf{Pa}_{X^i}\}_{i \in \mathbf{V}} \leftarrow \text{CAUSAL\_ALG}(\mathcal{D}, \mathbf{B}_{\text{CD}}) \tag{3.1}$$

It is crucial to highlight that the outcome of all the considered time-series CD methods are *lagged causal graphs*, that not only serve as an expressive representation of the causal model for training our LCMs, but also able to preserve the temporal structure of data for further use. We describe TCS in Algorithm 3. It should also be noted that as causal discovery algorithms are governed by a set of causal assumptions, sampled data from TCS, as well as input data to the method, are assumed to follow these assumptions. The algorithms and their assumptions are selected to align with the causal assumptions of our LCMs, in Chapter 4.

Phase 2: Estimation of Functional Dependencies After the causal structure of the data has been retrieved, we estimate the functional dependencies between variables by fitting forecasting methods on the past (lagged) parent values of each variable. We incorporate three model options for the forecasting task. The first consists of classic *Random Forest* (RF) [Breiman, 2001] models on lagged data representations according to the maximum discovered time lag $\ell_{\max}$ during the fist phase of the algorithm. The second option incorporates the *AD-DSTCN*, a 1-D Convolutional forecaster utilized by Nauta et al. [2019] in their proposed TCDF method. The last addition to our forecaster options is the TimesFM foundation model for time-series forecasting, introduced by Das et al. [2024]. Considering the time-series X^i, we now apply (or fit) a predictive model f on its lagged parents Pa_{X^i}, to obtain an approximation of the functional dependency as in the SCM formulation. Mathematically,

$$f(X^i \mid \text{Pa}_{X^i}) \leftarrow \text{FIT}(X^i, \text{Pa}_{X^i}, \mathbf{B}_{\text{pred}}) \tag{3.2}$$

where FIT corresponds to the predictive algorithm used. Like in the causal discovery phase, $\mathbf{B}_{\text{pred}}$ is the algorithm-dependent parameter configuration. We depict the set of the fitted functional dependencies for each node as $\mathcal{F}$.

Phase 3: Estimation of Noise Distribution The estimation of functional dependencies is followed by the estimation of the true noise distribution for each variable. The predictions of the fitted forecasting models are utilized to obtain the residual terms by subtracting the real data values of the corresponding time-steps. In case a variable at hand has no deduced causal parents from Phase 1, a trivial predictor can be used, as the mean of the observed values, a solution dependent on the assumption of time-series stationarity. The result of this process corresponds to the empirical distribution of the noise, which is then utilized along additional noise estimation methods. That is, for a variable X_i and a noise configuration $\mathbf{B}_{\text{noise}}$, fit the noise estimator $\epsilon_i \leftarrow \text{FIT}(\mathbf{B}_{\text{noise}})$.

Algorithm 3 Temporal Causal-Based Simulation (TCS-SIMULATE)

Input: Real time-series dataset $\mathcal{D} = \{\mathbf{X}_t\}_{t=1}^{T}$ over variable set $\mathbf{V}$; sample size n; causal discovery configs $\{\mathbf{B}_{CD}\}$; functional dependency estimator configs $\{\mathbf{B}_{pred}\}$; noise estimation configs $\{\mathbf{B}_{noise}\}$; ACT parameters $(D, \{C_d\}, \alpha, n_{perm})$

Output: Optimal causal model s_{best}, generated dataset $\mathcal{D}'_{s_{\text{best}}}$, and discriminator score AUC_{best}

$\mathcal{S} \leftarrow$ GENERATE_TSCM_CONFIG_SPACE($\{\mathbf{B}_{CD}\}, \{\mathbf{B}_{pred}\}, \{\mathbf{B}_{noise}\}$)
$\mathcal{R} \leftarrow \emptyset$
for each configuration $s \in \mathcal{S}$ **do**
  Obtain $\mathbf{b}_{CD}$, $\mathbf{b}_{pred}$, $\mathbf{b}_{noise}$ from s
  $(\mathcal{G}_s, \mathbf{Pa}_s) \leftarrow$ CAUSAL_ALG($\mathcal{D}, \mathbf{b}_{CD}$) {Phase 1: Causal discovery}
  Initialize $\mathcal{F}_s \leftarrow \emptyset$, $\mathcal{E}_s \leftarrow \emptyset$
  for each variable $X^i \in \mathbf{V}$ **do**
    if $\mathbf{Pa}_{X^i} \neq \emptyset$ **then**
      $f_{X^i} \leftarrow \text{FIT}(X^i, \mathbf{Pa}_{X^i}, \mathbf{b}_{pred})$
      $R_{X^i} \leftarrow X^i - f_{X^i}(\mathbf{Pa}_{X^i})$
    else
      $R_{X^i} \leftarrow X^i - \mathbb{E}[X^i]$
    end if
    $\mathcal{F}_s \leftarrow \mathcal{F}_s \cup \{f_{X^i}\}$
  end for
  for each variable $X^i \in \mathbf{V}$ **do**
    $\epsilon^i \leftarrow \text{FIT}(R_{X^i}, \mathbf{b}_{noise})$
    $\mathcal{E}_s \leftarrow \mathcal{E}_s \cup \{\epsilon^i\}$
  end for
  $\mathcal{D}'_s \leftarrow$ ANCESTRAL($\mathcal{G}_s, \mathcal{F}_s, \mathcal{E}_s, n, T$) {Ancestral sampling}
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{(s, \mathcal{G}_s, \mathcal{F}_s, \mathcal{E}_s, \mathcal{D}'_s)\}$
end for
$(s_{\text{best}}, \text{AUC}_{\text{best}}) \leftarrow \text{ACT}(\mathcal{S}, \mathcal{D}, D, \{C_d\}, \alpha, n_{perm})$
return $s_{\text{best}}, \mathcal{D}'_{s_{\text{best}}}, \text{AUC}_{\text{best}}$

Additionally, we also consider two neural approaches for the estimation of the noise distribution. These consists of Neural Spline Flows [Durkan et al., 2019] and a simplistic version of the RealNVP model [Dinh et al., 2017]. Finally, a known parametric distribution for the noise of the data is utilized (e.g., Normal) and the empirical noise distribution is used to fit its parameters (e.g., mean and variance). The set of all fitted noise estimators for each node is depicted as $\mathcal{E}$.

Table 3.5 provides an overview of the implemented methods for each phase, along with their hyperparameter configurations. Their successive application results in the creation of a TSCM $(\mathcal{G}, \mathcal{F}, \mathcal{E})$, consisting of the discovered causal structure $\mathcal{G}$ (*phase 1*), functional dependencies $f^i \in \mathcal{F}$ (*phase 2*) and noise distributions $\epsilon_t^i \in \mathcal{E}$ (*phase 3*), all of which are derived from the provided original time-series using the selected methods per phase. We now describe the methodology for obtaining the optimal configuration of the TSCM that best fits the original data through ACT.

Table 3.5: Search spaces for TSCMs and Discriminators (C2STs).

Method	Hparam	Values
(a) TSCM Search Space		
PCMCI [Runge, 2018]	n_lags	{2, 3}
	n_reps	{10}
DYNOTEARS [Pamfil et al., 2020]	n_lags	{1, 3}
	λ_w, λ_a	{0.1}
	max_iter	{100}
	n_reps	{10}
	Thresholded / Threshold	{True / 0.05}
DENSE (FC Baseline)	Graph Type	Fully Connected
Random Forest [Breiman, 2001]	n_estimators	{100, 500, 1000}
AD-DSTCN (TCDF) [Nauta et al., 2019]	num_levels	{0, 2}
	epochs	{1000}
	kernel_size / dilation_c	{(2,2), (3,3)}
	lr	{0.01, 0.001}
TimesFM [Das et al., 2024]	Model Type	Foundational Forecaster
Noise Models	noise_approx.	{`est`, `normal`, `uniform`, `spline`, `nvp`}
(b) Discriminator (C2ST) Search Space		
SVM	C	{1.0, 0.75, 0.5, 0.25}
	Kernel	{`linear`, `poly`, `rbf`}
	Degree	{3}
	Gamma	{`auto`, `scale`}
LSTM	Batch Size	{32, 64}
	Hidden Size	{128, 256}
	Layers	{2, 3}
	Dropout	{0.05, 0.1}
	Seq. Len	{10, 20}
	Epochs	{10, 50}
	LR	{0.0001, 0.001}

3.4.2 Adversarial Causal Tuning (ACT)

We leverage on the multiple available options per phase during the TSCM generation, to create a search space C for the optimal configuration of TCS. At its highest capacity, this search space consists of all possible unique combination of methods (adjusting also for different hyperparameter settings) for all three phases, resulting in the creation of thousands of candidate temporal causal models. To adjust for running efficiency, we restrict the possible simulation configurations to orders of magnitude of tens. To identify the causal model that best describes the original data, each created instance must be assessed based on the similarity of its sampled data empirical distribution to the empirical distribution of the original data.

Recall that concerning the use of complex parameterized evaluation methods such as C2STs, hyperparameter configuration plays a significant role, especially in case of high-dimensional data with complex functional dependencies (e.g., Figure 3.1). In general, different metrics may prove to have independent behavior, especially in high dimensions, as shown in Goudet et al. [2018]. Inspired by the AutoML [Hutter et al., 2019] domain, a configurable discriminator search space is created, with a grid-based search performed to find the C2ST that bests discriminates between generated and original data.

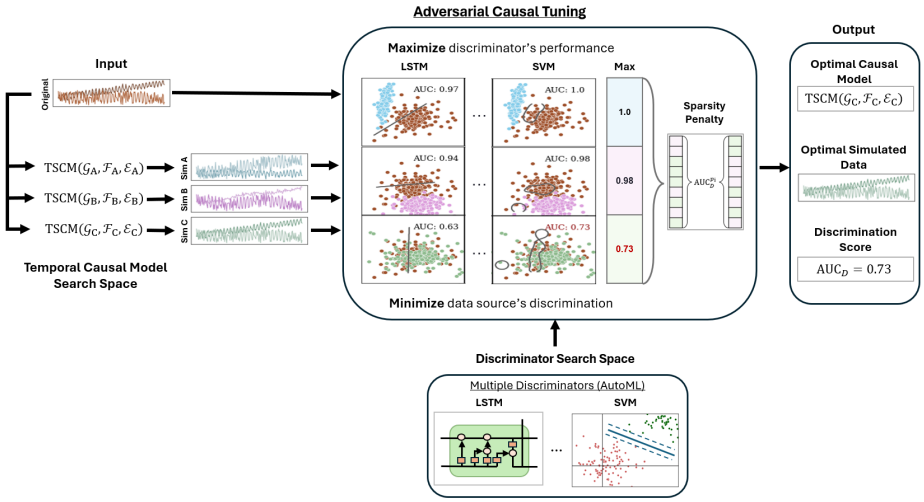


Figure 3.3: Inspired by AutoML practices and the adversarial learning paradigm of GANs, TCS leverages C2ST as discriminators to assess the quantitated temporal causal models. Each causal model generates synthetic data, which are discriminated against the real data by a robust AutoML pipeline of C2ST models. TCS returns the causal model with the *minimum* discrimination score (AUC_D) against the *maximum* performance discriminator.

The classifiers that we consider as possible C2ST discriminators include the *Support Vector Classifier (SVC)* [Hearst et al., 1998] and the *Long Short-Term Memory-based Classifier (LSTMC)* [Hochreiter and Schmidhuber, 1997], both optimized though hyperparameter tuning [Hutter et al., 2019], in order to obtain the discriminator that yields the highest AUC_D.

By running this process for each candidate causal model, we thus avoid underfit discrimination. This approach highly resembles a discreet, non-differentiable version of adversarial learning in GANs [Goodfellow et al., 2014], as both the temporal SCM and the discriminator are optimized in parallel while competing with each other. The C2ST problem is deduced to a Min-max selection scheme as follows: Define a set of classifiers D, each with a set of configurations $C_d = \{c_{d,1}, c_{d,2}, \ldots, c_{d,n}\}$. For each classifier $d \in D$ and configuration $c \in C_d$, we compute the discriminative score d_c of the classifier. From these configurations, the maximum discriminative score is selected across the set of lowest scores. That is, optimize with Min-max selection as

$$\min_{c \in C} \max_{d \in D} d_c$$

The maximization step can be interpreted as *"For each TSCM configuration c, obtain the highest discriminative score across the classifiers"*, while the minimization step as *"Among all TSCM configurations, select the configuration with the lowest maximum score, which depicts the least discriminative, yet optimal case"*.

In the end, TCS returns the configuration of the optimal temporal causal model, along with the discrimination score and the optimized configuration of the discriminator. This is shown in Figure 3.3, and in pseudocode form in

Algorithm 4 Adversarial Causal Tuning (ACT)

Input: Candidate TCS configurations $\mathcal{S} = \{s_1, \ldots, s_m\}$; real test set $\mathcal{D}$; discriminator space D; discriminator configuration grids $\{C_d\}_{d \in D}$; significance level α; number of permutations n_{perm}

Output: Optimal TCS configuration s_{best} and discrimination score AUC_{best}

$\mathcal{R} \leftarrow \emptyset$ {Store best discriminator result for each TCS configuration}
for each configuration $s \in \mathcal{S}$ **do**
  $\mathcal{D}'_s \leftarrow \text{SIMULATE}(s)$ {Generate dataset via TCS}
  $\text{AUC}^s_{\text{best}} \leftarrow -\infty$, $\text{Disc}^s_{\text{best}} \leftarrow$ None, $\text{Cfg}^s_{\text{best}} \leftarrow$ None
  for each discriminator $d \in D$ **do**
    for each configuration $c \in C_d$ **do**
      $\text{AUC}^c_d \leftarrow \text{TRAIN\_EVAL}(d, c, \mathcal{D}, \mathcal{D}'_s)$
      if $\text{AUC}^c_d > \text{AUC}^s_{\text{best}}$ **then**
        $\text{AUC}^s_{\text{best}} \leftarrow \text{AUC}^c_d$
        $\text{Disc}^s_{\text{best}} \leftarrow d$
        $\text{Cfg}^s_{\text{best}} \leftarrow c$
      end if
    end for
  end for
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{(s, \text{AUC}^s_{\text{best}}, \text{Disc}^s_{\text{best}}, \text{Cfg}^s_{\text{best}})\}$
end for
Select configuration $s_o \in \mathcal{R}$ with the lowest $\text{AUC}^s_{\text{best}}$
Retrieve associated $\text{Disc}_o, \text{Cfg}_o, \text{AUC}_o$
$\mathcal{E} \leftarrow \{s_o\}$ {Set of statistically equivalent configurations}
for each configuration $s \in \mathcal{R} \setminus \{s_o\}$ **do**
  Obtain $probs_o, probs_s, \text{AUC}_o, \text{AUC}_s$ from their discriminators
  $p \leftarrow \text{PERM\_TEST}(y_{\text{test}}, probs_o, probs_s, \text{AUC}_o, \text{AUC}_s, \alpha, n_{perm})$
  if $p \geq \alpha$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{s\}$ {Add statistically equivalent configuration}
  end if
end for
$s_{\text{best}} \leftarrow$ configuration in $\mathcal{E}$ with fewest edges in its causal graph
return $s_{\text{best}}, \text{AUC}^{s_{\text{best}}}_{\text{best}}$

Algorithm 4. The considered hyperparameter search spaces for the classifiers (adversarial space) are shown in Table 3.5.

3.4.2.1 Sparsity Penalty

A drawback of the proposed method is that it may occasionally favor dense graph outputs to more realistic candidate solutions of phase 1, being statistically beneficial during the next phases. To avoid this, we adopt a *sparsity penalty strategy* following Biza et al. [2022]. Specifically, a permutation-based statistical hypothesis test is introduced on whether two classifiers have statistically equivalent scores. The permutation test assumes as null-hypothesis that two given classifiers have statistically equivalent AUC_D scores and uses as its statistic the difference between these two discriminative scores. At its core, the permutation test swaps the per sample predicted probabilities of two classifiers and then

Algorithm 5 AUC Statistical Equivalence Test (PERM_TEST)

Input: Test set labels y_{test}; discriminator test predicted probabilities on the optimal TCSM output $probs_o$; discriminator test predicted probabilities on a candidate TCSM output $probs_i$; discrimination scores AUC_o and AUC_i; significance level α; number of permutations n_{perm}

Output: *True* or *False* for statistical equivalence

$t_{obs} \leftarrow |\text{AUC}_o - \text{AUC}_i|$
$ctr \leftarrow 0$
for $p = 1$ to n_{perm} **do**
  $w_l^p \leftarrow$ SAMPLE_PERMUTATION_INDICES() {Sample indices to swap uniformly}
  $(probs_w^o, probs_w^i) \leftarrow \text{SWAP}(probs_o, probs_i, w_l^p)$ {Permute sampled indices}
  $\text{AUC}_o^w \leftarrow \text{EVALUATE}(y_{\text{test}}, probs_w^o)$
  $\text{AUC}_i^w \leftarrow \text{EVALUATE}(y_{\text{test}}, probs_w^i)$
  $t_p \leftarrow |\text{AUC}_o^w - \text{AUC}_i^w|$
  if $t_p > t_{obs}$ **then**
    $ctr \leftarrow ctr + 1$
  end if
end for
$pvalue \leftarrow \frac{ctr}{n_{perm}}$
if $pvalue \geq \alpha$ **then**
  return True
else
  return False
end if

re-calculates and compares the permuted outcomes to the ground-truth. If the $p-$value computed by the permutation test is larger than the significance level α, then the comparing AUC_Ds are considered equivalent. By default, a significance level of $\alpha = 0.05$ and 1000 permutations is used to compute the $p-$value. It should be noted that no significant computational overhead is introduced with sparsity penalty, as the only operations performed is the swapping of predictions and the calculation of the permuted AUC_D scores. Detailed steps are described in Algorithm 5. Through this, candidate causal models from the TSCM search space that are statistically equivalent to the best-performing one are filtered. All such candidate models are compared against their sparsity, with the sparsest one being selected, thus avoid the problem of favoring the trivial solution of dense graphs during the search for the optimal causal model.

3.4.3 Evaluation of Simulation Quality

In this section, the capabilities of the proposed *ACT* mechanism are evaluated across two main axes: (i) the ability to accurately recover causal structures from temporal data and (ii) the capacity to simulate realistic temporal dynamics across synthetic and real-world cases. These evaluation experiments aim to assess the robustness of ACT in causal discovery and its competitiveness in

temporal data generation against both causal[5] and non-causal baselines.

The main evaluation metrics for these experiments are: (i) the *Structural Hamming Distance (SHD)* [Tsamardinos et al., 2006] for assessing causal structure discovery, computed betweeen the estimated and ground-truth graphs. SHD quantifies the number of graph operations (edge additions, deletions, or reversals) required to match the true graph, where a lower value indicates higher structural accuracy. A reversal accounts for two operations, while a deletion or addition for one. This metric is used to evaluate TCS in Figure 3.4. (ii) *Adjacency-based AUC*, notated AUC_{adj}. Following Cheng et al. [2024] and Stein et al. [2024], we compute an AUC score based on adjacency matrices of predicted and ground-truth DAGs. The predicted adjacency probabilities are thresholded to $\tau = 0.05$ and compared against the true structure; both matrices are flattened before computing their AUC. This score evaluates the ML efficacy of learned causal structures and is distinct from the sample-level discrimination score AUC_D introduced previously.

3.4.3.1 Evaluation of the Estimated Causal Structure

To assess the ability of ACT to correctly infer causal relationships, we generate synthetic datasets with varying levels of complexity. Each dataset consists of ten (10) variables, modeled as an additive noise TSCM with non-linear functional dependencies and Gaussian noise. The underlying causal graphs differ in edge density, ranging from three (3) to thirty (30) edges, thus simulating increasing structural difficulty. These datasets are generated as elaborated in Section 3.3. Two experimental scenarios are considered: (i) *Fully Connected Candidate Graphs* and (ii) *Oracle Ground Truth Candidate Graphs*.

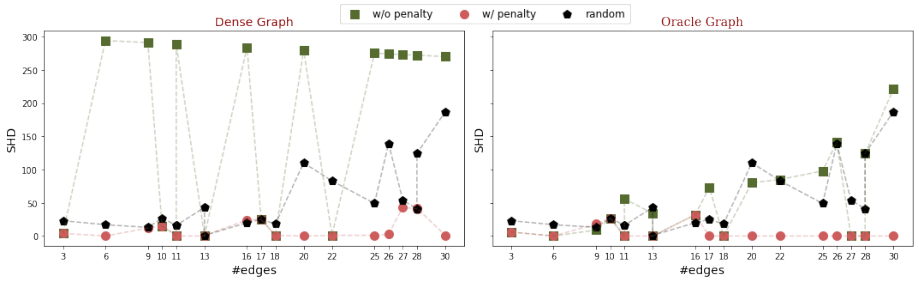
The first setup examines whether ACT avoids the trivial fully-connected causal structure when such a configuration is included in the search space. Nineteen (19) synthetic datasets are analyzed *(C1)*, each processed by ACT both *with* and *without* the sparsity penalty term. Our results indicate that the sparsity-regularized runs successfully avoided the trivial fully connected solution in all 19 cases, while 8 out of 19 non-regularized runs incorrectly selected it, especially in the case of denser graphs (more than 10 edges). This demonstrates the critical role of sparsity regularization in guiding ACT toward parsimonious causal graphs.

In the second setup, the true causal graph is included as a candidate solution (oracle) in Phase 1, to test whether ACT is able to recover it correctly. With sparsity penalty enabled, ACT successfully identifies the correct causal graph in 16 out of 19 cases (SHD = 0). Without the sparsity penalty, the true structure is recovered in only 8 out of 19 cases. However, even in mismatched cases, the estimated graphs exhibit low SHD scores, indicating that our method is able to recover solutions that are close to the ground truth.

3.4.3.2 Comparison to the State-of-the-Art

Next, we evaluate the ability of ACT to generate realistic temporal data perform a comparison against both causal and non-causal simulation approaches.

[5]Although it has been discussed that CausalTime is not based on a causal model, it is placed in this category for our comparison due to its context.



(a) (b)

Figure 3.4: **(a)**: Parallel calls of TCS with (w/) and without (w/o) the sparsity penalty, while providing a fully-connected lagged causal graph (referred as *dense graph*) in the 1^{st} phase outputs. The selected causal graphs are compared to the ground truth through SHD, where low SHD scores mean the graphs resemble each other. Calls with sparsity penalty successfully avoid the fully connected graph as a candidate solution at all cases. On the other hand, calls without sparsity penalty end up selecting the fully connected graph 8 out of 19 times, especially for denser cases with 11 or more edges in the ground truth. **(b)**: Parallel calls of TCS with and without the sparsity penalty, while providing the ground-truth as an oracle lagged causal graph in the 1^{st} phase outputs. Similarity measured through SHD, as in (a). In the vast majority of 16 out of 19 cases, calls with the sparsity penalty successfully chose the oracle graph as the optimal solution. In contrast, calls without the sparsity penalty were able to identify the oracle graph only in 7 out of 19 cases. The results in both (a) and (b) establish the sparsity penalty as a necessary part of TCS.

The considered methods are (i) *CausalTime* [Cheng et al., 2024], a predictive-modeling-based simulator that captures feature importance but does not perform explicit causal discovery, (ii) *CPAR* [Zhang et al., 2022], a conditional probabilistic auto-regressive generative model implemented within the *SDV* framework [Patki et al., 2016] and (iii) *TimeVAE* [Desai et al., 2021], a variational autoencoder model for time-series generation, implemented via the *SynthCity* library [Qian et al., 2023].

All considered methods are evaluated on both synthetic and real-world datasets, created in accordance with Section 3.6. For each dataset, 2000-sample sequences are drawn using a block bootstrap approach [Gonçalves and Politis, 2011, Härdle et al., 2003], repeated 10 times in order to compute standard errors. Additionally, the preprocessing protocol of [Cheng et al., 2024] is adopted, including linear interpolation for missing values. It should be once more noted that a comparison is inherently unfair, as causal simulation faces a much complex task: ACT generates complete Temporal Structural Causal Models (TSCMs), while CausalTime and other baselines operate at the correlation level without any causal interpretability.

Despite addressing the challenging task of causal data generation, ACT achieves AUC_D values on par with or better than purely statistical generators, especially on synthetic and semi-synthetic datasets. While non-causal models such as CPAR and TimeVAE focus solely on reproducing correlations, ACT is unique in its ability to produce a complete TSCM. As such, although causal

Table 3.6: Benchmarks against non-causal simulation: Comparison of TCS/ACT against causal (CausalTime) and non-causal (CPAR, TimeVAE) simulation methods. Despite solving a more complex causal task, ACT achieves comparable discrimination scores (AUC_D) across datasets, often outperforming CausalTime. Lower AUC_D values (in red) indicate more realistic simulations.

Dataset	**Method**			
	TCS (ACT)	**CausalTime**	**CPAR**	**TimeVAE**
WTH	$0.85_{\pm 0.01}$	$0.80_{\pm 0.00}$	$0.99_{\pm 0.00}$	$0.98_{\pm 0.00}$
AirQualityUCI	$0.99_{\pm 0.01}$	$0.83_{\pm 0.00}$	$0.99_{\pm 0.00}$	$0.95_{\pm 0.00}$
AirQualityMS	$0.97_{\pm 0.00}$	$0.84_{\pm 0.00}$	$0.98_{\pm 0.00}$	$0.99_{\pm 0.00}$
ETTh1	$0.92_{\pm 0.00}$	$0.80_{\pm 0.00}$	$0.95_{\pm 0.00}$	$0.91_{\pm 0.00}$
ETTm1	$0.93_{\pm 0.00}$	$0.82_{\pm 0.00}$	$0.96_{\pm 0.00}$	$0.91_{\pm 0.00}$
Bike Usage	$0.88_{\pm 0.01}$	$0.77_{\pm 0.00}$	$0.90_{\pm 0.00}$	$0.83_{\pm 0.00}$
Outdoors	$0.99_{\pm 0.00}$	$0.83_{\pm 0.00}$	$0.99_{\pm 0.00}$	$0.81_{\pm 0.00}$
fMRI	**0.71** $_{\pm 0.01}$	$0.78_{\pm 0.01}$	**0.74** $_{\pm 0.02}$	$0.75_{\pm 0.00}$
C1	**0.57** $_{\pm 0.00}$	$0.77_{\pm 0.00}$	**0.63** $_{\pm 0.01}$	$0.81_{\pm 0.00}$

data generation remains a challenging task, our proposed framework remains competitive while being the only approach that returns the estimated TSCM from data.

A shortcoming of using the Min-max search scheme of TCS, together with the sparsity penalty when attempting to generate hundreds of thousands of instances for training an LCM model is the significant computational overhead. For instance, considering three options for causal discovery, five for functional relationship estimation and two for noise estimator (not taking into account any configurable hyperparameters of each method), this would result in generating a search space of 30 configurations, from which one would select a single data instance. This greatly reduces the amount of training pairs one is able to generate, thus deeming it inefficient for our end scope regarding training pair generation for LCMs. It remains however a significant contribution of this work towards realistic causal data generation itself, as well as for curating datasets for benchmarking causal discovery algorithms.

3.5 Generation of Interventional Samples

So far our discussion concerned generation of temporal SCMs and observational data from their causal structure. As introduced methodologies (Sections 3.3 and 3.4) output causal models. From these causal models, observational data are then generated. The final pillar of data generation concerns the creation of interventional samples. As elaborated in Chapter 2, an intervention $\text{do}(V_t^i) = x_t$ corresponds to replacing the functional dependency of V_t^i in the time-lagged SCM at timestep t with $V_t^i := x_t$, in the case of a hard intervention and $V_t^i := f(\text{Pa}(V_t^i), \epsilon_{V^i}) + x_t$ in the case of a soft intervention. The general method for performing anestral sampling in the TSCM to obtain interventions is illustrated in Algorithm 6.

Algorithm 6 Temporal SCM Interventional Sampling (INTERVENTION)

Input: Temporal Causal Graph $\mathcal{G}$, number of timesteps T, max lag $\ell_{\max}$, number of warmup steps W, intervention probability $p_{\text{int}} \in [0,1]$, variable-wise value ranges $\mathcal{R} = \{[v_i^{\min}, v_i^{\max}]\}_{i=1}^{N}$

Output: Interventional time-series sample $\{\mathbf{X}_t\}_{t=1}^{T}$, intervention mask $\mathbf{B} \in \{0,1\}^{T \times N}$

Initialize X_0^i for all $i \in \{1, \dots, N\}$ with random noise
Initialize intervention mask $\mathbf{B} \leftarrow \mathbf{0} \in \{0,1\}^{T \times N}$
for $t = W + \ell_{\max} + 1$ to T **do**
  for each variable X_t^i **do**
    if Bernoulli$(p_{\text{int}}) = 1$ **then**
      Sample intervention value $x_{\text{int}}^i \sim \mathcal{U}(v_i^{\min}, v_i^{\max})$
      Set $X_t^i \leftarrow x_{\text{int}}^i$
      Set $B_{t,i} \leftarrow 1$
    else
      Determine lagged parents:
        $\text{Pa}(X_t^i) \leftarrow \{X_{t-k}^j \mid (X^j, X^i) \in \mathcal{G},\ 1 \le k \le \ell_{\max}\}$
      Compute $X_t^i \leftarrow f_i(\text{Pa}(X_t^i)) + \epsilon_t^i$
    end if
  end for
end for
return $\{\mathbf{X}_t\}_{W+\ell_{\max}}^{T}$, $\mathbf{B}$

In this work, data collections containing interventions are performed by hard interventions, uniformly sampled from the observational domain of each variable, i.e. $v^i \sim \mathcal{U}(V_i^{\min}, V_i^{\max})$. Such sampling is performed at each generated timestep and thus accounts for multiple interventional targets. The purpose of the above is two-fold: First, it ensures that interventions remain realistic and within the natural range of the variables, avoiding extreme or implausible values that could distort the dynamics or violate the underlying causal constraints. Second, it promotes diversity and coverage in the interventional data themselves, allowing the model to observe and learn causal effects across the full suport of each variable's distribution.

Each of these interventional samples is also paired with a *binary intervention mask*, which indicates which variables were intervened on at each timestep. Formally, the mask is $\mathbf{B} \in \{0,1\}^{T \times N}$, where $B_{t,i} = 1$ if variable X^i was intervened on at timestep t, and $B_{t,i} = 0$ otherwise. This formulation accounts for both single as well as multiple interventional targets. This does not only serve as a way to test performance of LCMs when training with interventional data as well, but also provide important information for a possible meta-analysis of these data, as an important component of the generation pipeline. The way interventional samples along with the intervention mask are handled by LCMs in particular is explained in Chapter 4.

3.6 Curating Training & Evaluation Data Pairs

In this section we provide an overview of the data curation process used to generate the data pairs used for training and evaluation of our LCMs. Three different types of datasets are considered: synthetic (following Section 3.3), simulated (following Section 3.4) and semi-synthetic. Sections 3.6.1, 3.6.2 and 3.6.3 provide a detailed description of each category of data corpora considered.

3.6.1 Semi-synthetic

For inference and evaluation purposes, we additionally consider *semi-synthetic* data collections, which combine mechanistic or physically motivated processes, constructed using either domain knowledge or known physical models. Such datasets provide an intermediate bridge between fully synthetic causal structures (where ground truth is known by construction) and real-world observational data (where causal ground truth is unknown or uncertain).

The *fMRI* datasets are based on the collection introduced by Smith et al. [2011], simulating blood-oxygen-level dependent (BOLD) responses across multiple brain regions. Each dataset models the neural activation time-series of a set of interconnected brain regions, with inter-regional causal connectivity defined according to a ground-truth directed network and the hemodynamic response modeled by the non-linear balloon model of Buxton et al. [1998]. This results in temporally smooth, non-linearly coupled time-series that realistically emulate the noise and temporal dependencies observed in empirical fMRI recordings.

From the full available collection of 27 datasets with 5, 10 and 15 variables, with sample lengths ranging from 50 to 5000 time steps, we restrict attention to datasets containing at least 500 observations, and exclude the 15-dimensional case to align with our input dimension constraints. This yields a final subset of 26 datasts, denoted as the *fMRI* collection. The *fMRI5* collection represents a smaller subset including only 5-dimensional networks.

To complement the limited number of fMRI datasets and further assess the ability of our LCMs to capture complex, nonlinear dynamics, we also include semi-synthetic data generated from the Kuramoto model of coupled phase oscillators [Kuramoto, 1975]. Briefly, the Kuramoto model describes the time evolution of oscillator phases $\theta_i(t)$ as

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N} \sum_{j=1}^{N} a_{ij} \sin(\theta_j - \theta_i), \tag{3.3}$$

where ω_i represents the intrinsic frequency of oscillator i, K controls the coupling strength, and a_{ij} encodes the adjacency (causal) structure of interactions among oscillators. The implementation from Löwe et al. [2022] is adopted, generating both 5-variable and 10-variable configurations with 1000 realizations each and a fixed maximum lag of 1.

For all considered semi-synthetic datasets, the known causal connectivity is provided in the form of directed adjacency matrices, which we convert into the lagged adjacency tensor representation described in Section 2.6 and Chapter 4 to enable correct comparison of the ground truth with the output of our LCMs. An overview of all semi-synthetic dataset collections is presented in Table 3.7.

Table 3.7: Overview of semi-synthetic time-series dataset collections.

Collection	Datasets	Vars	Timesteps	Max Lag	Rels.
fMRI5	21	5	1200–5000	1	Non-linear
fMRI	26	5–10	1200–5000	1	Non-linear
Kuramoto5	1000	5	500	1	Non-linear
Kuramoto10	1000	10	500	1	Non-linear

3.6.2 Synthetic

In addition to the proposed synthetic generation pipeline (Section 3.3), the synthetic data generator proposed by Stein et al. [2024] is also considered as an evaluation test set. Their formulation is based on what they refer to as a *time-invariant SCM with additive noise* (independent Gaussian noise) terms and time-invariant functional mechanisms (thus enforcing causal stationarity), where functional relationships are determined by the lagged adjacency tensor $\mathbb{A}$ and remain fixed across time. Their generation protocol begins by sampling a random lagged adjacency tensor $\mathbb{A}$ of predefined dimensionality. It then randomly selects non-zero elements of $\mathbb{A}$ to generate a single training example, based on a *link threshold parameter* $\rho \in (0, 1)$, which determines the probability $1 - \rho$ of including a causal edge in the adjacency tensor. A higher threshold would thus enforce sparser graphs. Each non-zero element is assigned a functional dependency f_{ij}^{ℓ} drawn from a predefined collection of functional families, including the non-linear transformations shown in Table 3.8 while linear cases are created with a *Vector Autoregressive Model (VAR)* form. From there, observational samples are generated.

