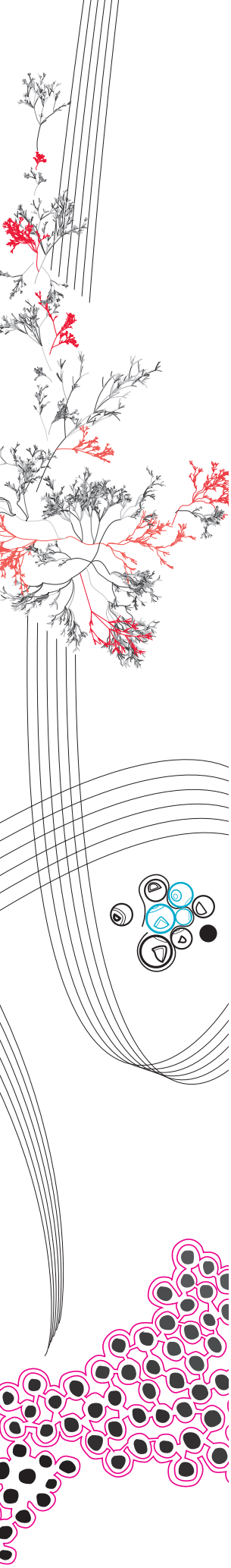


Temporal Causal Discovery and Structure Learning with Attention-Based Convolutional Neural Networks

Master's Thesis by

Meike Nauta

August 2018

Graduation Committee
Dr. Christin Seifert
Dr.ir. Maurice van Keulen
Dr. Doina Bucur

University of Twente
Enschede, The Netherlands

Preface

This graduation project was an unpredictable but exciting journey that resulted in two different research projects, both on the topic of *causal discovery and structure learning from observational data*. This thesis presents the results of both projects to obtain the Master's degree in Computer Science.

My first research project resulted in a paper named 'LIFT: Learning Fault Trees from Observational Data', that got accepted at the 15th International Conference on Quantitative Evaluation of SysTems[1] (QEST), that will be held from September 4-8 2018 in Beijing. This paper can be found on page 51 of this thesis. My main contribution however is the report of my second research project, called 'Temporal Causal Discovery and Structure Learning with Attention-Based Convolutional Neural Networks', which starts on page 1 of this thesis. In the following paragraphs, I will give a chronological overview of my graduation process that explains how I ended up with two different research projects.

My first research project started as part of the course 'Research Topics', which is intended to serve as a preliminary investigation for the final graduation project. I decided to join SEQUOIA[2], a project funded by the Dutch STW that studies smart maintenance optimization for railroads by deploying machine learning techniques as well as fault tree analysis. A fault tree graphically models probabilistic causal chains of events that end in a global system failure. Since constructing a fault tree was largely a manual process, I chose to study the possibility to machine-learn a fault tree from observational data.

Since the decision tree is a tree formalism that is commonly machine-learnt from data, decision tree learning algorithms appeared to be a natural starting point for the design of a fault tree learning algorithm. However, I soon found out that decision trees model only correlations which would not provide root causes for faults. Since a fault tree needs to model causal relationships, I got into the topic of *Causal Discovery* that aims to discover causal relationships from observational data. At the end of the course, I had designed and implemented a causal discovery algorithm that could successfully learn a static fault tree. In my paper I not only present this algorithm, but I also introduce formal definitions for all elements in a fault tree.

Together with my supervisor dr. Doina Bucur and SEQUOIA project leader prof.dr. Mariëlle Stoelinga, we decided that is was worthwhile to improve (and shorten) the paper such that it could be submitted to a conference. Since I already finished the Research Topics course, I spent the first 180 hours ($\pm$6 ECTS) of my final graduation project (credited for 30 ECTS in total) improving both the paper and the algorithm. In collaboration with Doina that wrote the main part of the related work section, and with the feedback of Mariëlle, the paper got accepted at the International Conference on Quantitative Evaluation of SysTems (QEST).

After finishing this paper, I wanted to stay in the causal discovery domain but switched from learning fault trees to learning more generic causal graphs. Furthermore, whereas my fault tree algorithm was mainly based on statistics, I decided to change my scope to deep learning. Neural networks and their state-of-the-art performance in many classification and prediction tasks sparked my interests. Since most existing causal discovery methods use statistical measures, I had to be creative on how to apply deep learning for causal discovery. I very much enjoyed the design part of my project where I had to think out-of-the-box. I also learnt all kinds of deep learning techniques and causality jargon that I didn't know about before.

Thanks to the critical yet valuable feedback of Doina Bucur, Christin Seifert, Maurice van Keulen and Yannick Donners, I have written a report that presents a deep learning framework based on attention-based convolutional neural networks that can successfully construct a temporal causal graph by discovering causal relationships in time series data. I am looking forward to continuing the research on this topic during my upcoming PhD at the University of Twente.

- *Meike Nauta*
August 2018

[1] `http://www.qest.org/qest2018/`
[2] `http://fmt.cs.utwente.nl/research/projects/SEQUOIA/`

Contents of this Master's Thesis

Temporal Causal Discovery and Structure Learning with Attention-Based Convolutional Neural Networks

Meike Nauta, m.nauta@alumnus.utwente.nl

Abstract

We present the Temporal Causal Discovery Framework (TCDF), a deep learning framework that learns a causal graph structure by discovering causal relationships in observational time series data. TCDF uses attention-based convolutional neural networks to detect correlations between time series and subsequently performs a novel validation step to distinguish causality from correlation. By interpreting the internal parameters of the convolutional networks, TCDF can also discover the time delay between a cause and the occurrence of its effect. Our framework can learn both cyclic and acyclic causal graphs, which can include confounders and instantaneous effects. The graph reduction step in TCDF removes indirect causal relationships to improve readability of the constructed graph. Using the representational power of deep learning, TCDF removes idealized assumptions upon the data that existing, usually statistical, causal discovery methods make. Experiments on actual and simulated time series data show state-of-the-art performance of TCDF on discovering causal relationships in continuous, noisy and (non-)stationary time series data. Furthermore, we show that TCDF can circumstantially discover the presence of hidden confounders. Our broadly applicable framework can be used to gain novel insights into the causal dependencies in a complex system, which is important for interpretation, explanation, prediction, control and policy making.

Contents

Summary of Notation

Notation	Meaning
$\mathbf{X}$	Dataset containing N time series all having the same length T, with $N \geq 2$ and $T \geq 2$.
$\mathbf{X}_i$	The i^{th} row in dataset $\mathbf{X}$ corresponding to one time series of T time steps, with $0 < i \leq N$.
$\hat{\mathbf{X}}_i$	Predicted time series for $\mathbf{X}_i$.
$\mathbf{X}_{-i}$	All time series in $\mathbf{X}$ except $\mathbf{X}_i$.
X_i^t	Value of time series $\mathbf{X}_i$ at time step t, with $0 < t \leq T$.
$\hat{X}_i^t$	Predicted value for X_i^t.
$\mathcal{N}_i$	Attention-based Dilated Depthwise Separable Temporal Convolutional Network (AD-DSTCN) that receives $\mathbf{X}$ as input and outputs $\hat{\mathbf{X}}_i$.
$\mathcal{G}$	Temporal causal graph with a set of vertices and edges (V, E), denoting the causal relationships between time series in $\mathbf{X}$ and their delays.
$\overline{\mathcal{G}}$	Complement of $\mathcal{G}$.
v_i	Vertex in V representing time series $\mathbf{X}_i$.
$e_{i,j}$	Directed edge in E from v_i to v_j.
$E(\mathcal{G})$	Set of edges in $\mathcal{G}$.
$d(e_{i,j})$	Delay corresponding to edge $e_{i,j}$ denoting the number of time steps between the occurrence of cause $\mathbf{X}_i$ and the occurrence of effect $\mathbf{X}_j$.
$s(e_{i,j})$	Causal strength score of cause $\mathbf{X}_i$ on effect $\mathbf{X}_j$.
$p = \langle v_i, ..., v_j \rangle$	Path from vertex v_i to vertex v_j.
$\|p\|$	Length of a path $p = \langle v_i, ..., v_j \rangle$, defined by the number of edges between v_i and v_j.
$d(p)$	Delay of a path $p = \langle v_i, ..., v_j \rangle$, defined as the sum of delays of all edges in p.
$\mathcal{G}_{\mathcal{G}}$	Temporal causal graph denoting the ground truth causal relationships and their delays.
$\mathcal{G}_{\mathcal{F}}$	Temporal causal graph denoting the full ground truth causal relationships and their delays, which contains an edge $e_{i,j}$ for each directed path from v_i to v_j in $\mathcal{G}_G$.
$\mathcal{G}_{\mathcal{L}}$	Temporal causal graph that is learnt by a causal discovery method.
K	Kernel size of an AD-DSTCN.
L	Number of hidden layers in an AD-DSTCN.
R	Receptive field of a Convolutional Neural Network.
c	Dilation coefficient.
f	Dilation factor (i.e. step size), which equals c^l for layer l.
λ	Learning rate of a neural network.
$\mathbf{a}_i$	Attention vector $\mathbf{a_i} = [a_{1,i}, a_{2,i}, ..., a_{i,i}, ..., a_{N,i}]$.
$a_{i,j}$	Attention score denoting how much $\mathcal{N}_j$ attends to input time series $\mathbf{X}_i$.
$\mathcal{W}_i$	The kernel weights of $\mathcal{N}_i$.
τ_i	Threshold to select the attention scores in $\mathbf{a}_i$ for causal validation.
$\mathbf{h}_i$	Attention vector $\mathbf{a}_i$ to which our HardSoftmax function is applied.
$h_{i,j}$	Attention score $a_{i,j}$ to which our HardSoftmax function is applied.
$\mathbf{G}$	List of gaps $[g_0, ..., g_{N-1}]$ denoting the gaps between an ordered list of attention scores.
g_i	Gap at position i in $\mathbf{G}$, denoting the value difference between two attention scores.
α_i	Learnable parameter used in the PReLU activation function in network $\mathcal{N}_i$.
$\mathcal{F}(x)$	Function applied by a convolutional layer to transform input x to $\mathcal{F}(x)$.
o	Output of a convolutional layer after the activation function is applied.
$\mathbf{P}_i$	Set of potential causes for time series $\mathbf{X}_i$.
$\mathbf{C}_i$	Set of true causes for time series $\mathbf{X}_i$.
$\mathcal{L}$	Loss of a network, i.e. the error between $\mathbf{X}_i$ and $\hat{\mathbf{X}}_i$.
$\mathcal{L}_{\mathcal{G}}$	Loss of a network when the real input distribution for CQII is used.
$\mathcal{L}_{\mathcal{I}}$	Loss of a network when the intervened distribution for CQII is used.
$\epsilon_{d(e_{i,j})}$	The error of an incorrectly learnt delay, calculated as the distance from the ground truth delay to the learnt delay relative to the receptive field.
μ_ϵ	Average error of all incorrectly learnt delays.
TP, **FP**	Set of True Positives, resp. False Positives.
TN, **FN**	Set of True Negatives, resp. False Negatives.

1 Introduction

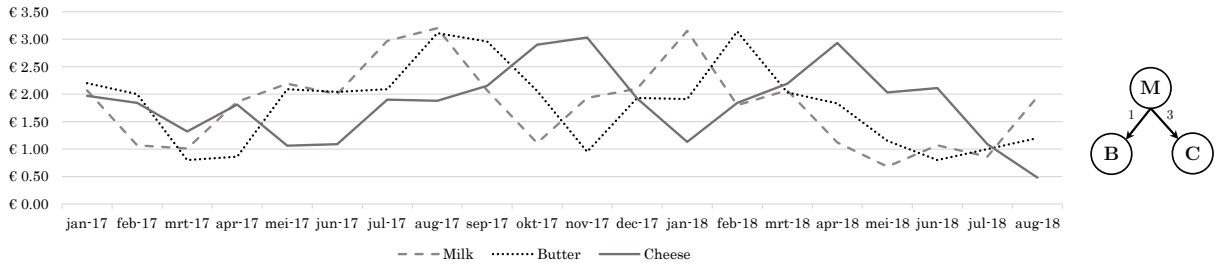
What makes a stock's price increase? What influences the water level of a river? Although machine learning has been successfully applied to predict these variables, most machine learning models cannot answer those questions. Existing machine learning models make predictions on the basis of correlations alone, but correlation does not imply causation [Kleinberg, 2015]. Measures of correlation are *symmetrical*, since correlation only tells us that there exists a relation between variables. In contrast, causation is usually *asymmetrical* and therefore tells us the directionality of a relation between variables. For example, since height is correlated with age, age is also correlated with height. Only a causality measure can conclude if either age causally influences height, or that height has a causal influence on age. It could also be that, although height and age are correlated, there is no causal relationship between these variables. Correlation which is not causation often arises if two variables have a common cause, or if there is a spurious correlation such that the values of two unrelated variables are coincidentally statistically correlated.

Predictive models, e.g. decision trees and neural networks, do not make a distinction between correlation and causation and only learn correlations to increase their prediction accuracy [Nauta et al., 2018]. The relationships learnt by the model may be unstable if they are not causal, such that the model might stop working in the future. This will lead to wrong predictions, which is undesired if these predictions are used for decision making. If a model would learn *causal* relationships, we can make more robust predictions. In addition to making forecasts, the goal in many sciences is often to *understand* the mechanisms by which variables come to take on the values they have, and to predict what the values of those variables would be if the naturally occurring mechanisms were subject to outside manipulations [Spirtes, 2010]. Those mechanisms can be understood by discovering causal associations that explain the relationship between and occurrence of events. Knowledge of the underlying causes allows us to develop effective policies to prevent or produce a particular outcome [Kleinberg, 2013].

The traditional way to discover causal relations is to manipulate the value of a variable by using interventions or real-life experiments. In an experimental setting, all other influencing factors of the target variable can be held fixed, such that it can be tested if a manipulation of a potential cause changes the target variable. However, such experiments and interventions are often costly, too time-consuming, unethical or even impossible to carry out. With the current advances in sensoring and Internet of Things, the amount of observational data grows extensively, allowing us to reveal (hypothetical) causal information by analysing this observational data, known as *causal discovery* [Zhang et al., 2017]. Causal discovery aims to help in interpreting data and formulating and testing hypotheses, which can be used to prioritize experiments and to build and improve theories or simulation models. Furthermore, causal discovery from observational data can be used to ensure the validity of experimental results that are often collected in restricted laboratory settings.

In this report, we focus on causal discovery from time series data, as the notion of time facilitates the discovery of the directionality of a causal relationship. After all, a cause generally happens before the effect. Many algorithms have been developed in the last years to discover causal relationships from multivariate temporal observational data, particularly in the area of graphical causal modeling [Singh et al., 2018]. However, these usually statistical measures tend to rely on idealized assumptions that rarely hold in practice. Existing approaches assume that the time series data is linear, stationary or without noise [Runge et al., 2017], [Huang and Kleinberg, 2015]. Furthermore, many methods assume that the underlying causal structure has no (hidden) common causes, is acyclic or does not have instantaneous effects [Budhathoki and Vreeken, 2018], [Entner and Hoyer, 2010]. Furthermore, existing methods are only designed to discover causal associations, and they cannot be used to predict a variable's value based on these discovered causal variables.

We suggest to use *Deep Learning* for causal discovery, since Deep Neural Networks (DNNs) achieve state-of-the art performance in many classification and prediction tasks. DNNs are able to discover complex underlying phenomena by learning and generalizing from examples without knowledge of generalization rules, while having a high degree of *error resistivity* making them almost insensitive to errors in a dataset [Müller et al., 2012]. Moreover, DNNs that combine linear and nonlinear feature transformations are able to capture long-term temporal correlations, in contrast to conventional linear or nonlinear predictive models [Müller et al., 2012].



(a) Input (b) Output

Figure 1: The plot (left) shows a fictive history of the prices of milk, butter and cheese. The data suggests that the price of milk causes the price of butter (with a delay of 1 month) and the price of cheese (with a delay of 3 months). Our framework receives the dairy prices as input and discovers the causal relationships and their delays to construct a temporal causal graph (right) showing the causal relationships and their delays between the prices of milk (M), butter (B) and cheese (C).

More specifically, in this report we present the Temporal Causal Discovery Framework (TCDF) that uses deep learning to:

1. Discover **causal relationships** in time series data,
2. Discover the **time delay** between cause and effect of each causal relationship,
3. Learn a **temporal causal graph** based on the discovered causal relationships with delays.

Our ambition is to provide a unified approach that exploits the representational power of deep learning and does not require any strong assumption on the data or underlying causal structure, in contrast to existing approaches.

For example, given a history of the prices of milk, butter and cheese, our framework could detect the causal relationships between these dairy products. Since milk is an ingredient of butter and cheese, the data should suggest that the price of milk causes the price of butter and that the price of milk causes the price of cheese. Furthermore, a change in the price of milk will probably be reflected in the price of butter resp. cheese with a certain delay. Figure 1 (left) shows a simple example containing fictive prices of milk, butter and cheese. It can be seen that the data suggests that the price of milk causes both the price of butter (with a delay of 1 month) and the price of cheese (with a delay of 3 months). Based on a dataset containing dairy prices, our framework can discover these causal relationships and their delays and can then visualize its findings in a temporal causal graph as shown in Figure 1 (right).

Thus, we require a dataset $\mathbf{X}$ containing N time series of the same length. These time series can be anything, from stock prices and weather data to water levels and heart rates. The goal is to discover the causal relationships between all N time series in $\mathbf{X}$, including the delay between cause and effect, and model this in a temporal causal graph.

Our framework, called Temporal Causal Discovery Framework (TCDF), consists of N convolutional neural networks, where each network receives all N observed time series as input. One network is trained to predict one time series. Thus, the goal of the j^{th} network $\mathcal{N}_j$ is to predict each time step of its *target* time series $\mathbf{X}_j \in \mathbf{X}$ based on past values. While a network performs supervised prediction, it trains its internal parameters using backpropagation.

We suggest to use these internal parameters for *unsupervised* causal discovery and delay discovery. More specifically, TCDF applies a successful explanation-producing method, *attention mechanisms*, to extract explanations from each network [Gilpin et al., 2018]. An attention mechanism in network $\mathcal{N}_j$ enables $\mathcal{N}_j$ to focus on a subset of its inputs. This allows us to learn to which time series the network *attends to* when predicting the target time series $\mathbf{X}_j$. Since neural networks can only learn correlations instead of causation, the attended time series will be at least correlated with the predicted time series $\mathbf{X}_j$ and might have a causal influence on $\mathbf{X}_j$.

After training the attention-based convolutional networks, TCDF distinguishes causation from correlation by applying a causal validation step. In this validation step, we intervene on a correlated time series to test if it is causally related with a predicted target time series. Each discovered time series that proves to be causal is included in a temporal causal graph that graphically shows the causal relationships between all time series.

In addition, we present a novel method to extract the learnt *time delay* between cause and effect of each relationship from a network by interpreting the network's internal parameters. TCDF then constructs a temporal causal graph in which both the discovered causal relationships and the discovered delays are included.

The rest of this report is organized as follows. Section 2 provides background information on temporal causal discovery with deep learning, and causal structure learning. Section 3 gives an overview of the existing, usually statistical, temporal causal discovery methods and presents the recent advances in non-temporal causal discovery with deep learning. Subsequently, Section 4 presents our Temporal Causal Discovery Framework (TCDF) based on attention-based convolutional neural networks. Our framework is evaluated on both simulated time series data and actual data in Section 5. Section 6 discusses some well-known issues that one should be aware of when interpreting the results of a causal discovery method. The conclusions, including future work, are discussed in Section 7.

2 Background

This section provides background information on temporal causal discovery (Section 2.1), gives a global overview on how deep learning can be used for temporal causal discovery (Section 2.2) and discusses the most important challenges faced by causal structure learning methods (Section 2.3).

2.1 Temporal Causal Discovery

Causality has been an important concept for decades, as it is important for interpretation, explanation, prediction, control and policy making. Although the notion of causality has shown to be evasive when trying to formalize, we assume that a causal relationship should comply with two aspects [Eichler, 2012]:
1) *Temporal precedence*: the cause precedes its effect,
2) *Physical influence*: manipulation of the cause changes its effect.

The first assumption can only be validated when the time steps of all time series in the dataset are physically aligned (i.e. time step t for one time series corresponds in the real world to time step t for another time series). Using temporal data with the temporal precedence assumption also allows for *delay discovery*, meaning that a model does not only detect the existence of a causal relationship but also the time delay between cause and effect.

The second aspect is usually defined in terms of interventions. Since a cause is a way of bringing about an effect, it can be understood in terms of how the probability or value of the effect changes when manipulating the cause. More specifically, an observed time series $\mathbf{X}_i$ is a cause of another observed time series $\mathbf{X}_j$ if there exists an intervention on $\mathbf{X}_i$ such that if all other time series $\mathbf{X}_{-i} \in \mathbf{X}$ are held fixed, $\mathbf{X}_i$ and $\mathbf{X}_j$ are associated [Woodward, 2005]. However, such controlled experiments in which other time series are held fixed may not be feasible in for example stock markets and many other time series applications. In those cases, researchers may be reluctant to test for physical influence. Another possibility, which is applied by TCDF, is to intervene on $\mathbf{X}_i$ while assuming that all other time series $\mathbf{X}_{-i} \in \mathbf{X}$ behave as usual.

2.2 Deep Learning for Temporal Causal Discovery

The following paragraphs present some background information on the main concepts of our Temporal Causal Discovery Framework: Convolutional Neural Networks, Attention Mechanisms and the Causal Quantitative Input Influence.

Convolutional Neural Networks for Time Series Prediction A Convolutional Neural Network is a type of feed-forward neural network, consisting of a sequence of convolutional layers. A convolutional layer of a CNN limits the number of connections to only some of the input neurons by sliding a *kernel* (a weight matrix) over the input and at each time step it computes the dot product between the input and the kernel. The kernel will then learn specific repeating patterns in the input series to forecast future values of the target time series. Intuitively, these learnt patterns denote correlations (and possibly causal relations) between the input series and the target output series which is important for causal discovery.

Until recently, Recurrent Neural Networks (RNNs), and in particular the Long-Short Term Memory unit (LSTM), were regarded as the default starting point to solve sequence learning. Because RNNs propagate forward a hidden state, they are theoretically capable of having infinite memory [Bai et al., 2018]. However, long-term information has to sequentially travel through all cells before getting to the present processing cell, causing the well-known *vanishing gradients* problem [Bengio et al., 1994], [Glorot and Bengio, 2010]. Other issues with RNNs are the high memory usage to store partial results and the impossibility of parallelism which makes them hard to scale and not hardware friendly [Bai et al., 2018].

RNNs are therefore slowly falling out of favor for modern convolutional architectures for sequence data. Convolutional Neural Networks (CNNs) are already successfully applied for sequence to sequence problems that need to convert sequences from one domain to sequences in another domain, including machine translation ([Gehring et al., 2017]) and image generation from text ([van den Oord et al., 2016]). However, although sequence to sequence modeling is related to our time series problem, the nature of those sequences is too dissimilar to apply the same architectures. The main difference is that in the aforementioned models, the entire input sequence (including "future" states) is used to predict each output which does not satisfy the

causal constraint that there can be no information 'leakage' from future to past. Furthermore, observational time series data usually contains only one repetition of the time series instead of observing a sequence several times. Convolutional architectures for time series are still scarce but recently successfully applied to financial time series. [Borovykh et al., 2017] created a deep convolutional model for noisy financial time series forecasting and [Binkowski et al., 2017] presented a deep convolutional network for predicting multivariate asynchronous time series.

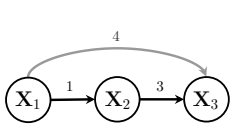
Attention Mechanism Recently, there has been a surge of work in *explaining* deep neural networks. One approach is to create an explanation-producing system that is designed to simplify the representation of its own behavior [Gilpin et al., 2018]. One successful explanation-producing method is the so-called *attention mechanism*. An attention mechanism (or 'attention' in short) equips a neural network with the ability to focus on a subset of its inputs. The concept of 'attention' has a long history in classical computer vision, where an attention mechanism selects relevant parts of the image for object recognition in cluttered scenes [Walther et al., 2004]. Only recently attention has made its way into deep learning. The idea of today's attention mechanism is to let the model learn what to *attend to* based on the input data and what is has learnt so far. Besides the increased accuracy by using attention, an important advantage is that it gives us the ability to *interpret* and visualize where the model attends to. This allowance of interpretability is why we propose attention mechanisms as a way to discover correlations.

Causal Validation To distinguish causation from correlation, we apply a causal validation step in which we intervene on the correlated time series to test the *Physical influence* assumption mentioned in Section 2.1. Only the time series that satisfy this constraint are considered to be causal and are included in the temporal causal graph.

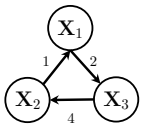
Since we only use observational data and will not physically intervene in a system to do experiments, we apply the *Causal Quantitative Input Influence (CQII)* measure of [Datta et al., 2016] to allow for causal reasoning. This measure models the difference in the "quantity of interest" between the real input distribution and an intervened distribution for a specific "input variable of interest". This hypothetical intervened distribution is constructed by retaining the marginal distribution over all other inputs and sampling the input of interest from its prior distribution. In this way, the influence of an input can be quantified by measuring the difference between the quantity of interest of the real input distribution and the intervened distribution.

