

Embracing the black box: Heading towards foundation models for causal discovery from time series data

Gideon Stein, Maha Shadaydeh, Joachim Denzler

Computer Vision Group, Friedrich Schiller University Jena
gideon.stein@uni-jena.de, maha.shadaydeh@uni-jena.de, joachim.denzler@uni-jena.de

Abstract

Causal discovery from time series data encompasses many existing solutions, including those based on deep learning techniques. However, these methods typically do not endorse one of the most prevalent paradigms in deep learning: End-to-end learning. To address this gap, we explore what we call Causal Pretraining. A methodology that aims to learn a direct mapping from multivariate time series to the underlying causal graphs in a supervised manner. Our empirical findings suggest that causal discovery in a supervised manner is possible, assuming that the training and test time series samples share most of their dynamics. More importantly, we found evidence that the performance of Causal Pretraining can increase with data and model size, even if the additional data do not share the same dynamics. Further, we provide examples where causal discovery for real-world data with causally pretrained neural networks is possible within limits. We argue that this hints at the possibility of a foundation model for causal discovery.

Introduction

The investigation of causal discovery from time series data encompasses many existing solutions, including, of course, the integration of deep learning techniques. Despite the widespread utilization of these techniques, many methods do not endorse one of the most prevalent paradigms in deep learning: End-to-end learning. This paradigm suggests that it is often beneficial to only provide neural networks with raw input data instead of handcrafting features. Additionally, making assumptions on how a particular task should be solved can hinder performance (Bojarski et al. 2016), (Amodei et al. 2016), (Glamachers 2017). Contrary to that, neural networks are typically embedded into well-established causal discovery frameworks such as Granger causality, (Tank et al. 2018), (Ahmad, Shadaydeh, and Denzler 2022), (Teodora Trifunov, Shadaydeh, and Denzler 2022), (Löwe et al. 2022) or score-based methods (Zheng et al. 2018), (Ng et al. 2019). Through this, the solution space is naturally restricted. To address this disparity, we explore what we call **Causal Pretraining**, a methodology that aims at learning a direct mapping from multivariate time series to causal graphs as depicted in Figure 1. We propose to parameterize this mapping by a deep neural network and train it in a supervised manner. In the training

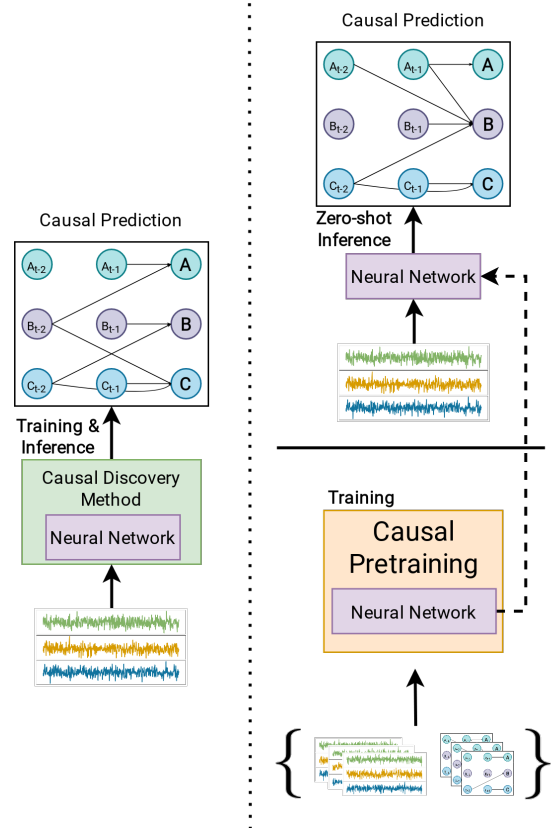


Figure 1: Comparison between a general depiction of causal discovery methods (left) and our Causal Pretraining methodology (right). Instead of inferring causal graphs from data directly, Causal Pretraining produces neural networks that can be deployed for inference directly.

phase, we sample from a distribution of synthetic time series with known corresponding causal graphs. Besides leaving the solution space completely open, Causal Pretraining has another distinct advantage over many other deep-learning approaches. Since it does not require fitting any parameters during inference and natively allows for effective parallel processing, it makes the analysis of large sets of time series exceptionally efficient. While this idea is, to

our surprise, largely unexplored, we determined that substantial challenges are stabilizing training and encouraging behavior beyond learning correlational patterns. Therefore, we conducted experiments involving training multiple deep-learning architecture archetypes with differing training techniques on increasingly more complex synthetic datasets. We specifically evaluate the test performance for synthetic data samples that mimic the training dynamics (in-distribution) and synthetic data samples with slightly altered dynamics (out-of-distribution). To demonstrate the applicability of our approach to real-world data, we also test our approach on in-the-wild datasets with completely unknown data distributions in a zero-shot setting.

While much work still lies ahead, we empirically show that Causal Pretraining can consistently uncover causal relationships for unseen time series, assuming that the training and test time series samples share most of their dynamics. More importantly, we also find that the performance of Causal Pretraining increases with model size and the amount of training data, even when the additional training samples extend the data distribution. In these scenarios, we show that Causal Pretraining can outperform alternative simple approaches without fitting parameters for new data. We further find potential to truly extrapolate to real-life data. We argue that this together hints at the possibility of a foundation model for causal discovery. Finally, we emphasize that this paper focuses on empirical results and the applicability of CP to real-world data. We retain theoretical insights, e.g., identifiability statements, for future work. In summary, we present the following contributions:

1. We explore supervised learning for causal discovery from time series data, aiming at learning it end-to-end from synthetic data with shared dynamical properties.
2. We introduce and evaluate several helper techniques to support the performance of Causal Pretraining.
3. We evaluate the ability of causally pretrained neural networks to predict real-world causal graphs.
4. We observe relations to foundation models, as supported by our empirical results.

Background and Related Work

The discovery of causal relationships from observational time series can be tackled in various ways (Gong et al. 2023), (Assaad, Devijver, and Gaussier 2022), (Vowels, Camgoz, and Bowden 2022), (Runge et al. 2023). Here, we give an overview of these methods and elaborate on how they relate to causal pretraining.

Score-based approaches aim to find a causal graph that maximizes a defined scoring function (e.g., AIC score (Akaike 1992)). With the introduction of formulating the scoring as a continuous optimization problem (Zheng et al. 2018), (Pamfil et al. 2020), the possibility of deploying neural networks arose quickly. For causal discovery in data without temporal dimension (sample data), a multitude of methods that deploy deep-learning architectures was developed ((Lachapelle et al. 2020), (Ng et al. 2019), (Kyono, Zhang, and van der Schaar 2020)). Concerning time series data, (Sun et al. 2023) adapts the previously used

vectorized autoregressive model (VAR) from (Pamfil et al. 2020) to handle nonlinear relationships by deploying a 1D convolution-based architecture.

Being exclusively applicable to time-series data, Granger causality (Granger 1969) has a substantial history. At its core, it evaluates whether a certain variable’s history helps with predicting another target variable. Deep learning-based Granger causality methods mostly deploy neural networks as forecasting architectures and infer Granger causality in various ways. (Montalto et al. 2015) and its extension (Wang et al. 2018) deploy a greedy search over which input variable to use with neural networks to determine Granger-causes. (Tank et al. 2018) and (Tank et al. 2021) alternatively use the weights of the first layer of an MLP under $L2$ regularization of the same layer to determine Granger-causes. Other methods infer from attention weights (Guo, Lin, and Antulov-Fantulin 2019), (Dang, Shah, and Zefos 2018) or the parametrization of variational autoencoders (Li, Yu, and Principe 2023), (Meng 2019). Furthermore, approaches such as (Ahmad, Shadaydeh, and Denzler 2022) propose to generate in-distribution intervention variables and determine Granger causes using the model invariance assumption. Importantly, in all approaches mentioned so far, neural networks are optimized for a single specific dataset.