As mentioned in Section 3.2, the authors attempt to guarantee stability of generated data samples by forcing boundness of the generated samples into a predefined interval. We found that the non-linear functions $\text{clamp}_{-0.5,0.5}(x)$, x^2 and $\frac{1}{x}$ contribute negatively to the stability and computational efficiency of the generator and were thus not considered. To populate each non-zero entry, the authors sample unifomly from a pre-defined range of causal strength values, while various variances are selected for the independent noise terms. Min-max normalization is also applied just like in our generation mechanism.

Table 3.8: Function families used for nonlinear datasets, Linear cases are parameterized as VAR models, while the nonlinear set (NL) is drawn from a pool of transformations, with one sampled per edge.

Family	Functional forms
Linear	$f(x) = ax + b$
Nonlinear (basic)	e^x, $\sin(x)$, $\cos(x)$, $\lvert x\rvert$, x
Nonlinear (bounded/activations)	$\sigma(x)$, $\text{ReLU}(x)$, $\log(\sigma(x))$

Using the above generator, we curated a set of dataset collections spanning $3-5$ variables, with $\ell_{\max} = 3$ and both linear (L) and non-linear (NL) functional forms. These datasets are aggregated into the `S_Joint` corpus, which includes $82,000$ training, $9,000$ validation, and $9,000$ test samples. This collection serves as in-distribution, holdout datasets for ablation and generalization studies.

Beyond these external generators, the `Synth_230k` corpus is introduced using the synthetic generation pipeline from Section 3.3, which is used to train large-scale LCMs. The collection extends the diversity of created samples, supporting up to 12 variables and three lags, as well as and ensuring causal stationarity by construction, and consisting of an 80% training, 10% validation and 10% test set as before. Importantly, it surpasses prior training set fforts by more than twofold in total instance count.

Table 3.9: Overview of synthetic time-series dataset collections used in this work. L denotes linear and NL non-linear. Sizes correspond to 80% training, 10% validation, and 10% test splits.

Collection	Size	Vars	Timesteps	Max Lag	Rels.	Obs./Interv.
`S_joint`	100k	3–5	500	1–3	L, NL	Observational
`Synth_230k`	230k	3–12	500	1–3	L, NL	Observational

3.6.3 Simulated

For the selection of real-data to generate simulated samples, we adopt datasets well-used throughout the deep learning time-series forecasting literature [Hahn et al., 2023], from a variety of real-world domains. These include domains of power, weather, and transportation. The *ETTh1* and *ETTm1* data collections [Wu et al., 2021] record hourly and quarter-hourly measurements respectively of electricity transformers, as well as oil and load temperature in a two-year span between July 2016-2018. The *WTH* dataset [Jiang et al., 2023] consists of 35064 hourly atmospheric measurements, including temperature, humidity, wind speed, direction and pressure starting from January 1st 2010. The *AirQualityUCI* corresponds to 12 hourly measurements of metal oxide chemical sensors in an Italian city. The *bike-usage* dataset contains 52584 hourly wire and infrared sensor measurements of both bike and pedestrian data at the Burke-Gilman Trail in King County, Washington[6], for a total of 5 time-series, while the *Outdoors* dataset contains outdoor BME280 sensor measurements from Nanning, China between February 21st 2016 and November 25th 2016, for a total of 1440 time-steps. Lastly, we include the *Air_Quality* dataset, consisting of hourly data from 36 monitoring stations across China, as available in the work of Cheng et al. [2024]. An overview is provided in Table 3.10.

For any of these input time-series, we verify for stationarity using the *Augmented Dickey-Fuller test* [Dickey and Fuller, 1979] (ADF), described in Algorithm 7. In case non-stationarity is observed, it is normalized using finite differences up to second order. This step assures that data fed into training our LCMs do not possess extreme trends, possibly resulting in unstable training. We illustrate the pseudocode in Algorithm 7. We follow the python implementation of `statsmodels` with maximum lag order $12 \cdot \left(\frac{T}{100}\right)^{1/4}$ where T is the number of observed timesteps.

Ultimately, we introduce a large-scale *simulated* dataset collection, denoted `sim_45K`. These datasets are derived from real multivariate time-series sources by

[6]Data is publicly available by the City of Seattle Open Data Portal (last accessed Oct 26, 2024).

Table 3.10: Overview of real time-series datasets used as inputs to the TCS algorithm for generating simulated data pairs.

Dataset	Variables	Timesteps	Granularity	Start Date	Domain
WTH	12	35064	1 hour	01/01/2020	Weather
ETTh1	7	17420	1 hour	07/01/2016	Power
ETTm1	7	69680	15 min	07/01/2016	Power
Air_Quality	36	8760	1 hour	-	Weather
AirQualityUCI	12	9357	1 hour	03/10/2024	Weather
Bike-usage	5	552584	1 hour	01/01/2014	Transportation
Outdoors	3	1440	1 sec	21/02/2016	Environmental

Algorithm 7 Dickey-Fuller Test (ADF)

Input: Time-series data $\{y_t\}_{t=1}^{T}$, maximum lag order p
Output: Test statistic τ, p-value, decision on unit root

1: Compute first differences: $\Delta y_t \leftarrow y_t - y_{t-1}$ for $t = 2, \ldots, T$
2: Construct regression design matrix with:

1. Regressor y_{t-1}
2. Deterministic terms (constant, trend) if included
3. Lagged differences Δy_{t-k} for $k = 1, \ldots, p$

3: Estimate regression by OLS:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{k=1}^{p} \phi_k \Delta y_{t-k} + \epsilon_t$$

4: Extract test statistic $\tau \leftarrow \hat{\gamma}/\mathrm{SE}(\hat{\gamma})$
5: Compare τ against Dickey-Fuller critical values
6: **if** $\tau < \tau_{\text{critical}}$ **then**
7: Reject null hypothesis (series is stationary)
8: **else**
9: Fail to reject null hypothesis (series has a unit root)
10: **end if**
11: **return** Test statistic τ, p-value, decision

simulating causal models under the Temporal Causal-based Simulation (TCS) framework (Subsection 3.4.1). Starting from well-established real-world datasets in the time-series forecasting literature [Lai et al., 2018, Wu et al., 2023], we generate realistic causal variants using three main data augmentation techniques: (i) *Time-window subsampling:* extracting multiple overlapping or disjoint temporal segments from long sequences to simulate diverse initial conditions and phase shifts; (ii) *Node subset sampling:* selecting variable subsets to emulate local subsystem dynamics and improve structural diversity, and (iii) *Estimator variability:* varying the causal estimation and simulation parameters within ACT to produce multiple realizations of plausible causal graphs from a single

base dataset.

This procedure yields over 45,000 high-quality simulated instances that preserve the statistical structure of real data while offering known causal ground truth. Together with the synthetic corpora (`synth_230k`), these datasets form the combined large-scale training corpus `synth_230k_sim_45k`, which is *used for training large-scale LCMs*. The combined dataset totals *more than 275,000 examples*, more than double the size of previous causal time-series foundations. Additionally, the `AirQualityMS` dataset is introduced as a realistic out-of-distribution benchmark, from which we create 50 simulated instances by ACT for further evaluating our trained models.

In addition to purely synthetic and simulated collections, we curated a set of *mixture datasets* designed to empirically investigate the optimal balance between synthetic and realistic (simulated) data during large-scale model training, in line with recent findings by Das et al. [2024] as elaborated in Section 3.2. To this end, we curate four mixture datasets with varying ratios of synthetic to simulated data, based on the `synth_230k_sim_45k` dataset, each consisting of $50,000$ paired instances. All datasets preserve consistent temporal characteristics, with variable count in the range $[3, 12]$, maximum lag $\ell_{\max} = 3$ and 500 time steps. Only data composition is altered. Each collection follows an 80% / 10% / 10% split for training, validation, and test data as in the previous cases, as shown in Table 3.11.

Table 3.11: Mixture dataset subsets curated to explore the effect of varying synthetic-to-simulated ratios on training dynamics and generalization. Each subset contains 50k data pairs.

Collection	**Size (pairs)**	**Synthetic (%)**	**Simulated (%)**
`mix_100s_0sim`	50k	100	0
`mix_80s_20sim`	50k	80	20
`mix_50s_50sim`	50k	50	50
`mix_20s_80sim`	50k	20	80

These collections are introduced in Chapter 6 to explore the optimal combination of simulated and synthetic data instances for training large-scale LCMs and hether they benefit out-of-distribution performance compared to training on synthetic data alone.

Finally, we also introduce a synthetic corpus combining observational and interventional samples, `Synth_40k`, which is introduced for preliminary experiments in Chapter 6 to explore the effect of interventions on the performance of LCMs. As the name suggests, it consists of forty thousand synthetic instances, of the same splits as the previous collections, up to 5 variables and $\ell_{\max} = 3$. Each instance consists not only of the ground truth lagged causal graph and the observed time series, but also of the intervened samples, created as in Section 3.5.

Table 3.12: Overview of the large-scale synthetic and simulated dataset collections used for training, ablation, and evaluation of large causal models (LCMs). Each dataset follows a 80%/10%/10% split for training, validation, and test data. `Synthetic_230k` forms the primary synthetic corpus; `Sim_45K` comprises simulated (real-derived) data for realism evaluation; `Synth_230k_sim_45k` is their combined mixture used for large-scale training and mixture analysis; and `S_Joint` (adapted from Stein et al. [2024]) supports ablation studies under varying causal configurations.

Collection	Type	Instances	Vars	Lags
`Synthetic_230k`	Synthetic (Section 3.3)	230k	3–12	1–3
`Sim_45K`	Simulated (Section 3.4)	45k	3–12	1–3
`Synth_230k_sim_45k`	Mixture	275k	3–12	1–3
`S_Joint`	Synthetic (Stein et al. [2024])	100k	3–5	1–3

3.6.4 Data Shardification

Given the large scale of the combined dataset (totaling more than 275,000 paired instances), it becomes impractical to load the entire collection into memory simultaneously during training. To address this limitation, we adopt a *shardification* approach, in which the dataset is partitioned into multiple serialized files (`.pt` shards), each containing a manageable subset of samples. This design allows efficient storage, distribution, and access to data instances without compromising training throughput.

To support seamless access across all shards, a custom data loader is introduced, treating the collection of sharded files as a single, unified dataset. During training, shards are loaded on demand (only the file corresponding to the currently accessed index range is read into memory, while previous shards are released). This loading mechanism substantially reduces memory footprint and enables training on systems with limited resources, while maintaining constant-time indexing behavior for random sample retrieval.

Each shard is dynamically wrapped by the main data loader, which applies standardized preprocessing steps such as normalization, temporal alignment, and fixed-length padding or truncation to $L_{\max}$. To prevent data leakage across shards, all samples are deeply cloned before returning, guaranteeing memory isolation and reproducible sampling.

At the beginning of each training epoch, the shard order is randomized to promote data diversity and prevent potential overfitting to shard-specific distributions. The result is an efficient, scalable data pipeline that allows LCMs to be trained over hundreds of thousands of simulated and synthetic causal time-series pairs without exceeding hardware memory limits.

This page intentionally left blank

4

Architecture

"The design is not just what it looks like and feels like. Design is how it works."

– Steve Jobs

Previous chapters established the theoretical and methodological ground for a scalable, data-driven foundation-model approach to temporal causal discovery. This chapter focuses on the architectural design of LCMs. In contrast to explicit statistical or structural constraints, LCMs must intrinsically learn to represent complex temporal dynamics and generalize across heterogeneous data distributions. The goal is first, to formalize the key design considerations and characteristics that guide causal discovery under the foundation model paradigm, and second, to present the detailed architectural components that realize the above.

Section 4.2 outlines the guiding principles that foundation models must adhere to. Section 4.3 discusses the preprocessing and input mechanisms required to handle variable-length and multivariate time-series, including normalization and padding strategies for both the input sequences and the corresponding ground-truth causal graphs. Section 4.4 provides an overview of the model, describing its major computational blocks and their purpose. Section 4.5 introduces a novel LCM design that promises enhanced performance through various architectural refinements. Finally, Section 4.6 presents a preliminary exploration of two natural extensions that highlight the LCM's potential as a more versatile causal foundation model: the integration of interventional samples (Section 4.6.1) and the incorporation of prior knowledge (Section 4.6.2). These investigations serve as a proof-of-concept for future comprehensive work.

4.1 Introduction

4.2 Objectives & Considerations

The primary objective of the Large Causal Model (LCM) is to learn a robust and generalizable mapping from multivariate time-series to their corresponding lagged causal graphs. To achieve this, the architecture must satisfy several key desiderata that arise from the challenges of temporal causal discovery under the foundation model paradigm.

First, the model must be capable of learning from highly heterogeneous datasets that vary in size, dimensionality, and underlying temporal dynamics. This diversity demands flexible neural representations that can adapt to varying numbers of variables, time horizons, and sampling frequencies. Second, the model must exhibit strong *generalization* and zero-shot inference capabilities, performing reliably on unseen domains or data distributions outside the training corpus. Third, it must process variable-length input sequences, analogous to how large language models (LLMs) handle text sequences of differing lengths. Finally, inference should be computationally efficient and ideally constant-time with respect to the number of variables, contrasting with the exponential complexity of constraint-based causal discovery algorithms.

These requirements render traditional causal discovery algorithms and classical variational inference approaches inadequate for large-scale, heterogeneous settings. Variational methods, while conceptually elegant, typically operate on single datasets and rely on restrictive assumptions about noise distributions and graph sparsity, limiting their scalability. Consequently, we turn to neural architectures inspired by foundation models in time-series forecasting and natural language processing, where large-scale supervised pretraining has demonstrated powerful generalization properties. In particular, the *Transformer architecture* offers a compelling foundation due to its flexibility, parallelizability, and capacity to capture long-range dependencies.

A central design decision in the LCM is therefore the adoption of a Transformer-based backbone [Vaswani et al., 2017]. The self-attention mechanism underlying Transformers allows efficient modeling of pairwise dependencies across time and variables, while its modular structure scales effectively across dataset sizes and modalities. This flexibility aligns closely with the requirements of temporal causal discovery, where causal effects may occur across arbitrary time lags and between heterogeneous variables. To motivate this choice, we contrast the Transformer with prior neural architectures traditionally used in time-series modeling, namely recurrent networks, convolutional models, and variational inference-based approaches.

Among the architectures explored for temporal representation learning and causal discovery, recurrent models such as LSTMs and GRUs [Hochreiter and Schmidhuber, 1997, Cho et al., 2014] were historically favored for their ability to capture sequential dependencies, yet their sequential nature limits scalability and parallelism. Temporal convolutional networks (TCNs) [Van Den Oord et al., 2016, Bai et al., 2018, Nauta et al., 2019] introduced dilated convolutions to increase receptive fields, improving efficiency while still struggling with non-stationary or long-range effects. Variational approaches [Löwe et al., 2022, Yang et al., 2021, Deng et al., 2022, Gong et al., 2022] offer probabilistic interpretability but often suffer from computational inefficiency and restrictive prior assumptions. In contrast, Transformer-based architectures [Vaswani et al., 2017, Stein et al., 2024, Das et al., 2024, Woo et al., 2024] have emerged as the most flexible and scalable foundation for temporal causal modeling, capable of capturing heterogeneous, long-range dependencies and supporting large-scale training paradigms such as those employed in Large Causal Models (LCMs).

Recurrent Neural Networks (RNNs) and their gated variants, such as *Long Short-Term Memory (LSTM)* [Hochreiter and Schmidhuber, 1997] and *Gated Recurrent Units (GRUs)* [Cho et al., 2014], represent early deep learning approaches to temporal modeling. While well-suited for short-range dependencies,

Table 4.1: Comparison of candidate neural architectures for temporal causal discovery.

Architecture	**Strengths**	**Limitations**
RNNs / LSTMs / GRUs [Hochreiter and Schmidhuber, 1997, Cho et al., 2014]	Capture sequential order; effective for short-term dependencies.	Sequential updates hinder scalability; vanishing gradients limit long-range modeling; fixed input sizes.
Temporal CNNs [Van Den Oord et al., 2016, Bai et al., 2018, Nauta et al., 2019]	Parallelizable; dilated convolutions extend receptive field; efficient training.	Local receptive fields; difficulty modeling irregular or long-range causal effects.
Variational Methods [Löwe et al., 2022, Yang et al., 2021, Deng et al., 2022, Gong et al., 2022]	Probabilistic treatment of uncertainty; principled latent structure modeling.	Computationally expensive for large graphs; rigid noise assumptions; limited scalability.
Transformers [Stein et al., 2024, Das et al., 2024, Woo et al., 2024]	Highly scalable; capture long-range and irregular dependencies; adaptable to heterogeneous data.	High computational cost; careful design of embeddings and normalization required.

their inherently sequential computation precludes efficient parallelization and introduces training bottlenecks for large-scale data. Furthermore, despite improvements in gating mechanisms, RNNs suffer from vanishing gradient issues that hinder the modeling of long-range dependencies [Bengio et al., 1994], making them less suitable for discovering causal relations spanning multiple time lags or variable types.

Convolutional Neural Networks (CNNs) adapted to sequential data alleviate some of these limitations by enabling parallel computation and expanded receptive fields through dilated convolutions [Bai et al., 2018]. Temporal CNNs, such as *WaveNet* [Van Den Oord et al., 2016], have achieved strong performance in sequence modeling tasks. However, their inductive biases remain inherently local (the scope of dependencies captured is tied to kernel size and dilation rate) which restricts their ability to model irregular or long-range causal effects. Such locality biases are misaligned with causal discovery, where dependencies may exist across non-contiguous time steps or arbitrary variable pairs.

Variational approaches to causal discovery conceptualize causal structures as latent variables and optimize corresponding evidence lower bounds (ELBOs) [Yang et al., 2021, Löwe et al., 2022, Deng et al., 2022, Gong et al., 2022]. Although theoretically principled, they quickly become intractable as the number of variables or maximum lag increases. Moreover, their reliance on dataset-specific optimization and fixed-size input assumptions limits their applicability

to heterogeneous temporal data and precludes large-scale pretraining.

Graph Neural Networks (GNNs) provide a complementary framework for modeling structured relational data. Their permutation-invariant aggregation mechanisms (such as mean or sum pooling) allow learning representations that respect the symmetries of causal graphs. Recent studies in temporal-spatial causal discovery leverage GNN layers to aggregate information across variables while maintaining temporal coherence [Job et al., 2025, Langbridge et al., 2023, Geffner et al., 2024]. Inspired by this, our LCM architecture integrates GNN-like aggregation over the feature dimension within the Transformer's attention-based framework, enabling a balance between permutation invariance and expressive temporal modeling. This hybrid design harmonizes the strengths of Transformers and GNNs, offering a unified and scalable architecture for temporal causal discovery.

In contrast, the Transformer offers several properties that are directly aligned with the desiderata of constructing an LCM. The concept of *self-Attention* enables the model to capture long-range and irregular dependencies between variables without explicit assumptions of locality. Parallelization of the self-attention mechanism allows computational scalability to large and diverse datasets, a requirement for training foundation-like causal models. Moreover, by leveraging a large training corpus, Transformers have been shown to perform robust zero-shot generalization to new domains, addressing a core limitation of both classical algorithms and variational methods. For the above reasons, a Transformer backbone is selected, specifically designed for temporal causal discovery, for all introduced models. Table 4.1 summarizes the comparative architectural considerations discussed above.

Table 4.2: Design constraints of the LCM architecture. These are engineering limitations that bound input dimensionality and model structure.

Constraint	**Value / Assumption**
Maximum Variables	$V_{\max} = 12$
Maximum Timesteps	$L_{\max} = 500$
Maximum Assumed Lag	$\ell_{\max} = 3$

Table 4.3: Causal assumptions underlying the LCM architecture. These are standard assumptions in causal discovery that define the valid regime of learned graphs.

Assumption	**Description**
Causal Sufficiency	No hidden confounders for any pair of variables
Causal Stationarity	Graph structure does not change over time
No Contemporaneous Effects	$\forall i \neq j, V_i^t \not\to V_j^t$
Causal Markov Condition	$X \perp\!\!\!\perp \text{Non-descendants}(X) \mid \text{Pa}(X)$
Faithfulness	Observed independencies reflect the true graph

As with any causal discovery in general, LCMs remain subject to a set of assumptions and design constraints. The most important ones are *design constraints* and *causal assumptions*. Regarding design constraints, we are interested in the maximum allowed dimensions of the input. For instance, text-to-speech

neural networks assume a maximum number of tokens or frames that can be processed within an audio sequence, while image recognition models are typically designed for a fixed input resolution (e.g. 224x224 in Resnet [He et al., 2016]). For our causal discovery task, this corresponds to the maximum number of allowed time-series, timesteps (or sequence length) and the maximum assumed lag for causalities to exist. These causal assumptions and design constraints define the operational regime of our LCMs by determining both the feasible input space and the valid interpretations of discovered causal graphs. Tables 4.3 and 4.2 summarize these assumptions and constraints. Their consideration is pivotal not only for the construction of the training data in Chapter 3 and the model's structure, but also for correct interpretation of the inferred causal graphs and any further causal query. A challenge in this phase is that choosing these limits too conservatively restricts the applicability of the model, while overly generous limits may hurt generalization and reduce scalability due to increased computational demands.

4.3 Handling Variable-Length and Multi-variable Sequences

Before elaborating on each separate building block of the neural architecture, we clarify the ability of the model to handle inputs of variable number of time-series and variable number of timesteps. To this end, we adopt a set of *standardized padding strategies*, similarly to Stein et al. [2024], both for the input time-series and the ground truth lagged adjacency tensor. This enables consistent batch processing and generalization of trained models across heterogeneous dataset configurations, as the model is able to handle an input of fixed dimensions.

To meet the input dimension assumptions, generated time-series samples and ground truth causal graph pairs (both synthetic and simulated from Chapter 3) are padded to a maximum of $L_{\max}$ time-steps and $V_{\max}$ variables. The ground truth causal graph is padded up to $V_{\max}$ variables and $\ell_{\max}$ lags, while the input time series is padded up to $V_{\max}$ variables and $L_{\max}$ timesteps. As described in Table 4.2, our LCMs are designed to handle a maximum of $V_{\max} = 12$ variables and $\ell_{\max} = 3$ lags.

4.3.1 Time-Series Padding

When the number of observed time-steps (samples) L is less than the configured maximum number of samples $L_{\max}$, the time-series is *padded with Gaussian noise* $\mathcal{N}(0, 0.01)$ *along the time-step dimension.* This prevents introducing zero artifacts, preserving the marginal statistics, and makes the model robust to handling varying sequence lengths. When inputs are longer than the configured maximum (either in time-steps $L_{\max}$ or the number of time-series $V_{\max}$), the excess is *truncated* to fit the maximum dimensions. This is conceptually similar to how Foundation models like the GPT family [Brown et al., 2020] truncate input lengths that exceed the maximum allowed tokens. The same strategy is applied for interventional samples, illustrated in Figure 4.1.

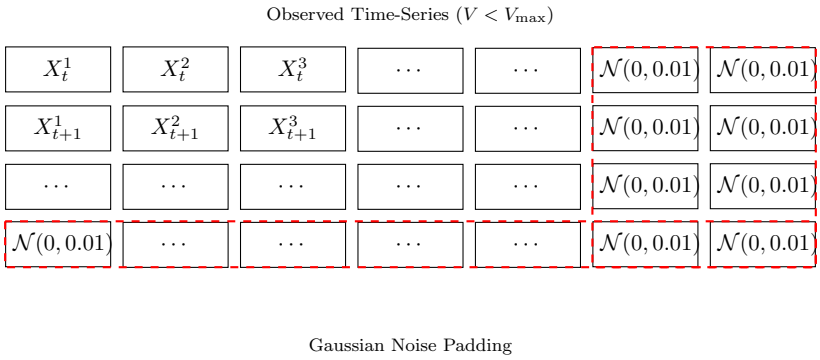


Figure 4.1: Observed time-series matrix padded with Gaussian noise to reach $L_{\max}$ (sequence length) and $V_{\max}$ (number of variables). Red dashed boxes indicate padded regions.

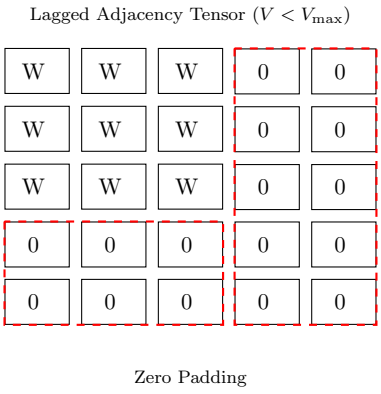


Figure 4.2: 2D slice of a lagged adjacency tensor (e.g. lagged adjacency matrix at lag 1) padded with zeros to reach $V_{\max}$. Zero-padding is applied when $V < V_{\max}$ or $\ell < \ell_{\max}$. Red dashed boxes indicate padded regions.

4.3.2 Causal Graph Padding

A similar procedure is applied to the padding of the lagged causal graph. When a dataset comprises fewer variables than $V < V_{\max}$ or when the underlying causal structure assumes a lower maximum lag $\ell < \ell_{\max}$, the corresponding lagged adjacency tensor is *zero-padded* along the unused variable and lag dimensions. This operation explicitly encodes the absence of nodes or temporal dependencies beyond the observed configuration and, together with the time-series padding described previously, constitutes a critical step for both the training and inference stages. It enables the model to uniformly process time-series and ground-truth causal graph pairs of varying time-step, node, and lag dimensionalities. During *training*, the foundation model receives a time-series input—typically observational—which is propagated through the sequential blocks of the architecture to produce a lagged adjacency tensor representing the confidence of edge existence, as detailed in Section 2.6. The predicted tensor is subsequently

compared to the appropriately padded ground-truth causal graph for loss computation, ensuring consistent alignment across lag dimensions. An illustration of the overall procedure is provided in Figure 4.2.

4.3.3 Min–max Normalization

To ensure consistent input scaling across heterogeneous datasets and to stabilize training, all time-series variables are subjected to min–max normalization prior to being fed into the LCM, both during training and inference. This normalization step rescales each variable to the range $[0, 1]$, thereby eliminating discrepancies in magnitude that may otherwise bias learning of the internal representations toward variables with larger numerical ranges. Formally, for each variable $x_v(t)$ in the dataset, the normalized value $\tilde{x}_v(t)$ is computed as

$$\tilde{x}_v(t) = \frac{x_v(t) - \min(x_v)}{\max(x_v) - \min(x_v) + \epsilon}, \tag{4.1}$$

where $\min(x_v)$ and $\max(x_v)$ denote the minimum and maximum values of variable v across all timesteps, and a small constant $\epsilon > 0$ is added for numerical stability.

This step ensures that all variables contribute comparably to the model's embedding and attention layers. During inference, i.e. with the pre-trained model weights at hand and with no ground-truth available, min–max normalization is applied using the statistics computed from the correspoding training set to maintain consistency between training and evaluation distributions.

4.4 Overview of the Informer-based LCM

In this section, we formally describe the underlying mechanisms and design of the LCM architecture, which corresponds to the Informer-based [Zhou et al., 2021] implementation of Stein et al. [2024]. Briefly, the building blocks revolve around an Encoder-only Transformer backbone[1], using self-Attention instead of the Sparse Attention of the Informer, with a final feed-forward block adapted for temporal causal discovery. As the scope is not autoregressive sequence generation (i.e. in time-series forecasting or natural language processing), the model does not require a decoder block and only uses the Encoder representations to make edge predictions. Accordingly, full attention is used instead of sparse attention as the time and spaceo complexity gains of the latter are not relevant in this context.

Recall from Section 2.6 that the goal of the LCM is to learn a mapping f_θ from multivariate time-series inputs to their corresponding lagged causal adjacency graphs, by predicting a *lagged adjacency tensor* $\hat{\mathbb{A}} \in [0, 1]^{V_{\max} \times V_{\max} \times \ell_{\max}}$ where each entry $\hat{\mathbb{A}}_{ji\tau}$ denotes the probability of edge existence $i \xrightarrow{\ell_{\max} - \tau} j$. As with any pre-trained neural architecture, we differentiate between the phases/steps:

1. *Training*, where the model is given as input a time-series and a ground truth causal graph pair, and outputs a lagged adjacency tensor of edge existence confidences

[1] An introduction to the Transformer is provided in Appendix B.

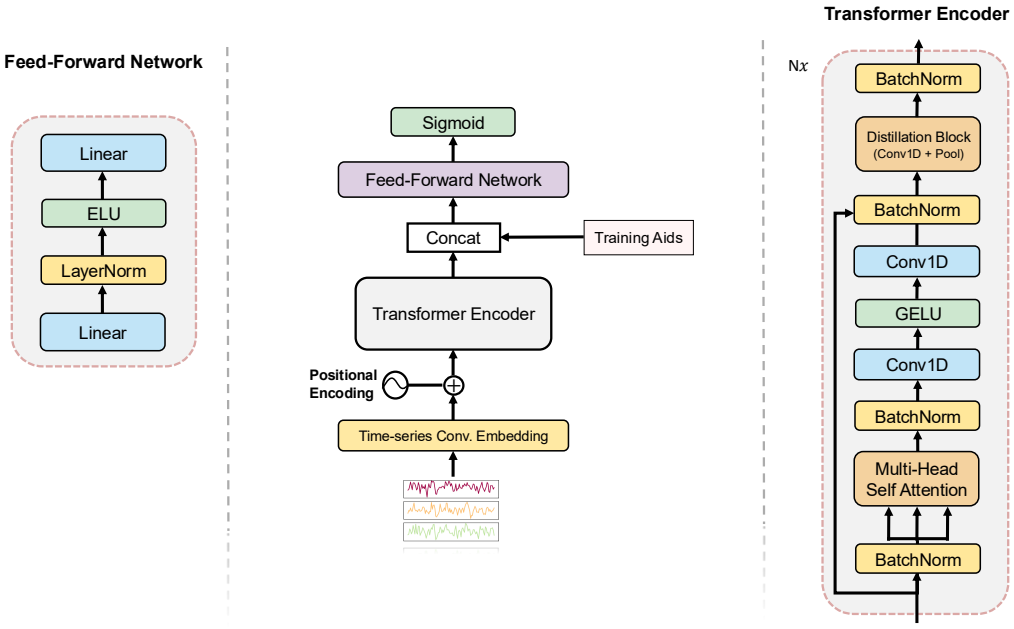


Figure 4.3: Overview of the Transformer-based architecture of the LCM, following the Informer [Zhou et al., 2021] variant of Stein et al. [2024]. Given a multivariate time-series input of L timesteps and V variables, noise padding is performing in the timestep and variable dimensions (Section 4.3) up to $L_{\max}$ and $V_{\max}$, which are then min–max normalized (Section 4.3.3). Input embeddings are created using *1D convolutions* and sinusoidal positional encodings. This is followed by a stack of transformer encoder blocks, performing batch normalization, multi-head self-attention and 1D convolutions, along with an *optional distillation block*. Finally, a feedforward block is used to infer a lagged adjacency tensor representation and a sigmoid activation is applied to obtain edge confidence scores.

2. *Inference* or *CD Phase*, where the model is given as input a time-series and outputs a lagged adjacency tensor $\hat{\mathbb{A}}$ of edge existence confidences, estimating the underlying true lagged causal graph $\mathcal{G}$ with adjacency tensor $\mathbb{A}$.

The Informer-based architecture of Stein et al. [2024] is now described in detail, which serves as the backbone of the Large Causal Models (LCMs) and is illustrated in Figure 4.3.

4.4.1 Input Representations

In the initial input phase, the model receives an (observational) time series sample $X \in \mathbb{R}^{B \times L \times V}$, where B denotes the *batch dimension*[2], L the *sequence length* (number of timesteps), and V the *number of variables*. Prior to any

[2]During causal discovery (CD) inference, the batch dimension is typically set to 1.

transformation, the inputs are min-max normalized. The normalized inputs are then padded along both the variable and temporal dimensions to ensure fixed dimensions $V = V_{\max}$ and $L = L_{\max}$, with sequences exceeding this length are truncated to $L_{\max}$, as detailed in Section 4.3.

Each variable's time series is embedded into a higher-dimensional latent space using a 1D convolutional projection, serving as a learnable feature extractor. The convolution operates along the temporal dimension, with $c_{\text{in}} = V_{\max}$ input channels and $c_{\text{out}} = d_{\text{model}}$ output channels, where d_{model} denotes the *model dimension*. The resulting *token embeddings* are represented as

$$X_{\text{tok}} = \text{Conv1D}(X), \tag{4.2}$$

yielding a tensor $X_{\text{token}} \in \mathbb{R}^{B \times L_{\max} \times d_{\text{model}}}$. This operation captures local temporal dependencies and smooths short-range fluctuations, providing robust contextual embeddings for subsequent attention mechanisms.

Since the Transformer architecture is permutation-invariant with respect to sequence order, explicit positional information must be added to the learned embeddings. As in the vanilla Transformer by Vaswani et al. [2017], sinusoidal positional encodings aim to inject absolute temporal position information into the convolutional embeddings. Specifically, for each timestep $t \in \{1, \ldots, L\}$ and embedding dimension index $j \in \{1, \ldots, d_{\text{model}}\}$, the positional encoding is defined as:

$$\begin{aligned} \text{PE}_{(t,2i)} &= \sin\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right), \\ \text{PE}_{(t,2i+1)} &= \cos\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right), \end{aligned} \tag{4.3}$$

which are added element-wise to the learnable token embeddings:

$$X_{\text{emb}} = X_{\text{token}} + \text{PE}. \tag{4.4}$$

Dropout is applied to X_{emb} to improve generalization, and the resulting embeddings are reshaped and organized as $[B, L, d_{\text{model}}]$, serving as the input to the Encoder block of the Transformer.