As an example, the authors consider the case where an automated system assists in hiring decisions for a moving company [Datta et al., 2016]. Suppose that the input features used by this classification system are *Weight Lifting Ability* and *Gender*. These variables are positively correlated with each other and with the hiring decisions made. In this example, the "quantity of interest" is the fraction of positive classification for women. To test for the influence of gender on positive classification for women, CQII replaces every woman's field for gender with a random value. This value replacement is the 'intervention' used to construct the intervened distribution. To check for a causal association, the classification outcome based on the real input distribution is compared with the classification outcome of the intervened distribution where each gender field has a random value. If the intervention leads to a significant change in the classification outcome, then *Gender* is causally associated with positive classification.

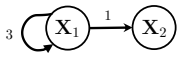
We propose to use CQII for causal discovery with deep learning, such that both the *temporal precedence* requirement and the *physical influence* requirement are satisfied. Details about the implementation of CQII in TCDF are described in Section 4.2.2.



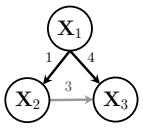
(a) $\mathbf{X}_1$ directly causes $\mathbf{X}_2$ with a delay of 1 time step, and indirectly causes $\mathbf{X}_3$ with a total delay of $1 + 3 = 4$ time steps.



(b) Feedback loop between $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ with a delay of 2, resp. 4 resp 1 time steps.



(c) Self-causation of $\mathbf{X}_1$ that repeatedly, with an interval of 3 time steps, causes $\mathbf{X}_2$ with a delay of 1 time step.



(d) $\mathbf{X}_1$ is a confounder of $\mathbf{X}_2$ and $\mathbf{X}_3$ with a delay of 1 resp. 4 time steps.

Figure 2: Temporal causal graphs showing causal relationships and their delays between cause and effect.

2.3 Causal Structure Learning and its Challenges

Structure learning is a model selection problem in which one estimates a graph that summarizes the dependence structure in a given data set [Drton and Maathuis, 2017]. In addition to performing temporal causal discovery which lists the causal relationships between observed time series, TCDF will perform *temporal causal structure learning* in which the discovered causal relationships between time series are visualized by a *causal* graph. This provides an intuitive understanding of the interrelations among the time series.

The starting point for causal graphical modeling is usually a directed graph in which an edge $e_{i,j}$ pointing from vertex v_i to vertex v_j represents a causal relationship from cause $\mathbf{X}_i$ to effect $\mathbf{X}_j$. The graphs learnt by non-temporal i.i.d. structure learning methods include a vertex for each variable in the used dataset. For the modeling of *temporal* structures, there are two visualizations methods, which we call *local* and *global* graphical methods.

Local temporal causal structure learning methods (e.g. [Peters et al., 2013], [Entner and Hoyer, 2010]) make an adaptation to the i.i.d. approach by replicating the set of variables by the number of time steps such that a vertex represents one time step in one time series $\mathbf{X}_i$. When no instantaneous effects are discovered, such learnt graphs will be acyclic by design. The lack of cycles follows from the *temporal precedence* assumption, such that an edge from an early time vertex cannot affect the past and will therefore always point to a later time vertex.

On the other hand, global graphical methods (e.g. [Budhathoki and Vreeken, 2018], [Jiao et al., 2013]) show that time series $\mathbf{X}_i$ causes time series $\mathbf{X}_j$ without taking a specific time step t into account . In such a learnt graph, a vertex denotes a time series $\mathbf{X}_i$ instead of referring to a specific time step in $\mathbf{X}_i$, which is comparable to a discovered graph by non-temporal i.i.d. structure learning methods in which a vertex corresponds to an i.i.d. variable. The acyclicity restriction does not apply here, since feedback loops and self-causation may be allowed. For better readability and the discovery of *global* causal relationships, our framework constructs a graph where a vertex denotes a time series.

More formaly, in the directed causal graph $\mathcal{G} = (V, E)$, vertex $v_i \in V$ represents an observed time series $\mathbf{X}_i$ and each directed edge $e_{i,j} \in E$ from vertex v_i to v_j denotes a causal relationship where time series $\mathbf{X}_i$ causes an effect in $\mathbf{X}_j$. Furthermore, we denote by $p = \langle v_i, ..., v_j \rangle$ a path in $\mathcal{G}$ from v_i to v_j. The length of a path (counted as the number of edges) is denoted as $|p|$.

In our *temporal* causal graph, every edge $e_{i,j}$ is annotated with a number $d(e_{i,j})$, that denotes the time delay between the occurrence of cause $\mathbf{X}_i$ and the occurrence of effect $\mathbf{X}_j$. An example of a simple temporal causal graph is shown in Figure 2a. The sum of the delays of all edges in path p is denoted as $d(p)$. The goal of this study is not only to perform temporal causal discovery, but also to learn a temporal causal graph from observational time series data.

However, structure learning methods are posed with major challenges when the underlying causal model is complex. First of all, the method should be able to distinguish **direct** causes from **indirect** causes (Fig. 2a). Vertex v_i is seen as an indirect cause of v_j if $e_{i,j} \notin \mathcal{G}$ and if there is a path $p = \langle v_i, ..., v_k, v_j \rangle \in \mathcal{G}$ with $|p| \geq 2$. Pairwise methods, i.e. methods that can only find causal relationships between *two* variables, are often unable to make this distinction and will include both $e_{i,j}$ (the grey edge in Fig. 2a) and $e_{k,j}$ in their graph $\mathcal{G}$, thus resulting in an incorrect inference [Hu and Liang, 2014]. In contrast, multivariate methods take all remaining variables into account to correctly distinguish between direct causality and

indirect causality [Papana et al., 2016], such that only $e_{k,j}$ would be included in their graph $\mathcal{G}$.

Secondly, it is relevant to correctly infer **instantaneous** causal effects, where the delay between cause and effect is 0 time steps. Neglecting instantaneous influences can lead to misleading interpretations of causal effects [Hyvärinen et al., 2008]. In practice, instantaneous effects mostly occur when cause and effect refer to the same time step that cannot be causally ordered a priori because of a too coarse time scale.

Moreover, it is important that a causal structure learning method does not rely on the idealized assumption that a directed graph should be acyclic. Since real-life systems may exhibit repeated behavior, there can be **feedback loops** (Fig. 2b) or **self-causation** (Fig. 2c) [Kleinberg, 2013].

Lastly, the presence of a **confounder**, a common cause of at least two variables, is a well-known challenge for structure learning methods (Fig. 2d). Although confounders are quite common in real-world situations, they complicate causal discovery since the confounder's effects are correlated with each other but they are not causally associated. Especially when the delays between the confounder and its effects are not equal, one should be careful to not incorrectly include a causal relationship between the confounder's effects (the grey edge in Fig. 2d). Going back to the milk price example from the Introduction, a machine learning model might incorrectly learn that the price of butter causes the price of cheese, since the delay between milk price ($\mathbf{X}_1$) and butter price ($\mathbf{X}_2$) is lower than the delay between the milk price and cheese price ($\mathbf{X}_3$) because of the long storage period of cheese.

It gets more complicated when such a confounder is not observed or measured, called a **hidden** (or latent) confounder. Although it might not even be known if or how many hidden confounders exist in the causal system, it is important that a structure learning method can hypothesise the existence of a hidden confounder to prevent learning an incorrect causal relation between its effects.

3 Related Work

Many different causal discovery algorithms have been developed to learn a causal graph from observational data. These algorithms are principally used to discover hypothetical causal relations between variables, in the context of other relevant or irrelevant variables. Most causal discovery methods construct a causal graph based on statistical tests. Pathways in the graph correspond to probabilistic dependence, and graphical non-adjacencies imply independence. These methods usually assume that the data satisfies the Causal Markov Condition, meaning that every variable in the dataset is independent of its non-effects conditional on its direct causes [Malinsky and Danks, 2018].

Literature distinguishes two common approaches to efficiently discover a causal graph structure given non-temporal i.i.d. (independent and identically distributed) observational data: score-based methods and constraint-based methods [Malinsky and Danks, 2018]. Score-based methods iteratively optimize a causal structure by scoring a specific structure on the basis of some measure of model fit, and return the causal structure with the best score. In contrast, constraint-based methods rule out all causal structures that are incompatible with a foreknown list of invariance properties, and return the set of causal graphs that imply exactly the (conditional) independencies found in the data [Kalisch and Bühlmann, 2014].

Temporal data present a number of distinctive challenges and can require quite different causal search algorithms [Malinsky and Danks, 2018]. Since there is no sense of time or prediction in the usual i.i.d. setting, causality as defined by the i.i.d. approaches is not philosophically consistent with causality for time series, as temporal data should also comply with the 'temporal precedence' assumption [Quinn et al., 2011]. Furthermore, an important difference is that in practice temporal observational data contains only one repetition of the time series instead of observing every variable several times [Peters et al., 2017].

For the scope of this chapter, we will introduce different categories for temporal causal discovery and give a selective overview of recent causal discovery algorithms for time series data in Section 3.1. We refer the reader to [Kalisch and Bühlmann, 2014] for an extensive review of non-temporal causal structure learning methods for non-temporal data. A more recent survey for non-temporal causal discovery techniques is [Singh et al., 2018] in which the authors present a comparative experimental benchmarking.

In Section 3.2, we propose the introduction of a new category for temporal causal discovery: Deep Learning. To the best of our knowledge, there does not yet exist a deep learning model for temporal causal discovery. However, we discuss some recent causal discovery methods for *non*-temporal observational data that use deep learning techniques.

3.1 Temporal Causal Discovery Methods

Table 1 shows recent temporal causal discovery models, categorised in five different approaches and assessed along various dimensions. Each approach is discussed in more detail in the paragraphs below Table 1. The table only reflects some of the most recent approaches for each type of model, since the amount of literature is very large.

Features The subcolumns in the 'Features' column in Table 1 denote if the algorithm can deal with confounders and cyclic graphs (i.e. feedback loops and self-causation), and if it can measure instantaneous effects, delay between cause and effect and the causal strength of a causal relationship. We consider causal strength as being some kind of quantitative measure that indicates how strongly one time series influences another.

Data The subcolumns in the 'Data' column in Table 1 denote if the algorithm can deal with specific types of data, namely multivariate, continuous, non-stationary, non-linear and noisy data. Stationarity is a common assumption in many time series techniques, meaning that the joint probability distribution of the stochastic process does not change when shifted in time [Papana et al., 2014]. Furthermore, some models require discrete data and cannot handle continuous values. Note that one may choose to discretize continuous variables, but different discretizations can yield different causal structures. Furthermore, discretization can also make non-linear causal dependencies difficult to detect [Malinsky and Danks, 2018].

Algorithm	Method	Features						Data					Output
		Confounders	Hidden Confounders	Cyclicity	Instantaneous	Delay	Causal Strength	Multivariate	Continuous	Non-Stationarity	Non-Linearity	Noise	
$\alpha(c, e)$ Huang and Kleinberg, 2015	Causal Significance	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	Causal relationships, delay and impact
CGC Hu and Liang, 2014	Granger	✗	✗	✓	✗?	✓	✓	✓	✓	✗	✓	✓	Causal relationships with causal influence
PCMCI Runge et al., 2017	Constraint-based	✓	✓?	✓	✓	✓	✓	✓	✓	✗	✓	✓	Causal time series graph, delay and causal strength
ANLTSM Chu and Glymour, 2008	Constraint-based	✓	✓[1]	✗[2]	✓	✓	✓	✓	✓	✓	✓	✓	Partial Ancestral Graph with node for each time step
tsFCI Entner and Hoyer, 2010	Constraint-based	✓	✓	✗	✓	✓	✗	✓	✓	?	✓	✓[3]	Partial Ancestral Graph with node for each time step
TiMINo Peters et al., 2013	Structural Equation Model	✓	✓[4]	✗[5]	✓	✓[6]	✗	✓	✓	✗	✓	✓	Graph with node for each time step (or remains undecided)
PSTE Papana et al., 2016	Information-theoretic	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	Causal Relationships
SDI Jiao et al., 2013	Information-theoretic	✗	✗	✓	✗	✓	✓	✗	✗	✓	✓	✓?	Causal relationships with a 'degree of causation'
CUTE Budhathoki and Vreeken, 2018	Information-theoretic	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	Causal graph

Table 1: Causal discovery methods for time series data, classified among various dimensions. A '?' indicates that we are unsure.

Granger Causality (GC) Granger, 1969 is one of the earliest methods developed to quantify the causal effects among two time series (therefore called a 'pairwise' method). It is based on the common conception that a cause occurs prior to its effect. More precisely, time series $\mathbf{X}_i$ *Granger causes* time series $\mathbf{X}_j$ if the future value of $\mathbf{X}_j$ (at time $t+1$) can be better predicted by using both the values of $\mathbf{X}_i$ and $\mathbf{X}_j$ up to time t than by using only the past values of $\mathbf{X}_j$ itself. However, in practice not all relevant variables may be observed or measured. This reveals an important shortcoming of GC; it cannot correctly deal with unobserved time series, including hidden confounders Bahadori and Liu, 2013.

Furthermore, although GC is successfully applied across many domains, it only captures the linear interdependencies among time series. Various extensions have been made to nonlinear and higher-order causality, e.g. Ancona et al., 2004, Marinazzo et al., 2008 and Luo et al., 2013. A more recent extension that outperforms other Granger causality methods is based on conditional copula, that allows to dissociate the marginal distributions from their joint density distribution to focus only on statistical dependence between variables for uncovering the temporal causal graph Hu and Liang, 2014.

[1] Requires hidden confounders to be instantaneous and linear.
[2] The authors present another version of the model that allows feedback loops, but only in the absence of hidden confounders.
[3] Assumes Gaussian noise.
[4] TiMINo stays undecided by not inferring a causal relationship in case of a hidden confounder.
[5] Cyclicity is theoretically shown, but the algorithm is only implemented to produce acyclic graphs.
[6] Although theoretically shown, the implemented algorithm does not explicitly output the discovered time delays.

Constraint-based Time Series approaches are often adapted versions of non-temporal causal graph discovery algorithms for random variables. As an additional advantage, the temporal precedence constraint helps reduce the search space of the causal structure [Spirtes and Zhang, 2016]. The well-known causal discovery algorithms PC and FCI both have a time series version: PCMCI [Runge et al., 2017] and tsFCI [Entner and Hoyer, 2010].

The PC algorithm (named after its authors, Peter and Clark) [Spirtes et al., 2000] makes use of a clever series of tests to efficiently explore the whole space of DAGs (Directed Acyclic Graphs). FCI (Fast Causal Inference) [Spirtes et al., 2000] is a constraint-based algorithm that, contrary to PC, can deal with hidden confounders by using independence tests on the observed data. However, both algorithms produce an acyclic graph and therefore do not allow feedback loops. Besides, [Chu and Glymour, 2008] developed ANLTSM (Additive Non-linear Time Series Model) for causal discovery in both linear and non-linear time series data, that can also deal with hidden confounders. It uses statistical tests based on additive model regression.

Structural Equation Model approaches assume that the causal system can be represented by a Structural Equation Model (SEM) that describes a variable $\mathbf{X_j}$ as a function of other substantive variables $\mathbf{X_{-j}}$ and a unique error term ϵ_X to account for additive noise such that $X := f(\mathbf{X_{-j}}, \epsilon_X)$ [Spirtes and Zhang, 2016]. It assumes that the set $\mathbf{X_{-j}}$ is jointly independent. SEM approaches are applied in the i.i.d. setting, but [Peters et al., 2013] presented TiMINo (Time Series Model with Independent Noise) for the case of stationary time series data. TiMINo associates the SEM with a directed graph that contains each time step $X_i^t \in \mathbf{X}_i$ as a node in the so-called *full time graph*. There is a directed edge from X_i^t to X_j^t, $i \neq j$, if the coefficient of X_i^t is nonzero for X_j^t. The resulting *summary time graph* contains all time series as vertices in which there is an edge from $\mathbf{X}_i$ to $\mathbf{X}_j$ if there exists an edge from X_i^{t-k} to X_j^t in the full time graph for some k.

Note, since TiMINo requires $i \neq j$, that self-causation is not allowed. Furthermore, TiMINo remains undecided if the direct causes of X_i are not independent, instead of drawing possibly wrong conclusions. However, the main disadvantage is that TiMINo is not suitable for large datasets, since even smallest differences between the true data and the model may lead to rejected independence tests. Furthermore, the authors state that the results from a high-dimensional dataset (more than ten time series) should be interpreted carefully.

There are also **Information-theoretic approaches** for temporal causal discovery such as (mutual) shifted directed information [Jiao et al., 2013] and transfer entropy [Papana et al., 2016]. Their main advantage is that they are model free and make no assumption for the distribution of the data, while being able to detect both linear and non-linear dependencies [Papana et al., 2014]. The universal idea is that $\mathbf{X}_i$ is likely a cause of $\mathbf{X}_j$, $i \neq j$, if $\mathbf{X}_j$ can be better sequentially compressed given the past of both $\mathbf{X}_i$ and $\mathbf{X}_j$ than given the past of $\mathbf{X}_j$ alone.

Compared to transfer entropy, directed information can be extended to more general systems, is not limited to stationary Markov processes and is able to quantify the instantaneous causality [Liu and Aviyente, 2012]. To solve the problem of transfer entropy not being able to deal with non-stationary time series, [Papana et al., 2016] introduced Partial Symbolic Transfer Entropy (PSTE). However, PSTE is not effective when only linear causal relationships are present in the underlying causal system.

Besides, [Budhathoki and Vreeken, 2018] introduced CUTE (Causal Inference on Event Sequences) that is claimed to be more robust than transfer entropy, but can only handle discrete data.

Causal Significance is a causal discovery framework that exploits the connections between each causal relationship's relative levels of significance [Huang and Kleinberg, 2015]. It calculates a causal significance measure $\alpha(c, e)$ for a specific cause-effect pair by isolating the impact of cause c on effect e. The advantage of this method is that it does not only discover a causal relationship, but also infers its delay and the impact (e.g. e's value raises with 2 units). However, the method assumes that causal relationships are linear and additive (i.e. the value of a variable at any time is given by the sum of the impact of its causes), and that all genuine causes are observed. But, the authors experimentally demonstrate that low false discovery and negative rates are achieved if some of these assumptions do not hold.

3.2 Causal Discovery Methods based on Deep Learning

Existing temporal causal discovery methods, as discussed in the previous section, are mainly based on statistical measures. Deep Learning, the approach we propose, is not yet used for temporal causal discovery. The only study we found that compared deep learning with existing causality measures, was [Guo et al., 2018] that proposed an interpretable LSTM network to characterize variable importance. Using an attention mechanism, the important variables found by the LSTM showed to be highly in line with those determined by the Granger causality test [Granger, 1969]. This exhibits the prospect that deep learning methods are suitable for causal discovery.

Although deep learning is not applied for temporal causal discovery, [Louizos et al., 2017] presented a method based on Variational Autoencoders to estimate causal effects for *non*-temporal observational data. Only recently, two methods were presented that aim to discover causal relationships from non-temporal observational data using neural networks. [Goudet et al., 2018] introduced Causal Generative Neural Networks (CGNN) to learn functional causal models from non-temporal observational data. Despite its good performances and lack of assumptions regarding confounders, CGNNs make the unrealistic assumption of a known graph skeleton for the graphical causal model such that only the directed edges need to be oriented.

The same authors later presented the Structural Agnostic Model (SAM) [Kalainathan et al., 2018]. They use Generative Adversarial Neural Networks for causal graph reconstruction from non-temporal continuous observational data. Each network is trained to predict one variable. Although called 'causal filters' by the authors, SAM uses a version of attention by multiplying each observed input variable by a trainable attention score. SAM estimates a causal relationship if this attention score is greater than a given threshold. However, this threshold should be specified by the user which can be hard to estimate. Besides, their loss function includes a penalty for the attention scores, which results in an unusual behaviour in which attention scores will decrease the longer the model is trained. Furthermore, SAM neglects to check for *physical influence* since no causal validation step is performed. A consequence is that SAM cannot distinguish between correlation and the presence of a hidden confounder.

4 Temporal Causal Discovery Framework

This chapter introduces and explains our Temporal Causal Discovery Framework (TCDF). Figure 3 gives a global overview of TCDF, showing that TCDF applies 4 steps to learn a Temporal Causal Graph from observational data: Correlation Discovery, Causal Discovery, Delay Discovery and Graph Construction.

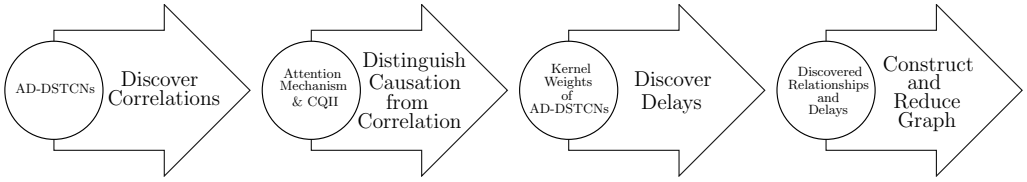


Figure 3: Overview of our Temporal Causal Discovery Framework (TCDF). The arrows describe the steps taken by TCDF, while the circles describe what TCDF uses to perform the corresponding step.

More specifically, our Temporal Causal Discovery Framework (TCDF) consists of N independent attention-based convolutional neural networks, all having the same architecture but a different target time series. An overview of TCDF containing multiple networks is shown in Figure 4. This shows that the goal of the j^{th} network $\mathcal{N}_j$ is to predict its *target* time series $\mathbf{X}_j$ by minimizing the loss $\mathcal{L}$ between the actual values of $\mathbf{X}_j$ and the predicted $\hat{\mathbf{X}}_j$. The input to network $\mathcal{N}_j$ consists of a $N \times T$ dataset $\mathbf{X}$ consisting of N equal-sized time series of length T. Row $\mathbf{X}_j$ from the dataset corresponds to the *target* time series, while all other rows in the dataset, $\mathbf{X}_{-j}$, are the so-called *exogenous* time series.

When network $\mathcal{N}_j$ is trained to predict $\mathbf{X}_j$, the attention scores $\mathbf{a}_j$ of the attention mechanism explain where network $\mathcal{N}_j$ attends to when predicting $\mathbf{X}_j$. Since the network uses the attended time series for prediction, the attended time series should contain information that is useful for prediction, implying that the attended time series are correlated with the predicted target time series $\mathbf{X}_j$. TCDF therefore use the attention scores to discover which of the exogenous time series are correlated with the target $\mathbf{X}_j$. By including the target time series in the input as well, the attention mechanism can also learn self-causation. We designed a specific architecture for these attention-based convolutional networks that allows TCDF to discover these correlations. We call our networks Attention-based Dilated Depthwise Separable Temporal Convolutional Networks (AD-DSTCNs). Based on the attention scores of AD-DSTCN $\mathcal{N}_j$, TCDF can discover which time series are correlated with $\mathbf{X}_j$.

The rest of this chapter is structured as follows: Section 4.1 describes the architecture of AD-DSTCNs and discusses in more detail how TCDF uses these to discover correlations in a dataset. Section 4.2 describes the second step of TCDF (shown in the middle of Figure 4) distinguishing causal relationships from all discovered correlations by interpreting the attention results and applying the Causal Quantitative Input Influence (CQII) to validate if a correlation is a causation. As a third step, TCDF discovers the time delay between the cause and effect of each discovered causal relationship. For this delay discovery, TCDF uses the kernel weights $\mathcal{W}_j$ of each AD-DSTCN $\mathcal{N}_j$, which will be discussed in more detail in Section 4.3. Lastly, TCDF merges the results of all networks to construct a Temporal Causal Graph that graphically shows the discovered causal relationships and their delays. For better readability, TCDF applies a graph reduction step that removes all discovered indirect causes. Section 4.4 describes the graph construction and reduction in more detail.

4.1 Correlation Discovery with AD-DSTCNs

We have designed a convolutional neural network architecture that can be used to predict a time series, while the included attention mechanism can be used to extract learnt correlations from the network. Paragraphs 4.1.1 to 4.1.7 describe all aspects of the AD-DSTCN architecture used by our framework. Subsequently, paragraph 4.1.8 describes how the attention scores from these networks are used by TCDF for correlation discovery.