Framing causal discovery as a supervised learning task was also explored to some extent. Concerning sample data, an early work that does not deploy deep learning is (Lopez-Paz et al. 2015), which aims at learning a binary classifier to direct a link between two variables from data. Some recent strain work that leverages deep learning methods to map from data correlation matrix to directed acyclic graph is (Li, Xiao, and Tian 2020) and (Petersen et al. 2023). (Geffner et al. 2022) aims at learning an autoregressive flow model (Huang et al. 2018) from data that can be used for causal discovery. Finally, as the work that is closest to our methodology, (Löwe et al. 2022) aims at learning a mapping from multivariate time series to causal graph by deploying a Variational autoencoder-based architecture to forecast the time series. The encoder output is then leveraged to infer Granger-causes for each variable. On the contrary, we aim to learn direct mappings without a surrogate task. Another crucial disparity from (Löwe et al. 2022) is that we frame Causal Pretraining to uncover the time lags of any causal relationship rather than only returning a summary graph that ignores temporal patterns. To the best of our knowledge, our proposed method is the first that aims to learn causal discovery for time series completely end-to-end.

Method

Here, we formally introduce our methodology and discuss assumptions. We subsequently detail the architectures and the data generation process we used to test our approach. Finally, we suggest various training techniques and assess their impact on Causal Pretraining.

Definitions & Assumptions

We define $\mathbb{X} = (x_i^t)_{i=1,\dots,V;t=0,\dots,T}$ to be a multivariate time series with $V \in \mathbb{N}^+$ variables and a length of $T \in \mathbb{N}_0$. We

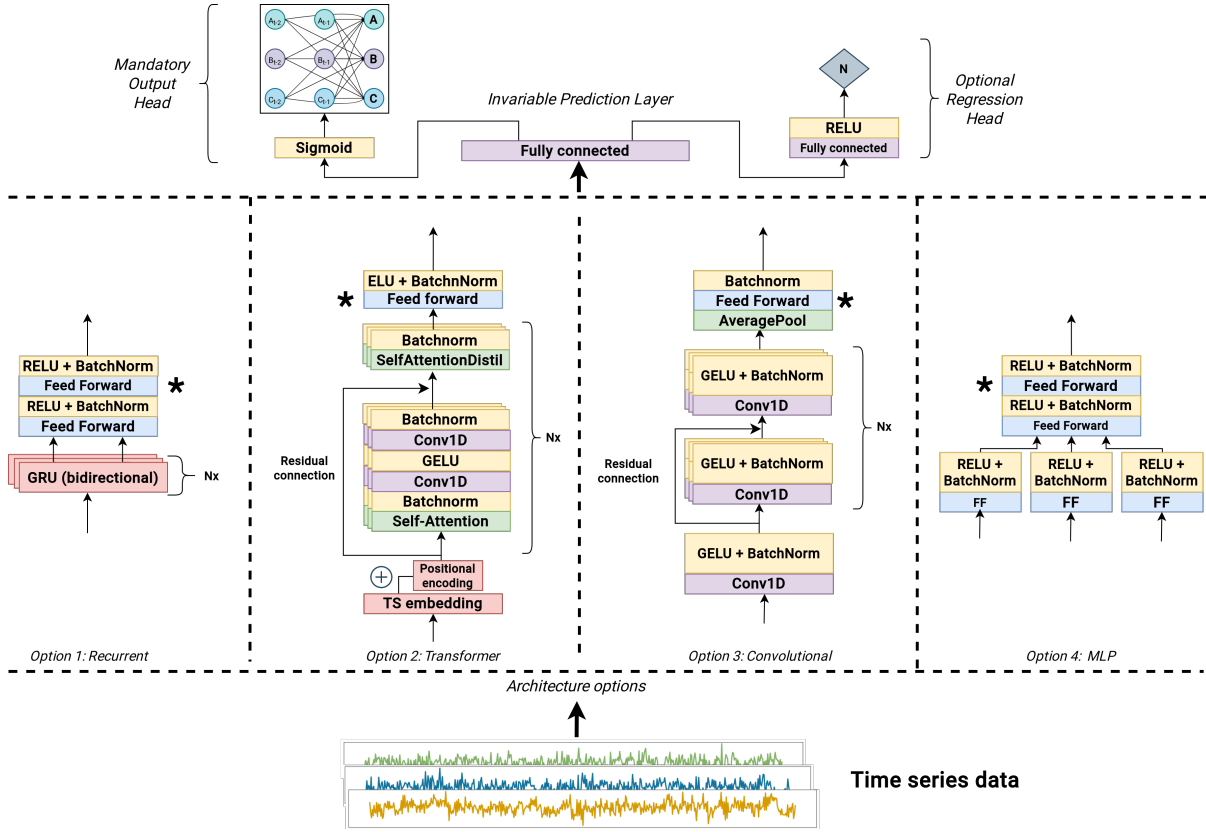


Figure 2: Depiction of architectures that we consider for Causal Pretraining. From left to right: **GRU, Transformer, ConVMixer, MLP**. We further mark locations for correlation injection with *.

assume, for now, that any x_i^t contained in \mathbb{X} can be described by the following time-invariant structural equation model:

$$x_i^t = \sum_{n=1}^N \sum_{j=1}^V f_{i,j}^n(\mathbb{A}^{i,j,n} \cdot x_j^{t-n}) + e_i^t \quad (1)$$

Here \mathbb{A} is a three-dimensional tensor where element $\mathbb{A}^{i,j,n}$ denotes a coefficient that describes the (causal) impact of a specific variable j at a specific time lag n , i.e., x_j^{t-n} , on x_i^t . N denotes the maximum time lag considered. Each linear effect is then transformed by an individual function $f_{j,i}^n$ to account for possible nonlinear relationships. Additionally, $(e_i^t)_{i=1,\dots,V}$ denotes uncorrelated Gaussian noise.

Causal discovery methods (CDMs) are concerned with uncovering the set of non-zero elements of \mathbb{A} from a given time series \mathbb{X} . Typically they either aim at uncovering a summary graph that is represented by an adjacency matrix $G \in \mathbb{Z}^V \times V$ where element (i, j) denotes whether any current or past observation of variable x_j has an impact on variable x_i . Formulated differently, it specifies whether the parameter vector $\mathbb{A}^{i,j}$ has any non-zero elements. Alternatively, methods that aim at uncovering a window causal graph return a 3-dimensional tensor $G \in \mathbb{Z}^{V \times V \times N}$ where element (i, j, n) specifies whether $\mathbb{A}^{i,j,n}$ is estimated to be non-zero. Based on this formalism, we can naively summa-

rize CDMs as: $\text{CDM}(\mathbb{X}) \rightarrow G$. Contrary to that, Causal Pretraining is not concerned with directly uncovering any causal graph G , but rather with producing a causally pretrained neural network (CPNN) in a supervised manner that can be applied in zero-shot settings (no further parameter optimization) to infer G . We define \mathcal{G} as a set of structural equations (Equation 1) that each specifies the causal dynamic of a multivariate time series. Additionally, we define \mathcal{X} to be a set of corresponding \mathbb{X} that originates from a specific element in \mathcal{G} . Formally, Causal Pretraining (CP) can be defined as follows:

$$\text{CP}(\mathcal{X}, \mathcal{G}) \mapsto \text{CPNN}, \quad \text{CPNN}(\mathbb{X}) \mapsto G \in \mathbb{Z}^{V \times V \times N}, \quad (2)$$

where CPNN refers to a causally pretrained neural network. We treat the elements of the network output G as probability estimates that the corresponding elements in \mathbb{A} exist. Throughout this paper, we make the following assumptions: 1. G is a Directed Acyclic Graph (DAG). No instantaneous effects. 2. Causal sufficiency: All causes are observed. 3. Theoretical insights such as (Shah and Peters 2020) suggest that learning a CPNN that can correctly identify the non-zero elements of \mathbb{A} for any \mathbb{X} is very likely impossible. Instead, we assume that train and test distribution share general dynamics, such as the functional space f in Equation 1 from which we draw elements for all synthetic data samples.