4.4.2 Transformer Encoder

The encoded sequence X_{emb} is then processed by a stack of Informer-style Transformer encoder blocks. Each encoder blocks consists of a self-attention layer followed by a convolutional feed-forward network. Let $E^{(l)}$ denote the input to the l-th encoder layer. Then each encoder layer computes

$$\begin{aligned} E'^{(l)} &= \text{LayerNorm}\big(E^{(l)} + \text{SelfAttn}(E^{(l)})\big), \\ E^{(l+1)} &= \text{LayerNorm}\big(E'^{(l)} + \text{FFN}(E'^{(l)})\big), \end{aligned} \tag{4.5}$$

where $\text{SelfAttn}(\cdot)$ denotes multi-head self-attention scaled by $1/\sqrt{d_{\text{model}}}$ as per Vaswani et al. [2017] and $\text{FFN}(\cdot)$ represents a two-layer convolutional feed-forward block defined as

$$\text{FFN}(x) = \text{Conv1D}_2\big(\text{GELU}\big(\text{Conv1D}_1(x)\big)\big), \tag{4.6}$$

applied independently at each temporal position. This convolutional replacement of the standard linear MLP (as introduced in the Informer model) allows for localized feature extraction and temporal smoothing, aiming to improve efficiency on long sequences. Each encoder block uses residual connections [He et al., 2016] and layer normalization [Ba et al., 2016, Xiong et al., 2020] after both the attention and feed-forward sublayers, ensuring stable gradient propagation across stacked layers.

Between successive encoder blocks, an optional *self-attention distillation layer* is optionally inserted to reduce the temporal resolution, as proposed in the Informer model. In this work, distillation layer is included in all trained models. Specifically, a convolutional downsampling followed by normalization and activation is applied:

$$X_{\text{distil}} = \text{MaxPool}\Big(\text{ELU}\big(\text{BatchNorm}\big(\text{Conv1D}(E^{(l)})\big)\big)\Big) \tag{4.7}$$

which halves the temporal dimension and improves computational efficiency. Let $E_{\text{enc}} \in \mathbb{R}^{B\times L'\times d_{\text{model}}}$ denote the encoder output after the final block, where $L' < L$ if temporal distillation is applied. Importantly, only the final timestep embedding is retained for causal prediction, $h = E_{\text{enc}}[:, -1, :] \in \mathbb{R}^{B\times d_{\text{model}}}$.

4.4.3 Correlation Injection

In an attempt to further assist the model in inferring directed dependencies, Stein et al. [2024] introduce a technique to inject prior statistical measures into the model, to improve performance and avoid inferring spurious relationships, as a form of *training aids*. These statistical measures take the form of lagged cross-correlations between variables. For each pair (X^i, X^j) and lag $\tau \in \{1, \dots, \ell_{\max}\}$, the empirical Pearson cross-correlation is computed as

$$\rho_{X^i,X^j}(\tau) = \frac{\text{Cov}(X^i_{1:T-\tau}, X^j_{\tau+1:T})}{\sqrt{\text{Var}(X^i_{1:T-\tau})\,\text{Var}(X^j_{\tau+1:T})}} \tag{4.8}$$

The resulting tensor of correlations $\mathbb{C} \in \mathbb{R}^{B\times V_{\max}\times V_{\max}\times \ell_{\max}}$ is flattened to $\mathbb{R}^{B\times (V^2_{\max}\cdot \ell_{\max})}$ and concatenated to the encoder representation (as shown in Figure 4.3):

$$h' = \text{Concat}(h, \text{Flatten}(\mathbb{C})) \in \mathbb{R}^{B\times (d_{\text{model}}+V^2_{\max}\cdot \ell_{\max})} \tag{4.9}$$

In Chapter 6, we thoroughly evaluate the effect of correlation injection features on the performance of LCMs, filling a gap that was previously unexplored in the literature, although the authors report that this approach is effective.

4.4.4 Feedforward Prediction Head

The concatenated representation h' is then processed by a two-layer feed-forward prediction head to map latent representations to *lagged adjacency predictions*. Formally,

$$\begin{aligned} h_1 &= \text{ELU}(\text{BatchNorm}(\text{Linear}_1(h'))), \\ h_2 &= \text{Linear}_2(h_1), \end{aligned} \tag{4.10}$$

where $\text{Linear}_1 : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{ff}}}$ and $\text{Linear}_2 : \mathbb{R}^{d_{\text{ff}}} \to \mathbb{R}^{V_{\max}^2 \cdot \ell_{\max}}$. The output is reshaped to form the predicted lagged adjacency tensor

$$\hat{\mathbb{A}} \;=\; \sigma\left(\text{Reshape}\big(h_2\big)\right) \in [0,1]^{V_{\max} \times V_{\max} \times \ell_{\max}} \tag{4.11}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid transformation and each element $\hat{\mathbb{A}}_{ji\tau}$ represents the confidence of the model that a directed edge $X^i_{t-\tau} \to X^j_t$ exists. The final sigmoid activation converts logits into interpretable probabilities of causal edge existence.

The cautious reader should notice that instead of a softmax to convert the final logits into probabilities (as in the vanilla transformer), a sigmoid is used. A sigmoid is appropriate for *binary* or *multi-label classification* problems where each output can be viewed as an independent Bernoulli event, while softmax is appropriate for multi-class classification where only one class is active at a time.

4.5 Towards a novel LCM architecture

The aim of this section is to propose a new neural architecture for temporal causal discovery, as a successor to the Informer-based LCM model introduced earlier. Our scope is to develop an expressive model that better captures both temporal and inter-variable dependencies based on novel advances in the state-of-the-art of neural networks, while also integrating auxiliary signals such as lagged crosscorrelations described previously.

An overview of the proposed model is shown in Figure 4.4. Conceptually, the design follows four sequential stages, as in the Informer-based LCM: (i) *Input preprocessing and patch embeddings* where the multivariate time-series is normalized, divided into overlapping temporal patches, and projected into a latent embedding space enriched with positional and variable identity information, (ii) *Transformer Encoder with decoupled temporal and spatial attention*, using alternating self-attention over time and variables to jointly learn dynamic and cross-variable dependencies, (iii) *auxiliary inputs* where lagged crosscorrelation is concatenated alongside the learned representations to guide causal learning and (iv) *prediction head* serving as a feedforward projection using a SwiGLU [Shazeer, 2020] activation and a sigmoid output to yield the lagged causal adjacency tensor $\hat{\mathbb{A}} \in [0,1]^{V_{\max} \times V_{\max} \times \ell_{\max}}$ representing the probability of causal influence across variables and lags.

4.5.1 Input representations

As in the Informer-based LCM, the model receives an (observational) time-series sample $X \in \mathbb{R}^{B \times L \times V}$ where B denotes the batch size, L the sequence length (timesteps), and V the number of variables. X is min-max normalized and padded as detailed in Sections 4.3 and 4.3.3.

Instead of processing individual timesteps directly as before, the model employs a *patching mechanism* [Nie et al., 2023], transforming each variable's sequence into overlapping windows of length ρ and stride s:

$$L' = \left\lfloor \frac{L - \rho}{s} \right\rfloor + 1 \tag{4.12}$$

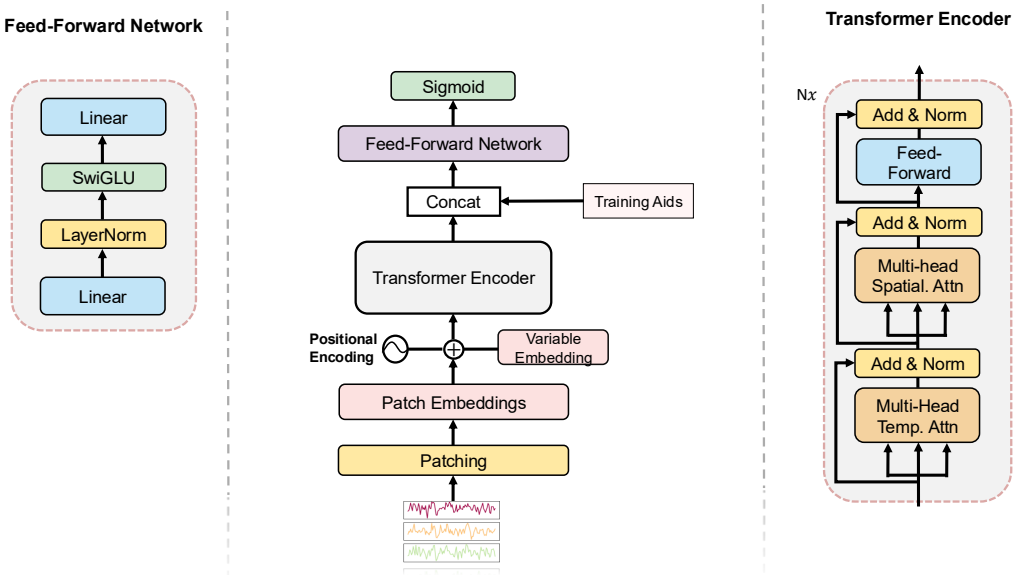


Figure 4.4: Overview of a novel architecture for temporal causal discovery. Each time-series is split into patches, which are then projected using linear embeddings *(Patch Embeddings)*. Standard sinusoidal positional encodings are added, together with variable embeddings (for carrying variable identity information) and passed through a stack of transformer encoder blocks, consisting of Temporal & Spatial multi-head attention. The resulted outputs are concatenated with crosscorrelation statistical measures *(training aids)* as a way to further increase the expressiveness of the model. They are finally processed through a feedforward head and a sigmoid activation to predict a lagged adjacency tensor $\hat{\mathbb{A}} \in \mathbb{R}^{V_{\max} \times V_{\max} \times \ell_{\max}}$ that represents the discovered lagged causal graph.

Each variable's sequence is split independently, yielding an intermediate tensor representation $[B, V_{\max}, L', \rho]$ which is afterwards reshaped into an embedding representation $[B \cdot V_{\max}, L', \rho]$. This operation reduces the effective sequence length while simultaneously allowing each patch to encapsulate short-term temporal dynamics, enabling efficient modeling of long-range dependencies.

Each patch $P \in \mathbb{R}^{\rho}$ is then projected to the model dimension d_{model} via a learnable linear transformation:

$$\hat{P} = W_P P + b_P, \quad W_P \in \mathbb{R}^{d_{\text{model}} \times \rho} \tag{4.13}$$

where W_P and b_P are learnable parameters. To preserve temporal order across patches, *sinusoidal positional encodings* [Vaswani et al., 2017] are added element-wise, as defined previously in Section 4.4.1:

$$\hat{P}^{\text{pos}} = \hat{P} + \text{PosEmb}(\ell) \tag{4.14}$$

where ℓ indexes the position of the patch within the sequence. Each variable is also assigned a learnable *variable embedding* $e_j \in \mathbb{R}^{d_{\text{model}}}$, shared across

patches, encoding variable identity information and facilitating inter-variable disambiguation. The final embedding sequence

$$X_{\text{emb}} \in \mathbb{R}^{B \times L' \times V_{\max} \times d_{\text{model}}} \tag{4.15}$$

is obtained after dropout regularization [Srivastava et al., 2014] and reshaping, and serves as the input to the Encoder block.

4.5.2 Encoder block

The encoder consists of (N) identical blocks. Each block alternates *temporal attention* (multi-head attention across patches within each variable) and *spatial attention* (multi-head attention across variables within a patch). This decoupling allows the model to explicitly disentangle intra-variable temporal dynamics from inter-variable dependencies. Specifically, for each variable i, temporal attention is computed as:

$$\text{TemporalAttn}^i = \text{MultiHeadAttn}(Q^i, K^i, V^i) \tag{4.16}$$

resulting in $X \in \mathbb{R}^{B \times P \times V_{\max} \times d_{\text{model}}}$. To enhance *permutation equivariance across features*, a permutation-equivariant transformation ϕ is introduced, followed by *mean pooling* across the variable dimension:

$$X_{\text{agg}} = \frac{1}{V_{\max}} \sum_{i=1}^{V_{\max}} \phi(X_i) \tag{4.17}$$

inspired by Deep Sets [Zaheer et al., 2017]. This global summary is added back to the temporally attended features and normalized:

$$X' = \text{LayerNorm}(X + X_{\text{agg}}) \tag{4.18}$$

The updated representations are then passed through *spatial attention*, *layer normalization* and a *feedforward network*:

$$\text{FF}(x) = W_2, \text{GeLU}(W_1 x + b_1) + b_2 \tag{4.19}$$

each wrapped with residual connections. It should be noted that since a feedforward network is applied, no permutation equivariance is preserved, although the transformation ϕ, together with mean pooling aim towards a permutation-equivariant model. Finally, temporal aggregation is performed by mean-pooling across patches:

$$\bar{X} = \frac{1}{L'} \sum_{p=1}^{L'} X_{:,p,:,:} \in \mathbb{R}^{B \times V_{\max} \times d_{\text{model}}} \tag{4.20}$$

Since our objective is not sequence generation, no decoder block is used (as in Section 4.4). The encoder's output logits directly represent causal feature interactions.

4.5.3 Auxiliary Inputs

$$\rho_{X^i,X^j}(\tau) = \frac{\mathrm{Cov}(X^i_{1:T-\tau}, X^j_{\tau+1:T})}{\sqrt{\mathrm{Var}(X^i_{1:T-\tau}), \mathrm{Var}(X^j_{\tau+1:T})}} \tag{4.21}$$

for $\tau = 1, \ldots, \ell_{\max}$. These are assembled into $\mathbb{C} \in \mathbb{R}^{B\times V_{\max}\times V_{\max}\times \ell_{\max}}$, flattened, and concatenated to the logits of the final encoder block

$$Z = \mathrm{Concat}(X_{\mathrm{enc}}, \mathbb{C}) \tag{4.22}$$

4.5.4 Feedforward Prediction Head

The concatenated representation is projected through a two-layer feedforward head with layer normalization and *SwiGLU activation* [Shazeer, 2020], followed by a sigmoid transformation:

$$\hat{\mathbb{A}} = \sigma\left(\mathrm{reshape}\left(W_2 \cdot \mathrm{SwiGLU}\big(\mathrm{LN}(W_1 Z + b_1)\big) + b_2 \in [0,1]^{B\times V_{\max}\times V_{\max}\times \ell_{\max}}\right)\right) \tag{4.23}$$

where the SwiGLU activation is defined as

$$\mathrm{SwiGLU}(x) = (xW_1) \odot \mathrm{SiLU}(xW_2), \tag{4.24}$$

where W_1 and W_2 are learned projection matrices, and $\odot$ denotes element-wise multiplication. The function $\mathrm{SiLU}(z) = z \cdot \sigma(z)$ denotes the *Sigmoid Linear Unit* (also known as the *Swish* activation), which combines linear gating with smooth nonlinear modulation, leading to improved gradient flow and expressivity compared to standard activations such as ReLU or GELU.

The output tensor $\hat{\mathbb{A}}$ is interpeted in the same manner as in the Informer-based model in Section 4.4.

4.6 Expanding the Scope of LCMs

This section explores two complementary extensions for causal foundation models, the incorporation of *interventional samples* and *prior knowledge*, in Subsection 4.6.1 and Subsection 4.6.2, respectively.

4.6.1 Towards Incorporating Interventional Samples

A central motivation in causal discovery is to move beyond mere association and towards identifiability of the underlying causal graph. While observational data allow the estimation of correlations and potential dependencies, they are insufficient to fully resolve causal directions in many cases due to the problem of statistical indistinguishability among Markov-equivalent structures. As an example in the i.i.d. setting, two variables X and Y may be highly correlated, yet share an unobserved common cause. If a CD algorithm assumes causal sufficiency (Section 1.1 & Table 4.3), then an edge from X to Y will be identified. To overcome this limitation, interventional data play a critical role: by intervening on X and observing Y, it is becomes possible to resolve the causal direction. Practical examples include controlled modifications in software configuration

parameters for AIOps, targeted marketing campaigns in economics or knockout experiments in biological networks. By explicitly introducing interventions in the model, the model is provided with more informative samples that aims to improve causal identifiability.

To accomplish this, we accompany each input batch with a *binary interventional matrix* $B_{K\times V} \in 0, 1$ such that

$$B_{ij} = \begin{cases} 1, & \text{if variable } V_j \text{ was directly intervened at timestep } i, \\ 0, & \text{if } V_j \text{ remained observational at timestep } i \end{cases} \tag{4.25}$$

where K denotes the number of interventional time-steps and V the number of variables. Thus, $B_{ij} = 1$ can be interpreted as a *masking operation*, indicating that the corresponding observation no longer reflects the natural dependency structure, as the variable's causal parents have been disrupted by an external intervention. As in the observational inputs, the interventional samples are padded as per Section 4.3, while the interventional matrix is padded with zeros where needed, to match the maximum number of time-steps $K_{\max} = L_{\max}$ and variables $V_{\max}$.

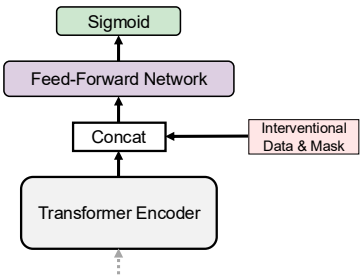


Figure 4.5: Integration of interventional samples in an LCM. Each input time series is accompanied by a binary interventional mask B, which encodes which variables were externally manipulated. The mask is concatenated with the encoder outputs, allowing the model to condition the following prediction layers based on the context of interventions.

During both training and inference, the (padded) binary interventional mask $B_{K_{\max}\times V_{\max}}$, together with interventional samples are concatenated with the encoded latent representation of the input time series (Figure 4.5). Formally, for a given encoder output $\mathbf{H}$, the augmented representation becomes

$$\mathbf{H}' = \text{Concat}(\mathbf{H}, B) \tag{4.26}$$

where B is broadcast or flattened to match the structure of $\mathbf{H}$. This augmentation allows the downstream attention and prediction layers to explicitly condition on which variables were intervened upon, thereby guiding the network to adjust its learned dependencies accordingly.

In essence, the model already learns, up to the final encoder block, to identify causal relationships. From there, the augmentation of the output is fed into

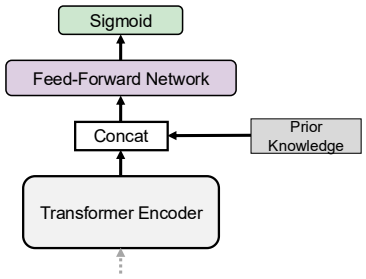


Figure 4.6: Integration of prior knowledge in an LCM. Prior tensors $\mathcal{P}$ and belief masks $\mathcal{B}$ are flattened and concatenated with the encoder output, aiming to provide soft inductive biases toward known or excluded edges and paths. This enables the model to learn causal structures that are consistent with provided knowledge while remaining data-adaptive.

the feedforward prediction head, which aims to further refine the prediction of the true causal directions by differentiating between observational and interventional examples. Over many such interventional samples, the model learn to internalize which dependencies are consistent under both observational and interventional conditions, yielding more accurate and identifiable causal graphs. The combination of interventional and observational data thus aims to extend the scope of neural foundation models beyond observational data alone.

4.6.2 Towards Incorporating Prior Knowledge

In many real-world domains, partial knowledge of causal relationships is often available or can be constructed manually by experts. This information can come in the form of known causal edges or paths (i.e. ("A causes B")) or exclusions ("A does not cause B"), and may include varying degrees of confidence. Incorporating such domain knowledge into a neural causal discovery model can improve interpretability and robustness, especially under limited data or noisy observational settings. However, unlike constraint-based algorithms such as PCMCI [Runge, 2018] which can directly encode hard constraints, neural architectures exhibit intrinsic stochasticity and non-linearity that make strict enforcement of such constraints impractical.

To incorporate domain information into the model, prior knowledge is injected after the last encoder block (in a similar manner to correlation injection and handling of interventions), allowing the model to refine its learned representations to more plausible causal structures without constraining it from the beginning. Prior information is provided either in edges and paths (edges are a special case of paths with length 1). By path, we mean a sequence of edges in the lagged causal graph of the form $X^i_{t-2} \to X^j_{t-1} \to X^k_t$ where $t \in \mathbb{N}$. The assumption of causal stationarity also applies to paths, making them time-invariant.

Specifically, information about priors is represented by two tensors: (i) a *prior tensor* $\mathcal{P} \in \{0,1,2\}^{V_{\max} \times V_{\max} \times \ell_{\max}}$ and (ii) a *belief strength mask* $\mathcal{B} \in [0,1]^{V_{\max} \times V_{\max} \times \ell_{\max}}$, associating each entry of $\mathcal{P}$ with a strength of knowledge

for each edge/path. Both $\mathcal{P}$ and $\mathcal{B}$ share the same semantics as the adjacency tensor representation in Section 2.6. Information in $\mathcal{P}$ is encoded as follows:

$$\mathcal{P}_{ji\ell} = \begin{cases} 0, & \text{no prior knowledge available,} \\ 1, & \text{if edge/path } X^i_{t-\ell} \to X^j_t \text{ is known to exist - inclusion prior,} \\ 2, & \text{if edge/path } X^i_{t-\ell} \to X^j_t \text{ is known to not exist - exclusion prior} \end{cases} \tag{4.27}$$

where the above formulation possesses the same semantics as our the adjacency tensor representation of the causal graph and accounts for paths up to $\ell_{\max}$.

The prior information is flattened and concatenated with the output of the final encoder block, similarly to the case of interventions, as illustrated in Figure 4.6. As it should be clear so far to the cautious reader, these two auxiliary prior inputs are provided during both inference and training, following a methodology described in Section 5.5 of Chapter 5.

This page intentionally left blank

5

Training

"Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise."

– *John Tukey*

This chapter provides a detailed treatment of the training process of our LCMs. It serves as an extension of our high-level discussion in Section 2.5, but tailored specifically towards the training of our LCMs. We begin by outlining the basic components of our training process and the associated challenges, before moving on to the main objectives of our work.

5.1 Optimization Strategies & Optimizers

As briefly mentioned in Section 2.5, a neural network model attempts to solve an optimization problem. Broadly speaking, this concerns finding the best possible values of its trainable parameters, over all feasible ones. The optimization problem is typically formulated as a *minimization problem*, where the goal is to find the set of parameters that minimizes a loss function. For a regression task, this means minimizing the expectation of the loss function over the parameters θ of a trainable neural model f_θ, where the expectation is taken with respect to the underlying joint distribution $p_{X,Y}$ that our input-output training data is assumed to follow. Mathematically, this corresponds to

$$\theta^* = \text{argmin}_{\theta \in \Theta} \mathbb{E}_{(X,Y)\sim p_{X,Y}} \left[\mathcal{L}(f(x;\theta), Y)\right] \tag{5.1}$$

where x is the input (in our case, the time-series samples) and y the corresponding label (the ground truth lagged adjacency tensor from the ground truth TSCM), Θ the parameter space, θ the optimal parameter set and $\mathcal{L}$ the loss function. Once the expectation is computed, all variables become fixed except for the parameters θ of the model. In this sense, only those are left to determine the functions value, which corresponds to the minimization of our quantity of interest. As may be already evident, this problem is *intractable* and is solved by an algorithm that dictates how and when the parameters θ are updated over time during training, such that convergence as close as possible

Algorithm 8 AdamW Optimizer

Input: Parameters θ, learning rate η, $\beta_1, \beta_2 \in [0, 1)$, weight decay λ, $\epsilon > 0$
Output: Updated parameters θ

1: Initialize $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$
2: **for** each iteration **do**
3: $t \leftarrow t + 1$
4: Sample mini-batch and compute gradient $g_t \leftarrow \nabla_\theta \mathcal{L}(\theta)$
5: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
6: $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
7: $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
8: $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
9: $\theta \leftarrow \theta - \eta \cdot \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta \right)$
10: **end for**

to the true minimum is achieved. This role is assigned to the *optimizer* that is used, which is a rigid part of the training protocol.

There are many optimizers to choose from depending on the implemented architecture and nature of the problem. It may even be the case that only one carefully tuned optimizer can adequately train the weights of a specific network. Nonetheless, some optimizers have proven to work well in most circumstances. A treatment of the most common ones is provided in Appendix C.

For training our LCMs, we use the Adam optimizer [Loshchilov et al., 2017] which serves as a popular extension of Adam [Kingma, 2015]. AdamW was introduced to fix a subtle but important flaw in Adam regarding weight decay (commonly used for regularization). In classical Adam, weight decay is typically implemented by adding an L_2 penalty term to the loss function. However, this approach causes the penalty to interact with Adam's adaptive learning rate mechanism in unintuitive ways: instead of applying a simple shrinkage to the parameters, the effective weight decay is scaled differently for each parameter depending on its learning rate. This leads to inconsistent and sometimes poor generalization performance. AdamW resolves this by decoupling weight decay from the gradient update rule. In practice, this means that the weight decay term is applied directly to the parameter update rule (line 9 in Algorithm 8), independently of the adaptive gradient correction. The effect is a more consistent and theoretically justified form of regularization. Empirically, AdamW has been shown to improve generalization, especially in large-scale deep learning models such as Transformers and convolutional networks, and it is now the optimizer of choice in many modern architectures, such as in BERT [Devlin et al., 2019].

5.2 Loss Functions

Since a neural network must learn the mapping from inputs to desired outputs, a fundamental requirement is a mechanism to quantify how "close" or "far" the network's predictions are from the ground truth. The loss function serves precisely this role—its magnitude measures the discrepancy between the model's estimates and the target causal structure. The optimization objective is therefore to minimize this loss across all training examples.

5.2.1 Binary Cross-Entropy Loss

For our LCMs, each possible directed edge $V_{t-\ell}^i \to V_t^j$ in the lagged causal graph corresponds to a Bernoulli random variable encoding the presence or absence of a direct causal relationship. As elaborated in Section 2.6, predicting causal edges naturally takes the form of a binary classification task, where the model outputs a confidence score $\hat{\mathbb{A}}_{ji\ell} \in [0, 1]$ representing the estimated probability that a direct lagged edge $V_{t-\ell}^i \to V_t^j$ exists.

Let $a_i \in 0, 1$ denote the ground-truth label for the i-th edge and $\hat{a}_i \in [0, 1]$ the model's predicted probability. The likelihood of the observed labels given the predictions can be modeled as a product of independent Bernoulli variables:

$$p(\mathbf{a} \mid \hat{\mathbf{a}}) = \prod_{i=1}^{N} \hat{a}_i^{a_i} (1 - \hat{a}_i)^{1-a_i} \tag{5.2}$$

where $N = V_{\max}^2 \times \ell_{\max}$ is the total number of potential lagged edges. Maximizing this likelihood is equivalent to minimizing the *negative log-likelihood (NLL)*:

$$\mathcal{L}_{\text{BCE}}(\mathbf{a}, \hat{\mathbf{a}}) = -\frac{1}{N} \sum_{i=1}^{N} [a_i \log(\hat{a}_i) + (1 - a_i) \log(1 - \hat{a}_i)] \tag{5.3}$$

This defines the *Binary Cross-Entropy (BCE)* loss, a standard choice for supervised binary prediction tasks. It penalizes incorrect predictions more heavily when the model is confident but wrong, owing to the logarithmic growth of the loss as $\hat{a}_i \to 0$ for positive labels ($a_i = 1$) or $\hat{a}_i \to 1$ for negative labels ($a_i = 0$).

The model produces logits $z_i \in \mathbb{R}$, which are converted to probabilities via the sigmoid function $\hat{a}_i = \sigma(z_i) = (1 + e^{-z_i})^{-1}$. Substituting this into Equation 5.3 yields the BCE loss in terms of logits:

$$\mathcal{L}_{\text{BCEWithLogits}}(\mathbf{a}, \mathbf{z}) = -\frac{1}{N} \sum_{i=1}^{N} [a_i \log \sigma(z_i) + (1 - a_i) \log(1 - \sigma(z_i))] \tag{5.4}$$

For numerical stability, training employs the fused `BCEWithLogitsLoss` formulation, which combines the sigmoid activation and BCE computation into a single operation to prevent overflow and ensure stable gradients. Differentiating with respect to z_i gives:

$$\frac{\partial \mathcal{L}_{\text{BCE}}}{\partial z_i} = \sigma(z_i) - a_i = \hat{a}_i - a_i \tag{5.5}$$

showing that the gradient corresponds directly to the difference between predicted and true probabilities. Denoting $\hat{\mathbb{A}}$ and $\mathbb{A}$ as the predicted and true lagged adjacency tensors, respectively, the BCE loss over a mini-batch is computed as:

$$\mathcal{L}_{\text{BCE}}(\mathbb{A}, \hat{\mathbb{A}}) = -\frac{1}{V_{\max}^2 \cdot \ell_{\max}} \sum_{i,j=1}^{V_{\max}} \sum_{\ell=1}^{\ell_{\max}} \left[\mathbb{A}_{ji}^{(\ell)} \log \hat{\mathbb{A}}_{ji\ell} + (1 - \mathbb{A}_{ji\ell}) \log(1 - \hat{\mathbb{A}}_{ji\ell}) \right] \tag{5.6}$$

5.2.2 Correlation Regularization

In addition to the primary BCE loss, we introduce a correlation-based regularization term designed to guide the model toward statistically plausible edge predictions. The intuition is to incorporate observable signal-level dependencies as auxiliary information, a form of *training aid*, to encourage the model to assign high causal confidence only where empirical evidence supports it.

This *Correlation Regularization* (CR) term penalizes predicted edges between variable pairs whose empirical lagged cross-correlation is low. Motivated by the formulation introduced by Stein et al. [2024], we define:

$$\mathcal{L}_{\text{CR}}(\hat{\mathbb{A}}, X) = \frac{1}{V_{\max}^2 \ell_{\max}} \sum_{i,j=1}^{V_{\max}} \sum_{\ell=1}^{\ell_{\max}} \left(\hat{\mathbb{A}}_{ji\ell} - \tilde{\text{CC}}(X_j, X_i, \ell)\right)^2 \left|\tilde{\text{CC}}(X_j, X_i, \ell)\right|^{3/2} \tag{5.7}$$

where $\tilde{\text{CC}}(X_j, X_i, \ell)$ denotes the min-max normalized empirical lagged cross-correlation between variables X_j and X_i at lag ℓ, computed over the observed time-series segment, and $\epsilon = 10^{-6}$ during normalization ensures numerical stability. The exponent $3/2$ downweights the penalty for near-zero correlations while preserving smooth gradients for moderately and strongly correlated pairs. In implementation, both the ground truth $\mathbb{A}$ and the predicted tensor $\hat{\mathbb{A}}$ are flattened to one-dimensional tensors of length $V_{\max}^2 \times \ell_{\max}$ before computing the BCE loss. The total loss used for training LCMs thus becomes:

where $\text{CC}(X_j, X_i, \ell)$ denotes the empirical lagged cross-correlation between variables X_j and X_i at lag ℓ, computed over the observed time-series segment, and $\epsilon = 10^{-6}$ ensures numerical stability. The exponent $3/2$ accentuates the penalty for near-zero correlations while preserving smooth gradients for moderately correlated pairs. In implementation, both the ground truth $\mathbb{A}$ and the predicted tensor $\hat{\mathbb{A}}$ are flattened to one-dimensional tensors of length $V_{\max}^2 \times \ell_{\max}$ before computing the BCE loss. The total loss used for training LCMs thus becomes:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \lambda_{\text{CR}} \cdot \mathcal{L}_{\text{CR}}, \tag{5.8}$$

where λ_{CR} is a tunable hyperparameter controlling the strength of the correlation-based penalty. In practice, this term discourages spurious edge predictions and promotes generalization by anchoring causal inference to observable statistical structure. Extensive ablation experiments are performed in Chapter 6 to quantify its effect on model stability and performance.

5.3 Training Protocol and Data Splits

All models are trained on the curated datasets described in Chapter 3. Each dataset follows an 80/10/10% split into training, validation, and test subsets to ensure a fair evaluation and prevent data leakage. Mini-batches are randomly sampled from the training set and optimization is performed using the AdamW optimizer (Section 5.1) with default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ for a maximum of 100 epochs.

5.4 Training Challenges & Practical Remedies

This section covers techniques in the optimizer and training protocol that help overcome challenges encountered when training not only our LCMs but deep neural architectures in general. As models become larger and datasets more complex, effective training is impeded by issues such as limited computational resources, data quality and quantity constraints, optimization difficulties like vanishing gradients, training instability, and overfitting. Remedies including advanced optimizer designs, learning rate scheduling, gradient accumulation, regularization, and adaptive training strategies enable more efficient and stable learning. By understanding and addressing these challenges with appropriate methods, it becomes possible to train larger, more accurate models that generalize well to unseen data.

5.4.1 Weight Initialization

To ensure stable gradient propagation and efficient convergence, all linear and convolutional layers in the LCM are initialized using the *Kaiming–He initialization* scheme [He et al., 2015]. This approach adjusts the variance of the initial weights to maintain a consistent signal magnitude across layers and prevent vanishing or exploding activations.

Formally, for a layer with weight matrix $W \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$, each element is initialized as

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right), \tag{5.9}$$

where n_{in} denotes the number of input units to the layer. Bias terms are initialized to zero:

$$b_i = 0 \quad \forall i. \tag{5.10}$$

The initialization method also provided a consistent performance baseline across training and fine-tuning stages, reducing sensitivity to the learning rate and batch size.

5.4.2 Gradient Accumulation

As deep neural networks become larger on the number of parameters, an important issue is training under memory constraints. For the most part, training deep neural architectures require large amounts of memory resources for storing model parameters and their gradients, even for just fine-tuning of pre-trained large scale models. A significant question is thus how to allow efficient training when memory resources are limited. In literature, primarily two memory reduction methods have been introduced to tackle the aforementioned problem: *gradient accumulation* [Huang et al., 2019] and *gradient release* [Pudipeddi et al., 2020].