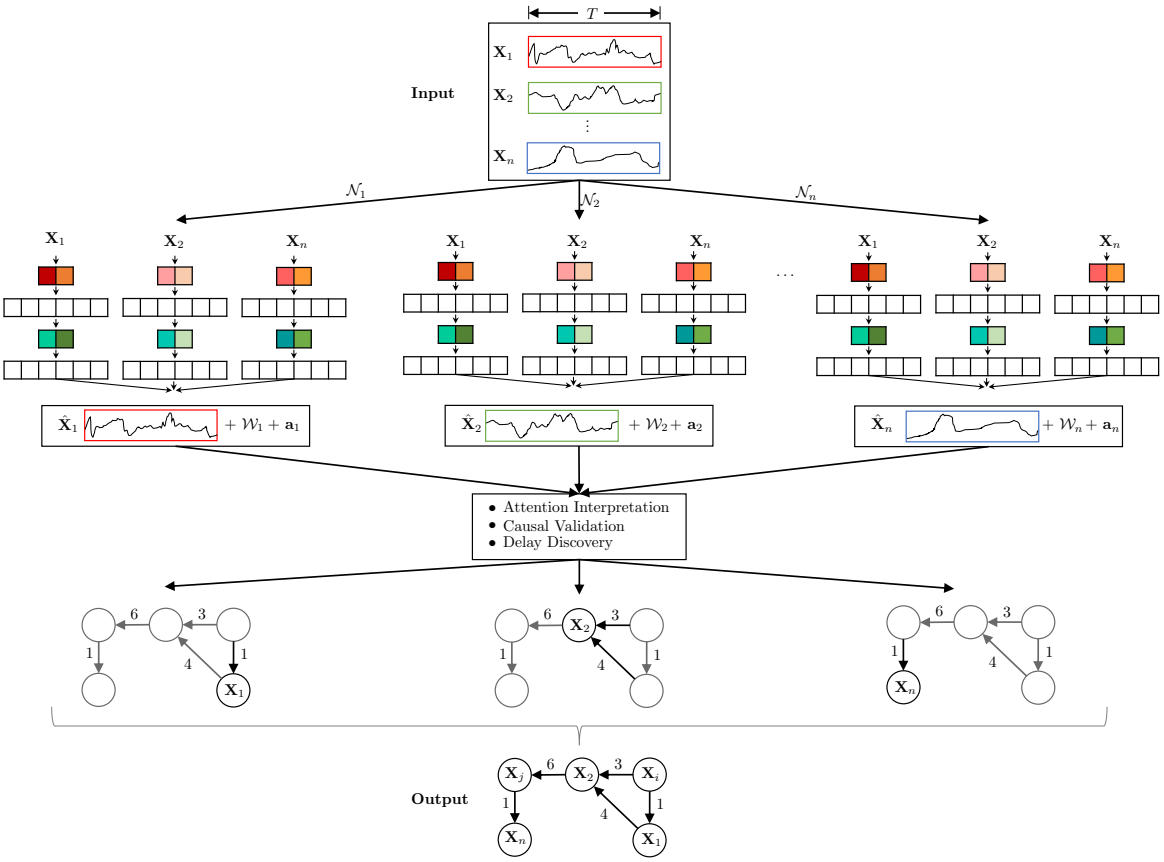


Figure 4: Temporal Causal Discovery Framework (TCDF) with N independent networks $\mathcal{N}_1 ... \mathcal{N}_n$, all having time series $\mathbf{X}_1 ... \mathbf{X}_n$ of length T as input. $\mathcal{N}_j$ predicts $\mathbf{X}_j$ and outputs besides $\hat{\mathbf{X}}_j$ also the kernel weights $\mathcal{W}_j$ and attention scores $\mathbf{a}_j$. After attention interpretation, causal validation and delay discovery, TCDF constructs a temporal causal graph.

4.1.1 Temporal Convolutional Network

An important restriction for time series prediction is that a neural network may not access future values when predicting the current value of a time series. Therefore, we use the generic Temporal Convolutional Network (TCN) architecture of [Bai et al., 2018] as a basis for our network architecture. This convolutional architecture has configurable settings that can be used for univariate time series modeling. Having one time series $\mathbf{X}_1$ (which could be the price of milk for example) as input, TCN can predict a different target time series $\mathbf{X}_2$ (say, the price of cheese). TCN consists of a 1-dimensional (1D) Convolutional Neural Network architecture in which each layer has length T, where T is the number of time steps of the input time series and the equally-sized target time series. TCN uses supervised learning by minimizing the loss $\mathcal{L}$ between the actual values of target $\mathbf{X}_2$ and the predicted $\hat{\mathbf{X}}_2$. A TCN predicts time step t of the target time series based on the past and current values of the input time series, i.e. from time step 1 up to and including time step t. Including the current value of the input time series enables the detection of instantaneous effects. Since no future values are used for prediction, it satisfies the causal time constraint that future information cannot cause an effect. Therefore, a TCN uses a so-called *causal convolution* in which there is no information 'leakage' from future to past.

A TCN predicts each time step of the target time series $\mathbf{X}_2$ by sliding a kernel over input $\mathbf{X}_1$ of which the input values are $[X_1^1, X_1^2, ..., X_1^t, ..., X_1^T]$. When predicting the value of $\mathbf{X}_2$ at time step t, denoted as X_2^t, the 1D kernel with a user-specified size K calculates the dot product between the learnt kernel weights $\mathcal{W}$, and the current input value plus its $K-1$ previous values, i.e. $\mathcal{W} \odot [X_1^{t-K+1}, X_1^{t-K+2}..., X_1^{t-1}, X_1^t]$.

However, when the first value of $\mathbf{X}_2$, denoted as X_2^1, has to be predicted, the input data only consists of X_1^1 and past values are not available. This means that the kernel cannot fill its kernel size if $K > 1$. Therefore, TCN applies *left zero padding* such that the kernel can access $K-1$ values that equal zero to replace the missing past values. For example, if $K = 4$, the sliding kernel first sees $[0, 0, 0, X_1^1]$, followed by $[0, 0, X_1^1, X_1^2]$, $[0, X_1^1, X_1^2, X_1^3]$, etc. until $[X_1^{T-3}, X_1^{T-2}, X_1^{T-1}, X_1^T]$.

4.1.2 Discovering Self-causation

Whereas the authors of TCN assume that the input time series is different than the output time series, we propose to allow the input and output time series to be the same. Having the target time series $\mathbf{X}_j$ as input data allows us to discover self-causation (i.e. the target time series causally influences itself, which enables the modeling of repeated behavior). However, in this case we have to slightly adapt the TCN architecture of [Bai et al., 2018], since we cannot include the current value of the target time series in the input. With an exogenous time series as input, the sliding kernel with size K can access $[X_i^{t-K+1}, X_i^{t-K+2}..., X_i^{t-1}, X_i^t]$ with $i \neq j$ to predict X_j^t for time step t. However, with the target time series as input, the kernel may only access the *past* values of the target time series $\mathbf{X}_j$, i.e. excluding the current value X_j^t since that is the value to be predicted.

Therefore, we have to make sure that this current value cannot be seen by the kernel. A simple solution to this is to shift the target input data one time step forward with left zero padding such that the input target time series in the dataset equals $[0, X_j^1, X_j^1, ..., X_j^{T-1}]$ and the kernel therefore can access $[X_j^{t-K}, X_j^{t-K+1}..., X_j^{t-2}, X_j^{t-1}]$ to predict X_j^t.

4.1.3 Multivariate Causal Discovery

A restriction of the TCN architecture is that it is designed for *univariate* time series modeling, meaning that there is only one input time series. Multivariate time series modeling in convolutional neural networks is usually achieved by merging multiple time series into a 2D-input. This architecture, having 1 channel, is shown in Figure 5. Instead of having a 1D-kernel with kernel size K as in the normal TCN architecture, there is a 2D-kernel in the first convolutional layer with a width of size K and a height that is equal to the number of input time series. The kernel slides over the 2D-input such that the kernel weights are element-wise multiplied with the input. This creates a 1D-output in the first hidden layer. For a deep TCN, 1D-convolutional layers can be added to the architecture. However, the disadvantage of this approach is that the output from each convolutional layer is always 1 dimensional, meaning that the input time series are mixed. This mixing of inputs hinders causal discovery.

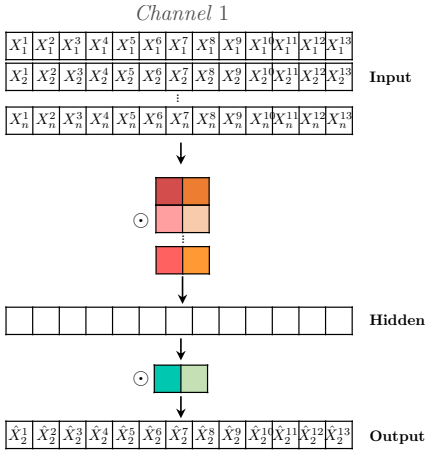


Figure 5: A Multivariate Temporal Convolutional Network that predicts $\mathbf{X}_2$ based on a 2D-input containing N time series with $T = 13$ time steps. The network has $L = 1$ hidden layer and 2 kernels with kernel size $K = 2$ (denoted by the colored blocks). The kernel in the first layer has a kernel height of N. The time series are element-wise multiplied with the kernel weights (denoted by $\odot$).

To allow for multivariate causal discovery, we therefore extend the univariate TCN architecture to a 1-dimensional **Depthwise Separable Temporal Convolutional Network** (DSTCN) in which the input time series stay separated. A DSTCN consists of one channel for each input time series making N channels in total. Thus, in network $\mathcal{N}_j$, channel j corresponds to the target time series $\mathbf{X}_j = [0, X_j^1, X_j^1, ..., X_j^{T-1}]$ and all other channels correspond to the exogenous time series $\mathbf{X}_{i \neq j} = [X_i^1, X_i^1, ..., X_i^{T-1}, X_i^T]$. An overview of this architecture is shown in Figure 6. This overview includes the attention mechanism that will be discussed in Section 4.1.7.

A depthwise separable convolution consists of *depthwise convolutions*, where channels are kept separate by applying a different kernel to each input channel, followed by a 1×1 *pointwise convolution* that merges together the resulting output channels [Chollet, 2017]. This is different than normal convolutional architectures that have just one kernel per layer. Because of the separate channels, the kernel weights relate to one specific time series which allows us to correctly interpret the relation between a specific input time series and the target time series, without any mixing of inputs. This shows to be useful for our delay discovery.

But, although a separate channel for each input time series is useful for correctly interpreting how one specific time series influences the target, it is not sufficient for accurate time series prediction. When predicting the target time series $\mathbf{X}_j$ conditional on another time series $\mathbf{X}_i$ where $i \neq j$, we should also include the past values of target $\mathbf{X}_j$. More formally, we aim at maximizing the conditional likelihood:

$$P(\mathbf{X}_j | \mathbf{X}_{i \neq j}) = \prod_{t=1}^{T} P(X_j^t | X_i^1, ..., X_i^t, X_j^1, ..., X_j^{t-1}). \tag{1}$$

Adopting the idea from [Borovykh et al., 2017] for time series prediction, the conditioning on past values of $\mathbf{X}_j$ is done by element-wise addition of the target convolutional output from the first layer to the convolutional outputs from the first layer of the other channels. The element-wise addition is indicated by $\oplus$ in Figure 6. With this addition, we ensure that each channel uses not only the past and current values of their input time series, but also the past values of the target time series.

4.1.4 Activation Functions

For non-linearity, a non-linear activation function is needed that transforms the output of a convolution. Although a Rectified Linear Unit (ReLU) is used in the original TCN architecture of [Bai et al., 2018], we use a Parametric Rectified Linear Unit (PReLU), defined as $\text{PReLU}(x) := max(x, 0) + \alpha_j \cdot min(x, 0)$ with α_j being a learnable parameter for input time series $\mathbf{X}_j$. Although any other non-linear activation function can be applied as well, we chose PReLU since it has found to be most efficient when applied to the

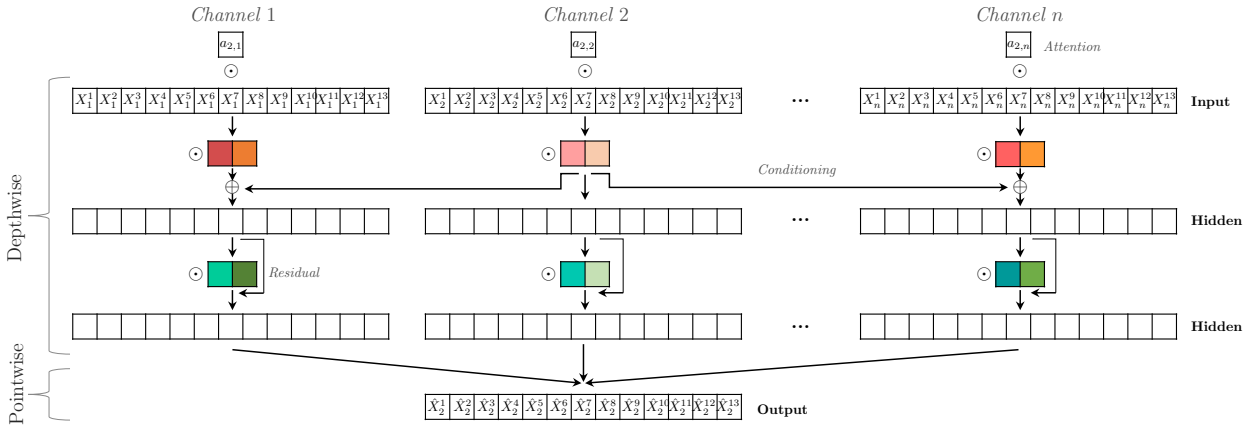


Figure 6: Attention-based Dilated Depthwise Separable Temporal Convolutional Network $\mathcal{N}_2$ to predict its target time series $\mathbf{X}_2$, having N channels with $T = 13$ time steps, $L = 2$ hidden layers and $N \times L$ kernels with kernel size $K = 2$ (denoted by the colored blocks). The attention scores a are element-wise multiplied with the input time series, followed by an element-wise multiplication with the kernel. The output of the first convolution from target $\mathbf{X}_2$ is element-wise added to the other outputs before being inputted to the next convolutional layer. In the pointwise convolution, all output channels are combined to construct the prediction $\hat{\mathbf{X}}_2$.

forecasting of non-stationary, noisy financial time series [Borovykh et al., 2017], and has shown to improve model fitting with nearly zero extra computational cost and little overfitting risk compared to the traditional ReLU [He et al., 2015].

But, since we have a regression task, the network needs to be able to approximate any real value, without being changed by a non-linear activation function. Therefore, we use the common setup that applies a linear activation function in the last hidden layer[7]. Moreover, it has been shown that neural networks that combine linear and nonlinear feature transformations are able to capture long-term temporal correlations [Müller et al., 2012].

4.1.5 Residual connections

An increasing number of hidden layers in a network usually results in a higher training error. This accuracy degradation, called the *degradation problem*, is not caused by overfitting, but because standard backpropagation tends to become unable to find optimal weights in a deep network [He et al., 2016]. The proven way around this problem is to use residual connections. A convolution layer transforms its input x to $\mathcal{F}(x)$, after which an activation function is applied. With a residual connection, the input x of the convolutional layer is added to $\mathcal{F}(x)$ such that the output o is:

$$o = \text{PReLU}(x + \mathcal{F}(x)) \tag{2}$$

We add a residual connection in each channel after each convolution from the input to the convolution to the output, as shown in Figure 6. We only exclude the first layer since here already the target conditioning takes place.

[7] If the network has exactly 1 layer (i.e. no hidden layers), only the PReLU activation is applied to allow for non-linearity.

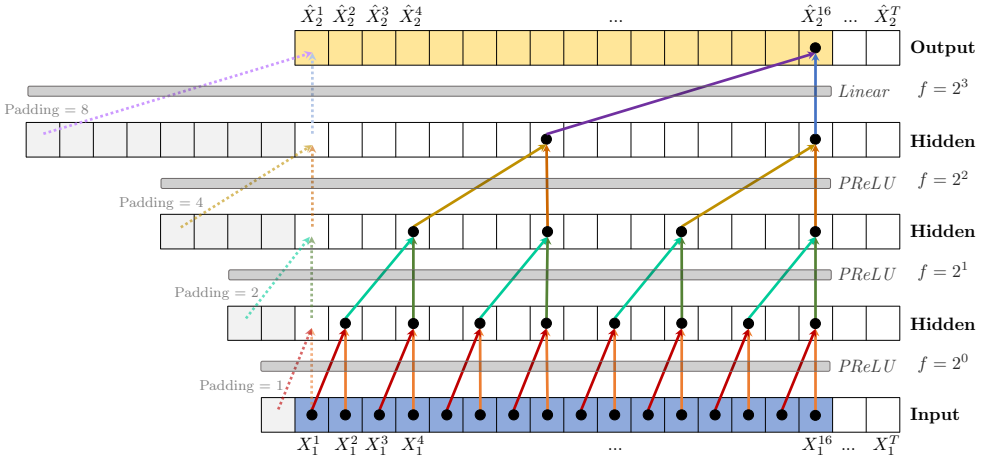


Figure 7: Channel 1 of a dilated DSTCN to predict $\mathbf{X}_2$, with $L = 3$ hidden layers, kernel size $K = 2$ (denoted by the arrows) and dilation coefficient $c = 2$, leading to a receptive field $R = 16$. A ReLU activation function is applied after each convolution. To predict the first values (shown by the dashed arrows), zero padding is added to the left side of the sequence. Weights are shared across layers, indicated by the identical colors.

4.1.6 Dilations

In a TCN with only one layer (i.e. no hidden layers), the *receptive field* (the number of time steps seen by the sliding kernel) is equal to the user-specified kernel size K. To successfully discover a causal relationship, the receptive field should be as least as large as the delay between cause and effect. To increase the receptive field, one can increase the kernel size or add hidden layers to the network.

A 1D convolutional network has a receptive field that grows linearly in the number of layers, which is computationally expensive when a large receptive field is needed. More formally, the receptive field R of a 1D CNN is

$$R_{CNN} = 1 + \sum_{l=0}^{L} (k - 1) \tag{3}$$

where K is the user-specified kernel size and L the number of hidden layers. ($L = 0$ corresponds to a network without hidden layers, where one convolution in a channel maps an input time series to the output.)

The well-known WaveNet architecture [Van Den Oord et al., 2016] therefore employed *dilated convolutions*. A dilated convolution is a convolution where a kernel is applied over an area larger than its size by skipping input values with a certain step size f. This step size f, called *dilation factor*, increases exponentially depending on the chosen *dilation coefficient* c, such that $f = c^l$ for layer l. An example of dilated convolutions is shown in Figure 7.

With an exponentially increasing dilation factor f, a network with stacked dilated convolutions can operate on a coarser scale without loss of resolution or coverage. We therefore implement the dilated convolutions to create a Dilated DSTCN (D-DSTCN). The receptive field R of a kernel in a 1D Dilated DSTCN is:

$$R_{D-DSTCN} = 1 + \sum_{l=0}^{L} (k - 1) \cdot c^l \tag{4}$$

This shows that dilated convolutions support exponential increase of the receptive field, while the number of parameters grows only linear.

4.1.7 Attention Mechanism

The D-DSTCN architecture as described in the previous paragraphs is suitable for time series prediction. However, we need to add a method to the network architecture that extracts explanations from the network, such that our framework TCDF can discover which time series are correlated with the predicted time series. We therefore add an explanation-producing method called 'attention mechanism' [Gilpin et al., 2018] to our network architecture. We call these attention-based networks 'Attention-based Dilated Depthwise Separable Temporal Convolutional Networks' (AD-DSTCNs).

An attention mechanism (or *attention* in short) equips a neural network with the ability to focus on a subset of its inputs. Prior work on attention in deep learning mostly addresses recurrent networks, but Facebook's FairSeq [Gehring et al., 2017] for neural machine translation and the Attention Based Convolutional Neural Network (ABCNN) [Yin et al., 2016] for modelling sentence pairs have shown that attention is very effective in CNNs as well. Besides the increased accuracy, attention allows us to interpret where the network attends to. Thus, after training a network on predicting the target time series $\mathbf{X}_j$, we can identify to which input time series network $\mathcal{N}_j$ attended to. These attended time series should be at least correlated with the predicted target time series and might have a causal influence on the target.

Typically, attention is implemented as a trainable $1 \times N$-dimensional vector $\mathbf{a}$ that is element-wise multiplied with the N input time series or with the output of another neural network. Each value $a \in \mathbf{a}$ is called an *attention score*. In our framework, each network $\mathcal{N}_j$ has its own attention vector $\mathbf{a_j} = [a_{1,j}, a_{2,j}, ..., a_{j,j}, ..., a_{N,j}]$. Attention score $a_{i,j}$ is multiplied with input time series $\mathbf{X_i}$ in network $\mathcal{N}_j$. This is indicated with $\odot$ at the top of Figure 6. Thus, attention score $a_{i,j} \in \mathbf{a_j}$ shows how much $\mathcal{N}_j$ attends to input time series $\mathbf{X}_i$ for predicting target $\mathbf{X}_j$. A high value for $a_{i,j} \in \mathbf{a_j}$ means that $\mathbf{X_i}$ is correlated with $\mathbf{X}_j$ and might cause $\mathbf{X}_j$. A low value for $a_{i,j}$ means that $\mathbf{X_i}$ is not correlated with $\mathbf{X}_j$. Note that $i = j$ is possible since we allow self-causation. The attention scores will be used after training of the networks to determine which time series are correlated with a target time series.

4.1.8 Correlation Discovery

Our framework has one AD-DSTCN for each time series $\mathbf{X}_j \in \mathbf{X}$. All N AD-DSTCNs have the same architecture, but only their target time series is different. Network $\mathcal{N}_j$ is trained to predict its target time series $\mathbf{X}_j$ with backpropagation by minimizing the error between the actual values of $\mathbf{X}_j$ and the predicted values $\hat{\mathbf{X}}_j$. The number of training epochs, loss function and optimization algorithm can be selected by the user.

When the training of the network starts, all attention scores are initialized as 1 such that $\mathbf{a}_j = [1, 1, ..., 1]$. While the networks use back-propagation to predict their target time series, the network also changes its attention scores such that each score is either increased or decreased in every training epoch. This means that after some training epochs, $\mathbf{a}_j \in [-\infty, \infty]^N$ although the boundaries depend on the number of training epochs and the specified learning rate.

Literature distinguishes between *soft* attention where $\mathbf{a}_j \in [0, 1]^N$, and *hard* attention where $\mathbf{a}_j \in \{0, 1\}^N$. Soft attention is usually realized by applying the Softmax function σ to the attention scores such that $\sum_{i=1}^{N} a_{i,j} = 1$. A limitation of the Softmax transformation is that the resulting probability distribution always has full support, i.e. $\sigma(a_{i,j}) \neq 0$ [Martins and Astudillo, 2016].

Intuitively, one would prefer hard attention for correlation discovery, since the network should make a binary decision: a time series is either correlated (and possibly causally related) or non-correlated. However, hard attention is non-differentiable due to its discrete nature and therefore cannot be optimized through back-propagation [Shen et al., 2018]. We therefore first use the soft attention approach by applying the Softmax function σ to each $a \in \mathbf{a}_j$ in each training epoch. After training network $\mathcal{N}_j$, we apply our straightforward semi-binarization function HardSoftmax that truncates all attention scores that fall below a threshold τ_j to zero:

$$h = \text{HardSoftmax}(a) = \begin{cases} \sigma(a) & \text{if } a \geq \tau_j \\ 0. & \text{if } a < \tau_j \end{cases} \tag{5}$$

We denote by $\mathbf{h}_j$ the set of attention scores in $\mathbf{a}_j$ to which the HardSoftmax function is applied. Time series $\mathbf{X}_i$ is considered to be *correlated* with the target time series $\mathbf{X}_j$ if $h_{i,j} \in \mathbf{h}_j > 0$.

Manually estimating this threshold τ_j for target $\mathbf{X}_j$ can be challenging. We therefore created an algorithm that determines τ_j by finding the largest *gap* between the attention scores in $\mathbf{a}_j$. The algorithm orders the

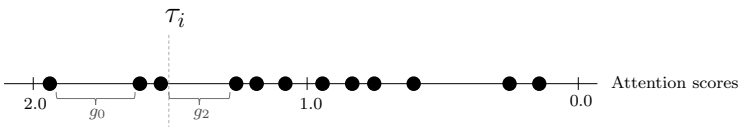


Figure 8: Threshold τ_j is set equal to the attention score at the left side of the largest gap g_k where $k \neq 0$ and $k < |\mathbf{G}|/2$. In this example τ_j is set equal to the third attention score.

attention scores from high to low and searches for the largest gap g between two adjacent attention scores $a_{i,j}$ and $a_{k\neq i,j}$, such that a split can be made between 'high' scores and 'low' scores. Threshold τ_j is set equal to the attention score on the left side of the gap. This approach is graphically shown in Figure 8. We denote by $\mathbf{G}$ the list of gaps $[g_0, ..., g_{N-1}]$.

However, we have set three requirements for determining τ_j (in priority order). First, we require that $\tau_j \geq 1$, since all scores are initiated as 1 and a score will only be increased through backpropagation if the network attends to that corresponding time series. Secondly, since a temporal causal graph is usually sparse, we require that the gap selected for τ_j lies in the first half of $\mathbf{G}$ (if $N > 3$) to ensure that the algorithm does not include low attention scores in the selection. This means that at most 50% of the input time series can be correlated with target $\mathbf{X}_j$. By defining this requirement, we ensure that not too many time series are labeled as *correlated*. Although this number can be changed by the user, we experimentally estimated that 50% gives good results in the evaluation of our framework.

Lastly, we require that the gap for τ_j cannot be in first position (i.e. between the highest and second-highest attention score). This requirement ensures that the algorithm does not truncate scores to zero of time series which were actually correlated, but weaker than the top scoring one. This means that at least 2 time series will be labeled as *correlated* for target $\mathbf{X}_j$.

After τ_j is determined, the HardSoftmax function can be applied. Thus, time series $\mathbf{X}_i$ is labeled as *correlated* with the target time series $\mathbf{X}_j$ if $a_{i,j} \in \mathbf{a}_j > \tau_j$, i.e. if $h_{i,j} \in \mathbf{h}_j > 0$. By applying HardSoftmax to the attention scores of all N networks, TCDF collects all correlations between time series discovered by the attention mechanisms.

4.2 Causal Validation

The second step of TCDF distinguishes causation from correlation. Recall that a causal relationship should comply with two aspects [Eichler, 2012]:

1. *Temporal precedence*: the cause precedes its effect,
2. *Physical influence*: manipulation of the cause changes its effect.