Architectures

To properly evaluate our methodology, we deem it essential to evaluate multiple differing architectures. We, therefore, select five different architectures, aiming at covering the prominent deep-learning archetypes up to this date and not necessarily focusing on fully optimizing the architecture for Causal Pretraining for now. We test an **MLP** (Taud and Mas 2018), a unidirectional GRU (**uGRU**), a bidirectional GRU (**bGRU**), a Conv Mixer **CM** (Trockman and Zico Kolter 2022), a recently introduced convolutional architecture that showed strong results on image tasks, which we adapt for time series and a Transformer (**Trf**) that borrows heavily from the Informer (Zhou et al. 2021) architecture excluding sparse attention but featuring Attention Distillation (Zhou et al. 2021). All architectures are depicted in Figure 2 (both GRU architectures are jointly depicted). We keep input and output dimensions for all architectures the same (input: \mathbb{X} , output: causal graph G , optional: regression surrogate task as described below). All networks return a vector with the length $V * V * N$ that we interpret as probabilities for specific elements of \mathbb{A} to be non-zero. We reshape this vector into $G^{V \times V \times N}$ and then calculate binary cross-entropy for the primary loss component with $\mathbb{A} > 0$ as the label. To make the comparison fair, we scale the architectures to have similar amounts of parameters, aiming at estimating the innate capabilities of these archetypes. In total, we deploy five sizes in our experiments (although not all sizes during every experiment), which we denote as "small" ($\sim 13k$ params), "medium" ($\sim 120k$ params), and "big" ($\sim 1.5M$ params) "deep" ($\sim 17M$ params), and "lcm" ("large causal model", $\sim 300M$ params), see Appendix 3.

Additional Training Techniques

While deep learning architectures are, in theory, universal function approximations (Hornik, Stinchcombe, and White 1989), in reality, learning a proper function from observational data is not always straightforward. Networks often converge to local minima or memorize data, which we attempt to counter by employing the following three techniques to help with the optimization process.

Regression Output An easy-to-achieve suboptimal solution in our training setup is predicting all elements of \mathbb{A} as zero. This often results in a small loss given that the true number of non-zero elements in \mathbb{A} ($|\mathbb{A} > 0|$) is small. To discourage this solution, we add an optional regression output that predicts the number of non-zero elements in \mathbb{A} . We formulate the loss related to this prediction as follows:

$$\text{regLoss} = (\hat{Y}_{reg} - |(\mathbb{A} > 0)|)^2 \quad (3)$$

where \hat{Y}_{reg} specifies the regression output. This task should be trivial when the correct edges are identified (counting), and this penalty term should be close to zero. However, predicting a graph with no edges leads to a high penalty.

Correlation Regularization Additionally, since no correlation typically implies no causation, we introduce a regularization term called correlation regularization (CR). This term punishes the case where a model is confident that a

specific element of \mathbb{A} is non-zero (high confidence in the model output G), but the lagged-crosscorrelation of the corresponding time series is low. CR is defined in the following way:

$$CR(G, \mathbb{X}) = \sum_{i=1}^V \sum_{j=1}^V \sum_{n=1}^N \left(\frac{G^{i,j,n}}{|\text{lcc}(\mathbb{X}_i, \mathbb{X}_j, n)| + \beta} \right)^\alpha, \quad (4)$$

$$\text{lcc}(a, b, n) = \frac{\sum_{t=0}^T (a^t - \bar{a}) \cdot L^n(b^t - \bar{b})}{\sqrt{\sum_{t=0}^T (a^t - \bar{a})^2} \sqrt{\sum_{t=0}^T (L^n(b^t - \bar{b}))^2}}, \quad (5)$$

where L^n is the lag operator (shifting the time series n steps). Furthermore, α and β are hyperparameters determining the exact shape of the function. We include a depiction of CR in the Appendix 5. Notable, the term does not reward high confidence for predictions with corresponding high cross-correlation. Nor does it punish low confidence for edges with corresponding high cross-correlation values. It thereby only encodes the idea that no correlation implies no causation.

Correlation Injection Many causal discovery methods take a skeleton graph that encodes correlations between variables as a starting point for causal discovery ((Spirtes et al. 2000),(Li, Xiao, and Tian 2020),(Petersen et al. 2023)). To mimic this and to help the learning process, we fuse (concatenate to the current hidden state) the lagged cross-correlation of all time series in \mathbb{X} into the network as depicted in Figure 2.

Data

Synthetic Data To perform Causal Pretraining, we require a large set of training examples. As real-world data with labels, i.e., ground-truth causal graphs, are typically rare, we generate sets of synthetic data (\mathcal{G}, \mathcal{X}) to train on. We randomly select elements of \mathbb{A} to be non-zero to create a single training example. We then randomly sample all values of \mathbb{A} that are non-zero from a specified uniform distribution and draw nonlinear relations $f_{i,j}^n$ from a set of specified functions: $\{e^x, x^2, \sigma(x), \sin(x), \cos(x), \text{relu}(x), \log(\sigma(x)), \frac{1}{x}, \|x\|, \text{clamp}(x, (-0.5, 0.5))\}$. For linear datasets, we generate data from vector autoregressive models, i.e., we set all $f_{i,j}^n(\cdot) = (\cdot)$ in Equation 1. Then we generate \mathbb{X} according to this dynamic. A detailed description of this process is included in Appendix 1. Since there is no guarantee that randomly sampled structural equations (Equation 1) lead to a stable system, we establish stability tests to exclude unstable systems from the datasets. Finally, we apply min-max normalization to the time series. We refer to Table 4 (Appendix) for further specifications of the datasets that we use for training. We also provide additional data information in the Experiment section.

Kuramoto A frequently used source for synthetic data is fully observable physical simulations. Similar to (Löwe et al. 2022) from which we also adapt the simulation, we

perform Causal Pretraining on data originating from a Kuramoto model (Kuramoto 1975) with five variables. Since we are primarily concerned with one-dimensional variables, we use the trajectories of the simulated variables as one-dimensional time series. Notably, the dynamics of this dataset do not strictly follow Equation 1. We, therefore, empirically evaluate with this dataset whether we can relax our initial assumption.

Zero-shot Inference Another capability of Causal Pretraining we aim to explore in this work is whether it can extrapolate to real-world data outside the training distribution. To test this, we perform zero-shot inference on two distinct benchmarks, meaning we do not perform any further parameter optimization for unseen data. First, we predict the causal relationships in a benchmark involving the discharge of rivers in Germany (Ahmad, Shadaydeh, and Denzler 2022). Secondly, we test the causal direction in a real-life dataset originating from (Jesson et al. 2021), involving the estimation of aerosol-cloud interactions. Since the dataset is originally concerned with estimating the average treatment effect of aerosol on different cloud parameters under weather confounding (meaning the direction of the effect is clear), we are concerned with confirming this causal direction by only evaluating the predicted direction between these two variables under the influence of other weather variables (we consider sea surface temperature, effective inversion strength, and relative humidity at 850 mb).