Gradient accumulation allows us to simulate training with a larger *effective batch size* by reducing the activation memory. As illustrated in Figure 5.1, instead of updating parameters after every mini-batch, we compute gradients for several small mini-batches (*microbatches*), accumulate them in memory and only

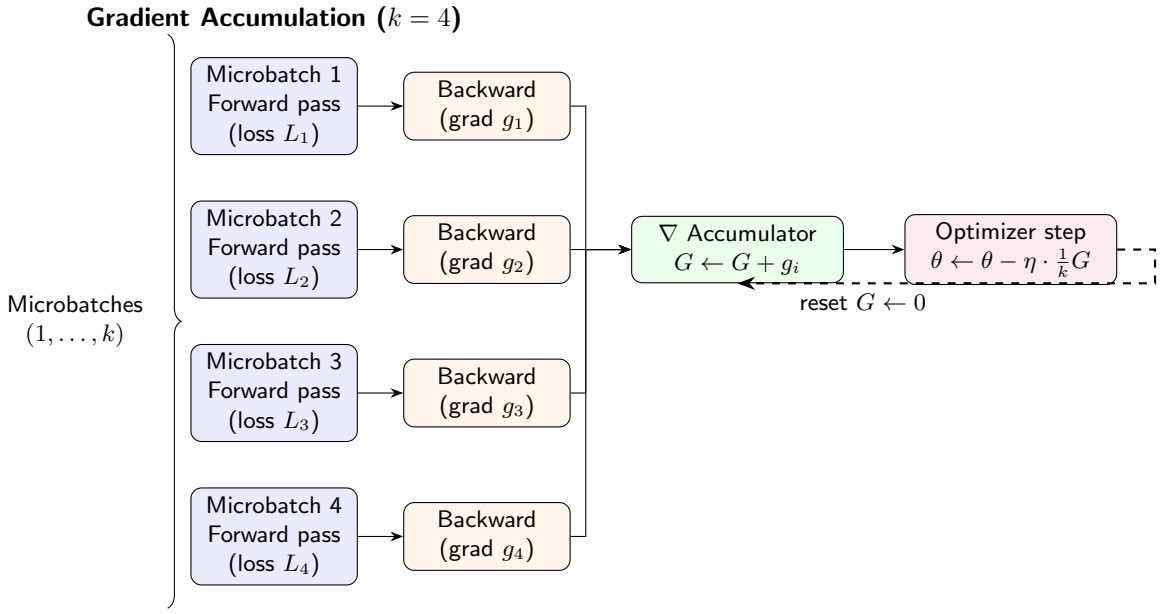


Figure 5.1: Illustration of the gradient accumulation method [Huang et al., 2019]. Instead of performing a parameter update after a single batch, gradients from multiple forward and backward passes of smaller batches (microbatches) are sequentially accumulated in a dedicated buffer (the *gradient accumulator*). After k microbatches, the aggregated gradient (summed or averaged) is applied in a single optimizer step to update the model parameters, followed by a reset of the accumulator. This approach allows training with effectively larger batch sizes while reducing memory usage, since only one microbatch needs to reside in memory at a time.

apply the optimizer update once after k steps (selecting $k = 4$ for large models). This means the effective batch size = (mini-batch size) $\times$ (accumulation steps). Gradient accumulation allows training of large-scale LCMs by simulating larger batch sizes when memory limitations prevent larger batch sizes. This is crucial since each batch may consist of hundreds of multivariate time-series samples with long temporal horizons. By accumulating gradients over $k = 4$ steps before applying an update, we approximate the stability of training with a larger batch (known as the *effective batch size*). Algorithm 9 describes the gradient accumulation method.

This approach has been experimentally shown to have no negative impact on training and convergence, although a small trade-off exists: In a sense, small batches can be noisy and thus lead to some training instability, or have a regularizing effect and lead to a better global minimum, while larger batches can be more stable but generalize less well.

In all training experiments, we utilize an *effective batch size of* $B = 64$. We thus select $k = 4$ when gradient accumulation is used for large-scale models ($> 10M$ parameters) to simulate an effective batch size of B.

Algorithm 9 Gradient Accumulation

Input: Accumulation steps k, optimizer, model f_θ, dataset $\mathcal{D}$
Output: Updated parameters θ

$step \leftarrow 0$
for each epoch **do**
  for microbatch $B \subseteq \mathcal{D}$ **do**
    Compute loss $L_B \leftarrow \mathcal{L}(f_\theta(B))$
    Compute gradients $g_B \leftarrow \nabla_\theta L_B$
    Accumulate gradients: $G \leftarrow G + g_B$
    $step \leftarrow step + 1$
    if $step \bmod k = 0$ **then**
      Update parameters: $\theta \leftarrow \theta - \eta \cdot G$
      Reset accumulator $G \leftarrow 0$
    end if
  end for
end for

5.4.3 Early Stopping

Early stopping (Algorithm 10) is a fundamental regularization technique when training of machine learning models. The main objective is to prevent overfitting (i.e. the model learns the training data's noise and idiosyncrasies rather than its underlying patterns) by monitoring the validation loss over epochs. The algorithm uses two key hyperparameters to achieve this goal: (i) the *minimum improvement threshold* δ, which is a small margin designed to prevent the algorithm from terminating prematurely due to minor, non-significant fluctuations in the validation loss and (ii) the *patience term* ρ which specifies the number of epochs to wait for a significant improvement in the validation loss before stopping. We adopt a patience value of $\rho = 20$ to accommodate the often slow and fluctuating convergence patterns observed in the training of deep neural networks, allowing the model to adapt through periods of stagnated improvement without premature termination.

5.4.4 Learning Rate Scheduler

A constant learning rate (independent of the epoch) can hinder effective training, as it can fail to adapt to the nature of the optimization landscape. In the initial stages of training, a high learning rate enables rapid traversal towards minima of the loss. However, as training progresses and the model approaches a local or global minimum, a large learning rate can lead to *overshooting* and instability, leading to oscillatory behavior or even divergence.

A learning rate scheduler (Algorithm 11) approach effectively addresses this problem by dynamically adjusting the learning rate η during training. The scheduler works by monitoring the validation loss and reducing the learning rate when there is no significant improvement for a specified number of epochs, known as *learning rate patience* p_{lr}. By decreasing the learning rate by a *reduction factor* γ, the optimizer takes smaller, more conservative steps, allowing for stable convergence towards the optimal causal structure. This prevents "over-updates" that could destabilize the discovery process and ensures the model can

Algorithm 10 Early Stopping for LCM Training

Input: Validation loss sequence $\{val\_loss^{(e)}\}$, patience p, minimum improvement δ
Output: Best epoch e^* or stop signal

$best\_val\_loss \leftarrow \infty$
$counter \leftarrow 0$
for epoch $e = 1, 2, \ldots$ **do**
  Compute $val\_loss^{(e)}$
  if $val\_loss^{(e)} < best\_val\_loss - \delta$ **then**
    $best\_val\_loss \leftarrow val\_loss^{(e)}$
    $e^* \leftarrow e$
    $counter \leftarrow 0$
  else
    $counter \leftarrow counter + 1$
  end if
  if $counter \geq p$ **then**
    stop training and return e^*
  end if
end for

effectively fine-tune its parameters. This approach aligns with well-established adaptive learning rate strategies in deep learning. Smith [2017] introduced cyclic and adaptive learning rate methods that help efficiently explore the loss landscape and lead to better convergence. Early work by Sutskever et al. [2013] also emphasized the importance of careful learning rate scheduling for stable optimization in deep networks.

In all our experiments, we apply a reduction factor $\gamma = 0.1$ after 10 epochs without improvement of the validation loss.

5.5 Training for Prior Knowledge

In many real-world domains, partial knowledge of causal relationships is often available or can be constructed manually by experts. This information can come in the form of known causal edges or paths (i.e. ("A causes B")) or exclusions ("A does not cause B"), and may include varying degrees of confidence. Incorporating such domain knowledge into a neural causal discovery model can improve interpretability and robustness, especially under limited data or noisy observational settings. However, unlike constraint-based algorithms such as PCMCI [Runge, 2018] which can directly encode hard constraints, neural architectures exhibit intrinsic stochasticity and non-linearity that make strict enforcement of such constraints impractical.

To address this challenge, we introduce a *soft prior-knowledge regularization* mechanism that complements the supervised edge classification task with an auxiliary stochastic objective. The objective is to gently bias the learning process toward plausible causal structures without rigidly constraining the optimization trajectory. Prior knowledge is injected both at the input level and in the loss function, allowing the model to be gently guided toward known or plausible causal structures without rigidly constraining learning.

Algorithm 11 Learning Rate Scheduler for LCM Training

Input: Initial learning rate η, reduction factor $\gamma < 1$, patience p_{lr}, min_lr
Output: Adaptively updated η

$best\_val\_loss \leftarrow \infty$
$counter \leftarrow 0$
for epoch $e = 1, 2, \ldots$ **do**
  Compute $val\_loss^{(e)}$
  if $val\_loss^{(e)} < best\_val\_loss$ **then**
    $best\_val\_loss \leftarrow val\_loss^{(e)}$
    $counter \leftarrow 0$
  else
    $counter \leftarrow counter + 1$
  end if
  if $counter \geq p_{lr}$ **then**
    $\eta \leftarrow \max(\gamma \cdot \eta, min\_lr)$
    $counter \leftarrow 0$
  end if
end for

5.5.1 Random Prior Knowledge Sampling

Prior knowledge is expressed either as edges (corresponding to paths of length one) or as paths of length $L \leq \ell_{max}$ in the lagged causal graph $\mathbb{A} \in \{0,1\}^{V_{\max} \times V_{\max} \times \ell_{\max}}$ as elaborated in Subsection 4.6.2. The ground-truth lagged adjacency tensor is first binarized, such that $\mathbb{A}_{ji\ell} = 1$ if a causal edge $X^i_{t-\ell} \to X^j_t$ exists, i.e. $\mathbb{A}_{ji\ell} > 0$ and zero otherwise. From this binary tensor, all directed paths of length up to $\ell_{\max}$ are extracted via a *depth-first search (DFS)* procedure, implemented in the FINDPATHS algorithm (Algorithm 12). Essentially, this unrolls the time-lagged causal graph forward in time, enumerating all (acyclic) causal paths of the form $\pi = \{(u_t, u_{t+1}, \ell_t)\}_{t=1}^{m}$, $m \leq \ell_{\max}$ so that

$$\prod_{t=1}^{m} \mathbf{1}[\mathbb{A}_{u_{t+1}, u_t, \ell_t} > 0] > 0 \tag{5.11}$$

Each valid path π is represented by its ordered node sequence, lag indices and its terminal edge (i, j, ℓ). These are then used to construct a *prior tensor* $\mathcal{P} \in \{0,1,2\}^{V_{\max} \times V_{\max} \times \ell_{\max}}$, encoding prior knowledge as:

$$\mathcal{P}_{ji\ell} = \begin{cases} 0, & \text{no prior knowledge available,} \\ 1, & \text{if edge/path } X^i_{t-\ell} \to X^j_t \text{ is known to exist,} \\ 2, & \text{if edge/path } X^i_{t-\ell} \to X^j_t \text{ is known to not exist} \end{cases} \tag{5.12}$$

5.5.2 Prior-Weighted Binary Cross Entropy

During training, both the ground-truth lagged graph $\mathbb{A}$ and the model's prediction $\hat{\mathbb{A}}$ are utilized to infer candidate paths, both for sampling prior knowledge from the ground truth as well as for comparing against predicted paths from

Algorithm 12 Path Extraction in Temporal Causal Graph (FINDPATHS)

Input: Lagged adjacency tensor $\mathcal{Y} \in \mathbb{R}^{V \times V \times \ell_{\max}}$, maximum path length L
Output: Set of valid paths $\mathcal{P}$ (each with `nodes`, `lags`, and `last_edge`)

Initialize edge map `edges` $\leftarrow \emptyset$
for each (i, j, k) with $\mathcal{Y}[j, i, k] > 0$ **do**
  $\ell \leftarrow \ell_{\max} - k$
  Append (j, ℓ) to `edges`$[i]$
end for
`paths` $\leftarrow \emptyset$
for $s = 1$ to V **do**
  `stack` $\leftarrow [([s], [], 0, 1.0)]$ {Each element: (nodes, lags, depth, strength)}
  while `stack` not empty **do**
    Pop (`nodes`, `lags`, `depth`, `strength`)
    $u \leftarrow$ `nodes`$[-1]$
    if $u \notin$ `edges` **then**
      continue
    end if
    for each $(v, \ell) \in$ `edges`$[u]$ **do**
      if $v \in$ `nodes` **then**
        continue {Avoid cycles}
      end if
      `new_nodes` $\leftarrow$ `nodes` $+ [v]$
      `new_lags` $\leftarrow$ `lags` $+ [\ell]$
      `new_strength` $\leftarrow$ `strength` $\times \mathbf{1}[\mathcal{Y}[v, u, \ell_{\max} - \ell] > 0]$
      if |`new_nodes`| ≥ 2 and `new_strength` > 0 **then**
        Append {`nodes` = `new_nodes`, `lags` = `new_lags`, `last_edge` = (u, v, ℓ)} to `paths`
      end if
      if `depth` $+ 1 < L$ **then**
        Push (`new_nodes`, `new_lags`, `depth` $+ 1$, `new_strength`) onto `stack`
      end if
    end for
  end while
end for
return `paths`

the model's prediction. These tensors are flattened across variables and lags to form two aligned tensors. A *prior-weighted binary cross-entropy loss* is then defined over the subset of indices where prior knowledge is available, i.e., where $\mathcal{P}_{ji\ell} \neq 0$, since absence of knowlege is encoded by zeros, including these terms in the loss would prove misleading. To handle the definition and properties of binary cross-entropy (as elaborated in Subsection 5.2.1), a remapped *target prior distribution* must be defined as $p_{\text{target}}(j, i, \ell) = 1$ if $\mathcal{P}_{ji\ell} = 1$ and $p_{\text{target}}(j, i, \ell) = 0$ if $\mathcal{P}_{ji\ell} = 2$. The total prior-weighted loss is then composed of two complementary parts, one for inclusion priors and one for exclusion priors:

$$\mathcal{L}_{\text{inc}} = \sum_{(j,i,\ell):\mathcal{P}_{ji\ell}=1} b_{ji\ell} \odot \text{BCE}(\hat{\mathbb{P}}_{ji\ell}, 1) \tag{5.13}$$

$$\mathcal{L}_{\text{exc}} = \sum_{(j,i,\ell):\mathcal{P}_{ji\ell}=2} b_{ji\ell} \odot \text{BCE}(\hat{\mathbb{P}}_{ji\ell}, 0) \tag{5.14}$$

where $\hat{\mathbb{P}}_{ji\ell}$ denotes the corresponding predicted paths based on the output $\hat{\mathbb{A}}$ of an LCM and $b_{ji\ell}$ are the corresponding belief strength weights of the belief strength tensor $\mathcal{B}$. Essentially, a loss between the predicted and true prior path distributions is computed for each valid prior entry. Weighted by the number of valid prior entries N_{priors}, the prior regularization term becomes

$$\mathcal{L}_{\text{prior}} = \frac{\lambda_{\text{prior}}}{N_{\text{priors}}} (\mathcal{L}_{\text{inc}} + \mathcal{L}_{\text{exc}}) \tag{5.15}$$

The term aims to softly enforce causal edges/path inclusions and exclusions, scaled by confidence weights. Consequently, the final loss function of a model trained with prior knowledge combines the base supervised objective, the correlation regularization, and the prior knowledge term, where the last two are scaled by regularization weights λ_{CR} and λ_{prior} respectively:

$$\mathcal{L}_{\text{LCM}} = \mathcal{L}_{\text{BCE}} + \lambda_{\text{CR}}\mathcal{L}_{\text{corr}} + \lambda_{\text{prior}}\mathcal{L}_{\text{prior}} \tag{5.16}$$

In summary, this formulation allows prior information to be used probabilistically while training rather than deterministically. Inclusion priors softly encourage the model to predict higher confidence for known causal connections, weighted by the strength of the prior. Exclusion priors gently penalize unlikely edges, while entries with no prior knowledge are ignored.

5.5.3 Staged Curriculum Learning

Curriculum learning [Bengio et al., 2009] represents a training paradigm inspired by the way humans learn progressively: starting from simple concepts before tackling more complex ones. Within the ML domain, it has been formalized as a strategy for optimizing non-convex functions by structuring the training process according to sample difficulty. Instead of exposing a model to the full complexity of a dataset from the initial training stages, curriculum learning advocates a staged exposure: the model is first trained on easier, cleaner, or more reliable examples, and only later is it challenged with noisier or harder samples. This gradual shift has been shown to improve optimization stability, convergence to more favorable local minima, and generalization performance across domains such as computer vision, natural language processing, and reinforcement learning.

This effectiveness is often attributed to its ability to allocate computational effort more efficiently: during the early stages, the model avoids being misled by noisy or ambiguous data, allowing it to establish strong inductive biases; later, the controlled introduction of harder examples prevents overfitting to simple patterns and pushes the model toward robustness. From an optimization standpoint, curriculum learning can be interpreted as a form of continuation method that smooths the loss landscape by deforming the training distribution over time.

An important open question regarding both neural-based approaches and foundation models for causal discovery, is an *efficient training regime to incorporate prior knowledge*. Motivated by these insights, we extend the curriculum

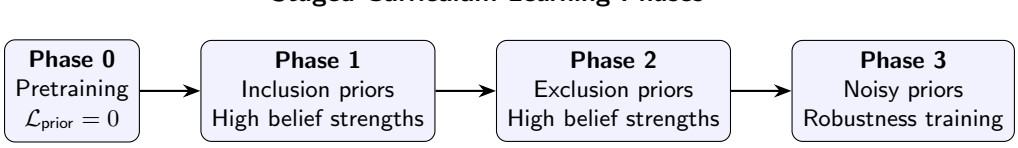


Figure 5.2: Overview of the proposed staged curriculum learning procedure. Training begins in Phase 0 with no prior knowledge, followed by gradually introducing true priors (Phase 1), exclusion priors (Phase 2), and noisy priors (Phase 3). Each stage builds on the model trained in the previous one, under a fine-tuning process, ensuring a smooth transition from data-only, learning to robust prior-informed causal discovery.

learning principle to the incorporation of prior knowledge in causal discovery. In particular, we design a staged curriculum that does not merely vary sample difficulty, but instead varies the quality, type, and reliability of prior knowledge provided during training. This aims to provide a principled way to integrate domain knowledge without overwhelming the model early on, while still enforcing robustness to noisy or even misleading priors at later stages.

Our curriculum learning strategy is organized into four distinct training phases. Each phase is designed to progressively refine the model's ability to leverage prior knowledge under varying belief strengths. For each Phase i, training begins from the pre-trained weights of Phase $i-1$ (except for Phase 0, where the model is trained from scratch) and is subsequently *fine-tuned*[1] to incorporate prior information. An overview of this staged process is shown in Figure 5.2.

Phase 0 - Pretraining (Data-Only Learning). The first stage, which we denote as Phase 0, establishes the foundation of the curriculum. At this point, the model is trained exclusively on the observational and interventional data without any form of prior knowledge. All prior-related inputs, such as prior tensors and belief strengths, are initialized and set to zero, resulting in the prior knowledge loss term vanishing, i.e., $\mathcal{L}_{\text{prior}} = 0$. This ensures that the model develops a baseline representation and learns to capture causal dependencies solely from the raw data. Importantly, although priors are not yet informative, the model architecture is already adapted to handle the additional input dimensions reserved for priors, thereby avoiding any architectural mismatch when subsequent phases introduce prior information. In this sense, Phase 0 provides a clean initialization point for the following stages.

Phase 1 - Incorporating Inclusion Priors. Once the baseline is established, we gradually expose the model to informative priors in Phase 1. Specifically, we sample a subset of ground-truth causal relations (edges or paths that

[1]In deep learning, *fine-tuning* refers to the process of reusing a pre-trained model and updating a subset of its parameters while keeping the remaining layers fixed. This approach allows adaptation to new data or objectives without overwriting previously learned representations. In our case, only the final feedforward block is unfrozen during fine-tuning.

exist) and inject them into the model as prior signals. These are provided with high belief strengths, drawn from the interval $\mathcal{B}_{ji\updownarrow}$ sampled from $\mathcal{U}(0.7, 1)$, reflecting strong confidence in their correctness. By combining the learned representations from Phase 0 with these informative signals, the model is encouraged to align its predictions with reliable causal structures. To balance the contributions of different objectives, we set the binary cross-entropy edge classification loss coefficient to 1.0 and the prior knowledge loss coefficient to 0.5. This weighting scheme ensures that while prior knowledge influences training, it does not dominate the optimization, allowing the model to maintain model flexibility.

Phase 2 - Introducing Exclusion Priors. After the model has successfully integrated inclusion priors, Phase 2 introduces exclusion priors, i.e., the absence of certain causal paths/edges. These are likewise injected with high belief strengths $\mathcal{U}(0.7, 1)$, serving as strong negative examples. From an optimization perspective, this phase provides an additional form of regularization by pruning spurious dependencies the model might otherwise overfit to. The same loss weighting strategy as in Phase 1 is maintained, striking a balance between edge classification accuracy and prior consistency. Taken together, Phases 1 and 2 guide the model through a stage of "structured learning," where it learns to incorporate both positive and negative causal evidence into its reasoning.

Phase 3 - Robustness to Noisy Priors. The final stage of the curriculum, Phase 3, aims to ensure robustness to imperfect prior knowledge. In practical applications, domain knowledge is often noisy, incomplete, or even contradictory. To prepare the model for such scenarios, we deliberately introduce noisy priors that include both wrong inclusions and false exclusions, drawn with varying belief strengths. Unlike earlier stages, where priors served primarily as reliable guidance, this phase trains the model to weigh prior information against the data more critically. To avoid model collapse, learning rate is halved during this stage. By exposing the model to uncertainty and potential misinformation, Phase 3 acts as a robustness training stage, preventing overreliance on priors and encouraging adaptive integration of knowledge and evidence. This is particularly important in large-scale causal discovery settings, where domain knowledge may be abundant but not uniformly reliable.

This page intentionally left blank

6

Results

"The great tragedy of science is the slaying of a beautiful hypothesis by an ugly fact."

– *Thomas Huxley*

This chapter contains the main experimental findings and results of LCMs. Section 6.1 elaborates on the experimental setup for assessing the performance of trained models. Section 6.2 is dedicated to the definition of the evaluation metrics for LCMs, while Section 6.3 contains the selection and description of established benchmarks. Section 6.4 concerns low to medium-scale results of ablation studies, while Section 6.5 covers results of large-scale LCMs. Trained models are evaluated on various dataset collections, both synthetic and simulated, as well as on in-distribution and out-of-distribution settings.

6.1 Evaluation Setup

Our evaluation setup can be briefly described as follows: We generate data collections as described in Chapter 3, train LCMs with varying capabilities as described in Chapter 4 and using a training protocol as in Chapter 5. We assess their predictive capabilities on various data collections, synthetic, semi-synthetic and simulated, both in-distribution and out-of-distribution, using performance metrics described in Section 6.2.

6.2 Measuring Causal Discovery Performance

We select standard metrics for predictive performance, as well as for causal graph distance. We evaluate the inferred lagged causal graph $\hat{G}$ based on the presence and absence of directed edges compared to the ground truth lagged causal graph $\mathcal{G}$. Recall that both the ground truth and inferred lagged causal graphs are padded both in the variable and lag dimension, so comparison is done on these padded graphs. Since causality is an asymmetric measure, and furthermore in the temporal case edges move forward in time, the directionality of an edge is taken into account. We evaluate the learnt edges in terms of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) [Sokolova and Lapalme, 2009]. Ultimately, our comparisons revolve around

the Area under the ROC curve (AUC) [Bradley, 1997] metric, which is the reported metric in the main text. Their known definitions are applied, but for comparing the edges on the ground truth and inferred lagged causal graphs (Table 6.1).

Table 6.1: Definitions of evaluation metrics for binary predictions on learned causal graphs.

Metric	**Definition**
TPR	Proportion of positives correctly identified.
FPR	Proportion of negatives incorrectly identified as positives.
TNR	Proportion of negatives correctly identified.
FNR	Proportion of positives incorrectly identified as negatives.
Precision	Proportion of positives correctly identified.
Recall	Proportion of positives correctly identified.
F1	Harmonic mean of precision and recall.
AUC	Area under the ROC curve (TPR vs. FPR) across different thresholds.

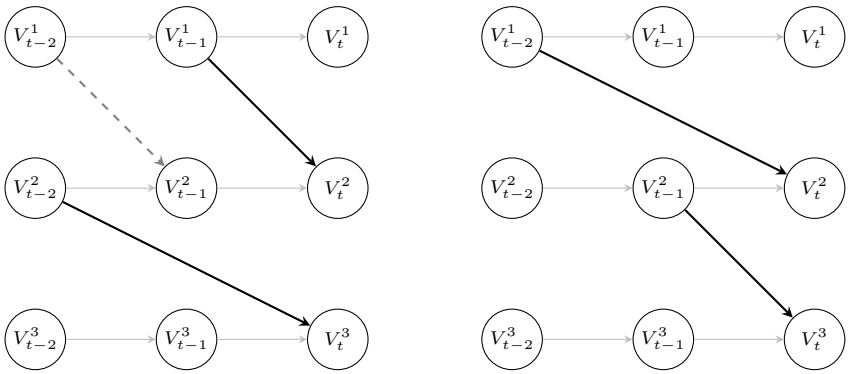
TPR, FPR, TNR, FNR. The *True Positive Rate (TPR)* (also known as *sensitivity*) measures the proportion of actual positives that are correctly identified by the model. $\text{TPR} = \frac{TP}{TP+FN}$. The *False Positive Rate (FPR)* measures the proportion of actual negatives that are incorrectly identified as positives, $\text{FPR} = \frac{FP}{FP+TN}$. The *True Negative Rate (TNR)* (also known as *Specificity*), measures the proportion of actual negatives that are correctly identified as negatives, $\text{TNR} = \frac{TN}{TN+FP}$. Finally, the *False Negative Rate (FNR)* measures the proportion of true edges that are incorrectly identified as non-existent, $\text{FNR} = \frac{FN}{FN+TP}$.

Precision, Recall, F1. We Additionally measure the *Precision* (also known as *Positive Predictive Value*) and *Recall* (also known as *Sensitivity*) metrics, which are defined as $\text{Precision} = \frac{TP}{TP+FP}$, $\text{Recall} = \frac{TP}{TP+FN}$ and $\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$.

AUC. We measure the AUC (Area under ROC Curve) [Bradley, 1997] using the area under the True Positive Rate and False Positive Rate at different prediction thresholds, based on the adjacency tensors of the ground truth and predicted DAGs. This is the same metric as the adjacency-based AUC introduced in Section 3.4.3, referred here simply as AUC. The predicted probabilities are obtained from the model and thresholded to $\tau = 0.05$, as well as the ground truth adjacencies (labels). Both time-lagged adjacency tensors are then flattened and concatenated along their batch dimension.

Figure 6.1 depicts an example of a metric comparison between the ground truth and a possibly inferred output graph.

Although LCMs output a soft adjacency tensor $\hat{\mathbb{A}}$ as discussed in Section 2.6, obtaining a binary causal graph from the soft predictions requires some sort of thresholding to the confidence scores. A thresholding operator is therefore defined, $\hat{\mathcal{G}} = 1\{\hat{\mathbb{A}} \geq \tau\}$, with τ either fixed (e.g., $\tau = 0.05$) or varied to compute ROC and AUC. Unless otherwise specified, reported results use $\tau = 0.05$. This



(a) Ground Truth (b) Inferred Example

Figure 6.1: Ground truth vs. inferred lagged causal graph with $\ell_{\max} = 2$ and 3 variables. Dashed edges encode causal stationarity, solid edges denote direct causal influence.

procedure separates the training objective (which operates directly on soft predictions) from the evaluation protocol (which relies on binarized graphs). For threshold-independent comparisons, we sweep τ to compute ROC and AUC.

6.3 Comparison with Existing Approaches

We compare the results of our framework with three other established works[1] for temporal causal discovery:

1. The Peter-Clark Momentary Conditional Independence (*PCMCI*) is a time-series causal discovery algorithm introduced by Runge [2018]. It is based on the constraint-based Peter-Clark (PC) algorithm by Spirtes et al. [2001], uncovering causal relationships using conditional independence tests of the form $X_i(t-\tau) \to X_j(t)$ for lag $\tau = 0, \ldots, \ell_{\max}$. The algorithm assumes the causal Markov condition, faithfulness, causal stationarity and causal sufficiency, while also accounting for contemporaneous ($\tau = 0$) causation. The process consists of two phases, (i) the PC phase where, starting from a fully connected time-lagged graph up to lag $\ell_{\max}$ iteratively prunes spurious relationships by testing conditional independences, given a conditioning set and (ii) the MCI phase where, using Momentary Conditional Independence, tests for causal relationships while conditioning on both lagged and contemporaneous variables. Specifically, for each remaining candidate edge $X^i_{t-\tau} \to X^j_t$, the Momentary Conditional Independence (MCI) test is performed, conditioning on both past and contemporaneous parents of X^j_t and parents of $X^i_{t-\tau}$. This reduces false positives caused by autocorrelation and contemporaneous confounding, yielding higher precision in estimated causal graphs. In our implementations we use the official implementation of PCMCI, available in the `Tigramite` Python package. We select $\alpha = 0.05$ as the threshold of the

[1] For an overview on causal discovery algorithms, we refer the reader to the review paper by Niu et al. [2024].

adjacency matrix's p-values exclude contemporaneous edges, $\ell_{\max} = 3$. For independence tests we utilize partial correlation, which is estimated through a linear Ordinary Least Squares (OLS) regression and testing for non-zero linear Pearson correlation on the residuals and GPDC, which is a conditional independence test based on Gaussian processes and distance correlation.

2. *DYNOTEARS* [Pamfil et al., 2020] is a time-series causal discovery algorithm, viewed as an extension of the NOTEARS [Zheng et al., 2018] method, which introduced causal discovery as a mathematical optimization problem with smooth acyclicity constraint. By leveraging the trace exponential function to enforce acyclicity and promoting sparsity using the ℓ_1 norm, this optimization program is solved using the augmented Lagrangian method, translating it into a series of unconstrained problems, in a similar setup to NOTEARS. Assuming M independent realizations of a *stationary* time-series $\{x_{m,t}\}_{t=0}^{T}$ for $\mathbf{x}_{m,t} \in \mathbb{R}^d$, the authors model the data using a Linear Structural Vector Autoregression (SVAR) model. The model is expressed as

$$\mathbf{x}_{m,t} = \mathbf{x}_{m,t}W + \sum_{i=1}^{p} \mathbf{x}_{m,t-i}A_i + \mathbf{z}_{m,t}, \tag{6.1}$$

where W represents contemporaneous (*intra-slice*) dependencies, A_i lagged (*inter-slice*) dependencies, and $\mathbf{z}_{m,t}$ independent error terms. In matrix form this translates to,

$$\mathbf{X} = \mathbf{X}W + \mathbf{Y}A + \mathbf{Z}, \tag{6.2}$$

where $\mathbf{X}$ contains the observations, $\mathbf{Y}$ the time-lagged versions of $\mathbf{X}$ and A is the concatenated matrix of lagged weights. The parameters of DYNOTEARS, apart from $\ell_{\max}$, are the regularization constants $\lambda_{\mathbf{W}}$, $\lambda_{\mathbf{A}}$ that control the sparsity (as causal graphs are in general sparse), and the weight thresholds $\tau_{\mathbf{W}}$, $\tau_{\mathbf{A}}$ that reduce the numerical error when computing $h(\mathbf{W}) = \text{trace}(e^{\mathbf{W}\circ\mathbf{W}}) - d$. In our implementations, we select $\lambda_{\mathbf{W}} = \lambda_{\mathbf{A}} = 0.1$, thresholds $\tau_{\mathbf{W}} = \tau_{\mathbf{A}} = 0.05$, maximum number of iterations in the augmented Lagrangian method $n = 100$ and $\ell_{\max} = 1$. Since DYNOTEARS' formulation is based on intra-slice and inter-slice dependencies for modeling the influence between variables in a contemporaneous and time-lagged setting respectively, we drop the contemporaneous predictions as we assume no contemporaneous effects in our experimental setup.

3. *VARLiNGAM* [Hyvärinen et al., 2010] extends the LiNGAM (Linear Non-Gaussian Acyclic Model) to time-series data by integrating it with vector autoregressive (VAR) models and infers both lagged and contemporaneous causal relations, whereas the classic VAR only analyzes lagged causal relationships. Specifically, it estimates a Structural Vector Autoregression (SVAR) model of the form

$$\mathbf{x}(t) = \sum_{\tau=0}^{k} B_\tau \mathbf{x}(t-\tau) + \mathbf{e}(t), \tag{6.3}$$

where B_τ are $n \times n$ coefficient matrices for lag τ, and $\mathbf{e}(t)$ is a vector of mutually independent, non-Gaussian disturbances. The matrix B_0 encodes instantaneous (contemporaneous) causal relations, while B_τ, $\tau > 0$, encode lagged causal relations.

The key assumptions are: (i) linearity of the causal relations, (ii) non-Gaussianity and independence of disturbances, (iii) acyclicity of contemporaneous relations, and (iv) absence of latent confounders. Unlike Gaussian SVARs, where identifiability requires strong priors, VARLiNGAM leverages Independent Component Analysis (ICA) to achieve identifiability purely from statistical properties of the data. Estimation proceeds in two stages: first, autoregressive coefficients M_τ are estimated via least squares; second, the residuals are subjected to LiNGAM analysis to recover instantaneous effects B_0. The full causal matrices are then reconstructed as

$$\hat{B}_\tau = (I - \hat{B}_0)\hat{M}_\tau, \quad \tau > 0 \tag{6.4}$$

This yields a consistent estimator of both instantaneous and lagged effects. The authors show that neglecting B_0 can severely bias lagged estimates, underscoring the importance of modeling contemporaneous effects. In our experiments, we implement VARLiNGAM using the publicly available code and set $\ell_{\max} = 3$, assuming first-order dependencies. Evaluation is restricted to lagged effects, discarding contemporaneous ones, in accordance with our experimental setup that excludes instantaneous causation.