Since we use a temporal convolutional network architecture, there is no information leakage from future to past. Therefore, we already comply with the temporal precedence assumption. To comply with the second assumption, we have to validate that a change in a potential cause will influence its effect. Since our method will be purely based on observational data, TCDF does not have the possibility to do a real-life experiment to check for physical influence. We therefore came up with a novel causal validation approach that uses only the observational dataset.

We divide the causal validation stage in two steps. TCDF first interprets the HardSoftmax scores $\mathbf{h}_j$ to find *potential causes*, described in Section 4.2.1. Secondly, TCDF applies the *Causal Quantitative Input Influence* (CQII) measure of [Datta et al., 2016] to check the physical influence assumption, described in Section 4.2.2. Potential causes that are validated by CQII will be called *true* causes. However, the existence of hidden confounders can complicate the correct discovery of true causes. Section 4.2.3 describes how TCDF handles a dataset in which not all confounders are measured.

4.2.1 Attention Interpretation

After each network $\mathcal{N}_j$ is trained for an equal number of training epochs, the HardSoftmax attention scores $\mathbf{h}_j$ for each network $\mathcal{N}_j$ are collected to distinguish causation from correlation. By interpreting these attention scores, we can create a set of *potential causes* $\mathbf{P}_j$ for each time series $\mathbf{X}_j \in \mathbf{X}$. This set $\mathbf{P}_j$ will serve as input for the CQII measure discussed in the next section.

We can observe the following cases between HardSoftmax scores $h_{i,j}$ and $h_{j,i}$:

1. $h_{i,j} = 0$ and $h_{j,i} = 0$: $\mathbf{X}_i$ is not correlated with $\mathbf{X}_j$ and vice versa.
2. $h_{i,j} = 0$ and $h_{j,i} > 0$: $\mathbf{X}_j$ is added to $\mathbf{P}_i$ since $\mathbf{X}_j$ is a potential cause of $\mathbf{X}_i$ because of:
   (a) (In)direct causal relation from $\mathbf{X}_j$ to $\mathbf{X}_i$, or
   (b) Presence of a (hidden) confounder between $\mathbf{X}_j$ and $\mathbf{X}_i$ where the delay from the confounder to $\mathbf{X}_j$ is smaller than the delay to $\mathbf{X}_i$.
3. $h_{i,j} > 0$ and $h_{j,i} = 0$: $\mathbf{X}_i$ is added to $\mathbf{P}_j$ since $\mathbf{X}_i$ is a potential cause of $\mathbf{X}_j$ because of:
   (a) (In)direct causal relation from $\mathbf{X}_i$ to $\mathbf{X}_j$, or
   (b) Presence of a (hidden) confounder between $\mathbf{X}_i$ and $\mathbf{X}_j$ where the delay from the confounder to $\mathbf{X}_i$ is smaller than the delay to $\mathbf{X}_j$.
4. $h_{i,j} > 0$ and $h_{j,i} > 0$: Time series $\mathbf{X}_i$ and $\mathbf{X}_j$ are correlated because of:
   (a) Presence of a 2-cycle where $\mathbf{X}_i$ causes $\mathbf{X}_j$ and $\mathbf{X}_j$ causes $\mathbf{X}_i$, or
   (b) Presence of a (hidden) confounder with equal delays to its effects $\mathbf{X}_i$ and $\mathbf{X}_j$.

Note that a HardSoftmax score > 0 could also be the result of a spurious correlation. However, since it is impossible to judge whether a correlation is spurious purely on the analysis of observational data, TCDF does not take the possibility of a spurious correlation into account. After causal discovery from observational data, it is up to a domain expert to judge or test whether a discovered causal relationship is correct. Section 6 presents a more extensive discussion on this topic.

By comparing all attention scores, we create a set of potential causes for each time series. Then, we will use the CQII measure to validate if a *potential* cause is a *true* cause. More specifically, TCDF will apply CQII to distinguish between case 2a and 2b, and between case 3a and 3b.

One could also use the CQII measure to distinguish between case 4a and 4b. In that case, $\mathbf{X}_i$ should be added to $\mathbf{P}_j$ and $\mathbf{X}_j$ should be added to $\mathbf{P}_i$. However, when we expect that a 2-cycle is non-existent (or at least sparse) based on domain knowledge, TCDF assumes that only the much more common case 4b occurs in order to save computational costs.

4.2.2 Causal Quantitative Input Influence

To allow for causal reasoning, we apply the *Causal Quantitative Input Influence (CQII)* measure of [Datta et al., 2016] as described in Section 2.2. This measure models the difference in the "quantity of interest" between the real input distribution and an intervened distribution. In our case, the quantity of interest is the loss $\mathcal{L}$ of the network between the predictions and the actual values of the target time series. As intervention, we change the input values of potential cause $\mathbf{X}_i \in \mathbf{P_j}$ to random values having the same mean and standard deviation as the true values of $\mathbf{X}_i$. Thus, the 'real input distribution' is the original input data, while the 'intervened distribution' is the input data where the intervention is applied.

Network $\mathcal{N}'_j$ is then trained with the same input dataset as for $\mathcal{N}_j$, except that $\mathbf{X}_i$ is replaced with random values having the mean and standard deviation as $\mathbf{X}_i$. If $\mathbf{X}_i$ would be a real cause of $\mathbf{X}_j$, the predictions of $\mathcal{N}'_j$ should be worse, since $\mathcal{N}'_j$ has no access to the ground truth values of $\mathbf{X}_i$ when predicting $\mathbf{X}_j$. Therefore, the intervention loss $\mathcal{L}_I$ of the network should significantly increase compared to the ground loss $\mathcal{L}_G$ when the real input distribution is used. If the intervention loss $\mathcal{L}_I$ is not significantly higher than $\mathcal{L}_G$ (and maybe even lower), then $\mathbf{X}_i$ is apparently no cause of $\mathbf{X}_j$ since $\mathbf{X}_j$ can be predicted without having access to the

ground truth values of $\mathbf{X}_i$. Only the time series in $\mathbf{P_j}$ that are validated by CQII are considered *true* causes of the target time series $\mathbf{X}_j$. We denote by $\mathbf{C}_j$ the set of all true causes of $\mathbf{X}_j$.

As an example, we consider the case depicted in Figure 2d. Suppose that both $\mathbf{X}_1$ and $\mathbf{X}_2$ are considered as potential causes for $\mathbf{X}_3$ based on the attention score interpretation. CQII is applied to validate if these causes are *true* causes of $\mathbf{X}_3$. When the intervened distribution of $\mathbf{X}_1$ is used in dataset $\mathbf{X}$ to predict $\mathbf{X}_3$, loss $\mathcal{L}_I$ will probably be higher than $\mathcal{L}_G$ since the network has no access to the ground truth values of the confounder $\mathbf{X}_1$. On the other hand, if CQII is applied to $\mathbf{X}_2$, the loss will probably not change significantly, since the network has still access to the confounder $\mathbf{X}_1$ to predict $\mathbf{X}_3$. Our TCDF will therefore conclude that only $\mathbf{X}_1$ is a true cause of $\mathbf{X}_3$.

The user can define a percentage to consider an increase in loss to be 'significant'. A low percentage will lead to a more tolerant algorithm, with the risk that some causes are incorrectly labeled as 'true cause'. A high percentage means a more strict algorithm, with the risk that some causes are incorrectly *not* labeled as 'true cause'. One should find a good balance between those two scenario's. In our experiments as described in Section 5, we require an increase of at least 5%.

The benefit of applying CQII, which is for the first time applied to neural networks to the best of our knowledge, is that the resulting true causes satisfy both the *temporal precedence* requirement and the *physical influence* requirement. However, the disadvantage is that computational costs increase significantly. Instead of training each network $\mathcal{N}_j$ only once, a network $\mathcal{N}'_j$ needs to be trained for each $\mathbf{X_i} \in \mathbf{P}_j$.

4.2.3 Dealing with Hidden Confounders

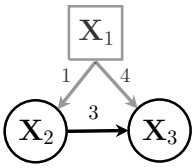
If we assume that all genuine causes are measured, the causal validation step of TCDF consisting of attention interpretation and CQII validation should in theory only discover correct causal relationships (according to the data). Namely, cases 2b, 3b and 4b from Section 4.2.1 then all arose because of a measured confounder. A time series $\mathbf{X}_i$ that was correlated with time series $\mathbf{X}_j$ because of a confounder would not be labeled as *true* cause by CQII, since only the presence of the confounder would be needed to predict $\mathbf{X}_j$.

However, our CQII approach might discover incorrect causal relationships if there exist hidden confounders, i.e. confounders that are not included in the dataset. This section describes how TCDF can successfully discover the presence of a hidden confounder with equal delays to its effects $\mathbf{X}_i$ and $\mathbf{X}_j$ (case 4b from Section 4.2.1). But, we also discuss that TCDF will probably *not* detect the presence of a hidden confounder when the hidden confounder has unequal delays to its effects (case 2b and 3b).

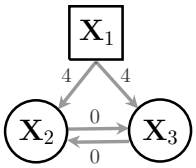
As is shown in Table 1, most temporal causal discovery methods cannot deal with hidden confounders in a dataset. Method ANLTSM can only deal with hidden confounders that are linear and instantaneous. The authors of TiMINo claim to handle hidden confounders by staying undecided instead of inferring any (possibly incorrect) causal relationship. Lastly, tsFCI handles hidden confounders by including a special edge type ($\mathbf{X}_i \leftrightarrow \mathbf{X}_j$) that shows that $\mathbf{X}_i$ is not a cause of $\mathbf{X}_j$ and that $\mathbf{X}_j$ is not a cause of $\mathbf{X}_i$. From this, the authors conclude that there should be a hidden confounder that causes both $\mathbf{X}_i$ and $\mathbf{X}_j$.

TCDF can discover this $\mathbf{X}_i \leftrightarrow \mathbf{X}_j$ relation in specific cases by applying CQII. Based on cases 2, 3 and 4, we can distinguish three reasons why two time series are correlated: existence of a causal relationship, presence of a measured confounder or presence of a hidden confounder. (We exclude the possibility of a spurious correlation). If there is a measured confounder, CQII should discover that the confounder's effects $\mathbf{X}_i$ and $\mathbf{X}_j$ are just correlated and not causally related. If there is a 2-cycle, CQII should discover that $\mathbf{X}_i$ causes $\mathbf{X}_j$ with a certain delay and that $\mathbf{X}_j$ causes $\mathbf{X}_i$ with a certain delay. If there is a *hidden* confounder of $\mathbf{X}_i$ and $\mathbf{X}_j$, CQII will find that $\mathbf{X}_i$ is a true cause of $\mathbf{X}_j$ and vice versa. Namely, if CQII uses an intervened distribution for $\mathbf{X}_i$, the loss $\mathcal{L}$ of network $\mathcal{N}_j$ to predict $\mathbf{X}_j$ will probably increase, since both the hidden confounder and the intervened $\mathbf{X}_i$ cannot give away any information about the future values of $\mathbf{X}_j$.

When the delay from the confounder to $\mathbf{X}_i$ is smaller than the delay to $\mathbf{X}_j$ (case 3b), TCDF will, based on the temporal precedence assumption, discover an incorrect causal relationship from $\mathbf{X}_i$ to $\mathbf{X}_j$. More specifically, TCDF will discover that the delay of this causal relationship will be equal to the delay from the confounder to $\mathbf{X}_i$ minus the delay from the confounder to $\mathbf{X}_j$. Figure 9a shows an example of this situation. The same reasoning applies when the delay from the confounder to $\mathbf{X}_i$ is *greater* than the delay to $\mathbf{X}_j$ (case 2b), since $\mathbf{X}_i$ and $\mathbf{X}_j$ are just interchanged such that TCDF will discover an incorrect causal relationship from $\mathbf{X}_j$ to $\mathbf{X}_i$.



(a) Example showing that TCDF will incorrectly discover a causal relationship from $\mathbf{X}_2$ to $\mathbf{X}_3$ when the delay from hidden confounder $\mathbf{X}_1$ to effect $\mathbf{X}_2$ is smaller than the delay from $\mathbf{X}_1$ to $\mathbf{X}_3$.



(b) Example showing that TCDF will discover a 2-cycle between $\mathbf{X}_2$ and $\mathbf{X}_3$ where both delays equal 0, such that it concludes that there should exist a hidden confounder between $\mathbf{X}_2$ and $\mathbf{X}_3$.

Figure 9: Example showing how TCDF deals with hidden confounders (denoted as a square). A black square indicates that the hidden confounder is discovered, whereas a grey square indicates that the hidden confounder is not discovered by TCDF. Black edges indicate causal relationships that will be included in the learnt temporal causal graph $\mathcal{G}_L$. Grey edges indicate that these causal relationships are not included in $\mathcal{G}_L$.

However, TCDF will *not* discover a causal relationship when the hidden confounder has equal delays to its effects $\mathbf{X}_i$ and $\mathbf{X}_j$ (case 4b), and can even conclude that there should be a hidden confounder that causes both $\mathbf{X}_i$ and $\mathbf{X}_j$. Because the confounder has equal delays to $\mathbf{X}_i$ and $\mathbf{X}_j$, both the delay from the discovered causal relationship from $\mathbf{X}_i$ to $\mathbf{X}_j$ will be 0, and the delay from the discovered causal relationship from $\mathbf{X}_j$ to $\mathbf{X}_i$ will be 0. The zero delays give away the presence of a hidden confounder, since there cannot exist a 2-cycle where both time series have an instantaneous effect on each other. Recall that an instantaneous effect means that there is an effect within 1 measured time step. If both time series cause each other instantaneously, there will be an infinite causal influence between the time series within 1 time step, which is impossible. Therefore, TCDF will in this case conclude that $\mathbf{X}_i$ and $\mathbf{X}_j$ are not causally related, and that there exists a hidden confounder between $\mathbf{X}_i$ and $\mathbf{X}_j$. Figure 9b shows an example of this situation.

The advantage of our approach is that TCDF not only concludes that two variables are not causally related, but can also detect the presence of a hidden confounder. This might be useful information for further research.

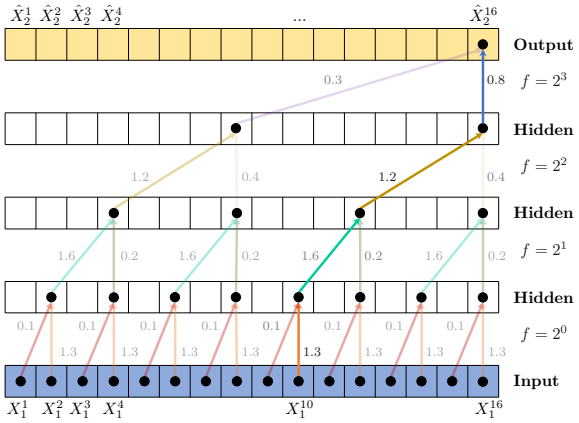


Figure 10: An example that discovers the delay between cause $\mathbf{X}_1$ and target $\mathbf{X}_2$, both having $T = 16$. Starting from the top convolutional layer, the algorithm traverses through the path with the highest kernel weights. Eventually, the algorithm ends in input value X_1^{10}, indicating a delay of $16 - 10 = 6$ time steps.

4.3 Delay Discovery

Besides discovering the existence of a causal relationship, it is useful to discover the number of time steps between the occurrence of a true cause and the occurrence of its effect. For example, the delay between a change in the milk price and a change in the butter price might be 1 month. We discover this time delay by interpreting the kernel weights $\mathcal{W}_i$ for a causal input time series $\mathbf{X}_i$ from a network $\mathcal{N}_j$ predicting target time series $\mathbf{X}_j$. Since we have a depthwise separable architecture where input time series are not mixed, the relation between the kernel weights of one input time series and the target time series can be correctly interpreted.

The kernel that slides over the N input channels is a weight matrix with N rows and K columns (where K is the kernel size), and outputs the dot product between the input channel and the weight matrix. Contrary to regular neural networks, all output values of a channel share the same weights and therefore detect exactly the same pattern, as indicated by the identical colors in Figure 7. These shared weights not only reduce the total number of learnable parameters, but also allow delay interpretation.

Since a convolution is a linear operation, we can measure the influence of a specific delay between cause $\mathbf{X}_i$ and target $\mathbf{X}_j$, by analyzing the weights of $\mathbf{X}_i$ in the kernel. The K weights of each channel output show the 'importance' of each delay with an intermediate step of dilation factor f.

TCDF creates a weighted tree of which the root corresponds to time step T (the last time step) in the output layer of the channel corresponding to cause $\mathbf{X}_j$. The time steps in the receptive field of the lower layer correspond to the children, and the leafs correspond to all time steps in the input layer. An example is shown in Figure 10, where the black dots are the vertices of the tree. The kernel weights are the weighted edges of the tree (depicted by the annotated colored edges in Figure 10). To discover the delay between cause and effect, our framework finds the path with the heaviest weight, starting from the output layer and ending up at a certain time step in the input layer, as shown by the bright colored edges in Figure 10. The difference between time step T and the end point of the heaviest path equals the discovered delay $d(e_{i,j})$. Since we also use the current values in the input data, the smallest delay can be 0 time steps, which indicates an instantaneous effect. The maximum delay that can be found equals the size of the receptive field.

When the network has more than 1 layer, it is advisable to select a dilation coefficient c that is equal to the kernel size K, to make sure that there is exactly one path for each delay. In the case of $c = K$, the receptive field R simply becomes:

$$R_{D-DSTCN_{c=K}} = k^{L+1} \tag{6}$$

where K is the kernel size that equals the dilation coefficient c, and L is the number of hidden layers.

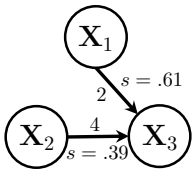


Figure 11: Example showing that TCDF can be used to estimate the relative causal strength of a true cause on its effect.

4.4 Graph Construction and Reduction

After training the network, interpreting the attention scores and discovering the delays, we can construct a temporal causal graph $\mathcal{G}_L$ by drawing a vertex for each time series in dataset $\mathbf{X}$ and drawing an edge $e_{i,j}$ from vertex v_i to v_j if there is a true causal relationship discovered from $\mathbf{X}_i$ to $\mathbf{X}_j$. Edge $e_{i,j}$ in the learnt graph $\mathcal{G}_L$ is annotated with the discovered delay $d(e_{i,j})$.

To improve readability, TCDF implements a *graph reduction* step that reduces the graph by removing all direct edges in $\mathcal{G}_L$ that denote an *indirect* causal relationship. A requirement for a causal relationship $e_{i,j}$ to be indirect, is that it should be weaker than the direct relationship $e_{k,j}$. Paragraph 4.4.1 describes how we can estimate the causal strength of a relationship based on the attention scores. Subsequently, paragraph 4.4.2 discusses how the causal strength estimation can be used to reduce the learnt temporal causal graph by removing edges from $\mathcal{G}_L$ that denote indirect causal relationships.

4.4.1 Causal Strength Estimation

It is useful to know how strongly a true cause influences its effect. This can be done by using the dataset to estimate causal model parameters that provide quantitative strength information [Malinsky and Danks, 2018]. As shown in Table 1, most existing approaches have some model parameter to estimate causal strength. In our framework, the attention scores provide such a quantitative measure.

In a network $\mathcal{N}_j$, an attention score $a_{i,j} \in \mathbf{a}_j$ of a true cause $\mathbf{X}_i \in \mathbf{C}_j$ can to some extent be interpreted as a 'causal strength score' indicating how much it influences its effect $\mathbf{X}_j$. An edge $e_{i,j}$ in the learnt temporal causal graph $\mathcal{G}_\mathcal{L}$ can be annotated with this causal strength score s.

Note that attention scores between networks cannot be compared, since the value of an attention score depends on the learning process of its network. For example, network $\mathcal{N}_j$ might immediately discover that $\mathbf{X}_i$ is correlated with $\mathbf{X}_j$ and therefore it can increase the attention score of $\mathbf{X}_i$ in all training epochs. On the contrary, if a different network $\mathcal{N}_k$ starts with incorrectly decreasing the attention score of $\mathbf{X}_i$ before learning that it has to increase this score, $\mathcal{N}_k$ will end up with a lower attention score for $\mathbf{X}_i$ than $\mathcal{N}_j$.

Since each network learns their attention scores differently, we can only compare the attention scores within one network, i.e. compare all $a_{i,j} \in \mathbf{a}_j$. By normalizing the attention scores in $\mathbf{a}_j$ of all *true* causes of a time series, we can create a measure denoting an estimate of the relative causal strength.

Figure 11 shows an example in which both $\mathbf{X}_1$ and $\mathbf{X}_2$ are true causes of $\mathbf{X}_3$, with attention scores $a_{1,3} = 1.9$ and $a_{2,3} = 1.2$. We assume that the attention scores of all other time series in $\mathbf{X}$ are irrelevant since these are not a true cause of $\mathbf{X}_3$. By normalizing the attention scores, TCDF will find that $\mathbf{X}_1$ is a stronger cause of $\mathbf{X}_3$ than $\mathbf{X}_2$, because the causal strength score of $\mathbf{X}_1$ is $1.9/(1.9 + 1.2) = 0.61$, while $\mathbf{X}_2$ has a causal strength score of 0.39.

More formally, we can define the causal strength score of a true cause $\mathbf{X}_i$ as:

$$s(e_{i,j}) = \frac{a_{i,j}}{\sum\limits_{\mathbf{X}_k \in \mathbf{C}_j} a_{k,j}} \tag{7}$$

where $e_{i,j}$ is the edge in a temporal causal graph $\mathcal{G}$ denoting a causal relationship from $\mathbf{X}_i$ to $\mathbf{X}_j$, $a_{i,j}$ is the attention score of cause $\mathbf{X}_i$ on $\mathbf{X}_j$ and $\mathbf{C}_j$ is the set of all true causes of $\mathbf{X}_j$.

Thus, the causal strength score s denotes how strongly a true cause affects its effect compared to other true causes of that same effect. If a time series has only one true cause, causal strength score $s = 1$.

(a) Learnt graph $\mathcal{G}_L$ that includes edge $e_{1,4}$ that denotes an indirect causal relationship.

(b) Reduced learnt graph where edge $e_{1,4}$ denoting an indirect causal relationship is removed.

Figure 12: Example showing that an indirect causal relationship can be removed from the constructed graph to improve readability.

4.4.2 Graph Reduction

TCDF applies a graph reduction step that reduces the learnt graph $\mathcal{G}_L$ by removing all direct edges from $\mathcal{G}_L$ that denote an *indirect* causal relationship. Although the attention mechanism and validation step might already have removed some indirect causes, we have experienced that some indirect causes are still labeled as *true cause* by TCDF. These indirect causes have no added informative value since the direct causal relationships are already in the learnt graph, and can therefore safely be removed. This even improves the readability of the learnt graph. Figure 12 shows an example in which edge $e_{1,4} \in \mathcal{G}_L$ denotes an indirect causal relationship and is therefore removed.

To ensure that an edge $e_{i,j} \in \mathcal{G}_L$ is indirect, and not an extra *direct* causal relationship from $\mathbf{X}_i$ to $\mathbf{X}_j$, we have set up a set of requirements that an indirect discovered causal relationship should adhere to. First of all, we assume that a causal relationship is only indirect if its delay is equal to the sum of the delays of the direct causal relationships that form a directed path from $\mathbf{X}_i$ to $\mathbf{X}_j$. Note that an indirect cause from $\mathbf{X}_i$ to $\mathbf{X}_j$ with a different delay than the sum of the direct causes cannot be removed, since $e_{i,j}$ then denotes a different *direct* causal relationship from $\mathbf{X}_i$ to $\mathbf{X}_j$.

Secondly, we check that the indirect causal relationship from $\mathbf{X}_i$ to $\mathbf{X}_j$ via $\mathbf{X}_k$ is *weaker* than the direct causal relationship from $\mathbf{X}_k$ to $\mathbf{X}_j$, to make sure that $e_{i,j}$ is not an extra direct causal relationship that by chance has the same delay to $\mathbf{X}_j$ as another direct edge $_{k,j}$.

More formally, we consider a discovered true causal relationship *indirect* using three criteria:

1. There is a path $p_1 = \langle v_i, v_j \rangle \in \mathcal{G}_L$ with $|p_1| = 1$, and a path $p_2 = \langle v_i, ..., v_k, v_j \rangle \in \mathcal{G}_L$ with $|p_2| \geq 2$.
2. The delay of path p_1 equals the delay of path p_2: $d(p_1) = d(p_2)$.
3. The causal strength of v_i on v_j is smaller than the causal strength of v_k on v_j: $s(e_{i,j}) < s(e_{k,j})$.

From now on, we denote by $\mathcal{G}_L$ a learnt temporal causal graph in which the graph reduction step is applied (i.e. the edges corresponding to indirect causal relationships are removed).

5 Experiments

To evaluate our framework, we apply TCDF both to actual time series data and simulated time series data. When TCDF is applied to an actual data set, of which the ground truth is unknown, we evaluate if the temporal causal graph is in accordance with domain knowledge. In a simulated dataset, the true causal relationships are known which allows us to evaluate the accuracy of TCDF. We perform multiple experiments with simulated time series data such that all data types listed in Table 1 are covered: multivariate, continuous, non-stationary, non-linear and noisy. In all experiments, we compare the results of TCDF with results of three existing temporal causal discovery methods.