Experiments

To properly evaluate Causal Pretraining, we conducted four studies aiming at the step-by-step evaluation. During all experiments, we compare CPNNs with two simple baseline techniques and one popular causal discovery method for time series data (PCMCI, (Runge et al. 2019)). We additionally compare the results on the Kuramoto data with results provided in (Löwe et al. 2022). For baselines, we calculate the absolute lagged cross-correlation for all variables and use them directly as probability estimates for \mathbb{A} . We refer to this as Correlation Thresholding (CT). Additionally, we perform a linear form of Granger-causality **GVAR** by fitting a VAR model to the data and treating the absolute parameters of the model as probability estimates for \mathbb{A} . Further, we deploy the **PCMCI** algorithm using partial correlation as the conditional independence test.

Synthetic Data 1 - Impact of the Dataset

To evaluate the performance of CP, we conducted experiments on the precise data distribution we described in the Data section. We continuously evaluated two distinct test sets to understand extrapolation capabilities during this synthetic data experiment. Firstly, a test set, named *Test-Set 1*, includes test samples originating from the same distribution as the training data. Secondly, a test set, named *Test-Set 2*, which has an increased variance for the noise e and an altered (slightly lower, not overlapping) range from which we draw non-zero elements in \mathbb{A} (see Equation 1). To stop training, we perform early stopping based on the loss of an additional validation set that follows the \mathbb{A} range of *Test-Set 2* but

keeps the training variance of the noise e . We use AdamW (Loshchilov and Hutter 2018) as the optimizer during all experiments.

We trained all described architectures on six distinct datasets. Here, we generated 5000 samples for training and 500 samples for testing, which have different structural equations (Equation 1) with varying \mathbb{A} and differing f . We increased the complexity of the dataset step by step, increasing the number of variables, the number of lags, the coefficient range, and the set from which we draw the functions $f_{i,j}^n$. In short, we tested two linear datasets (**SL**, **ML**), two nonlinear datasets with a reduced nonlinear function set (**SNL**, **MNL**), and two datasets with the full function set, as described in the Data section, (**LNL**, **XLNL**). In all three cases, we considered three variables, a maximum lag of two, and the model size "small" for the first dataset. For the second dataset, we considered five variables, a maximum lag of three, and the model size "medium". To compare the performance of our method, we report the best-performing hyperparameter combination of each architecture. We searched the optimal parameters of batch size, learning rate, weight decay, and training addition (Regression Loss, CR, Correlation injection) by performing a full grid search and ran each hyperparameter combination twice. We then selected the combination that resulted in the lowest observed validation loss and rerun it ten times.

Since we were also interested in the performance of CP when trained on data that origins from different distributions, we conducted a second set of experiments where we doubled the size of the six training sets that we described above and joined them together, making the maximum number of lags and the number of variables flexible. By this, we also mixed linear and non-linear samples, effectively drawing from different data distributions. We performed a similar hyperparameter search as in our first experiment. To keep the search space reasonable, we omitted the regression head. We also always included correlation injection since these hyperparameter selections performed consistently better in our first synthetic data experiment. We independently trained model sizes 'medium' and 'big' and rerun the best-scoring hyperparameter combination ten times. We report the results of these studies in Table 1. We denote the results of the first study with **Single** and the results of the second study with **Joint**.

Surprisingly, we found that CP generally performed worse than our baselines in our first set of experiments. With some exceptions, the AUROC scores on both test sets were typically worse, even considering the best possible runs we performed. From our architectures, uGRU and Trf seemed to perform slightly better. However, these tendencies were too inconsistent to conclude a clear trend. Contrary to that, the performance of CP in our second experiment set outperformed all baselines, notably, while fitting no parameters on the test sets. Since CDNNs have to learn a much more extensive training distribution to perform comparably, we expected CDNNs to be less precise. We found the opposite to be true. Even if we kept the architecture size the same, e.g., 'medium', we observed that the performance of CDNNs increased. Furthermore, increasing the model size improved

		MLP	uGRU	bGRU	CM	Trf	CT	GVAR	PCMCI
		Causal Pretraining					Baselines		
		Test-Set 1							
Single	SL	.999 (.000)	1.00 (.000)	.999 (.000)	.999 (.000)	1.00 (.000)	.997	1.00	1.00
	ML	.498 (.005)	.621 (.053)	.626 (.185)	.513 (.014)	.621 (.125)	.997	1.00	1.00
	SNL	.950 (.001)	.955 (.001)	.948 (.002)	.949 (.002)	.955 (.003)	.918	.916	.915
	MNL	.525 (.004)	.829 (.102)	.555 (.104)	.519 (.004)	.545 (.009)	.937	.938	.941
	LNL	.933 (.001)	.937 (.001)	.937 (.001)	.935 (.002)	.934 (.001)	.936	.944	.943
	XLNL	.499 (.006)	.689 (.111)	.749 (.138)	.506 (.004)	.518 (.034)	.938	.944	.943
Joint	Medium	.928 (.023)	.907 (.023)	.920 (.012)	.887 (.003)	.950 (.015)	.958	.888	.946
	Big	.761 (.028)	.959 (.028)	.962 (.032)	.886 (.003)	.977 (.001)			
		Test-Set 2							
Single	SL	.999 (.000)	1.00 (.000)	.999 (.000)	.999 (.000)	1.00 (.000)	.999	.999	.999
	ML	.499 (.003)	.600 (.052)	.613 (.181)	.506 (.007)	.601 (.119)	.999	1.00	1.00
	SNL	.930 (.001)	.935 (.000)	.933 (.001)	.931 (.002)	.935 (.002)	.926	.928	.924
	MNL	.520 (.006)	.785 (.088)	.547 (.090)	.518 (.005)	.538 (.010)	.927	.927	.927
	LNL	.905 (.002)	.915 (.002)	.916 (.001)	.912 (.002)	.916 (.001)	.910	.912	.905
	XLNL	.502 (.006)	.659 (.096)	.710 (.124)	.503 (.006)	.518 (.027)	.913	.913	.910
Joint	Medium	.916 (.023)	.895 (.022)	.906 (.011)	.869 (.003)	.939 (.015)	.948	.862	.910
	Big	.752 (.027)	.946 (.038)	.952 (.033)	.867 (.002)	.970 (.001)			

Table 1: Mean AUROC scores for the synthetic data experiments **Single** and **Joint** (bottom two lines of each Test-set). For experiment **Single**, the first column specifies the dataset, while for experiment **Joint**, it specifies model size. Best results are denoted in **bold**. We report the corresponding standard deviation calculated over 10 runs in brackets.

the performance and, most importantly, the generalization (Test-Set 2). We interpret this in the following way: The broader the training distribution, the more robust/generally applicable CPNNs must become to solve the training distribution. This, in turn, helps with out-of-training performance and generalization. Further, the amount of samples required to optimize CP properly is much larger than we initially expected. We suggest that further improvements could be achieved by making the training distribution even broader, even for similarly sized architectures.