In addition to the above baselines, we also compare against the pre-trained Transformer model by Stein et al. [2024] on our 5-dimensional evaluation datasets.

For all baselines, the official implementations are used or adapted, where needed, and the inferred graphs are transformed into a lagged adjacency form to ensure a fair comparison. The lag hyperparameter $\ell_{\max}$ is set precisely to that of our LCMs, i.e. $\ell_{\max} = 3$. It should further be noted that not all methods output a lagged adjacency matrix of probabilities for edge existence (confidence scores) like our LCMs. For constraint-based methods like PCMCI, we use the inverse of the edge p-values as confidence scores, since the method outputs the p-values of the conditional independence tests for each lagged edge. For models that output an adjacency matrix of causal effect coefficients (like VARLiNGAM and DYNOTEARS), results are not directly comparable to our LCMs that output soft adjacency probabilities and computing AUCs is not directly applicable. For a fair comparison, we follow a bootstrap-based procedure to estimate the probability of each causal edge, as shown in the official documentation[2]. Specifically, we run both VARLiNGAM and DYNOTEARS $n = 10$ times with resampled datasets and compute the proportion of times each edge is discovered, resulting in a soft adjacency matrix of edge probabilities. Although

[2] `https://lingam.readthedocs.io/en/latest/tutorial/var.html`

Algorithm 13 Bootstrap-based Estimation of Edge Probabilities

Require: Time-series dataset $\mathcal{D}$, maximum lag $\ell_{\max}$, number of bootstraps n, causal discovery algorithm configuration $\mathbf{B}_{\text{CD}}$
Ensure: Soft adjacency matrix $\mathbb{A} \in [0,1]^{V \times V \times \ell_{\max}}$

Initialize accumulator $\mathbf{S} \leftarrow \not{0}^{V \times V \times \ell_{\max}}$
for $b = 1$ to n **do**
  {Bootstrap Sampling Phase}
  Draw bootstrap sample $\mathcal{D}^{(b)}$ from $\mathcal{D}$ by resampling with replacement
  {Causal Discovery Phase}
  $\mathbf{A}^{(b)} \leftarrow \text{causalAlg}(\mathcal{D}^{(b)}, \mathbf{B}_{\text{CD}})$
  {Edge Confidence Update}
  $\mathbf{S} \leftarrow \mathbf{S} + f(\mathbf{A}^{(b)})$
  {f extracts edge indicators or confidence scores}
end for
{Normalization Phase}
$\mathbf{A} \leftarrow \mathbf{S}/n$
return $\mathbf{A}$

computationally expensive, this approach allows for The pseudocode for this approach is shown in Algorithm 13.

To assess whether the differences in AUC scores between not only LCMs but benchmark methods as well are statistically significant, we employ the *Wilcoxon signed-rank test*, a non-parametric paired test suitable for non-normally distributed data such as AUC scores. To control the family-wise error rate from multiple comparisons, we apply the *Bonferroni correction* $\alpha_{\text{corrected}} = \frac{\alpha}{k}$ where k is the number of pairwise comparisons. Additionally, each dataset instance is evaluated 10 times with different random seeds to obtain standard errors and confidence intervals. In the main text, the mean AUC is reported (as defined in Section 6.2) with standard errors. A complete list of experimental results is provided in the Appendix.

6.4 Ablation Studies

6.4.1 Ablation on Training Aid Techniques

In Chapters 4 and 5, two approaches for enhancing the performance of LCMs were presented, as introduced by Stein et al. [2024]: *Correlation Injection (CI)* (Subsection 4.4.3) and *Correlation Regularization (CR)* (Subsection 5.2.2). The impact of both techniques is now quantitatively and thoroughly evaluated in terms of causal discovery performance, a gap that was previously unfilled. We report results for medium-sized LCMs ($\approx$ 1M parameters) trained on the `S_Joint` collection (Subsection 3.6.2, Table 3.9) for in-distribution testing.

A baseline LCM was trained without any training aids, a second variant with CI only, and several models combining CI and CR with a grid search over the space of weight coefficients $\lambda_{\text{CR}} \in \{0.25, 0.5, 0.75, 1.0\}$. The objective was two-fold: (i) to assess the relative contributions of CI and CR, and (ii) to identify a stable λ_{CR} value for subsequent large-scale experiments.

Table 6.2: Ablation of training aids on `S_Joint` (in-distribution). Statistical significance refers to improvement over the preceding variant, under a Bonferroni correction for multiple comparisons.

Model	AUC	Significant
LCM	0.830 ± .000	–
LCM + CI	0.845 ± .000	Yes
LCM + CI + CR, ($\lambda_{CR} = 0.25$)	0.870 ± .000	Yes
LCM + CI + CR, ($\lambda_{CR} = 0.5$)	0.876 ± .000	Yes
LCM + CI + CR, ($\lambda_{CR} = 0.75$)	0.875 ± .000	No (vs 1.0)
LCM + CI + CR, ($\lambda_{CR} = 1.0$)	0.875 ± .000	No (vs 0.75)

Table 6.2 illustrates that both CI and CR substantially enhance causal discovery accuracy. CI alone provides a statistically significant improvement over the baseline ($p < 10^{-7}$), while the addition of CR further increases AUC across all tested regularization strengths. The highest mean AUC (0.876) is achieved for $\lambda_{\text{CR}} \in [0.5, 0.75]$, with negligible variance.

A post-hoc significance analysis (Table E.2 in the Appendix) confirms no statistically significant difference between $\lambda_{\text{CR}} = 0.75$ and $\lambda_{\text{CR}} = 1.0$ ($p = 0.0192 > 0.01$, after correction), while all other pairwise comparisons remain significant. Consequently, $\lambda_{\text{CR}} = 0.75$ is adopted as the default configuration for subsequent large-scale experiments, balancing regularization strength and stability. Table E.1 in the Appendix lists the full hyperparameter configuration and extended results.

6.4.2 Optimal Mixture of Synthetic and Simulated Data

To evaluate the optimal balance between purely synthetic and physically simulated training instances, we trained medium-sized LCMs ($\approx$ 1M parameters) on varying mixtures of the two data sources. The synthetic datasets provide large-scale diversity and fundamental causal patterns, while simulated datasets (e.g., Kuramoto oscillators) introduce structured physical dynamics and noise realism. This experiment thus proves whether combining both data modalities enhances out-of-distribution (OOD) causal discovery, as findings from Das et al. [2024] illustrate, which also highlight the complementary role of synthetic and simulated data for generalization.

Models were trained on mixtures with synthetic-to-simulated ratios of 100/0, 80/20, 50/50, 20/80 and evaluated on the semi-synthetic Kuramoto_5 and Kuramoto benchmark collections. All runs used identical hyperparameters and incorporated both correlation injection and correlation regularization ($\lambda_{\text{CR}} = 0.75$).

Table 6.3 shows that training on purely synthetic data results in poor OOD performance (AUC < 0.65), reflecting overfitting to artificial data distributions. Adding a small proportion of simulated data (20%) yields a dramatic increase in causal accuracy across both Kuramoto test sets (+0.27-+0.31 AUC). Beyond this point, the effect plateaus, suggesting diminishing returns and a slight trade-off between generalization and over-regularization. The 80/20 mixture thus provides an optimal balance, achieving consistently high AUCs and statistical robustness across both benchmarks.

Table 6.3: Out-of-distribution causal discovery performance (AUC) of medium-sized LCMs trained on varying mixtures of synthetic and simulated data, evaluated on semi-synthetic Kuramoto benchmarks. Statistical significance refers to improvement over the preceding mixture, under a Bonferroni correction for multiple comparisons.

Mix (Synth / Sim)	**AUC** (Kuramoto$_5$)	**AUC** (Kuramoto)	**Significant**
100% / 0%	0.629 ± .000	0.652 ± .000	–
80% / 20%	**0.939** ± .000	**0.903** ± .000	Yes (vs. 100/0)
50% / 50%	0.925 ± .000	0.920 ± .000	No (vs. 80/20)
20% / 80%	0.872 ± .000	0.922 ± .000	No (vs. 50/50)

Table 6.4: Out-of-distribution causal discovery performance (AUC) of medium-sized LCMs trained on varying mixtures of synthetic and simulated data, evaluated on the `AirQualityMS` benchmark. Statistical significance refers to improvement over the preceding mixture, under a Bonferroni correction for multiple comparisons.

Mix (Synth / Sim)	**AUC** (AirQualityMS)	**Significant**
100% / 0%	0.886 ± .000	–
80% / 20%	**0.957** ± .000	Yes (vs. 100/0)
50% / 50%	0.961 ± .000	No (vs. 80/20)
20% / 80%	0.951 ± .000	No (vs. 50/50)

Additionally, Table 6.4 shows that for `AirQualityMS`, adding a small proportion of simulated data 20% substantially improves OOD performance compared to purely synthetic training instances. However, increasing the fraction of simulated data beyond 20% does not yield statistically significant gains, which is also consistent with the Kuramoto benchmarks. These results reinforce the complementary nature of synthetic and simulated data: *synthetic datasets enable efficient pretraining and coverage of structural variations, while simulated datasets enhance realism, stability, and transfer to real-world-like domains.* This hybrid strategy is adopted in all subsequent large-scale experiments. Additional results that support this finding are presented in the Appendix.

6.5 Performance of Large-Scale LCMs

This section evaluates large-scale LCMs, ranging from ∼2.5M to ∼24M parameters, on both in-distribution and out-of-distribution (OOD) causal discovery tasks. All models were trained with verified aiding techniques (CI and CR) and mixed synthetic-simulated datasets as described in Section 3.6. While shallow models (∼1M parameters) have already demonstrated strong generalization across higher-dimensional settings (up to 12 variables), this section investigates whether deeper models maintain or improve performance without degradation, a challenge previously observed by Stein et al. [2024].

6.5.1 In-distribution Performance

In-distribution generalization is evaluated on (i) the synthetic `S_Joint` benchmark (3-5 variables) and (ii) the mixed `Synth_230K_Sim_45K` holdout set of mixed synthetic-simulated data. The `S_Joint` set additionally includes comparisons against the pretrained CP model from Stein et al. [2024], which was trained on a similar small-scale synthetic distribution.

Table 6.5: In-distribution causal discovery performance (AUC) of large-scale LCMs and baselines.

Model	**S_Joint (Synthetic)**	**Synth_230K_Sim_45K (Holdout)**
LCM-`2.5M`	0.957 ± .000	0.799 ± .000
LCM-`9.4M`	0.962 ± .000	0.800 ± .000
LCM-`12.2M`	0.964 ± .000	0.800 ± .000
LCM-`24M`	0.936 ± .000	0.800 ± .000
CP (Stein et al.)	0.915 ± .000	–
PCMCI	0.672 ± .000	0.783 ± .000
DYNOTEARS	0.540 ± .000	0.562 ± .000
VARLiNGAM	0.801 ± .000	0.773 ± .000

Table 6.5 illustrates that large-scale LCMs substantially outperform all classical baselines and maintain stable performance across scales. Despite the simplicity of `S_Joint` (3-5 variables), the LCM-`12.2M` model achieves an AUC of 0.964, outperforming the CP model from Stein et al. [2024] which was explicitly trained on such data distributions, and at the same input dimensionality.

In the mixed `Synth_230K_Sim_45K` holdout test set, which combines synthetic and simulated data, LCMs achieve consistent AUC values around 0.80, confirming stable in-distribution generalization without any overfitting to training dynamics. This suggests that higher-capacity LCMs leverage their scale effectively when coupled with high-quality training data, as well as correlation-based injection & regularization.

6.5.2 Out-of-distribution / Zero-shot Performance

We next test zero-shot transfer on three distinct OOD domains: semi-synthetic Kuramoto networks, *f*MRI-derived datasets (*f*MRI_5), and the real-world AirQualityMS dataset, which is unseen during training and created using the ACT method (Section 3.4).

Table 6.6 demonstrates that large-scale LCMs achieve robust OOD transfer across diverse domains. While traditional temporal CD baselines (PCMCI, DYNOTEARS, VARLiNGAM) exhibit severe degradation, LCMs consistently exceed 0.90 AUC on all benchmarks. Notably, the LCM-`9.4M` and LCM-`12.2M` variants provide the most balanced trade-off between model scale and generalization, outperforming all baselines by wide margins.

Consequently, high-quality mixed training data, combined with correlation-based aiding techniques, enables LCMs to scale efficiently to higher dimensions without degradation. In both in-distribution and OOD regimes, the introduced

Table 6.6: Out-of-distribution (zero-shot) causal discovery performance (AUC) of large-scale LCMs and baselines across semi-synthetic and realistic benchmarks.

Model	Kuramoto	*f*MRI_5	AirQualityMS	Mean (OOD)
LCM-`2.5M`	0.896 ± .000	0.945 ± .000	0.955 ± .000	0.932
LCM-`9.4M`	0.926 ± .000	0.943 ± .001	0.914 ± .001	0.928
LCM-`12.2M`	0.902 ± .000	0.946 ± .000	0.914 ± .001	0.921
LCM-`24M`	0.913 ± .000	0.954 ± .000	0.813 ± .006	0.893
CP (Stein et al.)	–	0.775 ± .001	–	–
PCMCI	0.640 ± .000	0.739 ± .005	0.556 ± .000	0.645
DYNOTEARS	0.503 ± .000	0.522 ± .011	0.694 ± .000	0.573
VARLiNGAM	0.584 ± .000	0.687 ± .005	0.552 ± .000	0.608

models maintain or improve performance as capacity increases, demonstrating effective scaling behavior and robust generalization.

6.6 Running Times

This section provides a comparison of running times for various LCM variants. The fact that all pre-trained models operate on constant time complexity per sample is highlighted, in contrast to scaling issues present when fitting classic causal discovery algorithms, based on a one-model-per-dataset basis (Section 1.2).

Figure 6.2 shows the running times for the Synth_230K dataset, highlighting the fact that LCMs are faster than other baselines (e.g., PCMCI, DYNOTEARS), even for deeper architectures. Numerical results are provided in Table 6.7. To ensure a fair comparison, all models and baselines have been evaluated on a consumer-grade, 6-core, 12-thread CPU. Additional boxplots on running times are provided in the Appendix (Figures E.11 to E.14).

Table 6.7: Mean, minimum, and maximum elapsed time (in seconds) for trained LCMs in the `Synth_230K` dataset (in-distribution, synthetic, holdout). All trained models achieve superior running times compared to non-foundation model baselines.

Model	Mean ± Std.	Range (s)
LCM 2.5M	0.014 ± 0.001	(0.012, 0.047)
LCM 9.4M	0.027 ± 0.002	(0.023, 0.086)
LCM 12.2M	0.030 ± 0.003	(0.026, 0.124)
LCM 24M	0.041 ± 0.004	(0.036, 0.100)
PCMCI	0.749 ± 0.056	(0.596, 1.453)
DYNOTEARS	0.264 ± 0.274	(0.020, 4.532)
VARLINGAM	3.825 ± 0.327	(2.567, 5.032)

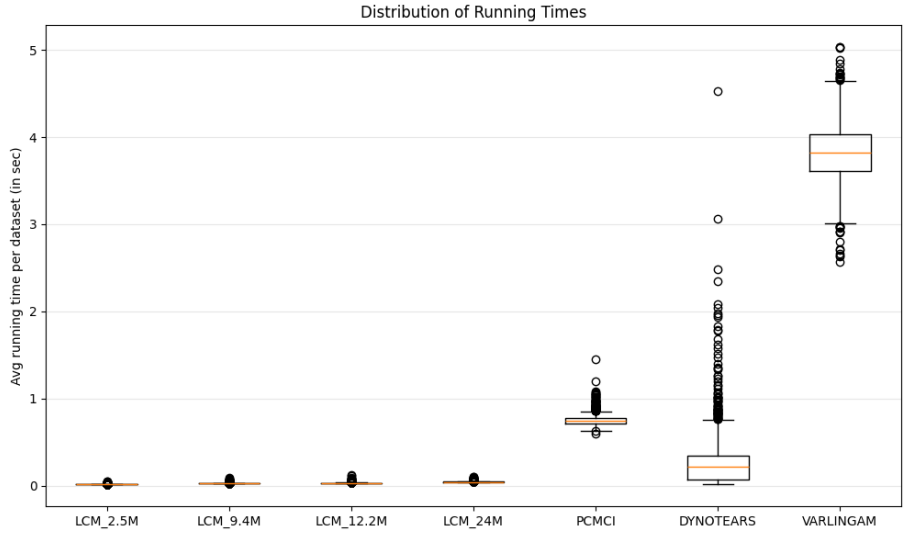


Figure 6.2: Running times on the `Synth_230K` dataset (in-distribution, synthetic, holdout): Mean, standard deviation, and range across model variants and baselines. All trained models achieve superior running times compared to non-foundation model baselines.

6.7 Preliminary Results on Interventional Data

To examine the direct impact of interventional samples on causal identifiability, we trained a medium-sized LCM (∼2M parameters) as in Subsection 4.6.1, but *without* correlation injection (CI) or correlation regularization (CR) terms. This design isolates the effect of interventional information itself, avoiding auxiliary inductive biases that could otherwise obscure its contribution.

Training was conducted on a synthetic dataset of 40K sample pairs generated as per Section 3.3, with an 80/10/10% split for training, validation, and testing respectively. Interventional samples were generated by random do-operations on either single or multiple interventional targets at each timestep, in accordance with the methodology outlined in Section 3.5.

Table 6.8: Preliminary causal discovery results on synthetic data of a medium-size model (∼2M parameters) with and without interventional samples, rounded to three decimal places.

Training Configuration	**AUC**
Observational data only	0.742
Observational + Interventional data	0.805

Table 6.8 shows that the inclusion of interventional data improves causal edge recovery. This result highlights that the observed gain in performance

originates purely from the additional interventional signal rather than any imposed structural priors.

These preliminary findings suggest that LCMs can effectively incorporate interventional data to enhance causal identifiability. Future work aims to investigate whether this benefit persists under more complex hybrid observational-interventional scenarios and by comparing against different approaches.

6.8 Preliminary Results on Prior Knowledge

This section investigates how the inclusion of prior causal knowledge and its associated belief strength affect inference in LCMs. Experiments were conducted on the `S_Joint` dataset using a medium-sized LCM ($\approx$ 1.8M parameters) and trained under the staged curriculum learning schedule described in Section 5.5.3. Similarly, no correlation injection (CI) or correlation regularization (CR) terms were used to avoid auxiliary inductive biases. Both the training and evaluation phases used the same dataset split to ensure consistency, while varying only prior information.

Prior knowledge is incorporated following the mechanism described in Subsection 4.6.2, where belief strengths are represented as scalar confidence values associated with each prior edge/path or exclusion. Two belief regimes were evaluated: (i) *Low belief*, where prior confidence values are uniformly sampled from $[0.1, 0.3]$ and (ii) *High belief* where prior confidence values are uniformly sampled from $[0.7, 0.9]$. These ranges align with the belief ranges used in the curriculum learning schedule defined in Section 5.5.3.

Table 6.9: Preliminary results on the effect of prior knowledge type and belief strength on causal discovery performance (AUC, rounded to two decimals) on the `S_Joint` dataset. Performance against the same model without any prior knowledge and the observationally trained model from stage 0 of training is also reported.

Prior Type	**Belief Strength**	**AUC**
True Edge Priors	Low	0.93
True Edge Priors	High	0.96
Path Exclusions	Low	0.91
Path Exclusions	High	0.92
False Edge Priors	Low	0.93
False Edge Priors	High	0.89
All Types Combined	Low	0.90
All Types Combined	High	0.92
No Priors (same model)	-	0.78
No Priors (weights from stage 0)	-	0.74

Table 6.9 shows that the presence of accurate priors enhances causal discovery accuracy, while incorrect priors or path exclusions can degrade performance. Importantly, the magnitude of these effects scales with the model's belief strength. High-confidence priors amplify both benefits and risks, improving results when priors are correct, but substantially reducing accuracy when

priors are false, as evident by the depicted drop in AUC.

Table 6.9 reports our preliminary findings on the efficacy of prior knowledge in LCMs. Empirically, the inclusion of correct priors yields the highest AUC, confirming that informative priors improve causal discovery. Importantly, introducing false or contradictory priors only marginally reduces performance, indicating that the LCM succesfully integrates prior information in a stochastic and belief-weighted manner, rather than enforcing it deterministically. Compared to non-prior baselines (AUCs of 0.78 and 0.74 for the same model without priors[3] and the observationally trained model from stage 0 of training[4], respectively), even weak priors improve performance. This demonstrates that the LCM's prior loss operates as a soft regularizer which, weighted by belief strengths and combined with evidence, allows discounting of implausible causal relations, as intended. When all prior types are combined, performance remains high, suggesting that the model effectively reconciles mixed-quality priors, thus avoiding implausible ones while leveraging consistent cues. The above observations highlight encouraging features for leveraging LCMs in real-world settings where expert knowledge is incomplete or unreliable.

[3]Prior knowledge and belief tensors given as zero tensors to the input.

[4]The model from stage 0 of training is trained as per the staged curriculum learning schedule described in Section 5.5.3.

This page intentionally left blank

7

Conclusion

"The best way to predict the future is to invent it."

– Alan Kay

In this work, we introduced *Large Causal Models* (LCMs), a family of foundation models for temporal causal discovery trained on large-scale datasets of time-series and corresponding ground-truth causal graphs. By leveraging a diverse mixture of synthetic and realistically generated datasets, our LCMs achieve strong generalization across domains and demonstrate competitive or superior performance to existing methods that require dataset-specific training, while operating in constant time, both at in-distribution and out-of-distribution scenarios.

7.1 Summary of Contributions

Our first research avenue focused on establishing a comprehensive data generation pipeline for causal discovery. We developed a scalable framework for sampling synthetic random temporal structural causal models (TSCMs), enabling the automatic generation of corresponding samples via ancestral sampling. In contrast to existing approaches that are limited in both generation diversity and expressivity, our framework supports the creation of hundreds of thousands of SCM instances with high variability and controlled complexity.

The second avenue addressed the integration of realistic datasets into training, aiming to increase model expressivity and predictive robustness. We referred to these datasets as simulated. Motivated by evidence that the inclusion of realistic data improves foundation model performance in out-of-distribution scenarios, we proposed a novel method, *Temporal Causal-based Simulation (TCS)*, to bridge the gap between purely synthetic and real-world data. TCS enables the generation of thousands of causal models grounded in empirical data, paired with a Min-Max optimization scheme, *Adversarial Causal Tuning (ACT)*, that facilitates optimal causal model selection. Together with subsampling across time and variable dimensions, it significantly expanded the available training corpus. Additionally, it serves as a method for benchmarking causal discovery algorithms, which is of vital importance in the Causality community.

Building on this foundation, the use of sample statistics within training and model inference was explored, showing that statistical regularization improves

learning stability and performance. The integration of synthetic training data with realistic data was shown to improve generalization and robustness, demonstrating that causal foundation models also benefit from a diverse training corpus. Additionally, the current state-of-the-art has been extended from five to twelve input variables. In contrast to prior work trained solely on synthetic data and prone to performance degradation under scaling, introduced models maintain stable and robust behavior across both in-distribution and out-of-distribution settings, while surpassing or performing on par with established causal discovery algorithms.

Finally, a novel neural architecture for LCMs was proposed, integrating mechanisms such as patch-based representations and alternating attention layers. Early experiments indicate promising behavior, although these findings remain preliminary in nature. Initial explorations into the incorporation of interventional data and prior knowledge suggest potential avenues for extending foundation models for causal discovery towards a more unified scope.

Overall, this work aims to establish the conceptual and technical frame for scalable, foundation-style models in temporal causal discovery. Through a combination of training data generation, design and pretraining strategies, the groundwork for neural causal discovery methods that are robust, generalizable and applicable to complex real-world systems has been paved.

7.2 Future Directions

Our work regarding LCMs results in many interesting avenues for further research. Approaches for expanding our models may arise from reducing the amount of causal assumptions needed for interpreting the output of our models. For instance, by relaxing the assumption of causal sufficiency, explicit modeling of latent confounders (e.g., indirect causes such as $X_t^i \leftarrow X_{t-1}^k \rightarrow X_t^j$) is enabled, while the same can be done for the handling of contemporaneous effects. Another interesting point of research is to the inclusion of direct causal effects in the output of our LCMs. As causal effects are available during training from the ground truth causal graph, it would be interesting to investigate estimating them using an auxiliary task within the proposed framework. With the rise of AI Agents fine-tuned to specific tasks, we envision the incorporation of the output of LCMs into agents for causal discovery *(Causal AI Agents)*. Such agents would actively reason about interventions, simulate outcomes, and refine their understanding of the environment through causal experimentation. By embedding LCMs as internal causal world models, such agents could autonomously generate and test causal hypotheses, while their natural language could serve as a natural medium for *AI Explainability*.

While our LCMs present advances in scalable temporal causal discovery, they remain primarily a graph-predicting model, without explicitly modeling or leveraging disentangled, causal representations of the underlying processes. Recent work by Schölkopf et al. [2021] highlights the potential of learning modular and invariant causal representations as a foundation for robust and generalizable reasoning. An exciting direction for future research would therefore be extending our LCMs to jointly learn structured representations that encode the independent mechanisms and interventions of the system, potentially improving interpretability and zero-shot generalization.

By bridging causal discovery, representation learning, and agentic decision-making, such systems could ultimately form the backbone of *causally grounded AI*, a new paradigm where models not only predict correlations but understand and manipulate the causes that generate them.

7.3 Epilogue & Final Thoughts

Our journey began with a simple question: can we create a foundation model on the dance of time-series, to uncover the hidden strings of causality that govern examined data? How can this be achieved without the need for re-training on each independent data input? Along the way, we wandered through the importance of causal assumptions, the complexity of real-world data, and the black-box beauty of deep learning. Each of these contributions, felt at times like bricks in a cathedral still under construction: each one necessary, yet incomplete on its own. Together, they form a sturdy ground on which future contributions may build higher.

To this end, much remains to be explored. Questions left unanswered remain as important as those addressed. In the end, as large language models continue to rise in prominence, their limitations in causal reasoning highlight a parallel challenge: models that excel at pattern recognition but lack causal understanding risk producing contextually plausible yet logically unsound outputs. If causality is essential for LLMs, then LCMs represent an analogous step for temporal data: embedding causal reasoning at the very core of time-series modeling.

> *"While LLMs excel at tasks involving language understanding, generation, and pattern recognition, they often struggle with tasks that require deeper causal reasoning. Without an understanding of causality, LLMs may produce outputs that are contextually relevant but not logically sound, leading to potential issues such as hallucinations, biased outputs, and an inability to perform well on decision-making tasks that depend on causal relationships. Incorporating causality into LLMs is essential for several reasons."*
>
> – Wu et al. [2024a]

This page intentionally left blank

Appendix A

d-Separation

The central tool connecting conditional independencies of data and their corresponding DAG is the concept of *d-separation* (directional separation) introduced by Pearl et al. [1988]. We provide its definition for completeness of our writing, as it represents a graphical criterion on whether a set of variables $\mathbf{X}$ is conditionally independent of another set $\mathbf{Y}$, given a third set $\mathbf{Z}$, based on the structure of the DAG. It is defined for the i.i.d. case, but it can also be extended intuit to the time series setting.

Definition 8 (d-Separation) *Let $\mathcal{G}$ be a DAG over a set of variables $\mathbf{V}$. A path p between nodes X and Y is said to be* blocked *by a set of variables $\mathbf{Z}$ if one of the following holds:*

1. *There exists a chain $A \to B \to C$ or a fork $A \leftarrow B \to C$ such that $B \in \mathbf{Z}$,*
2. *There exists a collider $A \to B \leftarrow C$ such that $B \notin \mathbf{Z}$ and no descendant of B is in $\mathbf{Z}$.*

If all paths between X and Y are blocked by $\mathbf{Z}$, then X and Y are said to be d-separated *given $\mathbf{Z}$, written $X \perp\!\!\!\perp Y \mid \mathbf{Z}$.*

D-separation serves as the graphical counterpart to conditional independence in the disentagled factorization $\mathbb{P}$, under the assumption of the Causal Markov Condition (Section 2.4). It plays a fundamental role in constraint-based causal discovery algorithms, such as the PC [Spirtes et al., 2001] algorithm (and importantly, its temporal extension PCMCI [Runge, 2018]), which rely on conditional independence tests to infer causal structure.

This page intentionally left blank

Appendix B

The Transformer

In this section we aim to provide a gentle introduction to the Transformer, as it is used in our LCM architecture, without extensively elaborating on its inner mechanism. Instead, we focus only on the essential components relevant to our LCMs. We refer the reader to Vaswani et al. [2017] for a more detailed description of the Transformer.

Up to 2017, the main strategy for approaching any NLP task has been with a reccurent neural network. The Transformer paper by [Vaswani et al., 2017] introduced a novel neural architecture addressing many shortcomings of RNN-based models, in the context of machine translation (e.g. translating a sentence from French to English). RNNs have their sequential nature presents issues in parallelization, due to the unfolding of their hidden states. Modern graphical processor units (GPUs) and Tensor Processing Units (TPUs) are capable of parallelizing matrix and tensor operations. For instance, when we compute the product $A \cdot B$ of $A \in \mathbb{R}^{n \times k}$ and $B \in \mathbb{R}^{k \times d}$, many sums and multiplies are not dependent on each other and can be efficiently computed congruently. For an RNN, each hidden state is dependent on the previous state h_{t-1} in the sequence, which is dependent on h_{t-2} and so on. Visually, this is depicted in Figure B.1.

Another issue concerning RNNs is the difficulty with which tokens interact with each other relative to their distance.[1] By interacting we refer to whether the presence of one token in the past affects the processing of another token in a future state. This can make it difficult to learn how distant words should impact the representation of the current word.

Instead, the Transformer entirely replaces the recurrent mechanism with only self-attention operations, which allows for efficient parallelization and processing of the input sequence. The concept of attention however is not new. Essentially, the notion of direct interaction between elements of a sequence in an RNN has already been introduced with context-based neural attention by [Bahdanau, 2014]. The Transformer presents an entire replacement for RNNs, revolving just around the concept of self-Attention which solves both parallelization and interaction distance issues. As already discussed, this innovation reduces significant bottlenecks in training large-scale SOTA models, leading to

[1] This is closely related to the notion of a *receptive field*, i.e., the effective context or span of input elements that can influence the representation at a given position. In RNNs this receptive field grows sequentially, whereas in Transformers it is global and uniform across the sequence.

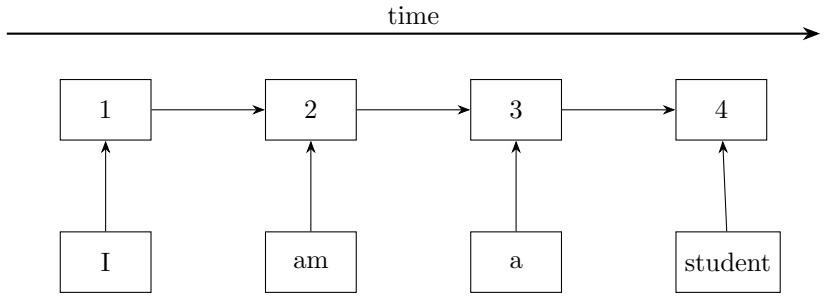


Figure B.1: An RNN unrolled over the time space. Rectangles correspond to intermediate states of the RNN. The first row corresponds to the embedding layer.

a paradigm shift of using Transformer-based models for foundation models to only for Natural Language Processing but for other domains as well.

High-Level Overview

We begin with a high-level overview of the vanilla Transformer architecture, as introduced in the seminal work of Vaswani et al. [2017], originally designed for machine translation. In this section, inputs and outputs are assumed to be sequences of words (not yet multivariate time-series). The architecture follows an *encoder-decoder* structure, with each side consisting of N identical blocks. In Figure B.2, the left part corresponds to the Encoder and the right part to the Decoder. This design replaces the recurrence in RNN/LSTM-based models with purely attention-driven computations, enabling full parallelization during training and more efficient scaling to large datasets.

Embeddings and Positional Encodings

Let the input sequence be $F = (w_1, \ldots, w_n)$, where each $w_i \in \mathcal{V}$, the vocabulary of size $|\mathcal{V}|$. The first step is to map discrete tokens into continuous representations. Each token w_i is projected into a dense vector $u_i \in \mathbb{R}^{d_{\text{model}}}$ via an *embedding layer*, yielding the matrix representation

$$X = \begin{bmatrix} u_1^\top \\ u_2^\top \\ \vdots \\ u_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d_{\text{model}}}.$$

where d_{model} is the dimensionality of the embeddings and known as the *model dimension*. It is a hyperparameter, tuned by the user, which controls the dimensionality of the representation and as such the capacity of the model. As will be made evident later on, a larger model dimension results in a more complex model with a higher number of learnable parameters. These embeddings may be randomly initialized and learned during training, pre-trained (e.g., word2vec, GloVe), or obtained from more advanced contextual models.

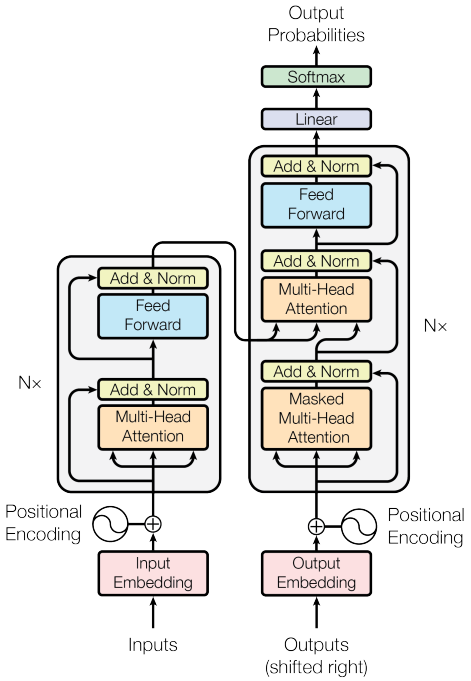


Figure B.2: Illustration of the Transformer Encoder-Decoder architecture by Vaswani et al. [2017].