In all experiments with simulated data, we evaluate the four steps of TCDF as shown in Figure 3, from back to front. First, we evaluate the output of TCDF by comparing the edges in the learnt (and reduced) temporal causal graph $\mathcal{G}_L$ with the edges in the ground truth graph $\mathcal{G}_G$ to evaluate how well TCDF discovered the existence of causal relationships. Secondly, we compare the discovered delays in $\mathcal{G}_L$ with the delays in $\mathcal{G}_G$ to evaluate how well our delay discovery algorithm in TCDF works. Thirdly, we compare the effectiveness of CQII to check if it correctly distinguishes causation from correlation. The evaluation measures for these evaluations are described in the next section. Lastly, we want to evaluate how the architecture of AD-DSTCNs influence the discovery of correct causal relationships. However, since it would be impractical to test all parameter settings, we only vary the number of hidden layers L.

As a side experiment, we evaluate how TCDF handles hidden confounders. As discussed in Section 4.2.3, TCDF should correctly discover the presence of a hidden confounders when the delays to its effects are equal. However, we expect that TCDF will discover an incorrect causal relationship if the delays from the hidden confounders to its effects are unequal.

In all experiments, we apply Mean Squared Error as loss function in our AD-DSTCNs. Furthermore, in all our AD-DSTCNs we use the Adam optimization algorithm which is an extension to stochastic gradient descent [Kingma and Ba, 2014]. This optimizer computes individual adaptive learning rates for each parameter which allows the gradient descent to find the minimum more accurately. Furthermore, in all experiments, we train our AD-DSTCNs for 2,000 training epochs, with learning rate $\lambda = 0.01$, dilation coefficient $c = 4$ and kernel size $K = 4$. We chose K such that the delays in the ground truth fall within the receptive field R. The number of training epochs was chosen so as to achieve convergence in the loss $\mathcal{L}$.

Our framework is implemented in Python and PyTorch. In all experiments, we measure the runtime of TCDF on a Ubuntu 16.04.4 LTS computer with an Intel® Xeon® E5-2683-v4 CPU (one thread used) and NVIDIA TitanX 12GB GPU. We leave the calculation of the time and space complexity of TCDF and other existing methods for future work.

5.1 Evaluation Measures

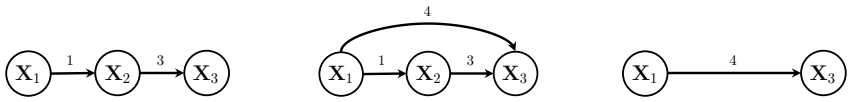
This section describes the evaluation measures to evaluate TCDF when applied to a simulated dataset of which the ground truth is known. It describes 1) the evaluation measures for discovered causal relationships, 2) the evaluation measures for discovered delays and 3) the evaluation measures for the effectiveness of CQII.

5.1.1 Evaluation Measure for Discovered Causal Relationships

We evaluate the learnt graph $\mathcal{G}_L$ by looking at the presence and absence of directed edges compared to the ground truth graph $\mathcal{G}_G$. Since causality is an asymmetric measure ($\mathbf{X}_1$ causing $\mathbf{X}_2$ is different than $\mathbf{X}_2$ causing $\mathbf{X}_1$), we take the directionality of an edge into account.

We evaluate the learnt edges in terms of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). We apply the usual definitions from graph comparison for the TP, FP, FN and TN evaluation measures, such that:
TP = $|\mathbf{TP}|$ with $\mathbf{TP} = E(\mathcal{G}_G) \cap E(\mathcal{G}_L)$,
FP = $|\mathbf{FP}|$ with $\mathbf{FP} = E(\mathcal{G}_G) \setminus E(\mathcal{G}_L)$,
FN = $|\mathbf{FN}|$ with $\mathbf{FN} = E(\mathcal{G}_G) \setminus E(\mathcal{G}_L)$,
TN = $|\mathbf{TN}|$ with $\mathbf{TN} = E(\overline{\mathcal{G}_G}) \setminus E(\overline{\mathcal{G}_L})$,
where $E(\mathcal{G})$ is the set of all edges in graph $\mathcal{G}$ and $\overline{\mathcal{G}}$ is the complement of $\mathcal{G}$.

Note that taking the directionality of edges into account means that a wrongly oriented edge counts for one False Positive and one False Negative.



(a) Ground truth $\mathcal{G}_G$ (b) *Full* ground truth $\mathcal{G}_F$ (c) Learnt $\mathcal{G}_L$

Figure 13: Example with three variables showing that $\mathcal{G}_L$ has TP = 0, FP = 1 ($e_{1,4}$), TP' = 1 ($e_{1,3}$), FP' = 0, FN = 2 ($e_{1,2}$ and $e_{2,3}$) and TN = 3 ($e_{2,1}$, $e_{3,1}$ and $e_{3,2}$). Therefore, Precision = 1, Recall = 0.333 and F1 = 0.5.

However, these TP and FP measures evaluate $\mathcal{G}_L$ only based on the *direct* causes in $\mathcal{G}_G$. This means that a direct edge in $\mathcal{G}_L$ which corresponds to a correct *indirect* causal relationship in the ground truth, will not be counted as True Positive. An example where an indirect causal relationship would not be counted as a True Positive is shown in Figure 13. Because of the graph reduction step of TCDF, the learnt graph $\mathcal{G}_L$ will only contain such indirect causal relationships when there is no direct causal relationship included in $\mathcal{G}_L$. Therefore, we think it is reasonable to count those indirect causal relationships as a True Positive as well, since an indirect cause does have, although indirectly, a causal influence on its effect.

For that reason, we introduce the TP' and FP' measures that extend the normal TP and FP measures by taking the edges into account that represent direct causal relationships in $\mathcal{G}_L$, but are indirect causal relationships in $\mathcal{G}_G$. Where we compared the learnt graph $\mathcal{G}_L$ with ground truth $\mathcal{G}_G$ to calculate the normal TP and FP measures, we compare $\mathcal{G}_L$ with the *full* ground truth graph $\mathcal{G}_F$ to calculate TP' and FP'. The full ground truth graph $\mathcal{G}_F$ contains a directed edge $e_{i,j}$ for each directed path $\langle v_i, ..., v_j \rangle$ in ground truth graph $\mathcal{G}_G$. Figure 13b shows an example of a full ground truth graph $\mathcal{G}_F$. The TP' and FP' are then measured as follows:
TP' = $|\mathbf{TP'}|$ with $\mathbf{TP'} = E(\mathcal{G}_F) \cap E(\mathcal{G}_L)$,
FP' = $|\mathbf{FP'}|$ with $\mathbf{FP'} = E(\mathcal{G}_L) \setminus E(\mathcal{G}_F)$.

To evaluate our framework, we use the TP', FP' and FN measures to calculate Precision (indicating the fraction of learnt edges that correspond to the full ground-truth edges) and Recall (indicating the fraction of full ground-truth edges that were learnt correctly). We define Precision and Recall as follows:

$$\text{Precision} = \frac{\text{TP'}}{\text{TP'} + \text{FP'}} \quad (8) \,, \qquad \text{Recall} = \frac{\text{TP'}}{\text{TP'} + \text{FN}} \quad (9)$$

We summarize our framework's accuracy using the F1'-score, that considers both Precision and Recall (based on our TP' and FP' scores):

$$\text{F1'} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)$$

5.1.2 Evaluation Measure for Discovered Delays

We evaluate a discovered delay $d(e_{i,j} \in \mathcal{G}_L)$ between cause $\mathbf{X}_i$ and effect $\mathbf{X}_j$ by comparing it to the full ground truth delay $d(e_{i,j} \in \mathcal{G}_F)$. We only evaluate the delay of edges in **TP** since the other edges do not exist in both $\mathcal{G}_F$ and $\mathcal{G}_L$.

We first measure the percentage of delays corresponding to edges in **TP** that are learnt correctly, i.e. where $d(e_{i,j} \in \mathcal{G}_L) = d(e_{i,j} \in \mathcal{G}_F)$. Secondly, we want to evaluate the error $\epsilon_{d(e_{i,j})}$ of an incorrectly learnt delay to measure how *off* the discovered delay is compared to the delay of the ground truth. This could be done by calculating the distance between $d(e_{i,j} \in \mathcal{G}_L)$ and $d(e_{i,j} \in \mathcal{G}_F)$. However, this absolute measure does not take the receptive field R of the AD-DSTCNs in TCDF into account. Recall that discovered delays will lie between 0 and the receptive field R. We argue that a TCDF with $R = 4$ which is for example 3 time steps off, should be more penalized than a TCDF with $R = 256$ that is 'just' 3 time steps off. Therefore, we propose to calculate the distance to the ground truth *relative* to the receptive field:

$$\epsilon_{d(e_{i,j})} = \frac{d(e_{i,j} \in \mathcal{G}_F) - d(e_{i,j} \in \mathcal{G}_L)}{R} \quad (11)$$

In this way, the error indicates how *off* the learnt delay was relative to the receptive field. A high error (close or equal to 1) denotes a learnt delay that could not be much more wrong, while a small error means

that the discovered delay was not equal but close to the ground truth delay. The mean of all non-zero errors, μ_ϵ, denotes the average error score of all incorrect delays.

5.1.3 Evaluation Measure for CQII effectiveness

Lastly, we evaluate the effectiveness of CQII. The goal of CQII is to label a subset of the potential causes as *true* causes. We first analyse how many potential causes were correctly *not* selected as true cause by comparing the number of False Positives when CQII is used to the number of False Positives when CQII is not applied. Secondly, we compare the number of True Positives with and without CQII to see how many potential causes were *not* labeled as true cause while they actually were a true cause according to the ground truth. We summarize the CQII effectiveness by calculating the relative increase (or decrease) of the F1'-score when CQII is applied compared to not applying CQII.

5.2 Comparison with Existing Approaches

We compare the results of our framework with three other methods for temporal causal discovery: PCMCI implemented by the authors in the Python Tigramite module [Runge et al., 2017], tsFCI implemented by TETRAD [Scheines et al., 1998] and TiMINo implemented by the authors in R [Peters et al., 2013]. Although the simulated financial dataset is provided by [Kleinberg, 2013], there is no public implementation available of her causal significance measure.

However, we cannot apply the TP' and FP' evaluation measures to the results of the existing methods like we do for TCDF. The TP' and FP' take the edges into account that represent direct causal relationships in $\mathcal{G}_L$, but are indirect causal relationships in $\mathcal{G}_G$. However, the existing methods do not implement any graph reduction step to remove edges that denote indirect causal relationships from $\mathcal{G}_L$. Therefore, their learnt graphs will probably include many edges that denote indirect causal relationships. If we would apply the TP' and FP' measures to the existing methods, we would overestimate their accuracy since we then count both the direct and indirect causal relationships as True Positives. In contrast, TCDF only contains an indirect causal relationship in $\mathcal{G}_L$ if there is no corresponding direct causal relationship existent in $\mathcal{G}_L$.

For a legitimate comparison, we therefore compare TCDF with the other methods in terms of the normal TP, FP and FN measures instead of TP', FP' and FN as we do for TCDF. We summarize the accuracy of each model by calculating the F1-score based on TP, FP and FN.

Parameters In all experiments, we apply the following parameters to the algorithms:
PCMCI: We set the maximum delay to 3 time steps and the minimum delay to 0, equivalent to the minimum and maximum delay that can be found by TCDF in our AD-DSTCNs with $K = 4$ and $L = 0$. We use the ParCorr independence test for linear partial correlation[8]. Furthermore, we chose a fixed threshold for p-values among $[0.01, 0.1, 0.2]$. Since all thresholds gave comparable results, we chose 0.01 such that all p-values > 0.01 are selected as causes.

tsFCI: We set the maximum delay to 3 time steps, equivalent to the maximum delay that can be found by TCDF in our AD-DSTCNs with $K = 4$ and $L = 0$. We chose the cutoff value for p-values among $[0.001, 0.01, 0.1]$ and use 0.01 as this value gave the best results (besides the fact that 0.01 is also the default setting). We only take the discovered *direct* causes into account and disregard other edge types which denote uncertainty or the presence of a hidden confounder.

TiMINo: We set the maximum delay to 3, equivalent to the maximum delay that can be found by TCDF in our AD-DSTCNs with $K = 4$ and $L = 0$. We assumed a linear time series model[9] including instantaneous effects and shifted time series, and we chose the significance level among $[0.05, 0.01, 0.001]$. However, TiMINo did not give any result for all of significance levels because it could not find a model that seemed to fit. Therefore, we set it to 0 such that TiMINo always obtains a DAG, even if the residuals are dependent.

[8] Besides the linear independence test, the author presents the non-linear GPACE test to discover non-linear causal relationships [Runge et al., 2017]. However, since GPACE scales $\sim T^3$, we apply for practical reasons the linear ParCorr test.

[9] The author presents two other variants besides the linear model, of which 'TiMINo-GP' was shown to be more suitable for 'long' time series (> 300 time steps) [Peters et al., 2013]. However, only the linear model was fully implemented by the author.

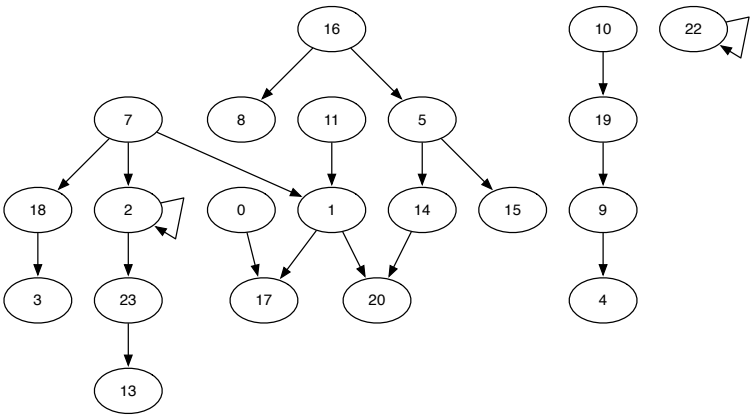
5.3 Experiment 1: Simulated Financial Time Series

5.3.1 Data

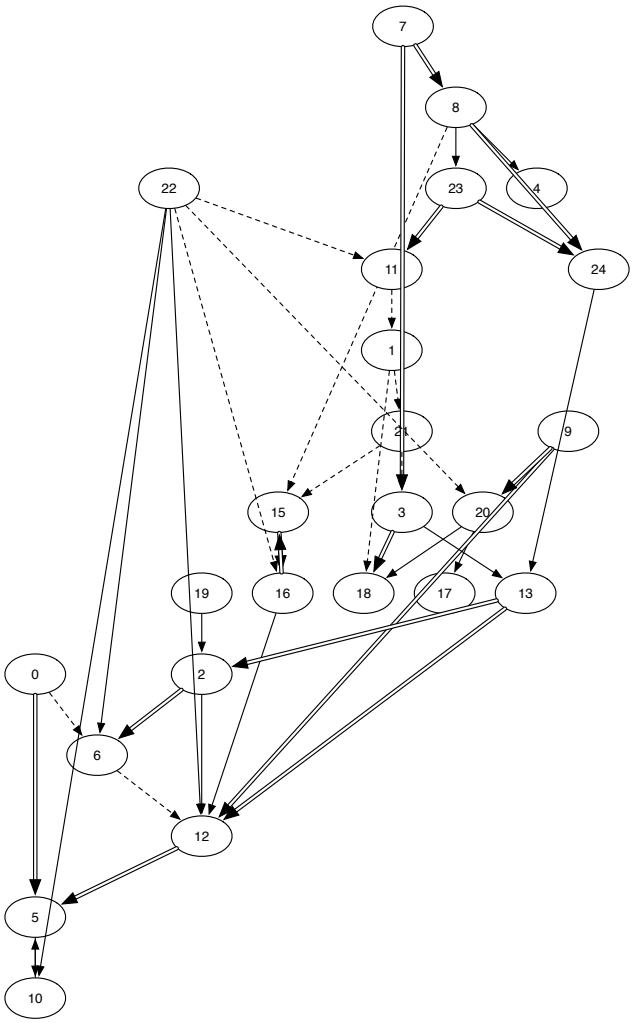
We apply our framework to a benchmark consisting of two different causal structures of simulated financial market data developed by [Kleinberg, 2013]. Both structures are based on a dataset with returns for 25 financial 'portfolios' (i.e. time series). All portfolios are stationary. Each simulated financial portfolio has a 4,000-day observation period (i.e. the size of $\mathbf{X}$ is $N \times T = 25 \times 4000$). Both datasets are created using the Fama-French Three-Factor Model [Fama and French, 1992] that can be used to describe stock returns based on the three factors 'volatility', 'size' and 'value'. A portfolio's return X_i^t depends on these three factors at time t plus a portfolio-specific error term [Kleinberg, 2013]. Thus, although the data is synthetic, it already includes some noise because of the error terms.

Structure 1 has 20 linear causal relationships, all having a delay of 1 day, and includes confounders and self-causation. Structure 2 is more complex. It has 40 linear causal relationships, each having a randomly generated delay of 1, 2 or 3 days, and includes confounders and 2-cycles. Both structures have some vertices which are not causally related to any other vertex. The ground truth graphs $\mathcal{G}_G$ of both structures are shown in Figure 14.

To evaluate the influence of hidden layers in the AD-DSTCN architecture for the TCDF accuracy, we vary the number of hidden layers in the AD-DSTCNs from $L = 0$ to $L = 3$ (and therefore also the receptive field R from 4 to 256). Recall that $L = 0$ corresponds to a network without hidden layers, where one convolution in a channel simply maps an input time series to the output.



(a) $\mathcal{G}_G$ of Structure 1 of the simulated financial time series



(b) $\mathcal{G}_G$ of Structure 2 of the simulated financial time series

Figure 14: Underlying causal structures of the simulated financial time series of Kleinberg, 2013. Both structures contain 25 vertices (numbered 0 to 24). Vertex 0 corresponds with time series $\mathbf{X}_0$. If a vertex is not present in the graph, then the corresponding time series has no causal relationships. Solid edges indicate a delay of one day, double edges mean two days and dashed edges mean a delay of 3 days.

5.3.2 Results and Discussion

For clarity, we have divided this evaluation section in multiple paragraphs, which discuss the evaluation of the causal discovery, delay discovery and CQII effectiveness, and present the comparison with existing methods.

Causal Discovery Evaluation The results of TCDF when applied to the simulated financial dataset corresponding to Structure 1 and Structure 2 are shown in Table 2. It can be seen that our framework discovered the causal relationships in both Structure 1 and the more complex Structure 2 reasonably well, with an F1'-score varying from 0.59 to 0.86.

The run times (which includes the comparison with the ground truth) varied from 14 to 30 minutes, with a mean run time of 24 minutes. The variance in run times can be explained by the fact that the number of potential causes that should be checked by CQII highly influences the run time.

Structure 1	$L=0$	$L=1$	$L=2$	$L=3$
Precision	0.83	0.70	0.79	0.66
Recall	0.80	0.76	0.96	0.76
F1'-score	0.82	0.73	**0.86**	0.70

Structure 2	$L=0$	$L=1$	$L=2$	$L=3$
Precision	0.96	0.88	0.83	0.72
Recall	0.56	0.65	0.65	0.50
F1'-score	0.71	**0.75**	0.73	0.59

Table 2: Evaluation results of TCDF applied to the financial data corresponding to Structure 1 resp. Structure 2, with a varying number of hidden layers L. Both structures are learnt with 2000 training epochs, learning rate $\lambda = 0.01$, kernel size $K = 4$ and dilation coefficient $c = 4$. The highest F1'-scores are highlighted in bold.

For Structure 1, Precision and Recall are nicely balanced since both measures are generally close to each other. The architecture with 2 hidden layers is a positive outlier with a Recall very close to 1, making it the best performing architecture for Structure 1. Apparently the TCDF could find almost all causal relationships that were present in the data by performing three convolutions (namely a convolution from the input layer and the two hidden layers). In practice, one does not know which number of layers would work best for a data set of which the ground truth is unknown. However, since the F1'-scores do not vary that much, these results suggest that the number of hidden layers is not decisive for the framework's accuracy.

For Structure 2, the Precision decreases when the number of hidden layers increases, whereas the Recall stays roughly the same. This reveals that the lower Precision is mainly caused by a higher number of False Positives. We expect that the increasing number of False Positives is caused by the increasing receptive field such that some spurious correlations with a high delay were discovered. Moreover, the Recall measures of Structure 2 are lower than the ones for Structure 1. We think that the worse Recall results come from the fact that the underlying causal structure of Structure 2 is much more complex, such that more causal relationships are not discovered.

In addition to judging the final outcome in terms of Precision, Recall and F1'-score, we are interested in the effectiveness of the graph reduction step. It turns out that our framework on average removes 3.8 edges from the learnt graph of Structure 1, and 1.5 edges for Structure 2, meaning that the graph reduction step was indeed effective. For example, for Structure 1, TCDF ($L = 0$) removed the indirect causal relationships $e_{10,9}$, $e_{10,4}$ and $e_{11,20}$.

Delay Discovery Evaluation As discussed in Section 5.1.2, we evaluate the discovered delays by calculating the percentages of delays that are correct (i.e. equal to the ground truth delay). Secondly, we evaluate the incorrectly learnt delays by calculating the mean distance to the ground truth delay relative to the receptive field R.

The evaluation results for Structure 1 and 2 are shown in Table 3. It can be seen that all discovered delays are correct when $L = 0$. This is not surprising, since the receptive field R is close to the maximum delays in $\mathcal{G}_G$ and the path length of the heaviest path to discover the delay is only 1. Interestingly, the percentage of correctly discovered delays stays relatively high when the number of hidden layers (and therefore the receptive field) is increased, meaning that our delay discovery algorithm works correctly on this benchmark. Furthermore, it can be seen that in most cases the incorrectly discovered delays have a rather small mean

error, meaning that an incorrect delay is still quite close to the ground truth delay. Only the TCDF with $L = 2$ and $L = 3$ for Structure 1 are quite off. Our hypothesis is that in these cases the weights in the last hidden layer were incorrectly learnt, such that the algorithm will follow a path that ends up in a much higher delay. For example, if the weights from the example shown in Figure 10 in the top layer were interchanged, the discovered delay would be 14 instead of 6.

Structure 1	$L = 0$	$L = 1$	$L = 2$	$L = 3$
% correct	100%	95%	95%	95%
μ_ϵ	-	0.063	0.810	0.890

Structure 2	$L = 0$	$L = 1$	$L = 2$	$L = 3$
% correct	100%	82%	93%	91%
μ_ϵ	-	0.250	0.039	0.023

Table 3: Evaluation results for the delay discovery for Structure 1 (left) and Structure 2 (right), where the first row shows the percentage of delays of the True Positives that were learnt correctly, and μ_ϵ denotes the mean distance of the incorrectly learnt delays to the ground truth, relative to the receptive field.

CQII Evaluation If we would not have applied CQII in TCDF ($L = 0$) to Structure 1, the number of False Positives would have been 59 and the number of True Positives 20. So, by applying CQII to Structure 1, 55 potential causes were correctly found to be no true cause and all true causes were correctly discovered, leading to a 112% increase of the F1'-score compared to not applying CQII (from 0.38 to 0.82).

For Structure 2, the number of False Positives would have been 28 without CQII (instead of 1 FP with CQII) and the number of True Positives would still be 24. Thus, applying CQII to Structure 2 led to a 40% increase of the F1-'score. The frameworks with other values for L gave comparable results, showing that CQII is very effective in reducing the number of False Positives.

However, the high number of False Positives when CQII is not applied indicates that the AD-DSTCNs have been trained for dozens of times to conclude that many time series are not a true cause. This might suggest that the selection of potential causes by the attention mechanism should be more strict, such that fewer time series have to be validated by CQII.

Comparison with Existing Approaches Table 4 shows a comparison between the results of our TCDF and the results of existing approaches applied to Structure 1 (top) and Structure 2 (bottom). We use TCDF ($L = 0$) in the comparison such that all methods have the same maximum delay. We mention the number of True Positives (TP), False Positives (FP) and False Negatives (FN), summarized by the F1-score. Since $\mathcal{G}_L$ and $\mathcal{G}_G$ will be sparse, we do not mention the, probably large number of, True Negatives. For TCDF, we also list the TP', FP' and F1'-score. A TP' corresponds to an edge denoting a direct causal relationship in $\mathcal{G}_L$ that corresponds to an indirect causal relationship in $\mathcal{G}_G$. FP' will not count these edges as a false positive.

Structure 1	TP(/TP')	FP(/FP')	FN	F1(/F1')	Correct delays
TCDF ($L = 0$)	15/20	9/4	5	0.68/**0.82**	100%
PCMCI	14	2	6	**0.80**	100%
tsFCI	5	1	15	0.38	100%
TiMINo	18	269	2	0.12	-

Structure 2	TP(/TP')	FP(/FP')	FN	F1(/F1')	Correct delays
TCDF ($L = 0$)	21/24	4/1	19	**0.64/0.71**	100%
PCMCI	14	5	26	0.47	100%
tsFCI	11	1	29	0.42	100%
TiMINo	34	254	6	0.20	-

Table 4: Results of our TCDF compared with results of PCMCI, TiMINo and tsFCI applied to Structure 1 (top) and Structure 2 (bottom). The highest F1-scores are highlighted in bold.