Synthetic Data 2 - Impact of Model Size

As we investigated the relationship between the training distribution and the performance of CP, we also wanted to explore the relationship between the number of parameters and the ability to learn increasingly more complex datasets. For this, we generated three synthetic linear datasets and stepwise increased the number of variables to 10. We performed Causal Pretraining on each dataset, trained varying model sizes independently, and performed the exact hyperparameter search as in the previous chapter. We selected the performance of the best-scoring hyperparameter combination of the best-performing architecture. We then retrained this configuration five times to calculate the final reference scores reported in Figure 4. Note that for the size "lcm" we restricted our search to the Trf and the CM architecture. We find that the parameter count of the model determines the ability to perform causal discovery properly. Specifically, there seems to be a parameter threshold under which no proper solution can be learned (AUROC 0.5). We also find that the extrap-

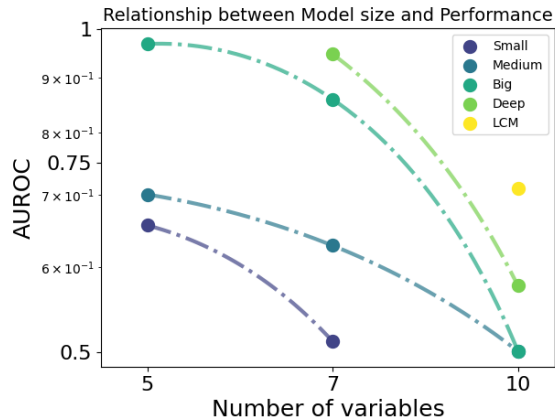


Figure 3: Relationship between the Model size and the performance on datasets with an increased number of variables. We display the performance on Test-set 2 for each data point.

olation capabilities increase even when keeping the same number of data samples (contrary to what one would typically expect concerning overfitting). Further, the required model size for certain variables was much bigger than expected. However, we suggest that this might be partly because discovering a causal graph G with multiple lags requires the estimation of $V \times V \times N$ values, which might be hard for CPNNs. This suggests that reducing the number of lags might reduce the required parameters in the future.

	ACD*	MLP	uGRU	bGRU	CM	Trf	CT	GVAR	PCMCI
<i>Test-Set</i>									
Kuramoto	.952	.825 (.025)	.905 (.053)	.852 (.076)	.954 (.010)	.888 (.070)	0.628	0.464	0.523
Rivers		0.425	1.00	0.5000	0.600	1.000	1.00	1.00	1.00
Aerosol-Cloud		0.00	0.017	0.793	1.00	0.150	0.473	1.00	0.363

Table 2: We here report mean AUROC scores for the Kuramoto dataset, AUROC scores for the highest scoring CPNNs on the River benchmarks, and Accuracy for the Aerosol-Cloud benchmark. The highest scores are denoted in **bold**. We additionally report the variance, calculated over 10 runs on the Kuramoto dataset. in brackets. * (Löwe et al. 2022).

Together, we conclude from this that to improve the performance of CP, we need to increase its scaling, especially considering that it is likely that learning much broader distributions will also require an even higher parameter count.

Kuramoto

To bridge the gap to (Löwe et al. 2022), we evaluate CP on data from a Kuramoto Model with five variables. We generated 50,000 samples for training and 5,000 samples for testing, originating from a simulation of this model. Further, we kept the same simulation settings as in (Löwe et al. 2022) and ignored the diagonal of G (autoregressive links) to make our experiments directly comparable. We again performed a hyperparameter search for each architecture and retrained the best hyperparameter combination ten times to report the standard deviation. We report the results of these experiments in Table 2. Here, the **CM** architectures performed best and achieved similar, if not slightly better, results than (Löwe et al. 2022). Further, all of our baselines are decisively outperformed on this dataset, which we attribute to the fact that the Kuramoto dataset does not follow Equation 1, which hinders methods that rely on linear relationships.

Zero-shot Inference

To test the performance of CPNNs when applied to truly out-of-distribution data, we reuse the CDNNs from Synthetic Data 1 on two benchmarks involving the discharge of rivers in Germany (Ahmad, Shadaydeh, and Denzler 2022) and aerosol-cloud interactions (Jesson et al. 2021). Since all of these datasets only specify a summary graph G , we took the causal link of the first time lag as the prediction for the summary graph. We report the results of this experiment in Table 2. Importantly, since we wanted to perform this experiment as close to a real application of CPNNs as possible, we do not report STD but inferred the summary graph once for each sample from the full-time series and with the highest-scoring models from Synthetic Data 1.

We found that the performance of CPNNs on unseen data distribution differs widely between model architectures and datasets. While the river benchmark is properly predicted by the **uGRU** and the **Trf** architecture, these architectures are specifically outperformed by **CM** on the aerosol dataset. As a side note, (Ahmad, Shadaydeh, and Denzler 2022) or originally (Gerhardus and Runge 2020) suggest that the river benchmark is more challenging than our scoring suggests. This is, since they report results for fixed p-values, which

is not required when calculating AUROC scores. While the performance of CP is improvable, we find it promising that models not trained on these datasets do not break down entirely but display some extrapolation capabilities. We believe this signifies that CPNNs incorporate some form of Causal Discovery test that can uncover correct causal structures in more general settings than the training distribution.

Discussion and Conclusion

This work introduced Causal Pretraining, a methodology for supervised deep end-to-end causal discovery from time series data. We conducted the first experiments, providing evidence that our causally pre-trained neural networks (CDNNs) can achieve similar results to other causal discovery methods without fine-tuning any parameters during inference and being highly computationally parallelizable. We provided evidence that CDNNs show some potential to extrapolate to unseen real-life data outside of the training distribution. Specifically interesting, the performance of CDNNs and its capability to generalize increases with data complexity and model size. Under the assumption that this trend continues beyond the scope of our experiments, we take this as evidence that foundation causal models trained through Causal Pretraining are possible. Recent developments in deep learning (OpenAI 2023) suggest that consistent generalization by scaling up through data and model dimensions is indeed possible. This idea also aligns with theoretical research such as (Bubeck and Sellke 2021), (Bartlett et al. 2020) or (Brutzkus and Globerson 2019), suggesting that greatly over-parametrizing neural networks can help with generalization. While we excluded these experiments in this paper since they are probably intuitive for deep-learning practitioners, we also found that both scalings must go hand-in-hand while performing Causal Pretraining. Simply scaling up the architectures without increasing the dataset’s complexity makes it arbitrarily easy for neural networks to remember the training dataset and not generalize at all. Further, keeping the network too small while increasing the data complexity in might make the learning task impossible and, with that, also prevent generalization. We reserve keeping this delicate balance while scaling up our methodology concerning data distribution and model size for future research.

Acknowledgments

We gratefully recognize the support of iDiv¹, which is funded by the German Research Foundation (DFG – FZT 118, 202548816). Special thanks from Gideon Stein to Yuanyuan Huang¹, Anne Ebeling¹, and Nico Eisenhauer¹ for supporting me in this project and giving me the freedom to be creative with my research ideas. Special thanks also to Christian Reimers² for the creative and helpful exchange of ideas.

Gideon Stein is funded by the iDiv flexpool (No 06203674-22). Maha Shadaydeh is funded by the Carl Zeiss Foundation within the scope of the program line “Breakthroughs: Exploring Intelligent Systems” for “Digitization—explore the basics (No P2017-01-003), use applications”.