Since there is no dependence of position in the Transformer mechanism, positional information must be injected explicitly. This is achieved by adding *positional encodings* to each embedding. These encodings can be either (i) fixed, such as the original sinusoidal functions which allow extrapolation to unseen sequence lengths, or (ii) learnable, updated during training. Formally, the effective input becomes

$$\tilde{u}_i = u_i + p_i, \quad i = 1, \ldots, n, \tag{B.1}$$

where $p_i \in \mathbb{R}^{d_{\text{model}}}$ encodes position i. Unlike recurrent models (RNNs, LSTMs, GRUs), which carry positional information through sequential state updates, Transformers rely entirely on these encodings. An illustration of the positional encodings is shown in Figure B.3.

Encoder

The encoder processes the input sequence through a stack of N identical blocks. Each block contains: (i) *Multi-Head Self-Attention*, which allows each token to attend to all others in the sequence in parallel, (ii) *Feed-Forward Network (FFN)*, a two-layer position-wise MLP with nonlinearity: $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$ and (iii) *Add & Norm*, consisting of residual connections and layer normalization to stabilize training. The encoder produces a sequence of contextualized representations that encode global dependencies across tokens. Only the final encoder layer output is directly passed to the decoder, though it is used by every decoder block.

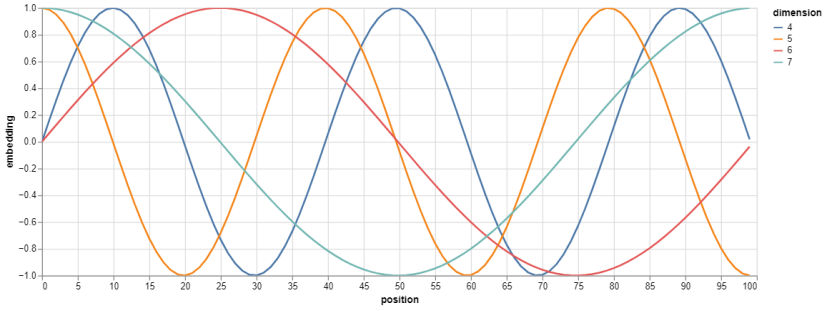


Figure B.3: Illustration of various sinusoidal positional encodings for $d_{\text{model}} = 4, 5, 6, 7$ and sequence length $L = 100$.

Decoder

The decoder operates analogously to the encoder but with two crucial differences. The target sequence $E = (e_1, \ldots, e_\ell)$ is first embedded (with a prepended start-of-sequence token $\langle \text{SOS} \rangle$ and shifted right by one position). Each decoder block then contains (i) *Masked Multi-Head Self-Attention*, where a causal mask ensures that position i can only attend to tokens $< i$, preventing information leakage from future positions, (ii) *Encoder-Decoder Attention*, a cross-attention mechanism (for which we won't elaborate) that allows the decoder to query encoder outputs, aligning target tokens with the source sequence and (iii) a *Feed-Forward Network (FFN)* with residual and normalization layers as in the encoder.

After passing through the stack of N decoder blocks, the output logits are transformed by a *final linear projection* (i.e. a feedforward linear layer) and a *softmax* function. The linear projection maps the hidden states into a vector of dimension $|\mathcal{V}|$, the vocabulary size. The softmax function then converts these scores into probabilities:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{|\mathcal{V}|} \exp(z_j)} \quad \text{for each } i \in \{1, \ldots, |\mathcal{V}|\},$$

where $z \in \mathbb{R}^{|\mathcal{V}|}$ are the logits produced by the final linear layer.

During training, the entire ground-truth sequence is available, and the model predicts all positions in parallel. During *inference*, however, the decoder generates tokens autoregressively, feeding each predicted token back as input until the end-of-sequence symbol is reached. Decoding can be done greedily or with beam search for better sequence-level optimization.

The Attention Mechanism

We can now elaborate on the internal mechanism of the Transformer, the *self-Attention* operation. Broadly speaking, it can viewed as a mechanism of taking a query (like in database search), looking-up information in a key-value store by selecting the values of the keys that most likely match the given query. In a sense, by "most likely match" we refer to putting more weight to those that correspond to the keys more like the query, and by "selecting" by averaging

over all values. As discussed before, many types of attention exist. The specific type of attention used by the Transformer is known as *Query-Key-Value self-Attention*. The main idea is that each token representation can *attend* to other tokens in the sequence and integrate their collective information. Given an input matrix $X \in \mathbb{R}^{L \times d_{\text{model}}}$ (with L tokens and model dimension d_{model}), the query, key and value projections are computed:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{B.2}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d}$ are learnable weight matrices for the query, key, and value projections. Often it is assumed that $d = d_K = d_V = d_Q$. Now consider a token $\mathbf{x}_i$ in the sequence $x_{1:n}$. The contextual representation $\mathbf{h}_i$ of $\mathbf{x}_i$ is the weighted sum of the values of the sequence $\mathbf{h}_i = \sum_{j=1}^{n} \alpha_{ij} \mathbf{v}_j$, where the weights α_{ij} control the strength of each value $\mathbf{v}_j$. The weights are obtained by computing the affinities between the keys and the query, which is the inner product $q_i^T k_j$ and then taking the softmax to obtain probablities:

$$\alpha_{ij} = \frac{\exp(q_i^T k_j)}{\sum_{j=1}^{n} \exp(q_i^T k_j)} \tag{B.3}$$

Regarding complexity, computing the attention scores involves the inner product $QK^\top \in \mathbb{R}^{L \times L}$ which costs $\mathcal{O}(L^2)$ in time and requires $\mathcal{O}(L^2)$ space for storing the pairwise similarities. Multiplying the score matrix with the values matrix V adds another $\mathcal{O}(L^2 d)$. A novelty by Vaswani et al. [2017] is the scaling of the product with respect to the square root of the dimensionality of the keys d_k, resulting in the *scaled dot-product attention*:

$$\text{Attention}(Q, K, V) = \text{softmax}(\left(\frac{QK^\top}{\sqrt{d_k}}\right))V \tag{B.4}$$

where d_k is the dimension of keys. The intuition is that when the dimensionality d_k of the dotted vectors grows large, the dot product of even random vectors (as during initialization) scales roughly as $\sqrt{d_k}$.

Each row of the resulting matrix is a weighted average of the values, where weights are determined by the similarity between queries and keys. The scaling by $\sqrt{d_k}$ prevents large dot products from destabilizing gradients. Note that for the algebra to work out, the number of keys and values n must be equal, but the number of queries m can vary.

Multi-Head Attention

Intuitively, a single self-attention operation is best at picking out a single value (on average) from the input value set. It does so softly, by averaging over all of the values, but it requires a balancing game in the key-query dot products in order to carefully average two or more things. Instead of performing a single attention operation, the Transformer uses h *independent attention heads* to capture multiple types of relationships i parallel, which applies self-attention multiple times with different query, key and value transformations on the same input, with the outputs concatenated and projected back:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{B.5}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

This design allows the model to represent diverse dependencies (syntactic, semantic, positional) simultaneously. While the theoretical cost is $\mathcal{O}(L^2 d_{\text{model}})$ in both time and space, optimized implementations (e.g., FlashAttention) can reduce memory usage to $\mathcal{O}(Ld_{\text{model}})$, improving scalability, but remaining outside the scope of our work.

Layer Normalization

Layer normalization [Ba et al., 2016] is crucial for stabilizing training. For token $h_i \in \mathbb{R}^d$, statistics are computed across the hidden dimension:

$$\hat{\mu}_i = \frac{1}{d}\sum_{j=1}^{d} h_{ij}, \quad \hat{\sigma}_i^2 = \frac{1}{d}\sum_{j=1}^{d}(h_{ij} - \hat{\mu}_i)^2, \tag{B.6}$$

and the normalized output is

$$\text{LayerNorm}(h_i) = \frac{h_i - \hat{\mu}_i}{\sqrt{\hat{\sigma}_i^2 + \epsilon}} \cdot \gamma + \beta, \tag{B.7}$$

with learnable scale and bias parameters $\gamma, \beta \in \mathbb{R}^d$.

Residual Connections

Residual connections [He et al., 2016] alleviate vanishing gradients and ease optimization in deep architectures. Formally,

$$\text{Residual}(x) = x + \text{Layer}(x). \tag{B.8}$$

The Transformer uses a combination of residual connections and normalization, known as *Add & Norm*. Two main variants exist:

$$h = \text{LayerNorm}(x + \text{Layer}(x)) \quad \text{(post-norm)}, \tag{B.9}$$

$$h = x + \text{LayerNorm}(\text{Layer}(x)) \quad \text{(pre-norm)}. \tag{B.10}$$

Empirically, pre-norm architectures exhibit better gradient flow and faster convergence in very deep Transformers [Xiong et al., 2020].

Dropout Layers

Dropout [Srivastava et al., 2014] is a regularization technique designed to mitigate overfitting in neural networks by randomly deactivating a subset of neurons during training. At each iteration, a binary mask $m \sim \text{Bernoulli}(p)$ is sampled where p denotes the probability of retaining a unit. The layer output is then given by

$$\text{Dropout}(x) = m \odot x, \tag{B.11}$$

where $\odot$ denotes element-wise multiplication. During inference, dropout is typically disabled, and activations are scaled by the retention probability p to preserve the expected activation magnitude. In Transformer models, dropout is

commonly applied to several components: (i) the outputs of attention weights before the residual addition, (ii) the outputs of feed-forward sublayers, and (iii) the input embeddings. Studies by Zhai et al. [2022] have shown that dropout contributes significantly to generalization in large-scale sequence models, particularly when combined with normalization and residual connections.

This page intentionally left blank

Appendix C

Optimizers

This chapter serves as complimentary material to the high level overview of deep learning in Section 2.5 for the interested reader, concerning the optimization of deep learning models.

Stochastic Gradient Descent

Named after its inherent randomness, *Stochastic Gradient Descent (SGD)* is one of the most well-known optimizers. Many consider it a cardinal pillar of modern artificial intelligence systems, as its core ideas remain highly relevant. Depending on the specific problem and the input dataset, using the original SGD algorithm as-is can present significant issues. For example, consider an input dataset x with a substantial amount of highly diverse samples, x_d. Since the gradient update in the original SGD is computed based on a *single drawn sample* per iteration, the gradient computed for these x_d samples may point in a direction completely opposite to the overall underlying distribution of x. This scenario leads to unstable, highly variant gradients, which consequently implies slower convergence or even failure to converge.

Mini-Batch SGD

The original SGD method described previously is often unstable because it updates the model's parameters using the gradient from only one sample at a time. To address this, a superior variant called Mini-Batch SGD is typically used. Instead of a single sample, Mini-Batch SGD computes the gradient using a batch of samples in each iteration. This approach offers significant benefits: (i) *stability*, as a batch of samples provides a much more representative estimate of the input data's distribution than a single sample. Consequently, the calculated gradients and the model's overall convergence are much more stable. (ii) *generalization*: Mini-Batch SGD is a generalization of the original SGD; the two are identical when the batch size B is equal to one. Nevertheless, it is almost always preferable to use a (non-zero) power of two as batch size [1].

[1] Generally speaking, since memory management in all modern operating systems is organized in powers of two, selecting a power of two for the batch size avoids fetching more pages than needed, therefore improving the algorithm's overall speed.

Algorithm 14 Mini-Batch Stochastic Gradient Descent (SGD)

Input: Parameters θ, learning rate η, batch size B, loss function $\mathcal{L}$
Output: Updated parameters θ

for each iteration $t = 1, 2, \ldots$ **do**
  Sample a mini-batch $\{(x^{(i)}, y^{(i)})\}_{i=1}^{B}$
  $g_t \leftarrow \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(f(x^{(i)}; \theta), y^{(i)})$
  $\theta \leftarrow \theta - \eta\, g_t$
end for

Algorithm 15 Adam Optimizer

Input: Parameters θ, learning rate η, $\beta_1, \beta_2 \in [0, 1)$, $\epsilon > 0$
Output: Updated parameters θ

Initialize $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$
for each iteration **do**
  $t \leftarrow t + 1$
  Sample mini-batch and compute gradient $g_t \leftarrow \nabla_\theta \mathcal{L}(\theta)$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (First moment)
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Second moment)
  $\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$ (Bias correction)
  $\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$ (Bias correction)
  $\theta \leftarrow \theta - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$
end for

Additionally, when implementing algorithms like this, biases are often omitted from the pseudocode for simplicity. This is because a bias term is mathematically and functionally just another trainable parameter (a weight) in the model. There's no practical need to update or treat biases separately from the other weights.

Adam Optimizer

The *Adaptive Moment Estimation (Adam)* optimizer [Kingma, 2015] enhances Mini-Batch SGD by incorporating two key momentum variables. A core limitation of standard Mini-Batch SGD is that updates from previous iterations are entirely disregarded when calculating new updates. This can lead to undesirable *oscillations*, where weights are updated consecutively in wildly different directions, thereby impeding convergence. Adam addresses this by introducing the first moment variable, **m**, which serves to maintain the momentum gained from previous updates. Adam further improves stability and efficiency using the second moment variable, **v**, which provides an *individual learning rate* for every weight in the model. Specifically, contrary to Mini-Batch SGD's fixed learning rate, when a weight receives a relatively large update (i.e., its gradient magnitude is high), its effective learning rate is reduced for subsequent iterations. This mechanism prevents consecutive large updates to the same weights, offering a crucial boost to the optimizer's stability and preventing slow convergence.

The success of Adam stems from its ability to combine the advantages of

two prior extensions of Mini-Batch SGD: the *Adaptive Gradient Algorithm (AdaGrad)* [Duchi et al., 2011] and *Root Mean Square Propagation (RMSProp)* [Tieleman, 2012]. Despite its prevalence, robustness, and efficacy across most tasks, Adam is not universally optimal. There are scenarios where its heightened stability can actually be detrimental, causing it to struggle in finding an adequate minimum or generalizing as well as simpler optimizers like Mini-Batch SGD. For instance, Mini-Batch SGD is known to generalize better for many *Convolutional Neural Networks (CNNs)*, while Adam is often the optimizer of choice for models like *Generative Adversarial Networks (GANs)* [Goodfellow et al., 2014]. Consequently, research continues to be dedicated to finding superior optimization algorithms [Chen et al., 2023].

This page intentionally left blank

Appendix D

Illustrative Example

To provide an intuitive demonstration of our LCMs, we construct a synthetic temporal causal process consisting of three variables V^1, V^2 and V_3. The example is governed by a simple temporal structural causal model:

$$V_t^1 := \epsilon_1(t), \quad V_t^2 := 3V_{t-1}^1 + \epsilon_t^2, \quad V_t^3 := V_{t-2}^2 + 5V_{t-3}^1 + \epsilon_t^3,$$

where $\epsilon_t^i \sim \mathcal{N}(0,1)\ \forall i \in \{1,2,3\}, t \in \mathbb{Z}$. This induces the following lagged causal relationships:

$$V^1 \xrightarrow[\text{lag}=1]{} V^2, \quad V^1 \xrightarrow[\text{lag}=3]{} V^3, \quad V^2 \xrightarrow[\text{lag}=2]{} V^3$$

The function `run_illustrative_example` generates 500 synthetic time series data and the corresponding ground-truth lagged adjacency tensor. We then feed this data through the `9.4M` LCM model. The predicted lagged adjacency tensor is visualized in Figure D.1, while the corresponding true and predicted lagged causal graphs, thresholded at $\tau = 0.05$, are shown in Figure D.2. As expected, the model successfully captures the underlying causal structure of this simple temporal dataset.

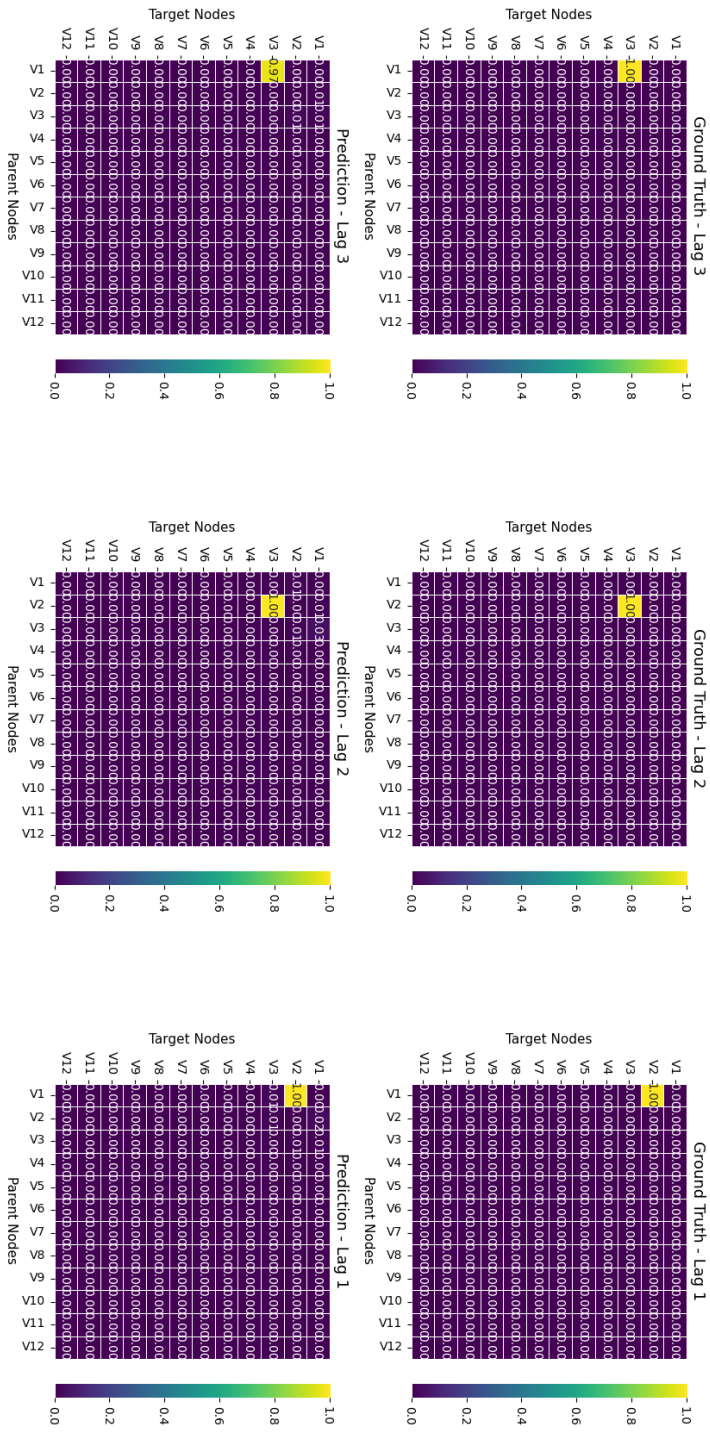


Figure D.1: Predicted (non-thresholded) lagged adjacency tensor $\hat{\mathbb{A}}_{j,i,\ell}$ by the 9.4M LCM on the illustrative example, against the ground truth.

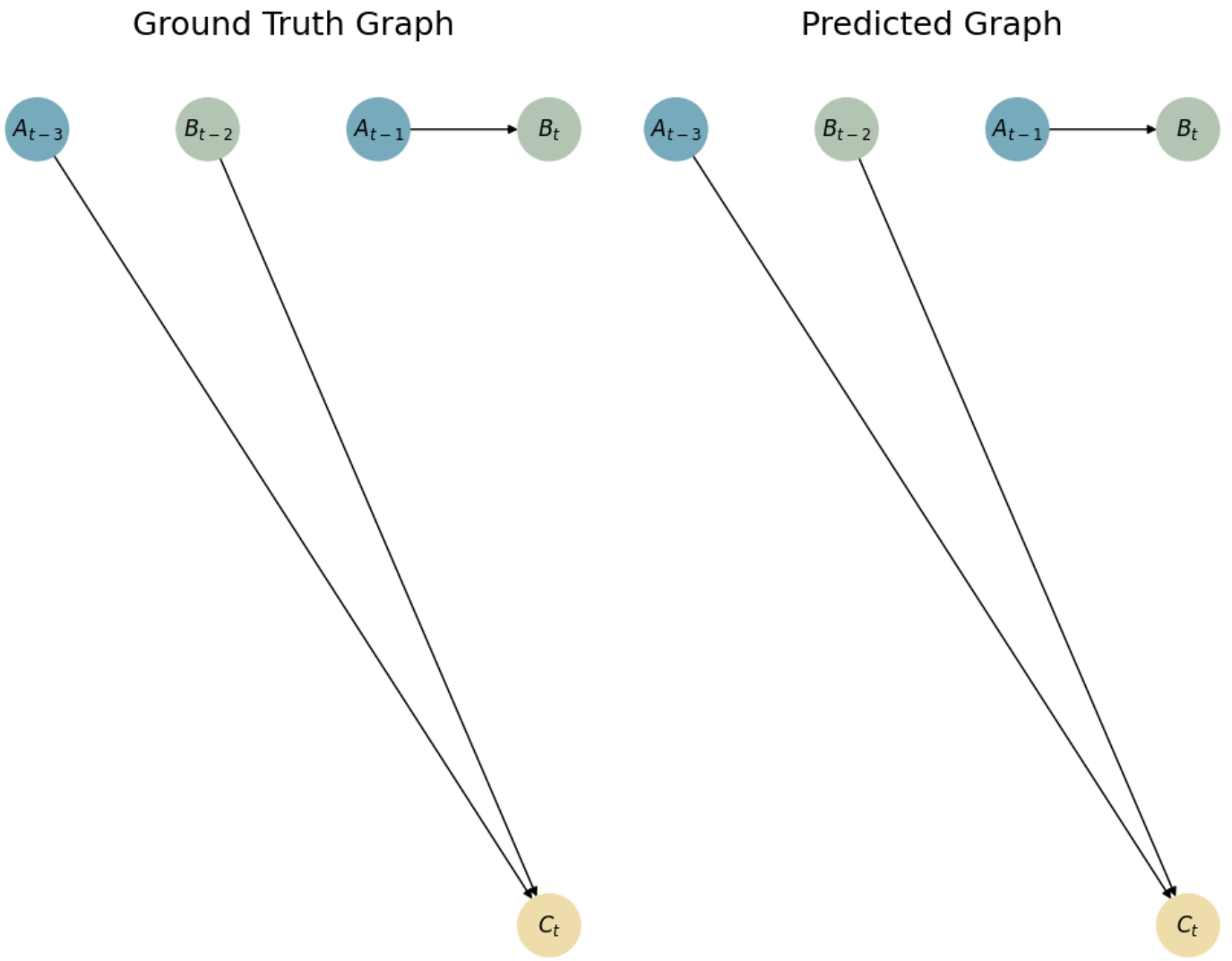


Figure D.2: Ground truth (left) and predicted (right) lagged causal graphs on the `9.4M` LCM on the illustrative example.

This page intentionally left blank

Appendix E

Additional Results

The following pages contain the complete set of experimental results reported in the main text, as well as additional results. We report the statistical significance of AUC differences between model variants, where each entry lists the mean AUC, its standard deviation, raw p-value, and whether the difference remains significant after correction. For the above tables, we use a Bonferroni correction for multiple comparisons. For data collections with less than 30 samples, such as the fMRI datasets, we do not report p-values as the sample size is too small to obtain a reliable estimate.

Tables E.1 & E.2 contain results for training aids ablations on the in-distribution synthetic test set. Tables E.3 to E.10 report results on selecting the optimal mixture of synthetic and realistic data for training of LCMs. Figures E.11 to E.14 illustrate representative running times across datasets and model variants. Finally, Tables E.15 to E.38 contain the complete set of experimental results covered in the main text.

Table E.1: Training aids ablations for LCMs on the in-distribution synthetic holdout test set of `S_Joint`.

Test Set: in-distribution (holdout)

Test Data	Lags	Variables	Func. Dep.	# Samples	# Datasets
S_Joint	1–3	3–5	L, NL	500	2000

Parameter	Value
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	256
n_{heads}	2
n_{blocks}	1
d_{ff}	128
Dropout	**0.05**
L_{max}	500
Training Aids	No / Yes
Total params	905K / 914K

Model	AUC_{mean}	TPR_{mean}	FPR_{mean}	TNR_{mean}	FNR_{mean}	Precision_{mean}	Recall_{mean}	F1_{mean}
$\text{LCM}_{(\text{Informer})}$	0.830± .000	0.972± .000	0.311± .000	0.689± .000	0.028± .000	0.743± .000	0.972± .000	0.841± .000
$\text{LCM}_{(\text{Informer, CI})}$	0.845± .000	0.977± .000	0.287± .000	0.713± .000	0.023± .000	0.800± .000	0.977± .000	0.826± .000
$\text{LCM}_{(\text{Informer, CI, } \lambda_{CR}=1)}$	0.875± .000	0.883± .000	0.134± .000	0.866± .000	0.116± .000	0.817± .000	0.883± .000	0.844± .000
$\text{LCM}_{(\text{Informer, CI, } \lambda_{CR}=0.25)}$	0.870± .000	0.859± .000	0.119± .000	0.881± .000	0.141± .000	0.816± .000	0.859± .000	0.830± .000
$\text{LCM}_{(\text{Informer, CI, } \lambda_{CR}=0.5)}$	0.876± .000	0.924± .000	0.172± .000	0.828± .000	0.076± .000	0.816± .000	0.924± .000	0.863± .000
$\text{LCM}_{(\text{Informer, CI, } \lambda_{CR}=0.75)}$	0.875± .000	0.893± .000	0.143± .000	0.857± .000	0.107± .000	0.819± .000	0.893± .000	0.849± .000

Table E.2: Significance of AUC differences between LCM variants trained on the in-distribution synthetic holdout test set of `S_Joint`. Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	$\text{AUC}_{\text{A}}^{\text{mean}}$	$\text{AUC}_{\text{B}}^{\text{mean}}$	$\textbf{p-value}_{\textbf{raw}}$	**Significant After Correction**
LCM(Informer)	LCM(Informer, CI)	0.8305 ± 0.0783	0.8450 ± 0.1126	$2.55e^{-08}$	Yes
LCM(Informer)	LCM(Informer, CI, λ_{CR}=1)	0.8305 ± 0.0783	0.8747 ± 0.1507	$3.75e^{-53}$	Yes
LCM(Informer)	LCM(Informer, CI, λ_{CR}=0.25)	0.8305 ± 0.0783	0.8699 ± 0.1603	$1.13e^{-38}$	Yes
LCM(Informer)	LCM(Informer, CI, λ_{CR}=0.5)	0.8305 ± 0.0783	0.8760 ± 0.1355	$5.24e^{-83}$	Yes
LCM(Informer)	LCM(Informer, CI, λ_{CR}=0.75)	0.8305 ± 0.0784	0.8748 ± 0.1483	$2.39e^{-59}$	Yes
LCM(Informer, CI)	LCM(Informer, CI, λ_{CR}=1)	0.8450 ± 0.1126	0.8747 ± 0.1507	$4.56e^{-58}$	Yes
LCM(Informer, CI)	LCM(Informer, CI, λ_{CR}=0.25)	0.8450 ± 0.1126	0.8699 ± 0.1603	$6.90e^{-42}$	Yes
LCM(Informer, CI)	LCM(Informer, CI, λ_{CR}=0.5)	0.8450 ± 0.1126	0.8760 ± 0.1355	$1.55e^{-79}$	Yes
LCM(Informer, CI)	LCM(Informer, CI, λ_{CR}=0.75)	0.8450 ± 0.1126	0.8748 ± 0.1483	$2.79e^{-61}$	Yes
LCM(Informer, CI, λ_{CR}=1)	LCM(Informer, CI, λ_{CR}=0.25)	0.8747 ± 0.1507	0.8699 ± 0.1603	$4.68e^{-05}$	Yes
LCM(Informer, CI, λ_{CR}=1)	LCM(Informer, CI, λ_{CR}=0.5)	0.8747 ± 0.1507	0.8760 ± 0.1355	$8.10e^{-28}$	Yes
LCM(Informer, CI, λ_{CR}=1)	LCM(Informer, CI, λ_{CR}=0.75)	0.8750 ± 0.1505	0.8748 ± 0.1483	0.0192	No
LCM(Informer, CI, λ_{CR}=0.25)	LCM(Informer, CI, λ_{CR}=0.5)	0.8699 ± 0.1603	0.8760 ± 0.1355	$1.98e^{-22}$	Yes
LCM(Informer, CI, λ_{CR}=0.25)	LCM(Informer, CI, λ_{CR}=0.75)	0.8698 ± 0.1604	0.8748 ± 0.1483	$2.31e^{-12}$	Yes
LCM(Informer, CI, λ_{CR}=0.5)	LCM(Informer, CI, λ_{CR}=0.75)	0.8760 ± 0.1356	0.8748 ± 0.1483	$8.22e^{-18}$	Yes

Table E.3: Optimal mixture of synthetic and simulated data for LCM training, evaluated on the out-of-distribution semi-synthetic test collection *f*MRI5. Reported are mean (± standard deviation) values across multiple runs.

Test Set: Out-of-Distribution (Semi-Synthetic)

Test Data	Lags	Variables	Func. Dep.	# Samples	# Datasets
*f*MRI5	1	5	NL	200-2400	21

Parameter	Value
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	256
n_{heads}	2
n_{blocks}	1
d_{ff}	256
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	1.01M

Model (Synth/Sim %)	AUC_{mean}	TPR_{mean}	FPR_{mean}	TNR_{mean}	FNR_{mean}	$\text{Precision}_{\text{mean}}$	$\text{Recall}_{\text{mean}}$	F1_{mean}
LCM (100/0)	$0.819_{\pm.000}$	$0.929_{\pm.002}$	$0.289_{\pm.001}$	$0.710_{\pm.010}$	$0.070_{\pm.002}$	$0.794_{\pm.000}$	$0.929_{\pm.002}$	$0.844_{\pm.003}$
LCM (80/20)	$0.877_{\pm.003}$	$1.000_{\pm.000}$	$0.245_{\pm.007}$	$0.754_{\pm.007}$	$0.000_{\pm.000}$	$0.813_{\pm.003}$	$1.000_{\pm.000}$	$0.894_{\pm.002}$
LCM (50/50)	$0.894_{\pm.002}$	$0.974_{\pm.002}$	$0.186_{\pm.004}$	$0.813_{\pm.004}$	$0.020_{\pm.002}$	$0.843_{\pm.003}$	$0.974_{\pm.000}$	$0.903_{\pm.002}$
LCM (20/80)	$0.893_{\pm.0001}$	$1.000_{\pm.000}$	$0.210_{\pm.003}$	$0.787_{\pm.003}$	$0.000_{\pm.000}$	$0.829_{\pm.002}$	$1.000_{\pm.000}$	$0.905_{\pm.001}$

Table E.4: Optimal mixture of synthetic and simulated data for LCM training, evaluated on the out-of-distribution semi-synthetic test collection *f*MRI. Reported are mean (± standard deviation) values across multiple runs.

Test Set: Out-of-Distribution (Semi-Synthetic)

Test Data	Lags	Variables	Func. Dep.	# Samples	# Datasets
*f*MRI	1	5,10	NL	200-2400	2

Parameter	Value
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	256
n_{heads}	2
n_{blocks}	1
d_{ff}	256
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	1.01M

Model (Synth/Sim %)	AUC_{mean}	TPR_{mean}	FPR_{mean}	TNR_{mean}	FNR_{mean}	$\text{Precision}_{\text{mean}}$	$\text{Recall}_{\text{mean}}$	F1_{mean}
LCM (100/0)	$0.745_{\pm.005}$	$0.927_{\pm.002}$	$0.436_{\pm.001}$	$0.563_{\pm.010}$	$0.070_{\pm.002}$	$0.733_{\pm.006}$	$0.927_{\pm.002}$	$0.795_{\pm.002}$
LCM (80/20)	$0.869_{\pm.002}$	$1.000_{\pm.000}$	$0.269_{\pm.005}$	$0.739_{\pm.005}$	$0.000_{\pm.000}$	$0.803_{\pm.003}$	$1.000_{\pm.000}$	$0.888_{\pm.002}$
LCM (50/50)	$0.888_{\pm.002}$	$0.979_{\pm.002}$	$0.20\ _{\pm.003}$	$0.795_{\pm.003}$	$0.020_{\pm.002}$	$0.832_{\pm.002}$	$0.979_{\pm.002}$	$0.898_{\pm.001}$
LCM (20/80)	$0.894_{\pm.0001}$	$1.000_{\pm.000}$	$0.210_{\pm.002}$	$0.788_{\pm.002}$	$0.000_{\pm.000}$	$0.829_{\pm.001}$	$1.000_{\pm.000}$	$0.905_{\pm.001}$

Table E.5: Optimal mixture of synthetic and simulated data for LCM training, evaluated on the out-of-distribution semi-synthetic test collection `Kuramoto5`. Reported are mean (± standard deviation) values across multiple runs.