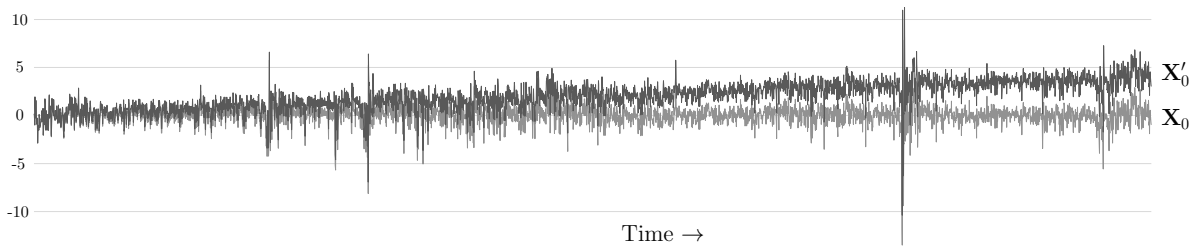


Figure 15: Plot showing the values of time series $\mathbf{X}_0$ from the financial dataset corresponding to Structure 1 and the transformed non-stationary time series $\mathbf{X}'_0$ that increases over time.

For Structure 1, it can be seen that TiMINo discovered the largest number of correct causal relationships. However, that method also discovered an incredibly large number of incorrect causal relationships and therefore has a low F1-score. This was in line with our expectations, since the authors already stated that TiMINo is not suitable for high-dimensional data [Peters et al., 2013]. The only two False Negatives correspond to the two cases of self-causation, since TiMINo assumes acyclicity. In contrast, where TiMINo discovers many incorrect causal relationships, tsFCI seems to be too conservative. It also missed both cases of self-causation. Our poor results of tsFCI correspond with poor results of tsFCI in experiments done by the authors on continuous data [Entner and Hoyer, 2010].

Overall, PCMCI performs best on the dataset of Structure 1, with an F1-score of 0.80, followed by our TCDF with 0.68. However, if we would count correct *indirect* causal relationships as True Positives (denoted by TP'), TCDF would have 20 True Positives and only 4 False Positives (FP'), leading to an F1'-score of 0.82. In addition, both PCMCI and TCDF discovered one of the two cases of self-causation.

The results for the more complex Structure 2 are comparable to the results of Structure 1. Although TiMINo found most True Positives, this method is not applicable in practical settings due to the high number of False Positives. Our method by far outperforms the existing methods with an F1-score and F1'-score that is much higher than the other ones.

When evaluating the delay discovery, one easily sees that all approaches perform equally well by measuring all delays correctly. Note that TiMINo only outputs causal relationships without an explicit notion of the underlying delays and could therefore not be taken into account in the delay evaluation.

5.4 Experiment 2: Non-stationary Simulated Financial Time Series

Stationarity is a common assumption in many temporal causal discovery methods (as shown in Table 1), meaning that those methods assume that the joint probability distribution of each time series does not change when shifted in time [Papana et al., 2014]. However, real data often possesses a non-constant mean and variance and hence such data is non-stationary [Papana et al., 2016]. We therefore test our framework on non-stationary data and evaluate how TCDF performs.

5.4.1 Data

We transform the financial datasets of [Kleinberg, 2013] for Structure 1 and Structure 2, described in Section 5.3.1, to model non-stationarity. We transform each time series $\mathbf{X}_i$ in dataset $\mathbf{X}$ to $\mathbf{X}'_i$, which includes a 'trend' that either goes up or down. More precisely:

$$\forall \mathbf{X}_i \in \mathbf{X}, \forall X_i^t \in \mathbf{X}_i, X_i^{t'} = \begin{cases} X_i^t + 0.001 \cdot t & \text{if } max(\mathbf{X}_i) \geq 0 \\ X_i^t - 0.001 \cdot t & \text{if } max(\mathbf{X}_i) < 0 \end{cases} \tag{12}$$

Thus, a time series that includes a positive value will be transformed to a time series that increases over time. If a time series consists of only negative values, it will be transformed to a decreasing time series. As an example, Figure 15 shows the values of the original time series $\mathbf{X}_0$ from the financial dataset corresponding to Structure 1 and the transformed non-stationary time series $\mathbf{X}'_0$.

5.4.2 Results and Discussion

The results of TCDF when applied to the non-stationary financial dataset corresponding to Structure 1 and Structure 2 are shown in Table 5. The run times (including the comparison with the ground truth) varied between 21 and 29 minutes, with a mean run time of 25 minutes.

Structure 1	$L = 0$	$L = 1$	$L = 2$	$L = 3$
Precision	0.43	0.74	0.43	0.33
Recall	0.83	0.71	0.48	0.43
F1'-score	0.57	**0.72**	0.45	0.38

Structure 2	$L = 0$	$L = 1$	$L = 2$	$L = 3$
Precision	0.66	0.81	0.73	0.67
Recall	0.47	0.65	0.24	0.35
F1'-score	0.55	**0.72**	0.37	0.46

Table 5: Evaluation results of TCDF applied to a non-stationary dataset derived from the financial dataset of Structure 1 (left) and Structure 2 (right), with a varying number of hidden layers L. Both structures are learnt with 2000 training epochs, learning rate $\lambda = 0.01$, kernel size $K = 4$ and dilation coefficient $c = 4$. The highest F1'-scores are highlighted in bold.

Although it has been shown that Convolutional Neural Networks can predict non-stationary time series [Borovykh et al., 2017] and also our used optimization algorithm (Adam) is appropriate for non-stationary objectives [Kingma and Ba, 2014], the F1'-scores are in general lower when TCDF is applied to non-stationary data compared to the results of TCDF applied to the original, stationary, financial time series. The F1'-scores vary between 0.37 and 0.72, compared to 0.59-0.86 with the stationary dataset. We can see that especially the Precision is worse, indicating that the number of False Positives increases when the non-stationary data is used. We suspect that this can be mainly explained by the fact that we artificially changed the time series values to include a certain 'trend' (either going up or going down) in the dataset. Since multiple time series will share the same trend, the data probably suggests that some of these time series are causally related.

Interestingly, for both Structure 1 and 2, the architecture with $L = 1$ gives the best results. We do not have a specific reason why TCDF with 1 hidden layer give the best results, and we therefore suspect that this result is dataset-specific. Apparently the TCDF had the highest F1'-score (and Precision) by performing two convolutions to these financial datasets.

The delay discovery results from TCDF applied to these non-stationary datasets are shown in Table 6. The results are in general comparable with the results from the stationary data. For Structure 1, the number of correctly discovered delays is at least 85% and in two cases even 100%. For Structure 2, the accuracy of the delay discovery varies between 81% and 87% and is therefore slightly lower than the results from the stationary data (82%-100%).

Structure 1	$L = 0$	$L = 1$	$L = 2$	$L = 3$
% correct	85%	100%	93%	100%
μ_ϵ	0.33	-	0.55	-

Structure 2	$L = 0$	$L = 1$	$L = 2$	$L = 3$
% correct	82%	85%	87%	81%
μ_ϵ	0.50	0.11	0.039	0.45

Table 6: Delay evaluation results of TCDF to non-stationary financial data for Structure 1 (left) and Structure 2 (right), where the first row shows the percentage of delays of the True Positives that were learnt correctly, and μ_ϵ denotes the mean distance of the incorrectly learnt delays to the ground truth, relative to the receptive field.

Besides, whereas the delay discovery was 100% correct when TCDF with $L = 0$ was applied to the stationary data, this is not the case with the non-linear data. We suspect that the AD-DSTCNs need at least two convolutions to learn the correct delays because of the more complex dataset. Namely, the transformation of the data from stationary to non-stationary could be seen as adding some noise to the time series, making it harder for the network to discover the correct delays.

Table 7 shows the results of TCDF compared with existing methods. For Structure 1, TCDF outperforms PCMCI, tsFCI and TiMINo in terms of F1-score when applied to this non-stationary dataset. It is not surprising that TCDF outperforms PCMCI and TiMINo, since both authors state that their method requires stationary data. This explains why the F1-score of PCMCI applied to Structure 1 decreases from 0.80 for the stationary dataset to 0.37 for the non-stationary dataset. However, the F1-score of PCMCI increases for

the non-stationary dataset of Structure 2, showing that PCMCI can still perform reasonably well with non-stationary data. Although TCDF has fewer False Positives, PCMCI outperforms our framework in terms of F1-score, with an F1-score of 0.51 compared to our 0.41. However, the F1'-score that includes correct indirect causal relationships of TCDF is a bit higher than PCMCI's accuracy. The results of TiMINo are very comparable to the results from the stationary data (and therefore still poor), despite the fact that its stationary-assumption was violated. The authors of tsFCI do not mention stationarity. Our results show that the accuracy of tsFCI decreases slightly, which might be caused by the extra 'noise' we've added.

When we evaluate the delay discovery of all methods, we see that tsFCI is the only method that discovered all delays from its True Positives correctly. However, the comparison with the other methods is a bit skewed because of the low number of True Positives of tsFCI. PCMCI shows to perform better than TCDF in terms of delay discovery in these non-stationary datasets, with 100% correct delays for Structure 1 and 95% correct delays for Structure 2.

Structure 1	TP(/TP')	FP(/FP')	FN	F1(/F1')	Correct delays
TCDF ($L = 0$)	16/20	26/22	4	**0.52/0.57**	85%
PCMCI	15	47	5	0.37	100%
tsFCI	4	2	16	0.31	100%
TiMINo	18	274	2	0.12	-

Structure 2	TP(/TP')	FP(/FP')	FN	F1(/F1')	Correct delays
TCDF ($L = 0$)	15/22	18/11	25	0.41/**0.55**	82%
PCMCI	22	24	18	**0.51**	95%
tsFCI	10	2	30	0.38	100%
TiMINo	36	252	4	0.22	-

Table 7: Results of our TCDF ($L = 0$) compared with PCMCI, tsFCI and TiMINo applied to a non-stationary dataset derived from the financial dataset of Structure 1 (top) and Structure 2 (bottom). The highest F1-scores are highlighted in bold.

Lastly, we evaluate the effectiveness of CQII for these non-stationary datasets. When we would not have applied CQII in TCDF ($L = 0$) to Structure 1, the number of False Positives would have been 50 and the number of True Positives would remain 20. So, by using CQII, 24 potential causes were correctly found to be no true cause and all true causes were correctly discovered, leading to an increase of the F1'-score of 34%. For Structure 2, the number of False Positives would have been 53 without CQII (instead of 11 FP with CQII). Interestingly, the number of True Positives would have been 27, instead of the 22 True Positives when CQII was applied to the non-stationary dataset of Structure 2. This means that 5 causes were incorrectly labeled as not being a true cause. Specifically, CQII denied $e_{7,3}$ since the loss of network $\mathcal{N}_3$ without $\mathbf{X}_7$ increased with only 4% which was less than the required significance threshold of 5%. The losses of the networks $\mathcal{N}_{16}$ and $\mathcal{N}_{18}$ stayed roughly the same when $\mathbf{X}_{22}$ resp. $\mathbf{X}_{20}$ were not included in the dataset. Surprisingly, $e_{22,11}$ and $e_{3,13}$ were denied because the network's loss decreased when $\mathbf{X}_{22}$ resp. $\mathbf{X}_3$ were not included in the dataset. We suspect that the losses did not increase significantly because of the fact that the time series are non-stationary and therefore easier to predict. However, applying CQII still increased the F1'-score of TCDF with 23%.

5.5 Experiment 3: Non-linear Simulated Financial Time Series

Some existing causal discovery methods, including the ones based on linear regression, assume that the causal relationships in a dataset are linear. This means that it is required that a time series $\mathbf{X}_j$ can be predicted by calculating:

$$\mathbf{X}_j = b_0 + b_1\mathbf{X}_i + ... + b_k\mathbf{X}_k + \epsilon \tag{13}$$

where b_0 is a constant, $b_1, ..., b_k$ are linear regression coefficients and ϵ is an error term [Geladi et al., 1999].

Thus, those methods cannot be used to reveal nonlinear causality, which inevitably hinders wide applications of the model in which non-linearity may occur [Hu and Liang, 2014]. Since we use a non-linear

activation function in our AD-DSTCN architecture, our framework should in theory be able to discover non-linear causal relationships. We evaluate this by applying TCDF to data with non-linear causal relationships.

5.5.1 Data

We transform the financial datasets of [Kleinberg, 2013], described in section 5.3.1, to model non-linear relationships. This can be done by applying a non-linear function to each time series $\mathbf{X}_i \in \mathbf{X}$ [Geladi et al., 1999]. Therefore, we replace each time series $\mathbf{X}_i \in \mathbf{X}$ by multiplying $\mathbf{X}_i$ with itself. More precisely:

$$\forall \mathbf{X}_i \in \mathbf{X}, \forall X_i^t \in \mathbf{X}_i, X_i^{t'} = (X_i^t)^2 \tag{14}$$

5.5.2 Results and Discussion

The results of TCDF when applied to the non-stationary financial dataset corresponding to Structure 1 and Structure 2 are shown in Table 8. The run times (including the comparison with the ground truth) varied between 14 and 28 minutes, with a mean run time of 21 minutes.

It can be seen that the accuracy of TCDF dropped vastly when applied to the non-linear financial time series compared to the results of TCDF applied to the original, non-linear, data. The F1'-scores from the non-linear dataset vary between 0.05 and 0.20, compared to 0.59-0.86 for the non-linear time series. Although our precision also decreased compared to the linear case, the F1'-score mainly decreased due to the decreased Recall, meaning that TCDF only discovered a fraction of all causal relationships in the non-linear dataset. From these poor results, we can conclude that TCDF is better at discovering linear causal relationships than non-linear ones. This suggests that although we apply non-linear activation functions in the AD-DSTCNs, the convolutional (and therefore linear) architecture of the networks dominates. Our results correspond with the findings of [Borovykh et al., 2017] that created a convolutional architecture for time series prediction and concluded that the error of a convolutional network applied to data with non-linear dependencies is higher than the error when the network is applied to linear data. We think it is worthwhile to test TCDF with other non-linear activation functions in the AD-DSTCN architecture, which might increase accuracy.

Structure 1	$L=0$	$L=1$	$L=2$	$L=3$
Precision	0.33	0.23	0.25	0.16
Recall	0.14	0.15	0.14	0.19
F1'-score	**0.20**	0.18	0.18	0.17

Structure 2	$L=0$	$L=1$	$L=2$	$L=3$
Precision	0.25	0.50	0.08	0.50
Recall	0.03	0.07	0.05	0.09
F1'-score	0.05	0.13	0.06	**0.16**

Table 8: Evaluation results of TCDF applied to a non-linear dataset derived from the financial dataset of Structure 1 (left) and Structure 2 (right), with a varying number of hidden layers L. Both structures are learnt with 2000 training epochs, learning rate $\lambda = 0.01$, kernel size $K = 4$ and dilation coefficient $c = 4$. The highest F1'-scores are highlighted in bold.

The delay discovery results from TCDF applied to the non-linear financial time series are shown in Table 9. It can be seen that the mean distance from the learnt delay to the ground truth (relative to the receptive field) is fairly low, meaning that TCDF still discovers delays that are close to the ground truths. However, the percentages of delays that are discovered correctly are a bit lower than the percentages when the algorithm is applied to the linear datasets. However, since the number of True Positives from the TCDF applied to non-linear data is much lower than the number of True Positives from the linear datasets, this comparison is slightly biased.

Structure 1	$L=0$	$L=1$	$L=2$	$L=3$
% correct	67%	100%	0%	50%
μ_ϵ	0.50	-	0.07	0.05

Structure 2	$L=0$	$L=1$	$L=2$	$L=3$
% correct	100%	50%	0%	0%
μ_ϵ	-	0.31	0.13	0.02

Table 9: Delay evaluation results of TCDF to non-linear financial data for Structure 1 (left) and Structure 2 (right), where the first row shows the percentage of delays of the True Positives that were learnt correctly, and μ_ϵ denotes the mean distance of the incorrectly learnt delays to the ground truth, relative to the receptive field.

Table 10 compares the results of TCDF ($L = 0$) with the results of three existing methods applied to the non-linear datasets. It can be seen that PCMCI did not discover any correct causal relationship in both structures and its F1-scores therefore equal 0. We suspect that the poor results are caused by the fact that we applied the linear 'ParCorr' correlation test in PCMCI, which is designed to only capture linear dependence. The author's GPACE non-linear dependence test will probably give better results (although it assumes that the time series are "sufficiently smooth functions" [Runge et al., 2017]). However, this implementation scales $\sim T^3$, making it impractical for datasets like ours with a relatively high number of time steps.

The delay discovery results of tsFCI show that the method is still conservative, but discovers less True Positives and more False Positives in the non-linear data than in the linear datasets. However, the discovered delays corresponding to its few True Positives are all correct. Lastly, TiMINo discovered many False Positives, which is in line with the results of Experiment 1 and 2. However, compared to the linear case, the number of True Positives has decreased from 18 to 7 for Structure 1 and from 34 to 21 for Structure 2. We suspect that this is mainly due to the linear fitting method of this TiMINo version. However, it is surprising that TiMINo still discovers some correct causal relationships, despite the fact that we violated the linearity assumption of this TiMINo version. The author also presented a non-linear variant of TiMINo, which should give better results, but this method was not fully implemented [Peters et al., 2013].

From this experiment we can conclude that all methods give poor results when applied to non-linear financial time series. However, TCDF scores best in terms of F1-score compared to the other methods when applied to Structure 1. For Structure 2, TiMINo scores best in terms of F1-score, closely followed by tsFCI and TCDF. But, TiMINo will not be applicable in practical settings due to the high number of False Positives.

Structure 1	TP(/TP')	FP(/FP')	FN	F1(/F1')	Correct delays
TCDF ($L = 0$)	2/3	7/6	18	**0.14/0.20**	67%
PCMCI	0	6	20	0.00	-
tsFCI	1	7	19	0.07	100%
TiMINo	7	270	13	0.05	-
Structure 2	TP(/TP')	FP(/FP')	FN	F1(/F1')	Correct delays
TCDF ($L = 0$)	1/1	3/3	39	0.05/0.05	100%
PCMCI	0	11	40	0.00	-
tsFCI	2	10	38	0.08	100%
TiMINo	21	267	19	**0.13**	-

Table 10: Results of our TCDF ($L = 0$) compared with PCMCI, tsFCI and TiMINo applied to a non-linear dataset derived from the financial dataset of Structure 1 (top) and Structure 2 (bottom). The highest F1-scores are highlighted in bold.

Lastly, we evaluate the effectiveness of CQII for these non-linear datasets. When we would not have applied CQII in TCDF ($L = 0$) to Structure 1, the number of False Positives would have been 56 and the number of True Positives would have been 10. This means that although CQII correctly labeled 50 causal relationships as not being causal, it also incorrectly labeled 7 causal relationships as being not causal. The F1'-score when CQII was not applied would have been 0.23 (instead of 0.20 when CQII is applied), meaning that applying CQII to the non-linear dataset of Structure 1 led to a 13% decrease in F1'-score.

For Structure 2, not applying CQII would result in 6 True Positives and 42 False Positives. Thus, for this structure CQII also incorrectly labeled some true causal relationships as being not causal, leading to an F1'-score of 0.14. Therefore, applying CQII decreased the F1'-score with 66% to 0.05. In most cases where CQII incorrectly denied a true causal relationship, the loss of the network did increase when the intervened distribution was used, but the increase was lower than our threshold of 5%. Therefore, it might be fruitful to lower this significance threshold when TCDF is applied to non-linear data.

5.6 Experiment 4: Hidden Confounders

TCDF should be able to discover the existence of a hidden confounder between two time series $\mathbf{X}_i$ and $\mathbf{X}_j$ when the confounder has equal delays to its effects $\mathbf{X}_i$ and $\mathbf{X}_j$. If the confounder has unequal delays to its effects, we expect that TCDF will discover an incorrect causal relationship between $\mathbf{X}_i$ and $\mathbf{X}_j$. To evaluate if TCDF deals with hidden confounders as expected, we apply TCDF to datasets in which there exists a hidden confounder.

5.6.1 Data

We apply TCDF to the simulated financial datasets of Kleinberg, 2013 in which we hide a confounder by replacing all the confounder's values by 0. We test TCDF on confounders with equal delays by hiding $\mathbf{X}_{16}$ in the dataset of Structure 1 and $\mathbf{X}_7$ in the dataset of Structure 2. Furthermore, we hide $\mathbf{X}_0$ in Structure 2 to evaluate how TCDF deals with a hidden confounder with unequal delays to its effects. Lastly, we hide in Structure 2 $\mathbf{X}_8$, which is a confounder that has equal delays to $\mathbf{X}_{23}$ and $\mathbf{X}_4$, and unequal delays to $\mathbf{X}_5$ and $\mathbf{X}_4$, and to $\mathbf{X}_5$ and $\mathbf{X}_{23}$. We apply TCDF with $L = 1$ since this architecture was most accurate for Structure 2 in Experiment 1 and also well-performing for Structure 1.

5.6.2 Results and Discussion

The results of TCDF applied to a simulated financial dataset in which a confounder is hidden, are shown in Table 11. We denote by $\rightarrow$ a causal relationship that is discovered by TCDF based on the method to discover hidden confounders as described in Section 4.2.3. The run times varied between 25 and 32 minutes, with a mean run time of 28 minutes.

It can be seen in Table 11 that TCDF discovered all hidden confounders with equal delays to the confounder's effects, which corresponds with our expectations. In two out of three cases, TCDF performed as expected by incorrectly learning a causal relationship between the effects of a hidden confounder with unequal delay. Interestingly, TCDF did not label $\mathbf{X}_{23}$ as a true cause of $\mathbf{X}_{15}$, because the attention mechanism did not discover $\mathbf{X}_{23}$ as potential cause of $\mathbf{X}_{15}$.

	Hidden Confounder	Effects	Equal Delays	Confounder Discovered	Learnt Causal Relationship(s)
Structure 1	$\mathbf{X}_{16}$	$\mathbf{X}_8$, $\mathbf{X}_5$	✓	✓	$\mathbf{X}_{16} \rightarrow \mathbf{X}_8$, $\mathbf{X}_{16} \rightarrow \mathbf{X}_5$
Structure 2	$\mathbf{X}_7$	$\mathbf{X}_8$, $\mathbf{X}_3$	✓	✓	$\mathbf{X}_7 \rightarrow \mathbf{X}_8$, $\mathbf{X}_7 \rightarrow \mathbf{X}_3$
Structure 2	$\mathbf{X}_0$	$\mathbf{X}_5$, $\mathbf{X}_6$	✗	✗	$\mathbf{X}_5 \rightarrow \mathbf{X}_6$
Structure 2	$\mathbf{X}_8$	$\mathbf{X}_{23}$, $\mathbf{X}_4$	✓	✓	$\mathbf{X}_8 \rightarrow \mathbf{X}_{23}$, $\mathbf{X}_8 \rightarrow \mathbf{X}_4$
Structure 2	$\mathbf{X}_8$	$\mathbf{X}_{15}$, $\mathbf{X}_4$	✗	✗	$\mathbf{X}_4 \rightarrow \mathbf{X}_{15}$
Structure 2	$\mathbf{X}_8$	$\mathbf{X}_{15}$, $\mathbf{X}_{23}$	✗	✗	-

Table 11: Results of our TCDF applied to a simulated financial dataset (Structure 1 or Structure 2) in which a confounder is hidden. 'Equal Delays' denotes whether the delays from the confounder to the confounder's effects are the same. Grey causal relationships denote that the discovered relationship was not causal according to the ground truth.

We also applied PCMCI, tsFCI and TiMINo to the datasets with a hidden confounder. The results are shown in Table 12. Whereas TCDF discovered 2 incorrect causal relationships because of a hidden confounder, PCMCI did not discover any incorrect causal relationship. However, in contrast to TCDF, PCMCI does not give any indication that two particular time series are correlated, or that there might be a hidden confounder between these time series.

tsFCI should handle hidden confounders by including a special edge type ($\mathbf{X}_i \leftrightarrow \mathbf{X}_j$) that shows that $\mathbf{X}_i$ is not a cause of $\mathbf{X}_j$ and that $\mathbf{X}_j$ is not a cause of $\mathbf{X}_i$. However, the results of tsFCI in our experiment are not in accordance with the theoretical claims. tsFCI discovered an incorrect causal relationship from $\mathbf{X}_5$ to $\mathbf{X}_8$ when $\mathbf{X}_{16}$ in Structure 1 was hidden, discovered the incorrect causal relationship from $\mathbf{X}_8$ to $\mathbf{X}_3$ when $\mathbf{X}_7$ in Structure 2 was hidden, and discovered the incorrect relationship from $\mathbf{X}_5$ to $\mathbf{X}_6$ when $\mathbf{X}_0$ was

	# Incorrect Causal Relationships	# Discovered Hidden Confounders
TCDF ($L = 0$)	2	**3**
PCMCI	**0**	0
tsFCI	3	0
TiMINo	6	0

Table 12: Results of the performance of TCDF compared with PCMCI, tsFCI and TiMINo. The middle column denotes how many incorrect causal relationships were discovered between the effects of the hidden confounder. The column on the right denotes how the number of hidden confounders that was discovered. The best results are highlighted in bold.

hidden. Furthermore, it did not find any correlation (i.e. no $\leftrightarrow$ edge and therefore no hidden confounder) between $\mathbf{X}_{15}$, $\mathbf{X}_{23}$ and $\mathbf{X}_4$ when $\mathbf{X}_8$ in Structure 2 was hidden.