¹German Centre of Integrative Biodiversity Research

²Max Planck Institute for Biogeochemistry Jena

References

- Ahmad, W.; Shadaydeh, M.; and Denzler, J. 2022. Causal Discovery using Model Invariance through Knockoff Interventions. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*. 1, 2, 5, 7, 10
- Akaike, H. 1992. Information Theory and an Extension of the Maximum Likelihood Principle. 610–624. New York, NY: Springer New York. ISBN 9780387940373 9781461209195. 2
- Amodei, D.; Ananthanarayanan, S.; Anubhai, R.; Bai, J.; Battenberg, E.; Case, C.; Casper, J.; Catanzaro, B.; Cheng, Q.; Chen, G.; Chen, J.; Chen, J.; Chen, Z.; Chrzanowski, M.; Coates, A.; Diamos, G.; Ding, K.; Du, N.; Elsen, E.; Engel, J.; Fang, W.; Fan, L.; Fougner, C.; Gao, L.; Gong, C.; Hannun, A.; Han, T.; Johannes, L.; Jiang, B.; Ju, C.; Jun, B.; LeGresley, P.; Lin, L.; Liu, J.; Liu, Y.; Li, W.; Li, X.; Ma, D.; Narang, S.; Ng, A.; Ozair, S.; Peng, Y.; Prenger, R.; Qian, S.; Quan, Z.; Raiman, J.; Rao, V.; Satheesh, S.; Seetapun, D.; Sengupta, S.; Srinet, K.; Sriram, A.; Tang, H.; Tang, L.; Wang, C.; Wang, J.; Wang, K.; Wang, Y.; Wang, Z.; Wang, Z.; Wu, S.; Wei, L.; Xiao, B.; Xie, W.; Xie, Y.; Yogatama, D.; Yuan, B.; Zhan, J.; and Zhu, Z. 2016. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *Proceedings of The 33rd International Conference on Machine Learning*, 173–182. PMLR. 1
- Assaad, C. K.; Devijver, E.; and Gaussier, E. 2022. Survey and Evaluation of Causal Discovery Methods for Time Series. *Journal of Artificial Intelligence Research*, 73: 767–819. 2
- Bartlett, P. L.; Long, P. M.; Lugosi, G.; and Tsigler, A. 2020. Benign Overfitting in Linear Regression. *Proceedings of the National Academy of Sciences*, 117(48): 30063–30070. ArXiv:1906.11300 [cs, math, stat]. 7
- Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L. D.; Monfort, M.; Muller, U.; Zhang, J.; Zhang, X.; Zhao, J.; and Zieba, K. 2016. End to End Learning for Self-Driving Cars. ADS Bibcode: 2016arXiv160407316B. 1
- Brutzkus, A.; and Globerson, A. 2019. Why do Larger Models Generalize Better? A Theoretical Perspective via the XOR Problem. In *Proceedings of the 36th International Conference on Machine Learning*, 822–830. PMLR. 7
- Bubeck, S.; and Sellke, M. 2021. A Universal Law of Robustness via Isoperimetry. In *Advances in Neural Information Processing Systems*, volume 34, 28811–28822. Curran Associates, Inc. 7
- Dang, X.-H.; Shah, S. Y.; and Zefos, P. 2018. seq2graph: Discovering Dynamic Dependencies from Multivariate Time Series with Multi-level Attention. ArXiv:1812.04448 [cs, stat]. 2
- Geffner, T.; Antoran, J.; Foster, A.; Gong, W.; Ma, C.; Kiciman, E.; Sharma, A.; Lamb, A.; Kukla, M.; Hilmkil, A.; Jennings, J.; Pawlowski, N.; Allamanis, M.; and Zhang, C. 2022. Deep End-to-end Causal Inference. 2
- Gerhardus, A.; and Runge, J. 2020. High-recall causal discovery for autocorrelated time series with latent confounders. In *Advances in Neural Information Processing Systems*, volume 33, 12615–12625. Curran Associates, Inc. 7
- Glasmachers, T. 2017. Limits of End-to-End Learning. In *Proceedings of the Ninth Asian Conference on Machine Learning*, 17–32. PMLR. 1
- Gong, C.; Yao, D.; Zhang, C.; Li, W.; and Bi, J. 2023. Causal Discovery from Temporal Data: An Overview and New Perspectives. ArXiv:2303.10112 [cs, stat]. 2
- Granger, C. W. J. 1969. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*, 37(3): 424–438. 2
- Guo, T.; Lin, T.; and Antulov-Fantulin, N. 2019. Exploring interpretable LSTM neural networks over multi-variable data. In *Proceedings of the 36th International Conference on Machine Learning*, 2494–2504. PMLR. 2
- Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5): 359–366. 4
- Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A. 2018. Neural Autoregressive Flows. In *Proceedings of the 35th International Conference on Machine Learning*, 2078–2087. PMLR. 2
- Jesson, A.; Manshausen, P.; Douglas, A.; Watson-Parris, D.; Gal, Y.; and Stier, P. 2021. Using Non-Linear Causal Models to Study Aerosol-Cloud Interactions in the Southeast Pacific. ArXiv:2110.15084 [physics]. 5, 7
- Kuramoto, Y. 1975. Self-entrainment of a population of coupled non-linear oscillators. *Mathematical Problems in Theoretical Physics*, 39: 420–422. ADS Bibcode: 1975LNP....39..420K. 5
- Kyono, T.; Zhang, Y.; and van der Schaar, M. 2020. CASTLE: Regularization via Auxiliary Causal Graph Discovery. In *Advances in Neural Information Processing Systems*, volume 33, 1501–1512. Curran Associates, Inc. 2
- Lachapelle, S.; Brouillard, P.; Deleu, T.; and Lacoste-Julien, S. 2020. Gradient-Based Neural DAG Learning. ArXiv:1906.02226 [cs, stat]. 2

- Li, H.; Xiao, Q.; and Tian, J. 2020. Supervised Whole DAG Causal Discovery. *ArXiv:2006.04697 [cs, stat]*. 2, 4
- Li, H.; Yu, S.; and Principe, J. 2023. Causal Recurrent Variational Autoencoder for Medical Time Series Generation. *ArXiv:2301.06574 [cs, eess]*. 2
- Lopez-Paz, D.; Muandet, K.; Schölkopf, B.; and Tolstikhin, I. 2015. Towards a Learning Theory of Cause-Effect Inference. In *Proceedings of the 32nd International Conference on Machine Learning*, 1452–1461. PMLR. 2
- Loshchilov, I.; and Hutter, F. 2018. Decoupled Weight Decay Regularization. 5
- Löwe, S.; Madras, D.; Zemel, R.; and Welling, M. 2022. Amortized Causal Discovery: Learning to Infer Causal Graphs from Time-Series Data. In *Proceedings of the First Conference on Causal Learning and Reasoning*, 509–525. PMLR. 1, 2, 4, 5, 7
- Meng, Y. 2019. Estimating Granger Causality with Unobserved Confounders via Deep Latent-Variable Recurrent Neural Network. *ArXiv:1909.03704 [cs, stat]*. 2
- Montalto, A.; Stramaglia, S.; Faes, L.; Tessitore, G.; Prevede, R.; and Marinazzo, D. 2015. Neural Networks with Non-Uniform Embedding and Explicit Validation Phase to Assess Granger Causality. *Neural Networks*, 71: 159–171. 2
- Ng, I.; Zhu, S.; Chen, Z.; and Fang, Z. 2019. A Graph Autoencoder Approach to Causal Structure Learning. ADS Bibcode: 2019arXiv191107420N. 1, 2
- OpenAI. 2023. GPT-4 Technical Report. *ArXiv:2303.08774 [cs]*. 7
- Pamfil, R.; Sriwattanaworachai, N.; Desai, S.; Pilgerstorfer, P.; Georgatzis, K.; Beaumont, P.; and Aragam, B. 2020. DYNOTEARS: Structure Learning from Time-Series Data. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 1595–1605. PMLR. 2
- Petersen, A. H.; Ramsey, J.; Ekstrøm, C. T.; and Spirtes, P. 2023. Causal Discovery for Observational Sciences Using Supervised Machine Learning. *Journal of Data Science*, 21(2): 255–280. 2, 4
- Runge, J.; Gerhardus, A.; Varando, G.; Eyring, V.; and Camps-Valls, G. 2023. Causal inference for time series. *Nature Reviews Earth & Environment*, 4(7): 487–505. 2
- Runge, J.; Nowack, P.; Kretschmer, M.; Flaxman, S.; and Sejdinovic, D. 2019. Detecting causal associations in large nonlinear time series datasets. *Science Advances*, 5(11): eaau4996. *ArXiv:1702.07007 [physics, stat]*. 5
- Shah, R. D.; and Peters, J. 2020. The Hardness of Conditional Independence Testing and the Generalised Covariance Measure. *The Annals of Statistics*, 48(3). *ArXiv:1804.07203 [math, stat]*. 3
- Spirtes, P.; Glymour, C. N.; Scheines, R.; and Heckerman, D. 2000. *Causation, Prediction, and Search*. MIT Press. ISBN 9780262194402. 4
- Sun, X.; Schulte, O.; Liu, G.; and Poupart, P. 2023. NTS-NOTEARS: Learning Nonparametric DBNs With Prior Knowledge. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, 1942–1964. PMLR. 2
- Tank, A.; Cover, I.; Foti, N. J.; Shojaie, A.; and Fox, E. B. 2018. An Interpretable and Sparse Neural Network Model for Nonlinear Granger Causality Discovery. *ArXiv:1711.08160 [stat]*. 1, 2, 10
- Tank, A.; Covert, I.; Foti, N.; Shojaie, A.; and Fox, E. 2021. Neural Granger Causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. *ArXiv:1802.05842 [stat]*. 2
- Taud, H.; and Mas, J. 2018. Multilayer Perceptron (MLP). In Camacho Olmedo, M. T.; Paegelow, M.; Mas, J.-F.; and Escobar, F., eds., *Geomatic Approaches for Modeling Land Change Scenarios*, Lecture Notes in Geoinformation and Cartography, 451–455. Cham: Springer International Publishing. ISBN 9783319608013. 4
- Teodora Trifunov, V.; Shadaydeh, M.; and Denzler, J. 2022. Time Series Causal Link Estimation under Hidden Confounding using Knockoff Interventions. ADS Bibcode: 2022arXiv220911497T. 1
- Trockman, A.; and Zico Kolter, J. 2022. Patches Are All You Need? ADS Bibcode: 2022arXiv220109792T. 4
- Vowels, M. J.; Camgoz, N. C.; and Bowden, R. 2022. D#x2019;ya Like DAGs? A Survey on Structure Learning and Causal Discovery. *ACM Computing Surveys*, 55(4): 82:1–82:36. 2
- Wang, Y.; Lin, K.; Qi, Y.; Lian, Q.; Feng, S.; Wu, Z.; and Pan, G. 2018. Estimating Brain Connectivity With Varying-Length Time Lags Using a Recurrent Neural Network. *IEEE Transactions on Biomedical Engineering*, 65(9): 1953–1963. 2
- Zheng, X.; Aragam, B.; Ravikumar, P.; and Xing, E. 2018. DAGs with NO TEARS: Smooth Optimization for Structure Learning. 1, 2
- Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11106–11115. Number: 12. 4