Test Set: Out-of-Distribution (Semi-Synthetic)

Test Data	Lags	Variables	Func. Dep.	Edge Prob.	# Samples	# Datasets
Kuramoto5	1	5	NL	–	500	1000 Models

Parameter	Value
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	256
n_{heads}	2
n_{blocks}	1
d_{ff}	256
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	1.01M

Model (Synth/Sim %)	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM (100/0)	$0.629_{\pm.000}$	$0.959_{\pm.000}$	$0.700_{\pm.001}$	$0.290_{\pm.004}$	$0.040_{\pm.000}$	$0.600_{\pm.000}$	$0.959_{\pm.000}$	$0.729_{\pm.000}$
LCM (80/20)	$0.939_{\pm.000}$	$0.998_{\pm.000}$	$0.110_{\pm.000}$	$0.881_{\pm.001}$	$0.000_{\pm.000}$	$0.895_{\pm.000}$	$0.998_{\pm.000}$	$0.943_{\pm.000}$
LCM (50/50)	$0.925_{\pm.000}$	$0.999_{\pm.000}$	$0.140_{\pm.000}$	$0.851_{\pm.000}$	$0.000_{\pm.000}$	$0.875_{\pm.000}$	$0.999_{\pm.000}$	$0.932_{\pm.000}$
LCM (20/80)	$0.872_{\pm.000}$	$1.000_{\pm.000}$	$0.250_{\pm.000}$	$0.744_{\pm.000}$	$0.000_{\pm.000}$	$0.802_{\pm.000}$	$1.000_{\pm.000}$	$0.880_{\pm.000}$

Table E.6: Significance of AUC Differences between model variants (`Kuramoto5`): Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	$\text{AUC}_{\mathbf{A}}^{\mathbf{mean}}$	$\text{AUC}_{\mathbf{B}}^{\mathbf{mean}}$	$\mathbf{p\text{-value}_{raw}}$	**Significant After Correction**
LCM (100/0)	LCM (80/20)	0.629 ± 0.000	0.939 ± 0.000	$7.37e^{-165}$	Yes
LCM (100/0)	LCM (50/50)	0.629 ± 0.000	0.925 ± 0.000	$9.63e^{-165}$	Yes
LCM (100/0)	LCM (20/80)	0.629 ± 0.000	0.872 ± 0.000	$2.90e^{-159}$	Yes
LCM (80/20)	LCM (50/50)	0.939 ± 0.000	0.925 ± 0.000	$5.25e^{-28}$	Yes
LCM (80/20)	LCM (20/80)	0.939 ± 0.000	0.872 ± 0.000	$1.58e^{-157}$	Yes
LCM (50/50)	LCM (20/80)	0.925 ± 0.000	0.872 ± 0.000	$2.25e^{-126}$	No

Table E.7: Optimal mixture of synthetic and simulated data for LCM training, evaluated on the out-of-distribution semi-synthetic test collection (`Kuramoto10`). Reported are mean (± standard deviation) values across multiple runs.

Test Set: Out-of-Distribution (Semi-Synthetic)

Test Data	Lags	Variables	Func. Dep.	Edge Prob.	# Samples	# Datasets
Kuramoto10	1	10	NL	–	500	1000 Models

Parameter	Value
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	256
n_{heads}	2
n_{blocks}	1
d_{ff}	256
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	1.01M

Model (Synth/Sim %)	AUC_{mean}	TPR_{mean}	FPR_{mean}	TNR_{mean}	FNR_{mean}	$\text{Precision}_{\text{mean}}$	$\text{Recall}_{\text{mean}}$	F1_{mean}
LCM (100/0)	$0.652_{\pm.000}$	$0.914_{\pm.000}$	$0.609_{\pm.000}$	$0.390_{\pm.000}$	$0.080_{\pm.000}$	$0.635_{\pm.000}$	$0.914_{\pm.000}$	$0.724_{\pm.000}$
LCM (80/20)	$0.903_{\pm.000}$	$0.959_{\pm.000}$	$0.151_{\pm.000}$	$0.848_{\pm.000}$	$0.040_{\pm.000}$	$0.865_{\pm.000}$	$0.959_{\pm.000}$	$0.907_{\pm.000}$
LCM (50/50)	$0.920_{\pm.000}$	$0.998_{\pm.000}$	$0.157_{\pm.000}$	$0.842_{\pm.000}$	$0.001_{\pm.000}$	$0.865_{\pm.000}$	$0.998_{\pm.000}$	$0.927_{\pm.000}$
LCM (20/80)	$0.922_{\pm.000}$	$0.997_{\pm.000}$	$0.152_{\pm.000}$	$0.847_{\pm.000}$	$0.002_{\pm.000}$	$0.868_{\pm.000}$	$0.997_{\pm.000}$	$0.928_{\pm.000}$

Table E.8: Optimal mixture of synthetic and simulated for LCM training, evaluated on the out-of-distribution semi-synthetic test collection `Kuramoto10`. Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	$\text{AUC}_{\mathbf{A}}^{\mathbf{mean}}$	$\text{AUC}_{\mathbf{B}}^{\mathbf{mean}}$	$\mathbf{p\text{-value}_{raw}}$	**Significant After Correction**
LCM (100/0)	LCM (80/20)	0.652 ± 0.000	0.903 ± 0.000	$7.78e^{-163}$	Yes
LCM (100/0)	LCM (50/50)	0.652 ± 0.000	0.920 ± 0.000	$6.38e^{-163}$	Yes
LCM (100/0)	LCM (20/80)	0.652 ± 0.000	0.922 ± 0.000	$6.40e^{-163}$	Yes
LCM (80/20)	LCM (50/50)	0.903 ± 0.000	0.920 ± 0.000	$4.41e^{-50}$	Yes
LCM (80/20)	LCM (20/80)	0.903 ± 0.000	0.922 ± 0.000	$3.92e^{-63}$	Yes
LCM (50/50)	LCM (20/80)	0.920 ± 0.000	0.922 ± 0.000	$4.07e^{-02}$	No

Table E.9: Optimal mixture of synthetic and simulated for LCM training, evaluated on the out-of-distribution simulated data collection `AirQualityMS`.

Model	**AUC**	**TPR**	**FPR**	**TNR**	**FNR**	**Precision**	**Recall**	**F1**
LCM synth/sim % 100/0	0.886±.000	0.801±.001	0.020±.000	0.975±.000	0.198±.001	0.960±.000	0.800±.001	0.850±.001
LCM synth/sim % 80/20	0.957±.000	0.955±.004	0.040±.000	0.959±.000	0.040±.004	0.959±.000	0.955±.004	0.954±.002
LCM synth/sim % 50/50	**0.961±.000**	**0.975±.001**	0.050±.000	0.947±.000	**0.002±.001**	0.949±.000	0.975±.001	**0.960±.001**
LCM synth/sim % 20/80	0.951±.000	0.959±.002	0.050±.000	0.943±.000	0.004±.002	0.944±.000	0.959±.002	0.949±.001

Table E.10: Significance of AUC differences between training mixture variants (`AirQualityMS`). Reported are mean (± standard deviation) values across multiple runs.

Model A	**Model B**	$\text{AUC}_{\mathbf{A}}^{\mathbf{mean}}$	$\text{AUC}_{\mathbf{B}}^{\mathbf{mean}}$	$\mathbf{p\text{-value}_{raw}}$	**Significant After Correction**
LCM (100/0)	LCM (80/20)	0.886 ± 0.000	0.957 ± 0.000	$6.48e^{-03}$	Yes
LCM (100/0)	LCM (50/50)	0.886 ± 0.000	0.961 ± 0.000	$2.43e^{-03}$	Yes
LCM (100/0)	LCM (20/80)	0.886 ± 0.000	0.951 ± 0.000	$5.63e^{-03}$	Yes
LCM (80/20)	LCM (50/50)	0.957 ± 0.000	0.961 ± 0.000	$1.47e^{-01}$	No
LCM (80/20)	LCM (20/80)	0.957 ± 0.000	0.951 ± 0.000	$7.43e^{-02}$	No
LCM (50/50)	LCM (20/80)	0.961 ± 0.000	0.951 ± 0.000	$2.94e^{-01}$	No

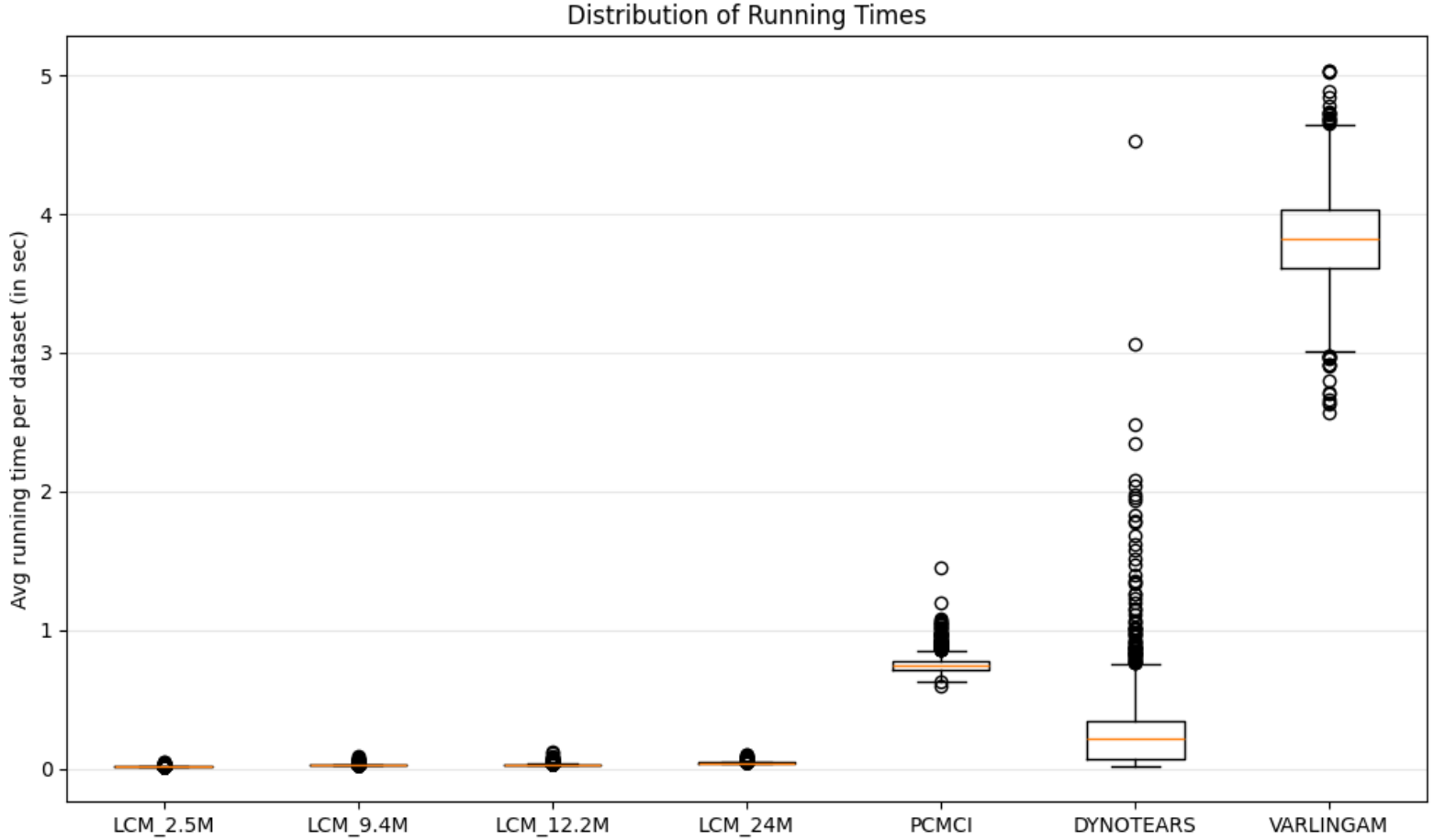


Table E.11: Running times on the `Synth_230K` data collection (in-distribution, synthetic): Mean, standard deviation, and range across model variants and baselines.

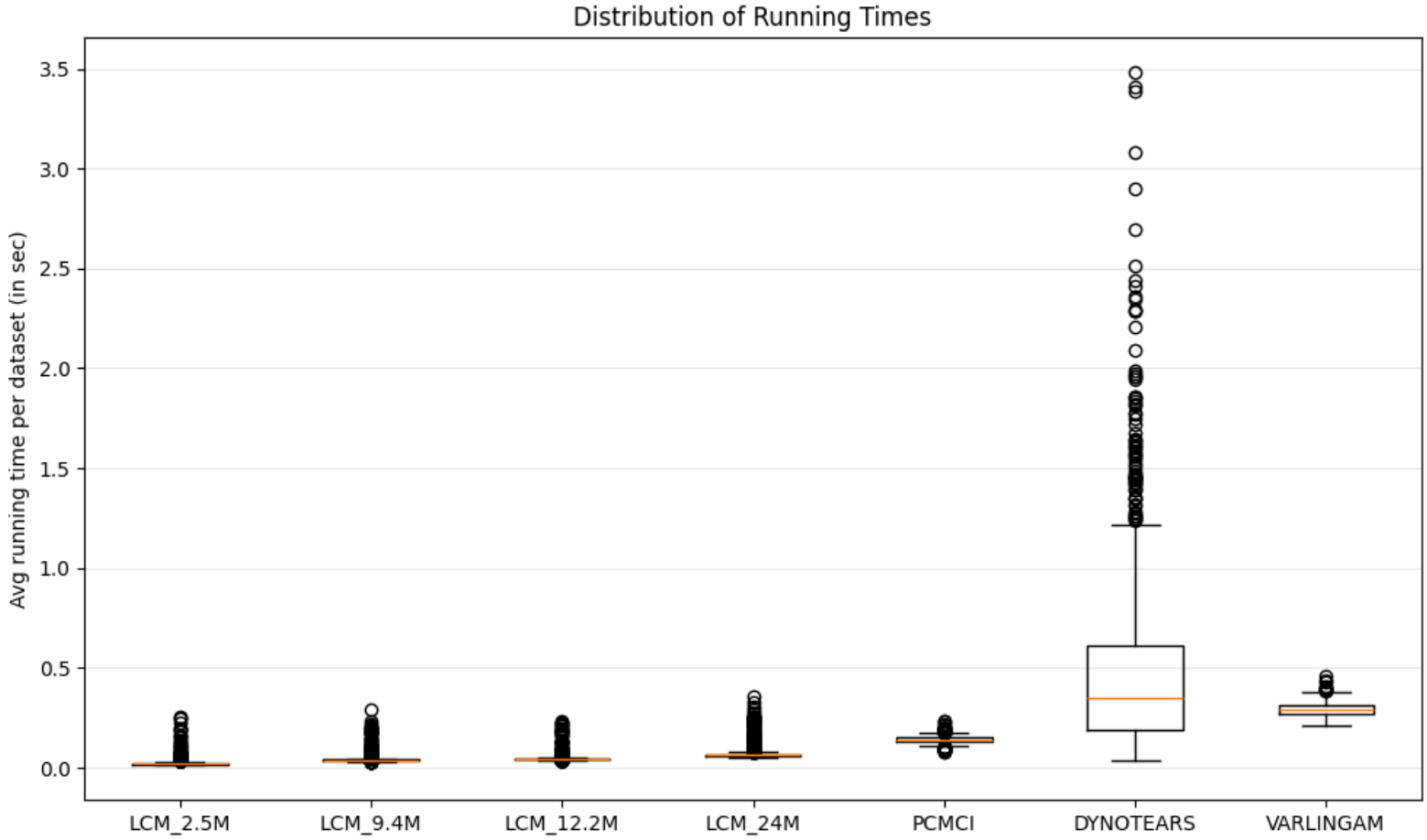


Table E.12: Running times on the `S_joint` data collection (synthetic): Mean, standard deviation, and range across model variants and classical baselines.

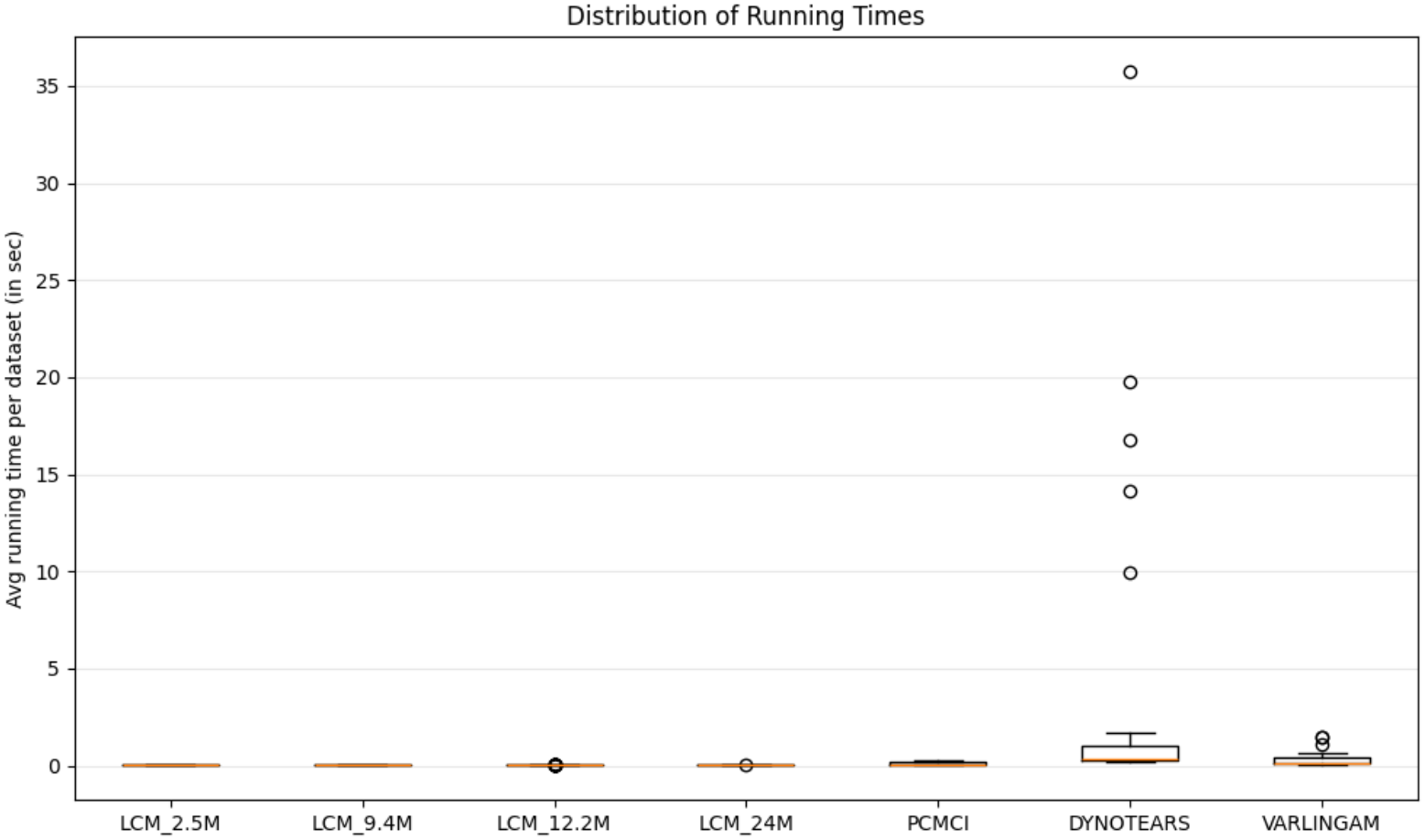


Table E.13: Running times on the fMRI data collection (semi-synthetic): Mean, standard deviation, and range across model variants and classical baselines.

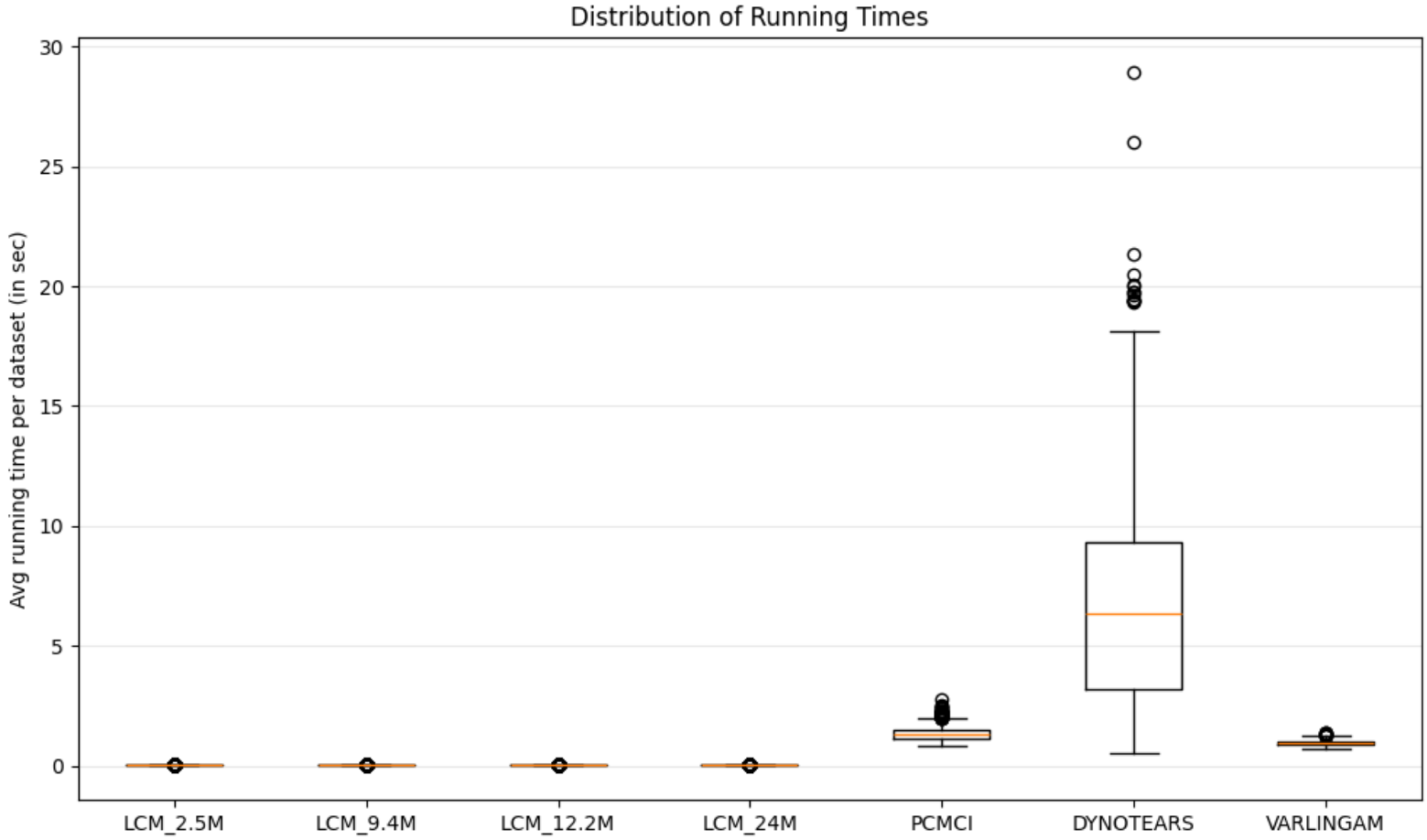


Table E.14: Running times on the `Kuramoto10` data collection (semi-synthetic): Mean, standard deviation, and range across model variants and classical baselines.

Table E.15: Large-scale results (Informer model) on the `S_Joint` data collection (synthetic). Reported are mean (± standard deviation) values across multiple runs.

Test Set: S_Joint (partially in-distribution, synthetic, holdout)

Test Data	**Lags**	**Variables**	**Func. Dep.**	Edge Prob.	# Samples	# Datasets
S_Joint	1-3	3-5	L, NL	0.2-0.8	500	2000

Parameter	**Values (4 Models)**
Max epochs	100
Patience	20
Learning rate	1e-4
d_{model}	256, 512, 512, 1024
n_{heads}	4, 4, 4, 8
n_{blocks}	4, 4, 4, 8
d_{ff}	256, 512, 1024, 1024
Dropout	0.05
$L_{max}/V_{mas}, \ell_{max}$	500 / 12 / 3
Training Aids	CI, CR
Total params	2.5M, 9.4M, 12.2M, 24M

Model	AUC_{mean}	TPR_{mean}	FPR_{mean}	TNR_{mean}	FNR_{mean}	Precision_{mean}	Recall_{mean}	F1_{mean}
LCM 2.5M	0.957 ± 0.000	0.936 ± 0.000	0.020 ± 0.000	0.978 ± 0.000	0.063 ± 0.000	0.944 ± 0.000	0.936 ± 0.000	0.936 ± 0.000
LCM 9.4M	0.962 ± 0.000	0.945 ± 0.000	0.020 ± 0.000	0.979 ± 0.000	0.054 ± 0.000	0.953 ± 0.000	0.945 ± 0.000	0.945 ± 0.000
LCM 12.2M	0.964 ± 0.000	0.951 ± 0.000	0.023 ± 0.000	0.976 ± 0.000	0.048 ± 0.000	0.951 ± 0.001	0.951 ± 0.001	0.948 ± 0.001
LCM 24M	0.936 ± 0.000	0.888 ± 0.000	0.010 ± 0.000	0.983 ± 0.000	0.111 ± 0.000	0.922 ± 0.000	0.888 ± 0.000	0.898 ± 0.000
CP (Stein et al.)	0.915 ± 0.000	0.894 ± 0.000	0.060 ± 0.000	0.935 ± 0.000	0.105 ± 0.000	0.878 ± 0.000	0.894 ± 0.000	0.882 ± 0.000
PCMCI	0.672 ± 0.000	0.685 ± 0.000	0.655 ± 0.000	0.344 ± 0.000	0.020 ± 0.000	0.427 ± 0.000	0.685 ± 0.000	0.520 ± 0.000
DYNOTEARS	0.540 ± 0.000	0.273 ± 0.000	0.191 ± 0.000	0.808 ± 0.000	0.726 ± 0.000	0.330 ± 0.000	0.273 ± 0.000	0.290 ± 0.000
VARLINGAM	0.801 ± 0.000	0.940 ± 0.000	0.330 ± 0.000	0.663 ± 0.000	0.059 ± 0.000	0.725 ± 0.000	0.940 ± 0.000	0.816 ± 0.000

Table E.16: Significance of AUC differences between large-scale LCMs (`S_Joint`). Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	**AUC_A**$_{mean}$	**AUC_B**$_{mean}$	**p-value (raw)**	**Significant after correction**
LCM 2.5M	LCM 9.4M	0.957±.103	0.962±.009	7.33e-08	Yes
LCM 2.5M	LCM 12.2M	0.957±.103	0.936±.009	0.9485	No
LCM 2.5M	LCM 24M	0.957±.103	0.934±.136	2.55e-14	Yes
LCM 2.5M	PCMCI	0.957±.103	0.947±.188	6.59e-22	Yes
LCM 2.5M	DYNOTEARS	0.957±.097	0.541±.018	1.24e-167	Yes
LCM 2.5M	VARLINGAM	0.957±.0971	0.802±.122	3.45e-166	Yes
LCM 9.4M	LCM 12.2M	0.962±.0956	0.967±.0929	1.67e-10	Yes
LCM 9.4M	LCM 24M	0.962±.0956	0.935±.136	0.0252	No
LCM 9.4M	PCMCI	0.962±.0966	0.947±.133	3.76e-19	Yes
LCM 9.4M	DYNOTEARS	0.964±.0929	0.541±.188	9.38e-18	Yes
LCM 9.4M	VARLINGAM	0.964±.0929	0.802±.121	1.82e-166	Yes
LCM 12.2M	LCM 24M	0.963±.0929	0.935±.136	0.00010	Yes
LCM 12.2M	PCMCI	0.963±.092	0.947±.133	3.93e-23	Yes
LCM 12.2M	DYNOTEARS	0.966±.088	0.541±.188	2.49e-168	Yes
LCM 12.2M	VARLINGAM	0.966±.088	0.802±.122	5.52e-169	Yes
LCM 24M	PCMCI	0.934±.136	0.946±.133	3.09e-31	Yes
LCM 24M	DYNOTEARS	0.942±.120	0.541±.188	1.17e-164	Yes
LCM 24M	VARLINGAM	0.942±.120	0.802±.121	5.25e-150	Yes
PCMCI	DYNOTEARS	0.957±.119	0.541±.188	9.60e-166	Yes
PCMCI	VARLINGAM	0.956±.119	0.802±.122	2.29e-151	Yes
DYNOTEARS	VARLINGAM	0.541±.188	0.802±.122	1.12e-138	Yes

Table E.17: Large-scale results (Informer model) on the Synth_230k data collection (in-distribution, synthetic, holdout test set). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.792±.000	0.711±.000	0.127±.000	0.872±.000	0.289±.000	0.819±.000	0.710±.000	0.740±.000
LCM 9.4M	0.789±.000	0.684±.000	0.105±.000	0.895±.000	0.316±.000	0.822±.000	0.683±.000	0.729±.000
LCM 12.2M	0.793±.000	0.693±.000	0.106±.000	0.893±.000	0.306±.000	0.830±.000	0.693±.000	0.736±.000
LCM 24M	0.771±.000	0.693±.000	0.149±.000	0.850±.000	0.306±.000	0.787±.000	0.693±.000	0.717±.000
PCMCI	0.801±.000	0.760±.000	0.402±.000	0.597±.000	0.233±.000	0.648±.000	0.760±.000	0.677±.000
DYNOTEARS	0.569±.000	0.189±.000	0.052±.000	0.947±.000	0.810±.000	0.504±.000	0.189±.000	0.254±.000
VARLINGAM	0.789±.000	0.694±.000	0.115±.000	0.886±.000	0.305±.000	0.815±.000	0.694±.000	0.733±.000

Table E.18: Significance of AUC Differences between large-scale LCMs (`Synth_230k`). Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	**AUC_A**$_{mean}$	**AUC_B**$_{mean}$	**p-value (raw)**	**Significant after correction**
LCM 2.5M	LCM 9.4M	0.792±.148	0.789±.0148	9.63e-08	Yes
LCM 2.5M	LCM 12.2M	0.792±.148	0.793±.0145	2.73e-15	Yes
LCM 2.5M	LCM 24M	0.792±.148	0.772±.147	1.69e-45	Yes
LCM 2.5M	PCMCI	0.792±.148	0.801±.181	1.02e-11	Yes
LCM 2.5M	DYNOTEARS	0.792±.148	0.567±.127	4.42e-277	Yes
LCM 2.5M	VARLINGAM	0.792±.148	0.790±.141	0.0102	No
LCM 2.5M	LCM 12.2M	0.792±.148	0.793±.146	0.0001	Yes
LCM 9.4M	LCM 24M	0.789±.148	0.772±.147	8.91e-49	Yes
LCM 9.4M	PCMCI	0.789±.148	0.801±.181	5.49e-14	Yes
LCM 9.4M	DYNOTEARS	0.789±.148	0.569±.127	5.79e-275	Yes
LCM 9.4M	VARLINGAM	0.789±.148	0.790±.141	0.0039	No
LCM 12.2M	LCM 24M	0.793±.146	0.772±.147	1.41e-55	Yes
LCM 12.2M	PCMCI	0.793±.146	0.801±.181	1.95e-11	Yes
LCM 12.2M	DYNOTEARS	0.793±.146	0.569±.127	4.30e-279	Yes
LCM 12.2M	VARLINGAM	0.793±.146	0.790±.141	4.61e-06	Yes
LCM 24M	PCMCI	0.772±.147	0.801±.181	6.38e-31	Yes
LCM 24M	DYNOTEARS	0.772±.147	0.569±.127	3.35e-263	Yes
LCM 24M	VARLINGAM	0.772±.147	0.790±.141	3.10e-13	Yes
PCMCI	DYNOTEARS	0.801±.181	0.569±.127	7.80e-237	Yes
PCMCI	VARLINGAM	0.801±.181	0.790±.141	1.53e-34	Yes
DYNOTEARS	VARLINGAM	0.569±.127	0.590±.141	1.36e-277	Yes

Table E.19: Large-scale results (Informer Model) on the `Synth_230k_Sim_45k` dataset. (in-distribution, mixed, holdout test set). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.799±.000	0.752±.000	0.155±.000	0.845±.000	0.248±.000	0.818±.000	0.752±.000	0.761±.000
LCM 9.4M	0.800±.000	0.727±.000	0.133±.000	0.866±.000	0.272±.000	0.823±.000	0.728±.000	0.752±.000
LCM 12.2M	0.800±.000	0.729±.000	0.136±.000	0.863±.000	0.271±.000	0.822±.000	0.729±.000	0.751±.000
LCM 24M	0.800±.000	0.733±.000	0.191±.000	0.809±.000	0.267±.000	0.777±.000	0.732±.000	0.733±.000
PCMCI	0.783±.000	0.768±.000	0.439±.000	0.560±.000	0.225±.000	0.636±.000	0.769±.000	0.675±.000
DYNOTEARS	0.562±.000	0.179±.000	0.053±.000	0.947±.000	0.821±.000	0.510±.000	0.179±.000	0.245±.000
VARLINGAM	0.773±.000	0.663±.000	0.116±.000	0.883±.000	0.336±.000	0.805±.000	0.663±.000	0.710±.000

Table E.20: Significance of AUC Differences between large-scale LCMs (`Synth_230k_Sim_45k`). Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	$\text{AUC}_{\text{A}}^{\text{mean}}$	$\text{AUC}_{\text{B}}^{\text{mean}}$	$\textbf{p-value}_{\text{raw}}$	**Significant After Correction**
LCM2.5M	LCM9.4M	0.781±.156	0.793±.147	0.2713	No
LCM2.5M	LCM12.2M	0.781±.156	0.795±.157	0.4798	No
LCM2.5M	LCM24M	0.781±.156	0.783±.146	0.0684	No
LCM2.5M	PCMCI	0.781±.156	0.764±.233	0.5091	No
LCM2.5M	DYNOTEARS	0.781±.156	0.558±.146	7.33e-14	Yes
LCM2.5M	VARLINGAM	0.781±.156	0.771±.172	0.5157	No
LCM9.4M	LCM24M	0.793±.147	0.783±.146	0.0045	No
LCM9.4M	PCMCI	0.793±.147	0.764±.233	0.6799	No
LCM9.4M	DYNOTEARS	0.793±.147	0.558±.146	1.38e-14	Yes
LCM9.4M	VARLINGAM	0.793±.147	0.771±.172	0.0496	No
LCM12.2M	LCM24M	0.795±.157	0.783±.146	0.0014	Yes
LCM12.2M	PCMCI	0.795±.157	0.764±.233	0.8689	No
LCM12.2M	DYNOTEARS	0.795±.157	0.558±.146	7.14e-14	Yes
LCM12.2M	VARLINGAM	0.795±.157	0.771±.172	0.0898	No
LCM24M	PCMCI	0.783±.146	0.764±.233	0.1536	No
LCM24M	DYNOTEARS	0.783±.146	0.558±.146	9.12e-16	Yes
LCM24M	VARLINGAM	0.783±.146	0.771±.172	0.8683	No
PCMCI	DYNOTEARS	0.764±.233	0.558±.146	2.03e-10	Yes
PCMCI	VARLINGAM	0.764±.233	0.771±.172	0.1237	No
DYNOTEARS	VARLINGAM	0.558±.146	0.771±.172	1.56e-14	Yes