Lastly, TiMINo discovered in all cases an indirect causal relationship. When the hidden confounders had equal delays to its effects, TiMINo found an instantaneous causal relationship from one effect to the other (but not vice versa). In the case of a hidden confounder with unequal delays, it discovered a causal relationship with a delay that was equal to the difference in delay between the two confounder's effects.

From this experiment, we can conclude that TCDF performs as expected by successfully discovering the presence of a hidden confounder when the delays to the confounder's effects are equal and by incorrectly discovering a causal relationship between the confounder's effects when the delays to the effects are unequal. Compared to other approaches, PCMCI performs better in terms of not discovering any incorrect causal relationships between the confounder's effects. However, only TCDF is able to detect the presence of hidden confounders.

5.7 Experiment 5: Prices of Dairy

Besides simulated data which is created in a controlled setting, we evaluate the accuracy of TCDF when applied to actual data. Since the ground truth causal relationships that might exist in the data are unknown, we evaluate the results by checking if the constructed temporal causal graph is in accordance with domain knowledge.

5.7.1 Data

We consider the actual Dutch monthly prices of milk, butter and cheese from the period January 2000 - June 2018[10], plotted in Figure 16. In this experiment, our framework discovers the causal relationships between these prices and their delays and includes this in a temporal causal graph. Although there is no ground truth of this dataset available, we can evaluate the results based on domain knowledge and compare our results with the graphs learnt by existing approaches.

We expect that the milk price causally influences both the price of butter and the price of cheese. The data might also incorrectly suggest that butter is a cause of cheese, due to the lower delay from milk to butter than from milk to cheese, as decribed in Section 2.3. However, since the prices are monthly, we can imagine that the time steps are too coarse to see this reflected in our dataset.

Besides evaluating the discovered causal relationships and their delays, we also analyse what *potential* causes were discovered, and which of these turned out to be *true* causes. This will give an indication on how our CQII validation step performs. For this analysis, in our learnt graphs $\mathcal{G}_L$ we draw a black edge for each *true* causal relationship, and a grey edge for each potential causal relationship that turned out to be *no true* causal relationship based on CQII.

We again use kernel size $K = 4$ and dilation coefficient $c = 4$. Since the dataset is rather small, we only evaluate AD-DSTCNs with $L = 0$ and $L = 1$.

[10]Source: Agrimatie - Wageningen Economic Research, prices for 'Melk, gemiddeld vet', 'Boter' and 'Boerenkaas' - https://www.agrimatie.nl/Prijzen.aspx

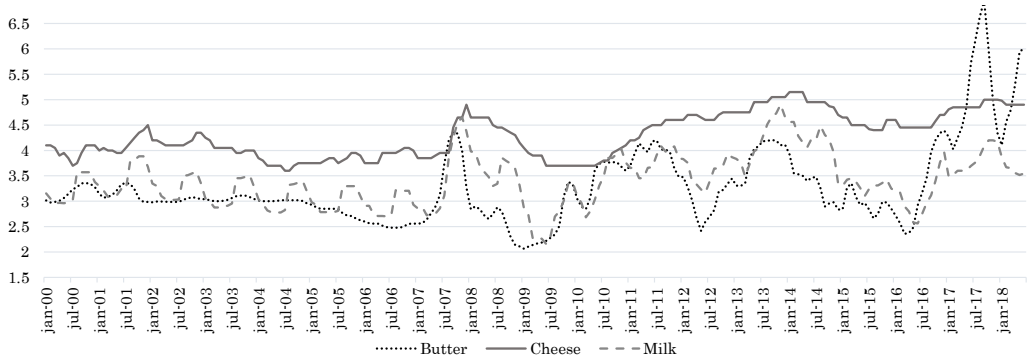


Figure 16: Dutch prices of milk (€/10kg), butter (€/1kg) and cheese (€/1kg).

5.7.2 Results

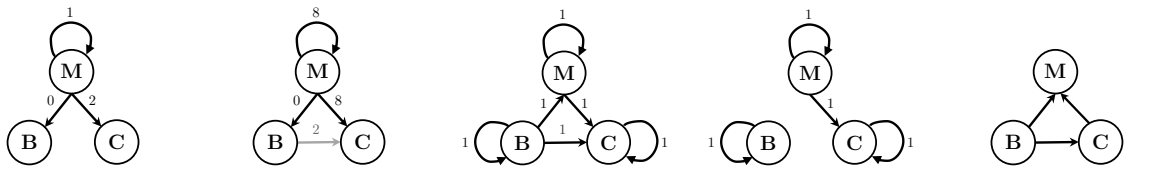
Both the TCDF with $L = 0$ and with $L = 1$ had a run time of 1.5 minute. The constructed temporal causal graphs by TCDF, PCMCI, tsFCI and TiMINo are shown in Figure 17. Note that we have excluded our causal strength scores since every time series has exactly one cause, such that all scores are equal to 1. The next paragraphs will discuss the results in more detail.

Figures 17a and 17b show that our framework with both $L = 0$ and $L = 1$ discovers that the price of milk causes both the price of butter and the price of cheese, which corresponds with our domain knowledge. In contrast, PCMCI, tsFCI and TiMINo did not discover that milk is a confounder and find causal relationships that are not in line with our domain knowledge.

In the constructed graph with $L = 0$ (Fig. 17a), the delay between milk and butter is 0 months, meaning that a change in the price of milk instantaneously (i.e. within 1 time step) influences the price of butter. Since butter is produced in less than a month (1 time step), this seems reasonable. The delay between milk and cheese is found to be 2 months with $L = 0$. Because of the long storage period of cheese, it seems reasonable that the delay from milk to cheese is higher than the delay from milk to butter. Lastly, our framework with $L = 0$ found that the price of milk causally influences itself with a delay of 1. The network probably attended most to the price of milk at time t to predict the next price at $t + 1$.

TCDF with $L = 1$ discovered the same causal relationships as with $L = 0$ (Fig. 17b), although the discovered delays are different due to a greater receptive field. The delay from milk to itself is discovered to be 8 months. This could denote a recurring pattern of 8 months, although we think a delay of 12 months would have been more logical because of a yearly cycle. The delay from milk to cheese was found to be 8 months. Since the ripening process of cheese varies from a few weeks to over a year, an average delay of 8 months might be correct, although 'Belegen' is considered as the average cheese type with a ripening time of 5-6 months [Roseboom et al., 2006]. We are therefore not sure if the discovered delay is correct (according to the data).

Most interesting however is the grey edge in Figure 17b, denoting that butter was found to be a *potential* cause of cheese, but not a *true* cause of cheese based on the CQII validation method. This is in accordance with our hypothesis that the data might suggest that butter is a cause of cheese, although this is not the case in reality. Both PCMCI and TiMINo included this incorrect causal relationship in their graph.



(a) $\mathcal{G}_L$ by TCDF ($L = 0$) (b) $\mathcal{G}_L$ by TCDF ($L = 1$) (c) $\mathcal{G}_L$ by PCMCI (d) $\mathcal{G}_L$ by tsFCI (e) $\mathcal{G}_L$ by TiMINo

Figure 17: Temporal causal graphs constructed by multiple temporal causal discovery methods, showing the causal relationships between the Dutch prices of milk (M), butter (B) and cheese (C).

5.7.3 Discussion

In contrast to the good results of TCDF applied to the actual dairy prices, the learnt temporal causal graphs of the existing approaches do not correspond with the domain knowledge. These poor results are probably caused by a violation of assumptions that the existing methods make on the data. The authors of PCMCI and TiMINo state that they assume stationarity, and we expect that tsFCI also makes this assumption because of its poor results. Figure 16 suggests that not all time series in this dataset are stationary. To formally check this, we use the Augmented Dickey-Fuller test [Dickey and Fuller, 1979] which tests the null hypothesis that a time series is non-stationary. In the test settings, we set the maximum lag to 3 (equivalent to the maximum delay used in TCDF($L = 0$) and in the existing methods). We find that in a 1% confidence interval, both the price of butter and cheese are non-stationary, which probably explains the poor results of the existing methods. To the contrary, in this experiment and in experiment 2, our TCDF has shown to be able to correctly handle non-stationary data.

6 Interpretation of Discovered Causality

Since a causal discovery method based on observational data cannot intervene in a system to check if manipulating the cause changes the effect, causal discovery methods are principally used to discover and investigate *hypotheses*. Therefore, a constructed temporal causal graph by TCDF should be interpreted as a hypothetical graph, learnt from observational time series data, which can subsequently be confirmed by a domain expert or experimentation. However, whereas most researchers are aware that real-life experiments are considered the "gold standard" for causal inference, manipulation of the independent variable of interest will often be unfeasible, unethical, or simply impossible [Rohrer, 2018]. Thus, causal discovery from observational data is often the better (or only) option.

However, drawing valid causal inferences on the basis of observational data always depends on assumptions that should be more or less plausible, and need to be accompanied by critical assessments. The following paragraphs will give an overview of well-known issues facing causal discovery that one should be aware of when interpreting results of any causal discovery method, including TCDF.

The first pitfall, called *imperfect regularities*, is that many causes are not *always* followed by their effects [Hitchcock, 2018]. TCDF, and most other causal discovery methods, aim to learn a type-level model describing causal relationships that hold in general. But the fact that a general causal relationship exists, does not necessarily mean that the effect occurs in all cases. For example, a bad diet is a cause of diabetes, even though some people that eat unhealthy will not develop diabetes, and some people that eat healthy might still develop diabetes. *Type-level* discoveries can therefore not directly be applied to *token-level* events that consider a specific individual.

A difficulty for many causal discovery methods is *irrelevance*. If a condition in the analysed dataset is often followed by some outcome, a causal discovery method might conclude that the condition and the outcome are causally related. However, the outcome might be completely irrelevant to this event. [Hitchcock, 2018] gives the example that hexed salt always dissolves when placed in water. Hexing does not cause the salt to dissolve (since salt would also have been dissolved if it was not hexed), but a causal discovery method can have difficulties with identifying that hexing is not a cause of dissolving salt. TCDF however tackles this problem by applying CQII. When an intervened distribution for the hexing variable is used (and assuming that both the positive and negative value of hexing are in the dataset), our framework will learn that hexing is no true cause of dissolving salt and that only water is, since the loss of the neural network will not increase.

The irrelevance issue relates to the problem of *sample selection bias*, that arises when the population representing the dataset is not representative of the population intended to be analyzed [Koller and Friedman, 2009]. It is important to critically assess which variables should be included in the dataset and to verify that the values of these variables are representative for the analysed population. If the dataset is not sufficient, new data collection approaches are needed such as repeated measures or collecting different samples [Rohrer, 2018]. In TCDF, this can be easily implemented by applying batch training, such that the neural networks learn to make correlations based on multiple samples.

Another difficulty important to take into account is the problem of *unmeasured variables*. TCDF can deal with unmeasured confounders (if the delays from the confounder to its effects are equal), but causal discovery methods will not correctly deal with unmeasured variables that are part of a causal chain of events. For example, the price of milk will influence the price of cream, which in turn influences the price of butter. However, if the price of cream is not measured, TCDF will conclude that the price of milk influences the price of butter. This result is correct, but the discovered direct causal relationship is in reality indirect.

Lastly, we discuss the occurrence of *spurious correlations*, where the values of two unrelated variables are coincidentally statistically correlated. For example, the number of lawyers in Amsterdam might be statistically correlated with the per capita consumption of cheese in Brazil with a delay of 2 months, according to a dataset. A causal discovery method will probably label this as a causal relationship if there are no counterexamples available. Only based on domain knowledge one can conclude that the discovered causal relationship is incorrect.

Thus, the practice of discovering causal relationships from observational data depends crucially on awareness of these aforementioned pitfalls. It might require careful planning before data collection begins to avoid sample selection bias. Furthermore, one should be critical when interpreting the causal discoveries and, if possible, do a real-life experiment to verify the findings.

7 Conclusions and Future Work

In this report, we introduced the Temporal Causal Discovery Framework (TCDF), a deep learning approach for causal discovery and structure learning from time series data. TCDF consists of multiple attention-based convolutional neural networks which we call *Attention-based Dilated Depthwise Separable Temporal Convolutional Networks* (AD-DSTCNs). These networks have an architecture that is optimized to predict a time series based on a multivariate temporal dataset. While an AD-DSTCN performs supervised prediction of a time series, it trains its attention-mechanism and internal parameters with backpropagation. Our experiments indicate that the implemented attention mechanism in an AD-DSTCN is accurate in discovering time series that are *correlated* with the predicted time series. Since correlation does not imply causation, TCDF subsequently applies a novel causal validation step to effectively distinguish *causality* from correlation. Our framework also interprets the internal parameters of each AD-DSTCN to discover the time delay between a cause and its effect. TCDF summarizes its findings by constructing a temporal causal graph that shows the discovered causal relationships between time series and their corresponding time delays. Our framework improves readability by implementing a graph reduction step that removes indirect causal relationships.

In an experiment based on actual dairy prices, we showed the excellent performance of TCDF. Compared to three existing temporal causal discovery methods (PCMCI, tsFCI and TiMINo), TCDF was the only method that successfully discovered that milk is a common cause of butter and cheese. The poor results of the existing methods can be explained by the fact that these statistical methods make idealized assumptions on a dataset that rarely hold in practice, such as acyclicity and non-stationarity.

Moreover, we evaluated TCDF on two simulated financial time series datasets with complex underlying causal structures that include confounders, feedback loops and self-causation. The learnt graphs by TCDF were close to the ground truth, with F1-scores that varied from 0.86 to 0.59. TCDF outperformed tsFCI and TiMINo in discovering causal relations, and had comparable or better accuracy than PCMCI for both stationary and non-stationary data. Our experiments also showed that TCDF discovered 82% to 100% of the delays correctly, which was comparable with the delay discovery results of existing methods. However, we found that the accuracy of TCDF decreases drastically for non-linear data. Although TCDF still performed comparable or better than existing methods when applied to non-linear financial data, we think it is worthwhile to test other activation functions in the AD-DSTCN architecture which might increase accuracy.

Lastly, TCDF includes a novel algorithm to detect the presence of hidden confounders. Our experiments show that, in contrast to existing temporal causal discovery methods, TCDF can successfully discover the presence of a hidden confounder when the time delays to the confounder's effects are equal.

A future work will be to use TCDF as a feature selector in order to compare the prediction results from a network that uses *all* observed variables, with results from a network that uses only *causal* variables. The latter has the benefit that the model is less complex because fewer variables are used, and that the learnt relationships by the model should be more robust. A future study has to show if the prediction accuracy will improve when all non-causal variables are removed. Our AD-DSTCN architecture already supports the prediction of time series relying only on causal variables, simply by fixing the attention scores of all non-causal variables to zero.

Secondly, we want to do an in-depth study of the time-complexity of our framework. Since many existing approaches do not report any complexity boundaries, it is useful to compare all approaches in terms of time-complexity or run time. We expect that TCDF will have a higher time-complexity than other methods because of our validation step by which each network may need to be trained multiple times. It is therefore worthwhile to study other validation approaches for TCDF that might have a lower time complexity.

In addition, our proposed framework summarizes the dependence structure of the dataset by learning a *static* temporal causal graph. In practice, causality could vary over time, because of local trends or seasonality. For future work, our structure learning method can be extended to flexibly accommodate these variations. A promising opportunity for this is the growing field of *multilayer networks* which enable to study causal networks on multiple interdependent levels in order to represent causal associations on different aggregation levels [Boccaletti et al., 2014]. Another possibility may be to implement a *state-space* model which can deal with time-varying influences [Brodersen et al., 2015]. We also see opportunities to learn these variations by interpreting the weight updates of a neural network that is being trained. Some large variations in weight updates could point to local trends or seasonality.

References

[Ancona et al., 2004] Ancona, N., Marinazzo, D., and Stramaglia, S. (2004). Radial basis function approach to nonlinear granger causality of time series. *Physical Review E*, 70(5):056221.

[Bahadori and Liu, 2013] Bahadori, M. T. and Liu, Y. (2013). An examination of practical granger causality inference. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 467–475. SIAM.

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). Convolutional sequence modeling revisited. Workshop paper at *International Conference on Learning Representations*.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Binkowski et al., 2017] Binkowski, M., Marti, G., and Donnat, P. (2017). Autoregressive convolutional neural networks for asynchronous time series. *arXiv preprint arXiv:1703.04122*.

[Boccaletti et al., 2014] Boccaletti, S., Bianconi, G., Criado, R., Del Genio, C. I., Gómez-Gardenes, J., Romance, M., Sendina-Nadal, I., Wang, Z., and Zanin, M. (2014). The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122.

[Borovykh et al., 2017] Borovykh, A., Bohte, S., and Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. In *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence*, pages 729–730.

[Brodersen et al., 2015] Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N., Scott, S. L., et al. (2015). Inferring causal impact using bayesian structural time-series models. *The Annals of Applied Statistics*, 9(1):247–274.

[Budhathoki and Vreeken, 2018] Budhathoki, K. and Vreeken, J. (2018). Causal inference on event sequences. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 55–63. Society for Industrial and Applied Mathematics.

[Chollet, 2017] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 1800–1807.

[Chu and Glymour, 2008] Chu, T. and Glymour, C. (2008). Search for additive nonlinear time series causal models. *Journal of Machine Learning Research*, 9(May):967–991.

[Datta et al., 2016] Datta, A., Sen, S., and Zick, Y. (2016). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 598–617. IEEE.

[Dickey and Fuller, 1979] Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431.

[Drton and Maathuis, 2017] Drton, M. and Maathuis, M. H. (2017). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393.

[Eichler, 2012] Eichler, M. (2012). Causal inference in time series analysis. *Causality: Statistical perspectives and applications*, pages 327–354.

[Entner and Hoyer, 2010] Entner, D. and Hoyer, P. O. (2010). On causal discovery from time series data using fci. *Probabilistic graphical models*, pages 121–128.

[Fama and French, 1992] Fama, E. F. and French, K. R. (1992). The cross-section of expected stock returns. *Journal of Finance*, 47(2):427–465.

[Gehring et al., 2017] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

[Geladi et al., 1999] Geladi, P., Hadjiiski, L., and Hopke, P. (1999). Multiple regression for environmental data: nonlinearities and prediction bias. *Chemometrics and Intelligent Laboratory Systems*, 47(2):165–173.

[Gilpin et al., 2018] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An approach to evaluating interpretability of machine learning. *arXiv preprint arXiv:1806.00069*.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

[Goudet et al., 2018] Goudet, O., Kalainathan, D., Caillou, P., Guyon, I., Lopez-Paz, D., and Sebag, M. (2018). Causal generative neural networks. *arXiv preprint arXiv:1711.08936v2*.

[Granger, 1969] Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438.

[Guo et al., 2018] Guo, T., Lin, T., and Lu, Y. (2018). An Interpretable LSTM Neural Network for Autoregressive Exogenous Model. Workshop paper at *International Conference on Learning Representations*.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Hitchcock, 2018] Hitchcock, C. (2018). Probabilistic causation. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2018 edition.

[Hu and Liang, 2014] Hu, M. and Liang, H. (2014). A copula approach to assessing granger causality. *NeuroImage*, 100:125–134.

[Huang and Kleinberg, 2015] Huang, Y. and Kleinberg, S. (2015). Fast and accurate causal inference from time series data. In *FLAIRS Conference*, pages 49–54.

[Hyvärinen et al., 2008] Hyvärinen, A., Shimizu, S., and Hoyer, P. O. (2008). Causal modelling combining instantaneous and lagged effects: an identifiable model based on non-gaussianity. In *Proceedings of the 25th International Conference on Machine Learning*, pages 424–431. ACM.

[Jiao et al., 2013] Jiao, J., Permuter, H. H., Zhao, L., Kim, Y.-H., and Weissman, T. (2013). Universal estimation of directed information. *IEEE Transactions on Information Theory*, 59(10):6220–6242.

[Kalainathan et al., 2018] Kalainathan, D., Goudet, O., Guyon, I., Lopez-Paz, D., and Sebag, M. (2018). Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*.

[Kalisch and Bühlmann, 2014] Kalisch, M. and Bühlmann, P. (2014). Causal structure learning and inference: a selective review. *Quality Technology & Quantitative Management*, 11(1):3–21.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. International Conference on Learning Representations, 2015.

[Kleinberg, 2013] Kleinberg, S. (2013). *Causality, probability, and time*. Cambridge University Press.

[Kleinberg, 2015] Kleinberg, S. (2015). *Why: A Guide to Finding and Using Causes*. O'Reilly.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[Liu and Aviyente, 2012] Liu, Y. and Aviyente, S. (2012). The relationship between transfer entropy and directed information. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 73–76. IEEE.

[Louizos et al., 2017] Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R., and Welling, M. (2017). Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, pages 6446–6456.

[Luo et al., 2013] Luo, Q., Ge, T., Grabenhorst, F., Feng, J., and Rolls, E. T. (2013). Attention-dependent modulation of cortical taste circuits revealed by granger causality with signal-dependent noise. *PLoS computational biology*, 9(10):e1003265.

[Malinsky and Danks, 2018] Malinsky, D. and Danks, D. (2018). Causal discovery algorithms: A practical guide. *Philosophy Compass*, 13(1):e12470.

[Marinazzo et al., 2008] Marinazzo, D., Pellicoro, M., and Stramaglia, S. (2008). Kernel method for nonlinear granger causality. *Physical review letters*, 100(14):144103.

[Martins and Astudillo, 2016] Martins, A. and Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

[Müller et al., 2012] Müller, B., Reinhardt, J., and Strickland, M. T. (2012). *Neural networks: an introduction*. Springer Science & Business Media.

[Nauta et al., 2018] Nauta, M., Bucur, D., and Stoelinga, M. (2018). LIFT: Learning fault trees from observational data. In *Quantitative Evaluation of Systems. 15th International Conference, QEST 2018, Beijing, China, September 4-7, 2018, Proceedings*. Springer International Publishing.

[Papana et al., 2016] Papana, A., Kyrtsou, C., Kugiumtzis, D., and Diks, C. (2016). Detecting causality in non-stationary time series using partial symbolic transfer entropy: evidence in financial data. *Computational Economics*, 47(3):341–365.

[Papana et al., 2014] Papana, A., Kyrtsou, K., Kugiumtzis, D., Diks, C., et al. (2014). Identifying causal relationships in case of non-stationary time series. Technical report, Universiteit van Amsterdam, Center for Nonlinear Dynamics in Economics and Finance.

[Peters et al., 2013] Peters, J., Janzing, D., and Schölkopf, B. (2013). Causal inference on time series using restricted structural equation models. In *Advances in Neural Information Processing Systems*, pages 154–162.

[Peters et al., 2017] Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. MIT Press.

[Quinn et al., 2011] Quinn, C. J., Coleman, T. P., Kiyavash, N., and Hatsopoulos, N. G. (2011). Estimating the directed information to infer causal relationships in ensemble neural spike train recordings. *Journal of computational neuroscience*, 30(1):17–44.

[Rohrer, 2018] Rohrer, J. M. (2018). Thinking clearly about correlations and causation: Graphical causal models for observational data. *Advances in Methods and Practices in Psychological Science*, 1(1):27–42.

[Roseboom et al., 2006] Roseboom, J., Gijsbers, G., and van der Zee, F. (2006). Kwaliteit als toekomst? Een verkenning van de Boerenkaasketen.

[Runge et al., 2017] Runge, J., Sejdinovic, D., and Flaxman, S. (2017). Detecting causal associations in large nonlinear time series datasets. *arXiv preprint arXiv:1702.07007*.

[Scheines et al., 1998] Scheines, R., Spirtes, P., Glymour, C., Meek, C., and Richardson, T. (1998). The tetrad project: Constraint based aids to causal model specification. *Multivariate Behavioral Research*, 33(1):65–117.

[Shen et al., 2018] Shen, T., Zhou, T., Long, G., Jiang, J., Wang, S., and Zhang, C. (2018). Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4345–4352. International Joint Conferences on Artificial Intelligence Organization.

[Singh et al., 2018] Singh, K., Gupta, G., Tewari, V., and Shroff, G. (2018). Comparative benchmarking of causal discovery algorithms. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CoDS-COMAD '18, pages 46–56, New York, NY, USA. ACM.

[Spirtes, 2010] Spirtes, P. (2010). Introduction to causal inference. *Journal of Machine Learning Research*, 11(May):1643–1662.

[Spirtes et al., 2000] Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.

[Spirtes and Zhang, 2016] Spirtes, P. and Zhang, K. (2016). Causal discovery and inference: concepts and recent methodological advances. In *Applied Informatics*, volume 3, page 3. Springer.

[Van Den Oord et al., 2016] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[van den Oord et al., 2016] van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798.

[Walther et al., 2004] Walther, D., Rutishauser, U., Koch, C., and Perona, P. (2004). On the usefulness of attention for object recognition. In *Workshop on Attention and Performance in Computational Vision at ECCV*, pages 96–103.

[Woodward, 2005] Woodward, J. (2005). *Making things happen: A theory of causal explanation*. Oxford university press.

[Yin et al., 2016] Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

[Zhang et al., 2017] Zhang, K., Schölkopf, B., Spirtes, P., and Glymour, C. (2017). Learning causality and causality-related learning: some recent progress. *National Science Review*.