Appendix

Here, we include additional specifications for the exact parameter counts for all used architectures and sizes in Table 3 and data that we use throughout this paper in Algorithm 1 and Table 4. Next, we include a study on inference speed in Inference Speed and a depiction of Correlation Regularization in Figure 5. While we expected that the importance of hyperparameter selection would decrease with model size, we found no consistent pattern in our experiments. To depict this, we display an example of performance distribution over hyperparameter in Figure 6. Further, in Preliminary studies, we describe two small experiments on the baseline capabilities of our architectures, which were historically performed before experiments in the Experiment section. In Probabilistic studies, we introduce a procedure that allows CPNNs to predict probability distributions instead of probability scalars that might be useful for certain applications. Finally, we provide a summary of all abbreviations used throughout the paper in Table 5.

Algorithm 1: An algorithm with caption

Require: $: maximum_lags(n), number_of_variables(v)$
Require: $: link_threshold, link_distribution$
Require: $: ts_length, noise_var, number_of_samples$
Require: $: nl_selection, nl_threshold$
Require: $: data_test(), graph_test()$
 $\mathcal{X} = \{\}$
 $\mathcal{G} = \{\}$
while $len(\mathcal{G}) < number_of_samples$ **do**
 $Draw : U(0, 1)^{V \times V \times N}$
 $PA : 1$ **if** $Draw > link_threshold$ **else** 0
 $A = U(link_distribution)$ **if** PA **else** 0
 if nonlinear **then**
 $Draw : U(0, 1)^{V \times V \times N}$
 $PA : 1$ **if** $Draw > nl_threshold$ **else** 0
 $f : U(nl_selection)$ **if** PA **else** 1
 else
 $f : 1$
 end if
 $\mathbb{X} = \{\}$
 while $len(\mathbb{X}) < ts_length$ **do**
 generate x_i^{new} according to Equation 1 (A, f)
 end while
 if $data_test(\mathbb{X})$ **then**
 $\mathcal{X} \leftarrow X$
 $\mathcal{G} \leftarrow G$
 end if
end while

Inference Speed

Since inference with CPNNs is highly parallelizable (batching), which we denote as a neat advantage of CPNNs, we provide some short comparisons on inference speed. For this, we reuse the dataset of the above section and measure the speed to predict the corresponding test set with 500 samples each. We perform inference on an Intel i7-7700

Size	Small	Medium	Big	Deep	LCM
MLP	13.4k	118k	1.5M	17.3M	-
uGRU	13.0k	126k	1.6M	17.2M	-
bGRU	12.7k	116k	1.5M	18.4M	-
CM	13.3k	128k	1.4M	17M	286M
Trf	12.3k	120k	1.4M	16.1M	391M

Table 3: Exact parameter count for all architectures and sizes we use throughout this paper.

CPU, omitting GPU support from which CPNNs would additionally benefit. We summarize the relationship between the number of variables and the inference speed as follows: While our simple benchmark techniques are still slightly faster since they come with closed-form solutions for their optimization (and we additionally implemented a batched version for **CT**), the inference speed of Causal Pretraining lies in the same order of magnitude for all number of variables tested. In contrast, **PCMCI** scales quadratically in our experiments. The number of conditional independence tests that must be performed grows quadratically with the number of variables, making it infeasible to infer from a large set of variables. Importantly, all our methods (including **PCMCI**) are very fast to compute in comparison to other approaches such as, e.g., (Tank et al. 2018) or (Ahmad, Shadaydeh, and Denzler 2022), which require fitting neural networks for each sample. We believe this feature of CPNNs makes it specifically beneficial in areas where many similar samples have to be processed, such as in Earth and neuroscience. Especially considering that they can outperform similarly fast benchmarks.

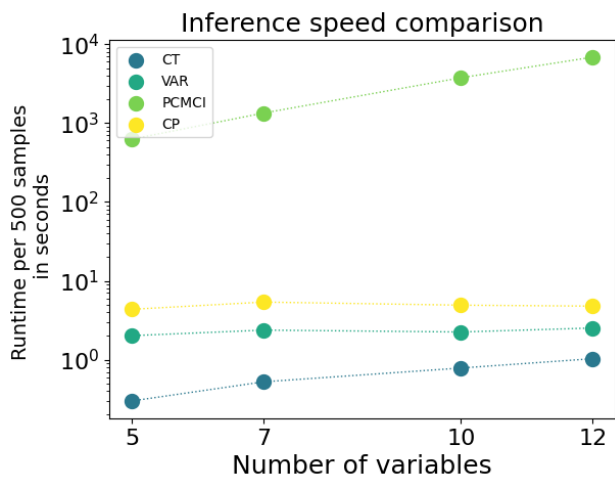


Figure 4: Comparison of inference speed. Here, we use the architecture with the highest AUROC score for each dataset to represent CPNNs. We report the speed of computing the solution for 500 samples and 100 repetitions.