Table E.21: Large-scale results (Informer model) on the `Sim_45k` data collection (in-distribution, simulated, holdout test set). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.834±.000	0.945±.000	0.271±.000	0.729±.000	0.050±.000	0.791±.000	0.945±.000	0.852±.000
LCM 9.4M	0.841±.000	0.951±.000	0.270±.000	0.723±.000	0.048±.000	0.793±.000	0.951±.000	0.857±.000
LCM 12.2M	0.839±.000	0.958±.000	0.280±.000	0.712±.000	0.042±.000	0.789±.000	0.958±.000	0.857±.000
LCM 24M	0.778±.000	0.935±.000	0.378±.000	0.623±.000	0.065±.000	0.721±.000	0.935±.000	0.805±.000
PCMCI	0.702±.000	0.803±.000	0.622±.000	0.377±.000	0.170±.000	0.554±.000	0.802±.000	0.646±.000
DYNOTEARS	0.542±.000	0.131±.000	0.040±.000	0.960±.000	0.864±.000	0.517±.000	0.131±.000	0.191±.000
VARLINGAM	0.688±.000	0.496±.000	0.115±.000	0.884±.000	0.499±.000	0.768±.000	0.496±.000	0.582±.000

Table E.22: Significance of AUC differences between large-scale LCMs (`Sim_45k`). Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	$\text{AUC}_{\mathbf{A}}^{\mathbf{mean}}$	$\text{AUC}_{\mathbf{B}}^{\mathbf{mean}}$	$\mathbf{p\text{-value}_{raw}}$	**Significant After Correction**
LCM2.5M	LCM9.4M	0.837±.119	0.841±.118	0.0004	Yes
LCM2.5M	LCM12.2M	0.837±.119	0.839±.117	0.8453	No
LCM2.5M	LCM24M	0.837±.119	0.770±.120	1.86e-168	Yes
LCM2.5M	PCMCI	0.837±.119	0.722±.158	3.74e-172	Yes
LCM2.5M	DYNOTEARS	0.837±.119	0.545±.092	2.83e-162	Yes
LCM2.5M	VARLINGAM	0.839±.117	0.602±.132	4.18e-132	Yes
LCM9.4M	LCM24M	0.841±.118	0.778±.120	1.28e-167	Yes
LCM9.4M	PCMCI	0.841±.118	0.722±.158	4.65e-179	Yes
LCM9.4M	DYNOTEARS	0.843±.116	0.545±.092	8.60e-163	Yes
LCM9.4M	VARLINGAM	0.843±.116	0.692±.132	2.48e-140	Yes
LCM12.2M	LCM24M	0.839±.117	0.778±.120	7.84e-172	Yes
LCM12.2M	PCMCI	0.839±.117	0.722±.158	2.36e-178	Yes
LCM12.2M	DYNOTEARS	0.841±.115	0.545±.092	1.18e-163	Yes
LCM12.2M	VARLINGAM	0.841±.115	0.692±.132	1.75e-138	Yes
LCM24M	PCMCI	0.770±.012	0.722±.158	5.19e-49	Yes
LCM24M	DYNOTEARS	0.781±.016	0.545±.092	9.71e-160	Yes
LCM24M	VARLINGAM	0.781±.116	0.692±.132	2.57e-78	Yes
PCMCI	DYNOTEARS	0.722±.016	0.543±.099	3.05e-136	Yes
PCMCI	VARLINGAM	0.722±.016	0.688±.141	1.55e-28	Yes
DYNOTEARS	VARLINGAM	0.543±.099	0.688±.141	6.25e-131	Yes

Table E.23: Large-scale results (Informer model) on the `Sim_45k` data collection (in-distribution, simulated, holdout test set). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.945±.000	1.000±.000	0.110±.001	0.891±.001	0.000±.000	0.904±.002	1.000±.000	0.949±.000
LCM 9.4M	0.943±.001	1.000±.000	0.113±.002	0.887±.002	0.000±.000	0.901±.002	1.000±.000	0.947±.000
LCM 12.2M	0.946±.000	1.000±.000	0.103±.001	0.897±.001	0.003±.000	0.901±.010	0.997±.000	0.949±.000
LCM 24M	0.954±.000	0.997±.000	0.084±.001	0.916±.001	0.008±.000	0.925±.001	0.992±.000	0.956±.000
CP (Stein et al.)	0.775±.001	0.681±.004	0.129±.001	0.870±.001	0.318±.004	0.839±.001	0.682±.004	0.742±.002
PCMCI	0.739±.005	0.994±.001	0.982±.003	0.018±.003	0.005±.000	0.504±.000	0.994±.000	0.668±.000
DYNOTEARS	0.522±.011	0.349±.031	0.305±.012	0.695±.012	0.651±.031	0.373±.022	0.349±.031	0.344±.030
VARLINGAM	0.687±.005	0.612±.009	0.243±.003	0.757±.003	0.383±.009	0.701±.004	0.617±.009	0.653±.007

Table E.24: Large-scale results (Informer model) on the `Kuramoto5` data collection (semi-synthetic, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.913±.000	0.997±.000	0.171±.000	0.829±.000	0.003±.000	0.855±.000	0.997±.000	0.920±.000
LCM 9.4M	0.919±.000	0.997±.000	0.161±.000	0.839±.000	0.001±.000	0.863±.000	0.999±.000	0.925±.000
LCM 12.2M	0.922±.000	0.999±.000	0.153±.000	0.847±.000	0.002±.000	0.868±.000	0.997±.000	0.928±.000
LCM 24M	0.932±.000	0.997±.000	0.060±.000	0.939±.000	0.076±.000	0.939±.000	0.923±.000	0.928±.000
CP (Stein et al.)	0.518±.000	0.235±.000	0.199±.000	0.801±.000	0.767±.000	0.488±.000	0.235±.000	0.312±.000
PCMCI	0.466±.000	0.981±.000	0.966±.000	0.034±.000	0.019±.000	0.504±.000	0.981±.000	0.665±.000
DYNOTEARS	0.499±.000	0.007±.000	0.008±.000	0.992±.000	0.993±.000	0.067±.000	0.007±.000	0.013±.000
VARLINGAM	0.504±.000	0.653±.000	0.645±.000	0.355±.000	0.347±.000	0.499±.000	0.653±.000	0.561±.000

Table E.25: Significance of AUC differences between large-scale LCMs (`Kuramoto5`). Reported are mean (± standard deviation) values across multiple runs.

Model A	**Model B**	AUC_A	AUC_B	**p-value**	**Significant after correction**
CP trf	LCM 2.5M	0.518±.078	0.913±.030	3.33e-165	Yes
CP trf	LCM 9.4M	0.518±.078	0.919±.029	3.33e-165	Yes
CP trf	LCM 12.2M	0.518±.078	0.923±.017	3.33e-165	Yes
CP trf	LCM 24M	0.518±.078	0.931±.046	3.33e-165	Yes
CP trf	PCMCI	0.518±.078	0.466±.103	1.35e-34	Yes
CP trf	DYNOTEARS	0.518±.078	0.499±.018	8.25e-10	Yes
CP trf	VARLINGAM	0.518±.078	0.504±.096	0.0007	Yes
LCM 2.5M	LCM 9.4M	0.913±.025	0.919±.030	1.95e-19	Yes
LCM 2.5M	LCM 12.2M	0.913±.025	0.923±.018	5.59e-46	Yes
LCM 2.5M	LCM 24M	0.913±.025	0.931±.046	1.70e-27	Yes
LCM 2.5M	PCMCI	0.913±.025	0.466±.103	3.33e-165	Yes
LCM 2.5M	DYNOTEARS	0.913±.025	0.499±.018	3.32e-165	Yes
LCM 2.5M	VARLINGAM	0.913±.025	0.504±.096	3.33e-165	Yes
LCM 9.4M	LCM 12.2M	0.919±.029	0.923±.018	0.0007	Yes
LCM 9.4M	LCM 24M	0.919±.029	0.931±.046	2.64e-14	Yes
LCM 9.4M	PCMCI	0.919±.029	0.466±.103	3.33e-165	Yes
LCM 9.4M	DYNOTEARS	0.919±.029	0.499±.018	3.32e-165	Yes
LCM 9.4M	VARLINGAM	0.919±.029	0.504±.096	3.33e-165	Yes
LCM 12.2M	LCM 24M	0.923±.018	0.931±.046	1.77e-13	Yes
LCM 12.2M	PCMCI	0.923±.018	0.466±.103	3.33e-165	Yes
LCM 12.2M	DYNOTEARS	0.923±.018	0.499±.018	3.32e-165	Yes
LCM 12.2M	VARLINGAM	0.923±.018	0.504±.096	3.33e-165	Yes
LCM 24M	PCMCI	0.931±.046	0.466±.103	3.33e-165	Yes
LCM 24M	DYNOTEARS	0.931±.046	0.499±.018	3.32e-165	Yes
LCM 24M	VARLINGAM	0.931±.046	0.504±.096	3.33e-165	Yes
PCMCI	DYNOTEARS	0.466±.103	0.499±.018	1.23e-21	Yes
PCMCI	VARLINGAM	0.466±.103	0.504±.096	8.89e-20	Yes
DYNOTEARS	VARLINGAM	0.499±.018	0.504±.096	0.171	No

Table E.26: Large-scale results (Informer model) on the `Kuramoto10` data collection (semi-synthetic, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.896±.000	0.950±.000	0.158±.000	0.842±.000	0.050±.000	0.858±.000	0.950±.000	0.901±.000
LCM 9.4M	0.926±.000	0.999±.000	0.147±.000	0.853±.000	0.001±.000	0.872±.000	0.999±.000	0.931±.000
LCM 12.2M	0.902±.000	0.992±.000	0.188±.000	0.812±.000	0.008±.000	0.844±.000	0.992±.000	0.911±.000
LCM 24M	0.913±.000	0.979±.000	0.153±.000	0.847±.000	0.022±.000	0.865±.000	0.978±.000	0.917±.000
PCMCI	0.640±.000	0.977±.000	0.950±.000	0.050±.000	0.233±.000	0.507±.000	0.977±.000	0.667±.000
DYNOTEARS	0.503±.000	0.019±.000	0.014±.000	0.986±.000	0.981±.000	0.370±.000	0.193±.000	0.036±.000
VARLINGAM	0.584±.000	0.618±.000	0.449±.000	0.551±.000	0.382±.000	0.578±.000	0.618±.000	0.591±.000

Table E.27: Significance of AUC differences between large-scale LCMs (`Kuramoto10`). Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	AUC_A	AUC_B	**p-value**	**Significant after correction**
LCM 2.5M	LCM 9.4M	0.896±.038	0.926±.014	1.17e-129	Yes
LCM 2.5M	LCM 12.2M	0.896±.038	0.902±.040	5.82e-10	Yes
LCM 2.5M	LCM 24M	0.896±.038	0.913±.024	1.58e-29	Yes
LCM 2.5M	PCMCI	0.896±.038	0.640±.046	3.34e-165	Yes
LCM 2.5M	DYNOTEARS	0.896±.038	0.503±.010	3.33e-165	Yes
LCM 2.5M	VARLINGAM	0.896±.038	0.584±.045	3.33e-165	Yes
LCM 9.4M	LCM 12.2M	0.926±.014	0.902±.040	9.65e-115	Yes
LCM 9.4M	LCM 24M	0.926±.014	0.913±.024	7.37e-95	Yes
LCM 9.4M	PCMCI	0.926±.014	0.640±.046	3.33e-165	Yes
LCM 9.4M	DYNOTEARS	0.926±.014	0.503±.010	3.32e-165	Yes
LCM 9.4M	VARLINGAM	0.926±.014	0.584±.045	3.33e-165	Yes
LCM 12.2M	LCM 24M	0.902±.040	0.913±.024	1.05e-06	Yes
LCM 12.2M	PCMCI	0.902±.040	0.640±.046	3.33e-165	Yes
LCM 12.2M	DYNOTEARS	0.902±.040	0.503±.010	3.33e-165	Yes
LCM 12.2M	VARLINGAM	0.902±.040	0.584±.045	3.33e-165	Yes
LCM 24M	PCMCI	0.913±.024	0.640±.046	3.33e-165	Yes
LCM 24M	DYNOTEARS	0.913±.024	0.503±.010	3.33e-165	Yes
LCM 24M	VARLINGAM	0.9128±.024	0.584±.045	3.33e-165	Yes
PCMCI	DYNOTEARS	0.6404±.046	0.503±.010	3.59e-165	Yes
PCMCI	VARLINGAM	0.6404±.046	0.584±.045	7.26e-110	Yes
DYNOTEARS	VARLINGAM	0.5025±.0104	0.584±.045	1.28e-159	Yes

Table E.28: Large-scale results (Informer model) on the `AirQualityMS` data collection (simulated, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs.

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 2.5M	0.955±.000	0.974±.001	0.060±.000	0.937±.000	0.030±.001	0.940±.000	0.974±.001	0.955±.000
LCM 9.4M	0.914±.001	0.872±.002	0.045±.000	0.955±.000	0.130±.003	0.952±.000	0.872±.000	0.898±.001
LCM 12.2M	0.914±.001	0.890±.002	0.060±.000	0.938±.000	0.110±.002	0.928±.004	0.890±.002	0.898±.002
LCM 24M	0.813±.006	0.693±.013	0.060±.000	0.933±.000	0.310±.012	0.800±.010	0.694±.012	0.710±.002
PCMCI	0.556±.000	0.913±.000	0.901±.000	0.098±.000	0.070±.000	0.517±.000	0.913±.000	0.639±.000
DYNOTEARS	0.694±.000	0.482±.000	0.482±.000	0.910±.000	0.520±.000	0.835±.000	0.482±.000	0.544±.000
VARLINGAM	0.552±.000	0.552±.000	0.552±.000	0.551±.000	0.448±.000	0.496±.000	0.552±.000	0.494±.000

Table E.29: Significance of AUC differences between large-scale LCMs (`AirQualityMS`). Reported are mean AUCs (± standard deviation), raw p-values, and significance after correction.

Model A	**Model B**	AUC_A	AUC_B	**p-value**	**Significant after correction**
LCM 2.5M	LCM 9.4M	0.954±.034	0.910±.094	0.15025	No
LCM 2.5M	LCM 12.2M	0.954±.034	0.913±.094	0.00723	No
LCM 2.5M	LCM 24M	0.954±.034	0.829±.175	1.29e-05	Yes
LCM 2.5M	PCMCI	0.954±.034	0.557±.215	1.63e-10	Yes
LCM 2.5M	DYNOTEARS	0.954±.034	0.695±.232	8.61e-07	Yes
LCM 2.5M	VARLINGAM	0.952±.036	0.552±.176	5.68e-14	Yes
LCM 9.4M	LCM 12.2M	0.910±.094	0.913±.094	0.76842	No
LCM 9.4M	LCM 24M	0.910±.094	0.829±.175	0.04917	No
LCM 9.4M	PCMCI	0.910±.094	0.5567±.215	1.72e-10	Yes
LCM 9.4M	DYNOTEARS	0.910±.094	0.695±.232	8.61e-07	Yes
LCM 9.4M	VARLINGAM	0.898±.098	0.552±.176	5.18e-09	Yes
LCM 12.2M	LCM 24M	0.913±.094	0.829±.175	0.0089	No
LCM 12.2M	PCMCI	0.913±.094	0.557±.215	1.72e-10	Yes
LCM 12.2M	DYNOTEARS	0.913±.094	0.695±.232	3.54e-06	Yes
LCM 12.2M	VARLINGAM	0.913±.076	0.552±.176	5.18e-09	Yes
LCM 24M	PCMCI	0.829±.175	0.557±.215	9.54e-09	Yes
LCM 24M	DYNOTEARS	0.829±.175	0.695±.232	0.00327	No
LCM 24M	VARLINGAM	0.823±.167	0.552±.176	1.79e-07	Yes
PCMCI	DYNOTEARS	0.557±.215	0.695±.232	0.03955	No
PCMCI	VARLINGAM	0.552±.194	0.552±.176	0.94224	No
DYNOTEARS	VARLINGAM	0.640±.212	0.552±.176	0.10671	No

Table E.30: Large-scale results (PatchTSTSpacetimeformer model) on the `S_Joint` data collection (synthetic). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**# Samples**	**# Datasets**
S_Joint	1–3	3–5	L, NL	500	2000

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.962±.000	0.945±.000	0.020±.000	0.979±.000	0.054±.000	0.953±.000	0.945±.000	0.945±.000
LCM 9.1M (updated arch.)	0.954±.000	0.925±.000	0.016±.000	0.983±.000	0.074±.001	0.943±.001	0.925±.000	0.929±.001

Pairwise AUC Comparison

Model A	**Model B**	$\text{AUC}_{\textbf{A}}$	$\text{AUC}_{\textbf{B}}$	**p-value**	**Significant after correction**
LCM 9.4M	LCM 9.1M (updated arch.)	0.962±.000	0.954±.000	5.8e–24	Yes

Table E.31: Large-scale results (PatchTSTSpacetimeformer model) on the `Synth_230k` data collection (in-distribution, synthetic, holdout test set). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**# Samples**	**# Datasets**
Synth_230K	1–3	3–12	L, NL	500	2000

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.789±.000	0.684±.000	0.105±.000	0.895±.000	0.316±.000	0.823±.000	0.684±.000	0.728±.000
LCM 9.1M (updated arch.)	0.805±.000	0.710±.000	0.100±.000	0.900±.000	0.290±.000	0.843±.000	0.710±.000	0.751±.000

Pairwise AUC Comparison

Model A	**Model B**	$\text{AUC}_{\textbf{A}}$	$\text{AUC}_{\textbf{B}}$	**p-value**	**Significant after correction**
LCM 9.4M	LCM 9.1M (updated arch.)	0.789±.000	0.805±.000	6.29e–16	Yes

Table E.32: Large-scale results (PatchTSTSpacetimeformer model) on the `Synth_230K_Sim_45K` data collection (in-distribution, mixed, holdout). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**# Samples**	**# Datasets**
Synth_230K_Sim_45K	1–3	3–12	L, NL	500	5500

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.797±.000	0.727±.000	0.134±.000	0.866±.000	0.272±.000	0.823±.000	0.723±.000	0.752±.000
LCM 9.1M (updated arch.)	0.810±.000	0.750±.000	0.130±.000	0.870±.000	0.250±.000	0.835±.000	0.750±.000	0.770±.000

Pairwise AUC Comparison

Model A	**Model B**	$\text{AUC}_{\textbf{A}}$	$\text{AUC}_{\textbf{B}}$	**p-value**	**Significant after correction**
LCM 9.4M	LCM 9.1M (updated arch.)	0.793±.000	0.799±.000	0.0146	Yes

Table E.33: Large-scale results (PatchTSTSpacetimeformer model) on the `Sim_45K` data collection (in-distribution, simulated, holdout). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**# Samples**	**# Datasets**
Sim_45K	1–3	3–12	L, NL	500	2000

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.840±.000	0.951±.000	0.270±.000	0.723±.000	0.049±.000	0.793±.000	0.951±.000	0.857±.000
LCM 9.1M (updated arch.)	0.849±.000	0.959±.000	0.262±.000	0.738±.000	0.041±.000	0.800±.000	0.959±.000	0.865±.000

Pairwise AUC Comparison

Model A	**Model B**	$\text{AUC}_{\textbf{A}}$	$\text{AUC}_{\textbf{B}}$	**p-value**	**Significant after correction**
LCM 9.4M	LCM 9.1M (updated arch.)	0.841±.000	0.849±.000	1.06e–13	Yes

Table E.34: Large-scale results (PatchTSTSpacetimeformer model) on the *f*MRI5 data collection (semi-synthetic, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Variables**	**Func. Dep.**	**Edge Prob.**	**# Samples**	**# Datasets**
*f*MRI5	5	NL	–	[200–2400]	21

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.944±.001	1.000±.000	0.113±.002	0.887±.002	0.000±.000	0.900±.001	1.000±.000	0.945±.000
LCM 9.1M (updated arch.)	0.960±.000	0.991±.001	0.072±.001	0.928±.001	0.008±.001	0.934±.001	0.991±.001	0.961±.000

Table E.35: Large-scale results (PatchTSTSpacetimeformer model) on the fMRI data collection (semi-synthetic, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**Edge Prob.**	**# Samples**	**# Datasets**
fMRI	1	5, 10	NL	–	[200–2400]	26

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

	Performance Metrics							
Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.934±.000	1.000±.000	0.131±.001	0.869±.002	0.000±.000	0.887±.001	1.000±.000	0.939±.000
LCM 9.1M (updated arch.)	0.950±.000	0.994±.000	0.098±.001	0.902±.001	0.006±.000	0.912±.001	0.994±.000	0.951±.000

Table E.36: Large-scale results (PatchTSTSpacetimeformer model) on the `Kuramoto5` collection (semi-synthetic, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**# Samples**	**# Datasets**
Kuramoto5	1	5	NL	500	1000

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.919±.000	0.999±.000	0.161±.000	0.839±.000	0.015±.000	0.863±.000	0.999±.000	0.925±.000
LCM 9.1M (updated arch.)	0.925±.000	0.952±.001	0.102±.000	0.898±.000	0.049±.001	0.908±.000	0.951±.001	0.924±.000

Pairwise AUC Comparison

Model A	**Model B**	$\text{AUC}_{\textbf{A}}$	$\text{AUC}_{\textbf{B}}$	**p-value**	**Significant after correction**
LCM 9.4M	9.1M	0.919±.000	0.925±.000	1.9e–07	Yes

Table E.37: Large-scale results (PatchTSTSpacetimeformer model) on the `Kuramoto10` collection (semi-synthetic, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	**Lags**	**Variables**	**Func. Dep.**	**# Samples**	**# Datasets**
Kuramoto10	1	5, 10	NL	500	1000

Parameter	**Value**
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	$\text{AUC}_{\textbf{mean}}$	$\text{TPR}_{\textbf{mean}}$	$\text{FPR}_{\textbf{mean}}$	$\text{TNR}_{\textbf{mean}}$	$\text{FNR}_{\textbf{mean}}$	$\text{Precision}_{\textbf{mean}}$	$\text{Recall}_{\textbf{mean}}$	$\text{F1}_{\textbf{mean}}$
LCM 9.4M	0.926±.000	0.999±.000	0.147±.000	0.853±.000	0.015±.000	0.872±.000	0.999±.000	0.931±.000
LCM 9.1M (updated arch.)	0.894±.000	0.998±.000	0.211±.000	0.788±.000	0.017±.000	0.826±.000	0.999±.000	0.904±.000

Pairwise AUC Comparison

Model A	**Model B**	$\text{AUC}_{\textbf{A}}$	$\text{AUC}_{\textbf{B}}$	**p-value**	**Significant after correction**
LCM 9.4M	LCM 9.1M (updated arch.)	0.926±.000	0.894±.000	4.06e–139	Yes

Table E.38: Large-scale results (PatchTSTSpacetimeformer model) on the `AirQualityMS` collection (simulated, out-of-distribution). Reported are mean (± standard deviation) values across multiple runs, raw p-values, and significance after correction.

Test Data	Lags	Variables	# Samples	# Datasets
AirQualityMS	1–3	3–12	500	54

Parameter	Value
Max epochs	100
Patience	**20**
Learning rate lr	1e-4
d_{model}	512 / 256
n_{heads}	4 / 4
n_{blocks}	4 / 6
d_{ff}	512 / 512
Dropout	**0.05**
L_{max}	500
Training Aids	CI, CR
Total params	9.4M / 9.1M

Performance Metrics

Model	AUC_{mean}	TPR_{mean}	FPR_{mean}	TNR_{mean}	FNR_{mean}	$Precision_{mean}$	$Recall_{mean}$	$F1_{mean}$
LCM 9.4M	0.914±.001	0.872±.002	0.045±.000	0.955±.000	0.128±.003	0.952±.000	0.872±.003	0.898±.002
LCM 9.1M (uprated arch.)	0.903±.002	0.865±.004	0.059±.000	0.940±.000	0.135±.005	0.934±.003	0.865±.004	0.879±.003

Pairwise AUC Comparison

Model A	Model B	AUC_A	AUC_B	p-value	Significant after correction
LCM 9.4M	LCM 9.1M (uprated arch.)	0.910±.000	0.898±.000	0.0359	Yes

This page intentionally left blank

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

Charles K Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv Preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

Konstantina Biza, Ioannis Tsamardinos, and Sofia Triantafillou. Out-of-sample tuning for causal discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):4963–4973, 2022.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

Leo Breiman. Random forests. *Machine Learning*, 45, 2001.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Richard B Buxton, Eric C Wong, and Lawrence R Frank. Dynamics of blood flow and oxygenation changes during brain activation: The balloon model. *Magnetic resonance in medicine*, 39(6):855–864, 1998.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in Neural Information Processing Systems*, 36:49205–49233, 2023.

Yuxiao Cheng, Ziqian Wang, Tingxiong Xiao, Qin Zhong, Jinli Suo, and Kunlun He. Causaltime: Realistically generated time-series for benchmarking of causal discovery. In *International Conference on Learning Representations*, 2024.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Martina Cinquini, Fosca Giannotti, and Riccardo Guidotti. Boosting synthetic data generation with effective nonlinear causal discovery. In *2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI)*, pages 54–63. IEEE, 2021.

Corinna Cortes and Vladimir Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

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

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. TimeVAE: A variational autoencoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a), 1979.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using RealNVP. *International Conference on Learning Representations*, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 2019.

Stephen E Fienberg. A brief history of statistical models for network analysis and open challenges. *Journal of Computational and Graphical Statistics*, 21 (4):825–839, 2012.

Ronald A Fisher. *The Design of Experiments*. Oliver and Boyd, 1935.

Juan L Gamella, Jonas Peters, and Peter Bühlmann. The causal chambers: Real physical systems as a testbed for AI methodology. *arXiv preprint arXiv:2404.11341*, 2024.

Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, et al. Deep end-to-end causal inference. *Transactions on Machine Learning Research*, 2024.

Sílvia Gonçalves and Dimitris Politis. Discussion: Bootstrap methods for dependent data: A review. *Journal of the Korean Statistical Society*, 40(4), 2011.

Chang Gong, Chuzhe Zhang, Di Yao, Jingping Bi, Wenbin Li, and Yongjun Xu. Causal discovery from temporal data: An overview and new perspectives. *ACM Computing Surveys*, 57(4):1–38, 2024.

Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. Rhino: Deep causal temporal relationship learning with history-dependent noise. *arXiv preprint arXiv:2210.14706*, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep Learning*, volume 1. MIT Press, 2016.

Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michele Sebag. Learning functional causal models with generative neural networks. *Explainable and Interpretable Models in Computer Vision and Machine Learning*, 2018.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in Neural Information Processing Systems*, 19, 2006.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

Riccardo Guidotti. Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery*, 38 (5):2770–2824, 2024.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Yannik Hahn, Tristan Langer, Richard Meyes, and Tobias Meisen. Time series dataset survey for forecasting with deep learning. *Forecasting*, 5(1), 2023.

Wolfgang Härdle, Joel Horowitz, and Jens-Peter Kreiss. Bootstrap methods for time series. *International Statistical Review*, 71(2), 2003.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international Conference on Computer Vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79 (8):2554–2558, 1982.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in Neural Information Processing Systems*, 32, 2019.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature, 2019.

Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O Hoyer. Estimation of a structural vector autoregression model sing non-gaussianity. *Journal of Machine Learning Research*, 11(5), 2010.

Luka Jakovljevic, Dimitre Kostadinov, Armen Aghasaryan, and Themis Palpanas. Towards building a digital twin of complex system using causal modelling. In *International Conference on Complex Networks and Their Applications*, pages 475–486. Springer, 2021.

Maowei Jiang, Pengyu Zeng, Kai Wang, Huan Liu, Wenbo Chen, and Haoran Liu. Fecam: Frequency enhanced channel attention mechanism for time series forecasting. *Advanced Engineering Informatics*, 58, 2023.

Simi Job, Xiaohui Tao, Taotao Cai, Haoran Xie, Lin Li, Qing Li, and Jianming Yong. Exploring causal learning through graph neural networks: An in-depth review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 15(2):e70024, 2025.

Nan Rosemary Ke, Silvia Chiappa, Jane Wang, Anirudh Goyal, Jorg Bornschein, Melanie Rey, Theophane Weber, Matthew Botvinic, Michael Mozer, and Danilo Jimenez Rezende. Learning to induce causal structure. *arXiv preprint arXiv:2204.04875*, 2022.

Nan Rosemary Ke, Silvia Chiappa, Jane X Wang, Jorg Bornschein, Anirudh Goyal, Melanie Rey, Theophane Weber, Matthew Botvinick, Michael Curtis Mozer, and Danilo Jimenez Rezende. Learning to induce causal structure. In *International Conference on Learning Representations*, 2023.

Jongseon Kim, Hyungjoon Kim, HyunGi Kim, Dongjun Lee, and Sungroh Yoon. A comprehensive survey of deep learning for time series forecasting: Architectural diversity and open challenges. *Artificial Intelligence Review*, 58(7): 1–95, 2025.

Diederik P Kingma. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.

Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In *International Symposium on Mathematical Problems in Theoretical Physics: Kyoto University, Kyoto/Japan*. Springer, 1975.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

Abigail Langbridge, Fearghal O'Donncha, Amadou Ba, Fabio Lorenzi, Christopher Lohse, and Joern Ploennigs. Causal temporal graph convolutional neural networks. *arXiv preprint arXiv:2303.09634*, 2023.

Andrew R. Lawrence, Marcus Kaiser, Rui Sampaio, and Maksim Sipos. Data generating process to evaluate causal discovery techniques for time series data. *Causal Discovery & Causality-Inspired Machine Learning Workshop at Neural Information Processing Systems*, 2020.

Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for backpropagation. In *Proceedings of the 1988 Connectionist Models Summer School*, volume 1, pages 21–28, 1988.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Transformers for time series analysis at scale. *arXiv CoRR*, 2024.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017.

Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35:13104–13118, 2022.

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5(5):5, 2017.

Sindy Löwe, David Madras, Richard Zemel, and Max Welling. Amortized causal discovery: Learning to infer causal graphs from time-series data. In *Conference on Causal Learning and Reasoning*, pages 509–525. PMLR, 2022.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Marvin Minsky and Seymour Papert. Perceptrons. *MIT Press*, 1969.

Kevin P Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

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

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *International Conference on Learning Representations*, 2023.

Wenjin Niu, Zijun Gao, Liyan Song, and Lingbo Li. Comprehensive review and empirical evaluation of causal discovery algorithms for numerical data. *arXiv preprint arXiv:2407.13054*, 2024.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. PMLR, 2020.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE DSAA*. IEEE, 2016.

J Pearl, F Bacchus, P Spirtes, C Glymour, and R Scheines. Probabilistic reasoning in intelligent systems: Networks of plausible inference. *Synthese*, 104 (1), 1988.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

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

Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

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

Bharadwaj Pudipeddi, Maral Mesmakhosroshahi, Jinwen Xi, and Sujeeth Bharadwaj. Training large neural networks with constant memory uing a new execution algorithm. *arXiv preprint arXiv:2002.05645*, 2020.

Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. Synthcity: Facilitating innovative use cases of synthetic data in different data modalities, 2023. URL `https://arxiv.org/abs/2301.07573`.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Hans Reichenbach. *The Direction of Time*. University of California Press, 1956.

Thomas Richardson and Peter Spirtes. Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. Do-pfn: In-context learning for causal effect estimation. *arXiv preprint arXiv:2506.06039*, 2025.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Jakob Runge. Causal network reconstruction from time series: From theoretical assumptions to practical estimation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(7), 2018.

Jakob Runge, Sebastian Bathiany, Erik Bollt, Gustau Camps-Valls, Dim Coumou, Ethan Deyle, Clark Glymour, Marlene Kretschmer, Miguel D Mahecha, Jordi Muñoz-Marí, et al. Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):2553, 2019.

Jakob Runge, Andreas Gerhardus, Gherardo Varando, Veronika Eyring, and Gustau Camps-Valls. Causal inference for time series. *Nature Reviews Earth & Environment*, 4(7):487–505, 2023.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Towards causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Edward H Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13, 1951.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

Stephen M Smith, Karla L Miller, Gholamreza Salimi-Khorshidi, Matthew Webster, Christian F Beckmann, Thomas E Nichols, Joseph D Ramsey, and Mark W Woolrich. Network modelling methods for fmri. *Neuroimage*, 54 (2), 2011.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4): 427–437, 2009.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT press, 2001.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Gideon Stein, Maha Shadaydeh, and Joachim Denzler. Embracing the black box: Heading towards foundation models for causal discovery from time series data. *AAAI Workshop (AI4TS)*, 2024.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.

Tijmen Tieleman. Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural networks for machine learning*, 4(2):26, 2012.

Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12:1, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Bingyang Wen, Luis Oliveros Colon, KP Subbalakshmi, and Rajarathnam Chandramouli. Causal-TGAN: Generating tabular data using causal generative adversarial networks. *arXiv preprint arXiv:2104.10680*, 2021.

Paul J Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 2002.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. 2024.

Anpeng Wu, Kun Kuang, Minqin Zhu, Yingrong Wang, Yujia Zheng, Kairong Han, Baohong Li, Guangyi Chen, Fei Wu, and Kun Zhang. Causality for large language models. *CoRR*, 2024a.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

Menghua Wu, Yujia Bao, Regina Barzilay, and Tommi Jaakkola. Sample, estimate, aggregate: A recipe for causal discovery foundation models. In *ICLR Workshop on Machine Learning for Genomics Explorations*, 2024b.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9593–9602, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in Neural Information Processing Systems*, 30, 2017.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

Jiaqi Zhang, Joel Jennings, Agrin Hilmkil, Nick Pawlowski, Cheng Zhang, and Chao Ma. Towards causal foundation model: On duality between causal inference and attention. *arXiv preprint arXiv:2310.00809*, 2023.

Kevin Zhang, Neha Patki, and Kalyan Veeramachaneni. Sequential models in the synthetic data vault. *arXiv preprint arXiv:2207.14406*, 2022.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.