Name	Pre	SL	ML	SNL	MNL	LNL	XLNL	Wide
Function set	L	L	L	NL1	NL1	NL2	NL2	NL2
Variables	3	3	5	3	5	3	5	7/10/15
Maximum lag	2	2	3	2	3	2	3	3
Coefficients (train)	br	br	± br	br	± br	br	± br	± br
Noise var (train)	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Coefficients (test 1)	br	br	± br	br	± br	br	± br	± br
Noise var (test 1)	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Coefficients (test 2)	sr	sr	± sr	sr	± sr	sr	± sr	± sr
Noise var (test 2)	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Average non-zero	9	2.7	7.5	2.7	7.5	2.7	7.5	14.7/30/67.5

Table 4: Dataset specifications. L denotes linear, NL1 denotes drawing only from (e^x, x^2) , and NL2 denotes the full function set specified in the method Method section . Furthermore, sr denotes the range (0.2-0.3), and br denotes the range (0.3-0.5). ± denotes that values drawn from a range can be positive or negative. **Wide** specifies the datasets that were used during Synthetic Data 2 . **Pre** specifies the dataset used during Preliminary studies .

Preliminary Studies: Architecture Capabilities

Initially, we conducted two experiments to estimate our architecture’s general capabilities. We used the model size ”small” for these three experiments and chose a learning rate of 0.0001, a weight decay of 0.01, and a batch size of 64, performing no hyperparameter search.

First, we tested whether our architectures can correctly distinguish between a linear and a nonlinear \mathbb{X} , which we deemed to be an essential capability. For this purpose, we fixed which elements of \mathbb{A} are non-zero and only alter their actual values. We then added a nonlinear $f_{j,i}^n$ for half of the samples as described in Algorithm 1. We temporarily repurposed the regression head by adding a Sigmoid function. We treated the output as a binary classification and used binary cross entropy as the loss function.

We found that the **uGRU** and the **Trf** architecture outperform the other approaches in this limited setting (Increased AUROC scores on Test data), framing them as the most promising candidates. While we found this to be rather intuitive, since these are archetypes frequently applied to time series, we also later found out that this advantage does not necessarily translate into more general setups. We conclude that this supports our approach of considering multiple architectures.

Second, we tested whether CPNNs can uncover the precise values in \mathbb{A} . We again fix which values of \mathbb{A} are non-zero and alter only its coefficients. To train, we remove the sigmoid activation from our output vector and treat it as a direct prediction for \mathbb{A} . We take the original \mathbb{A} as labels and optimize an MSE loss.

We found that none of our architectures can uncover the exact \mathbb{A} . They instead converge on predicting the mean of each element in \mathbb{A} . This behavior is consistent over all tested architectures, and we observed no change even when weighing the loss values of non-zero elements higher than those elements that are zero. We took this as a reason for our binary problem formulation, aiming only at uncovering which elements are non-zero but not the exact strength of the relationship. While we are confident that this behavior might be altered with more sophisticated loss schemes or larger

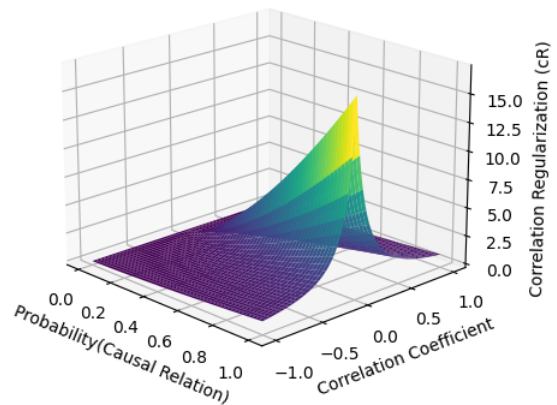


Figure 5: Visualization of CR for $\alpha = 1.5$ and $\beta = 0.15$. The penalty is only big when the confidence is high while the correlation coefficient is low.

model sizes, we keep this additional complexity for future research. Importantly, making binary decisions aligns with many other causal inference methods.

Probabilistic Predictions

Here, we tested whether we can adapt our method to produce distribution as outputs instead of a single probability. We do this by running many short samples of the same time series through a model and constructing a distribution over output probabilities for each edge. It might be interesting to analyze the form of these distributions in the future, which could help detect wrongly classified edges or determine the exact decision threshold. As an example of this procedure, Figure 6 holds the output distributions of a **uGRU** model with the size ”big” for a single sample from the **XLnL** dataset. Notably, some links have very high consistent confidence, while the prediction varies much more for others.

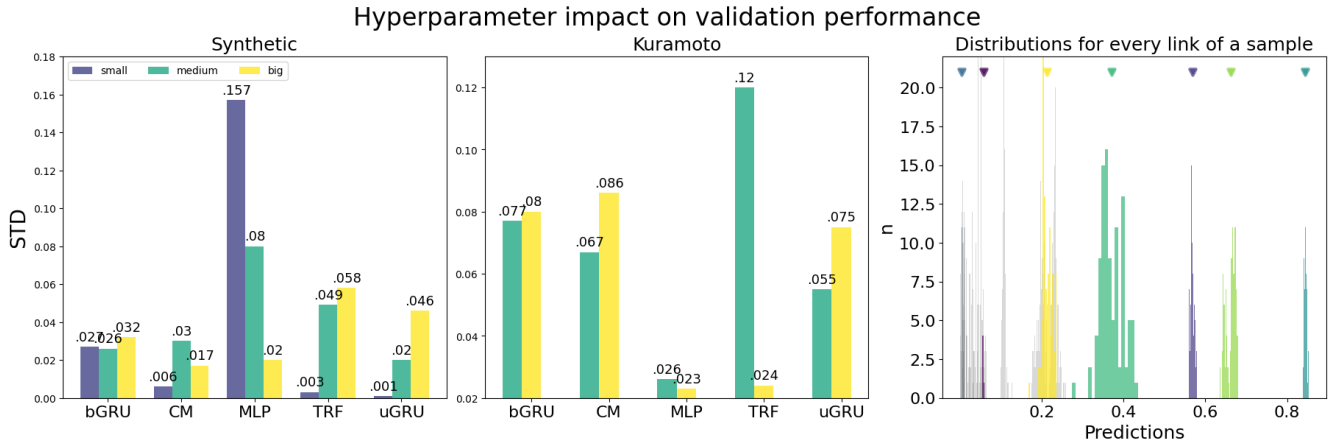


Figure 6: **Left:** Impact of hyperparameter selection. We report the standard deviation of the AUROC score on the validation set. For the synthetic data, we provide the joint dataset grid search results for model sizes "medium" and "big" and the grid search results for the dataset **LnL** for model size "small," ignoring all combinations that were not evaluated for the joint dataset. **Right:** Distributions over output probabilities for every edge in a single sample, sampled as described in Probabilistic studies . Colorful distributions denote true edges.

Abbreviation	Full Name	Type
CP	Causal Pretraining	Method
CPNN	Causally Pretrained Neural Network	Method
MLP	Multi-Layer Perceptron	Architecture
uGRU	unidirectional Gated Recurrent Unit	Architecture
bGRU	bidirectional Gated Recurrent Unit	Architecture
CM	ConvMixer	Architecture
Trf	Transformer	Architecture
SL	Small Linear	Dataset
ML	Medium Linear	Dataset
SNL	Small Nonlinear	Dataset
MNL	Medium Nonlinear	Dataset
LNL	Large Nonlinear	Dataset
XLNL	Extra Large Nonlinear	Dataset
CP	Correlation Thresholding	Baseline
GVAR	Granger Causal Vectorized Autoregression	Baseline
LCM	Large Causal Model	Model size
CR	Correlation Regularization	Technique

Table 5: Summary of abbreviations used throughout this paper.