LIFT: Learning Fault Trees from Observational Data

Meike Nauta, Doina Bucur, and Mariëlle Stoelinga

University of Twente, Enschede, The Netherlands
{m.nauta, d.bucur, m.i.a.stoelinga}@utwente.nl

Abstract. Industries with safety-critical systems increasingly collect data on events occurring at the level of system components, thus capturing instances of system failure or malfunction. With data availability, it becomes possible to automatically learn a model describing the failure modes of the system, i.e., how the states of individual components combine to cause a system failure. We present LIFT, a *machine learning method* for *static fault trees* directly out of *observational datasets*. The fault trees model probabilistic causal chains of events ending in a global system failure. Our method makes use of the Mantel-Haenszel statistical test to narrow down possible causal relationships between events. We evaluate LIFT with synthetic case studies, show how its performance varies with the quality of the data, and discuss practical variants of LIFT.

1 Introduction

Fault tree (FT) analysis [1] is a widely applied method to analyse the safety of high-tech systems, such as self-driving cars, drones and robots. FTs model how system failures occur as a result of component failures: the leaves of the tree model different failure modes, while the fault tree gates model how failure modes propagate through the system and lead to system failures. A wide number of metrics, such as the system reliability and availability, can then be computed to evaluate whether a system meets its dependability and safety requirements.

A key bottleneck is the construction of the FT. This requires domain knowledge, and the number of potential *failure causes* and contributing factors can be overwhelming: age, system loads, usage patterns and environmental conditions can all influence the failure mechanisms. It is thus appealing to learn FTs automatically from data, to assist reliability engineers in tackling the complexity of today's systems. This paper is a first step in this direction: we *learn* static FTs from *observational records*.

The fault-tree formalism. The nodes in an FT are either events or logical gates. Fig. 1 shows an example FT and the graphical notation. A part of the system is modelled by an *intermediate event*; a special intermediate event is the root node of the tree, called the *top event* or *outcome*, which models the global system failure. A set of *basic events*, distinct from the intermediate events, marks the most elementary faults in system components, may be annotated with a probability of occurrence, and form the leaves of the FT. Intermediate

events form the inputs and the output of any gate, and are the output of any basic event. The basic gates, AND and OR, denoted by the standard logic-gate symbols, model their standard logic meaning, in terms of causal relationships between the events in the input and the event in the output of any gate.

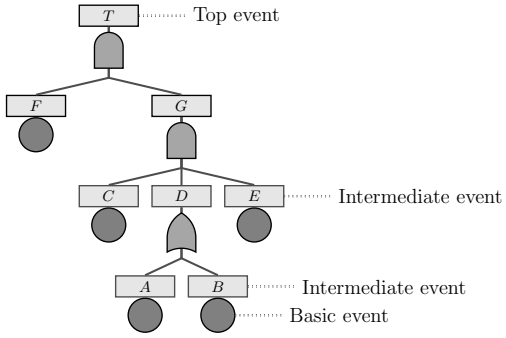


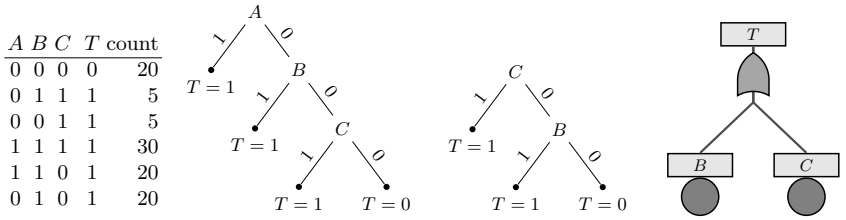
Fig. 1: Example fault tree with annotations

Summary of contribution. We learn static FTs with Boolean event variables (where an event variable has value True or 1 if that fault occurs in the system), n-ary AND/OR gates, and annotated with event failure probabilities. The input to the algorithm consists of raw, untimed observational data over the system under study, i.e., a dataset where each row is a single *observation* over the entire system, and each column *variable* records the value of a system *event*. All intermediate events to be included in the FT must be present in the dataset, but not all of those events in the dataset may be needed in the FT. We do not know what the basic events will be, nor which gates form the FT, nor which intermediate events are attached to the gates. We know the top event: the system failure of interest. Our main result is an algorithm that learns a statistically significant FT; we allow for a user-specified amount of noise, assumed uniformly distributed in the data. We evaluate the algorithm on synthetic data: given a "ground truth" FT, we synthesise a random dataset, apply the learning algorithm, and then compare the machine-learnt FT to the ground truth.

An example dataset is shown in Fig. 2a, in compact form: each row is a *count* (e.g., 20) of identical *records*, where each record is an untimed list of Boolean observations of events (denoted A, B, C and T, with T the global system failure, or outcome). The order of the records in a dataset is not significant.

A tree formalism commonly machine-learnt from such observational data is the Binary Decision Tree (BDT), a practical tool for the description, classification and generalisation of data [2]. The BDT learning algorithm appears to be a natural starting point for the design of an FT learning algorithm; however, we argue below why the BDT learning logic is unsatisfactory.

Detecting causality from data. The construction of a BDT is a greedy algorithm: a series of local optimum decisions determines subsequent branching nodes. Each decision uses a variable test condition (e.g., a classification error) to find the best split in the data records [3]. For example, when creating a BDT



(a) Dataset (b) Decision Tree (c) Causal Decision Tree (d) Fault Tree

Fig. 2: An example showing that a BDT does not encode causal relationships.

for the dataset in Fig. 2a to classify variable T, a naive approach is to first split the dataset on variable A, and obtain the BDT in Fig. 2b. However, decision trees model *correlations* (which are symmetric) and not the *causal* relationships (which are asymmetric) required for an FT. As a correlation between variables does not imply a causation, the knowledge represented in a decision tree does not provide root causes for faults, and thus cannot support decision making.

To overcome this problem, the *Causal* Decision Tree (CDT) [4] was recently introduced. A CDT differs from a BDT in that each of its non-leaf nodes has a causal interpretation with respect to the outcome T. CDTs find causal relationships automatically by using the Mantel-Haenszel statistical test [5]. A causal relationship between a variable and the outcome exists if the causal effect is statistically significant (i.e., is above random chance). A CDT for the dataset in Fig. 2a is in Fig. 2c; it shows that C has a causal relationship with T and that B has a causal relationship with T under the "context" (i.e., fixed variable assignment) $C = 0$. A is not included in this CDT. The path $(A = 1) \rightarrow (T = 1)$ in the BDT with probability $P(T = 1|A = 1) = 1$ correctly classifies half of the records in the dataset. However, the path does not code a causal relationship between A and T since, for example, given $C = 1$, $P(T = 1|A = 1) - P(T = 1|A = 0) = 0$. When fixing the value of C, a change in A does not result in a change in T. In fact, C causes T, and B causes T under the context $C = 0$.

CDTs on average achieve similar classification accuracy as decision trees, even though this is not a CDT objective; also, the size of CDTs is on average half that of decision trees [4], simplifying their analysis. Some aspects of CDT learning are useful in the automatic construction of an FT. However, while a CDT can only model the causal relationship between a variable and the outcome, the strength of an FT is the additional modelling of (a) multiple independent variables that may cause a failure, and (b) if-then Boolean logic. As shown in Fig. 2d, the CDT of Fig. 2c can be redrawn as an FT with a single OR gate.

In the following, Sect. 2 gives the related work on the automated synthesis of fault trees. Sect. 3 formally introduces FTs. Sect. 4 presents the LIFT algorithm and examples. Sect. 5 evaluates LIFT on datasets with noise or superfluous variables. Sect. 6 discusses possible LIFT variants. The conclusions, including future work, are presented in Sect. 7.

2 Related Work

System dependability evaluation via fault tree analysis is largely a manual process, performed over informal fault-tree models which will not accurately describe an evolving system [6]. Due to this, the *automatic synthesis of fault trees* has been of recent interest. However, we stress the fact that most of the existing contributions generate the necessary dependability information from existing, formal system models, and are thus Model-Based Dependability Analysis (MBDA) techniques [6–8]. In contrast, there is little research aiming to synthesise causal dependability information for *black-box systems*, for which formal models do not exist, or for which the quantity and quality of the available sensed data surpasses the quality and completeness of existing system models.

Learning fault trees from data. Observational data was used for machine-learning fault trees in the Induction of Fault Trees (IFT) algorithm [9], based on decision-tree learning. As in our method, all that is needed are observations of measurable quantities taking certain values. However, IFT completely disregards the matter of causality between events, and essentially learns a syntactically correct FT which encodes exactly the same information as a decision tree – so the FT is essentially a classifier, rather than a means of modelling causal effect.

Generating fault trees from formal system models. A diverse body of techniques is available for this; we refer to recent reviews on MBDA for a complete picture [6–8] and give here a brief overview of the most relevant generation methods. While these approaches cannot directly synthesise FTs from observational data (as in our work), other techniques able to learn the required system models from observational data could (indirectly) bridge this gap.

In the Hierarchically Performed Hazard Origin & Propagation Studies (HiP-HOPS) framework [10], any system model formalising the transactions among the system components, annotated with failure information for components (as Boolean expressions), may be used to synthesise an FT. Using these annotations, the synthesis is straightforward: it proceeds top-down from the top event and creates local FTs based on the component failure annotations; these are then merged into a global FT showing all combinations leading to system failure. If formal models in the AltaRica high-level system description language are available, they include explicit transitions modelling causal relations between state variables and events, which can similarly be used to synthesise classic FTs [11]. The Formal Safety Analysis Platform (FSAP/NuSMV-SA) generates, from NuSMV system models, FTs which show only the relation between top events and basic events, and not how faults propagate among the system components [12]. The Architecture Analysis and Design Language (AADL) includes an Error Model for the specification of fault information, and a number of techniques exist to translate an AADL model into static or dynamic FTs (recently, in [13]). AADL models have also been translated into models compatible with the HiP-HOPS and AltaRica frameworks, enabling cross-framework FT synthesis [6].

A process of FT generation with explicit reasoning about causality is described in [14]; however, this approach still requires a formal system model to exist. Given such a probabilistic system model, a set of probabilistic counterex-

amples (i.e., system execution paths of temporally ordered, interleaved events leading to a system fault) is obtained from the process of model-checking. As the system is concurrent, the counterexamples potentially, but not necessarily, model causality. Logical combinations of events are determined as causes of other events using a set of test conditions; the time complexity is cubic in the size of the set of counterexamples.

Other approaches. Causal Bayesian Networks (CBNs) [15] can also be learnt from observational data, as well as Boolean formulas (BFs) [16]; both models may be translated into FTs, and both learning problems are NP-hard or require exponential time [17, 16]. As our algorithm will also be shown to have a worst-case exponential complexity, both CBNs and BFs remain feasible alternatives to FT learning.

3 Background: Fault Trees

We define the basic components of an FT formally in Definitions 1–4.

Definition 1. *A **gate** G is a tuple $\langle t, \mathbf{I}, O \rangle$, where:*

- *t is the type of G, with $t \in \{And, Or\}$.*
- *$\mathbf{I}$ is a set of $n \geq 2$ intermediate events $\{i_1, ..., i_n\}$ that are inputs to G.*
- *O is the intermediate event that is output for G.*

We denote by $I(G)$ the set of intermediate events in the input of G, and by $O(G)$ the intermediate event in the output of G.

Definition 2. *An **AND gate** is a gate $\langle And, \mathbf{I}, O \rangle$ where output O occurs (i.e. O is True) if and only if every $i \in \mathbf{I}$ occurs.*

Definition 3. *An **OR gate** is a gate $\langle Or, \mathbf{I}, O \rangle$ where output O occurs (i.e. O is True) if and only if at least one $i \in \mathbf{I}$ occurs.*

Definition 4. *A **basic event** B is an event with no input and one intermediate event as output. We denote by $O(B)$ the intermediate event in the output of B.*

Intuitively, a basic event B models an elementary system fault in the real world; its output $O(B)$ is True when this elementary system fault occurs. Then, all system components modelled by the events in the input of an AND gate must fail in order for the system modelled by the event in the output to fail.

We then formalise the fault tree in Definition 5.

Definition 5. *A **fault tree** $\mathbf{F}$ is a tuple $\langle \mathbf{BE}, \mathbf{IE}, T, \mathbf{G} \rangle$, where:*

- *$\mathbf{BE}$ is the set of* basic events*; $\forall B \in \mathbf{BE}, O(B) \in \mathbf{IE}$. A basic event may be annotated with a probability of occurrence p.*
- *$\mathbf{IE}$ is the set of* intermediate events*, where $\mathbf{IE} \cap \mathbf{BE} = \emptyset$.*
- *T is the* top event*, $T \in \mathbf{IE}$.*
- *$\mathbf{G}$ is the set of* gates*; $\forall G \in \mathbf{G}, I(G) \subset \mathbf{IE}, O(G) \in \mathbf{IE}$.*

– *The graph formed by* $\mathbf{G}$ *should be connected and acyclic, with the top event* T *as unique root.*

Given fault tree $\mathbf{F}$*, we denote by* $IE(\mathbf{F})$ *the set of intermediate events in* $\mathbf{F}$*.*

The basic LIFT algorithm (Sect. 4) will learn trees rather than directed acyclic graphs (DAGs), i.e. an intermediate event can be the input of only one gate. Sect. 6 will then discuss a DAG variant of the LIFT algorithm.

Comparison FT-CDT. Unlike FTs, CDTs can be learnt from data, and also encode causal relationships between variables; an example CDT was given in Fig. 2c. However, there are major *syntactic* differences between the two formalisms. An FT can be n-ary, while a CDT can only be binary: every branching decision is based on a Boolean variable. Also, an FT is more concise: it models only the positive (failure) outcome, while the CDT must model both outcomes of any variable. Finally, the position of the outcome differs: while in FTs the top event models the system outcome, in a CDT this is modelled by leaf nodes.

4 Machine Learning Fault Trees

The dataset from which an FT can be learnt contains untimed, Boolean observations of system events; an FT *event* corresponds to a column *variable* in the dataset. A record and a dataset are formally defined in Definitions 6–7.

Definition 6. *A record* R *over the set of variables* $\mathbf{V}$ *is a list of length* $|\mathbf{V}|$ *containing tuples* $[\langle V_i, v_i \rangle]$*,* $1 \leq i \leq |\mathbf{V}|$*, where:*

– V_i *is a* variable name*,* $V_i \in \mathbf{V}$*.*
– v_i *is a* Boolean value *of* V_i*.*

Definition 7. *A dataset* $\mathbf{D}$ *is a set of* r *records, all over the same set of variables* $\mathbf{V}$*. Each variable name in* $\mathbf{V}$ *forms a column in* $\mathbf{D}$ *and each record forms a row. When* k *identical records are present in* $\mathbf{D}$*, a single such record is shown, with a new* count *column for the value* k*.*

A synthetic dataset (of 185 records in total, but only 11 unique records) is shown in Table 1. We assume the *sufficiency* of any dataset (i.e., all shared causes are measured [18]) and also its *faithfulness* (i.e., the data accurately represents the real-world dependencies [18]). However, because of either sensor glitches or human error, there may be some noise in the dataset (i.e., flipped bits).

From a dataset, causal relationships between (groups of) variables can be discovered to form an FT. For this, one can use the standard Mantel-Haenszel Partial Association test (PAMH) [5], a test used for the analysis of data that is *stratified*. When stratifying the dataset, the effect of other variables on the outcome variable T is eliminated, and hence the difference reflects the causal effect of one variable (say, E) on the outcome T. By this test, a causal relationship between two given variables is statistically significant if and only if the PAMH-score $\geq \tilde{\chi}^2_{\alpha,1}$, where $\tilde{\chi}^2_{\alpha,1}$ is the standard critical value of the chi-square

Table 1: Example dataset

A	B	C	D	E	F	G	T	count
0	0	0	0	0	0	0	0	30
1	0	1	1	0	1	0	0	20
0	1	0	1	1	0	0	0	20
0	1	0	1	1	1	0	0	20
1	0	0	1	0	0	0	0	15
0	0	1	0	0	1	0	0	15
0	0	0	0	1	0	0	0	15
0	0	1	0	1	1	0	0	15
1	1	1	1	1	0	1	0	20
0	1	1	1	1	1	1	1	10
1	0	1	1	1	1	1	1	5

distribution with 1 degree of freedom [19]. A significance level of $\alpha = 0.05$ or $\alpha = 0.01$ is often used in practice.

A stratum is formally defined in Definition 8 (and is a classic concept, as per [19]). A concrete example is given later in this section, in Example 1.

Definition 8. *Given a dataset* $\mathbf{D}$ *over the set of variables* $\mathbf{V}$*, a* ***stratum*** $\mathbf{S}_{E,T}$ *(where* E *and* T *are variables from* $\mathbf{V}$*) is a contingency table which shows the distribution of the values of variables* E *and* T *in the dataset, as counts, in the following format:*

	$T=1$	$T=0$	*Total*
$E = 1$	n_{11}	n_{12}	$n_{1.}$
$E = 0$	n_{21}	n_{22}	$n_{2.}$
Total	$n_{.1}$	$n_{.2}$	$n_{..}$

where n *denotes the number of records that satisfy a given valuation of* E *and* T*.*

The PAMH-score can be calculated over multiple strata (as done in CDTs [4] and first formalised in [19]); here we have a single stratum, as follows:

$$\mathrm{PAMH}(E,T) = \left(\left| \frac{n_{11}n_{22} - n_{21}n_{12}}{n_{..}} \right| - \frac{1}{2} \right)^2 \Bigg/ \frac{n_{1.}n_{2.}n_{.1}n_{.2}}{n_{..}^2(n_{..}-1)}$$

Using these concepts of strata and the PAMH-score, the LIFT algorithm[1], shown in Alg. 1, synthesises an FT from a dataset $\mathbf{D}$. A variable in $\mathbf{D}$ corresponds to an intermediate event. All intermediate events to be included in the FT must be present in the dataset, but not all of those events in the dataset may be needed in the FT. We do not know what the basic events will be, nor which gates form the FT, nor which intermediate events are attached to the gates.

[1] Code can be found at `https://github.com/M-Nauta/LIFT`

Algorithm 1: LIFT: Learning a Fault Tree from a dataset

Input: $\mathbf{D}$, a data set containing r records over $\mathbf{V}$;
T, the intended top event with $T \in \mathbf{V}$;
α, the significance level for the Mantel-Haenszel test
Result: Fault Tree $\mathbf{F}$

```
1  Function CheckANDGate(E, I):
2      result = False, pamh = 0.0
3      v = [v_1, ..., v_r] in which v_j is 1 if every i ∈ I in record j of D is 1
4      if n_12 of S_{v,E} < αn.. && n_21 of S_{v,E} < αn.. then
5          pamh = PAMH(v, T)
6          if pamh ≥ χ̃²_{α,1} then
7              result = True
8      return result, pamh
9  Function CheckORGate(E, I):
10     Similar to lines 2-8
11 Function CreateLevel(F, Leaves):
12     for l ∈ Leaves do
13         k = 2, gate = False
14         while not gate and k ≤ |V \ IE(F)| do
15             for a in generator of combinations of size k from V \ IE(F) do
16                 compute isGate, pamh = CheckANDGate(l, a)
17                 compute isGate, pamh = CheckORGate(l, a)
18             if at least one a exists where isGate was True then
19                 select that a and gate type t where pamh was maximum
20                 add gate ⟨t, a, l⟩ to F
21                 gate = True
22             else
23                 k++
24         if not gate then
25             p = ratio of records in D where l = 1
26             create basic event B as input for l in F, and annotate B with p
27     return F
28 let F = Fault Tree ⟨∅, {T}, T, ∅⟩
29 while at least one new event is added to F do
30     let Leaves = set of all intermediate events at the lowest level of F
31     F = CreateLevel(F, Leaves)
```

Checking a proposed gate. To create a fault tree $\mathbf{F}$, the LIFT algorithm iteratively adds a level to $\mathbf{F}$, starting with just the top event (line 28-31 in Alg. 1). Each time `CreateLevel` is called, the depth of the FT increases with one level. For each intermediate event E at the lowest level of $\mathbf{F}$, sets (of size ≥ 2) containing intermediate events not yet in $\mathbf{F}$ are proposed as input of a new (AND or OR) gate whose output is E. This is done by checking the gate for correctness according to the properties of Definitions 2–3. If both gates are correct, any design choice can be made, as discussed later in Sect. 6.

Example 1. Using the data set shown in Table 2a, a significance level α, the outcome variable T and the set of variables $\mathbf{I} = \{A, B\}$, one can check if gate G of the form $\langle And, \mathbf{I}, T\rangle$ meets the property specified in Definition 2 and is statistically significant using function `CheckANDgate`.

Table 2: Dataset for Example 1, over variables A, B and outcome variable T; stratum and fault tree learnt.

A	B	T	count
0	0	0	30
1	0	0	25
0	1	0	20
0	1	1	1
1	1	1	15

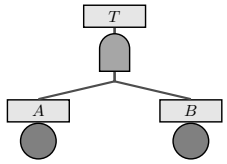
(a) Dataset

A	B	$\mathbf{v}$	T	count
0	0	0	0	30
1	0	0	0	25
0	1	0	0	20
0	1	0	1	1
1	1	1	1	15

(b) Dataset incl. var. $\mathbf{v}$ (AND gate)

	T=1	T=0	Total
$\mathbf{v} = 1$	15	0	15
$\mathbf{v} = 0$	1	75	76
Total	16	75	91

(c) Stratum $\mathbf{S}_{\mathbf{v},T}$



(d) FT learnt

A temporary new variable $\mathbf{v}$ is added to the dataset (Table 2b). $\mathbf{v}$ encodes an AND relation between A and B; $\mathbf{v}$ occurs (is 1) only when both A and B occur. This variable $\mathbf{v}$ can then be compared with top event T to measure if there is a causal relationship between T and A AND B. The stratum $\mathbf{S}_{\mathbf{v},T}$ is computed by counting the corresponding records in Table 2b, as shown in Table 2c.

The user can specify the ratio of noise allowed by LIFT per stratum. (If the user can assume that the flipped bits are uniformly distributed in the dataset, the expected per-stratum noise ratio is equal to the global noise ratio.) For simplicity, we set this noise "allowance" equal to the significance level α; the algorithm is easily modified for any other level. In the dataset shown in Table 2a, one can see that one record may be noise. In this example, we will set $\alpha = 0.05$, so we allow 5% noise in a stratum. It then follows that the proposed AND gate G of the form $\langle And, \{A, B\}, T\rangle$ meets the property of Definition 2 because in stratum $\mathbf{S}_{\mathbf{v},T}$ we have $n_{12} = 0$, $n_{21} = 1$, meaning one record where the values of $\mathbf{v}$ and T differ. We do allow a ratio α out of $n_{..} = 91$ to differ, but $1 < 0.05 \cdot 91$ holds. However, if we would have selected a significance level $\alpha = 0.01$, since $1 < 0.01 \cdot 91$ does not hold, the FT couldn't include this gate.

If the proposed gate has less noise than allowed (which is in this case true for $\alpha = 0.05$), we can determine if the causal relation between T and $\mathbf{v}$ is significant, by calculating the PAMH-score:

$$\mathrm{PAMH}(\mathbf{v}, T) = \left(\frac{15 \cdot 75 - 1 \cdot 0}{91} - \frac{1}{2}\right)^2 \Bigg/ \frac{15 \cdot 76 \cdot 16 \cdot 75}{91^2(91-1)} = 76.66\ .$$

For $\alpha = 0.05$, the critical value $\tilde{\chi}^2_{\alpha,1} = 3.84$. Since the PAMH-score is higher than $\tilde{\chi}^2_{\alpha,1}$, $\mathbf{v}$ and T can be concluded to have a significant causal relationship.

Similarly, a proposed OR gate is checked. By creating a temporary new variable $\mathbf{v}$ for the OR gate, one can create a stratum to calculate the noise and the PAMH-score in a similar way. This OR gate will have too much noise and is

therefore not correct. So, $\langle And, \mathbf{I}, T\rangle$ is added to $\mathbf{F}$. Table 2d shows the final FT learnt from the original dataset in Table 2 for $\alpha = 0.05$.

An FT may have a path containing two subsequent gates of the same type; in this case, the FT solution is not unique, and one may optimise for either *minimal gate sizes*, or *minimal tree depth*. We choose here the former, i.e., select the smallest input sets for all gates. LIFT is easily modified for another aim. Example 2 below clarifies this situation.

Example 2. Take the dataset in Table 1 at the beginning of this section. The LIFT algorithm starts with an FT containing only the top event T (line 28 in Alg. 1). For this top event, the algorithm generates all combinations (sets) of intermediate events, in order of increasing size (line 15). For each set $\mathbf{a}$ containing intermediate events, LIFT tests whether a gate $\langle Or/And, \mathbf{a}, T\rangle$ (either AND or OR) meets the property in Definitions 2–3 and does not exceed the noise allowance (line 4). If true, LIFT checks if the PAMH score is higher than the threshold for the Mantel-Haenszel test.

For this dataset, there is no correct OR gate $\langle Or, \mathbf{a}, T\rangle$. However, there are 9 sets of intermediate events that can act as input for a correct and significant AND gate $\langle And, \mathbf{a}, T\rangle$: $\mathbf{a} = \{F, G\}$, $\mathbf{a} = \{E, F, G\}$, $\mathbf{a} = \{D, F, G\}$, $\mathbf{a} = \{C, F, G\}$, $\mathbf{a} = \{D, E, F, G\}$, $\mathbf{a} = \{C, E, F, G\}$, $\mathbf{a} = \{C, D, F, G\}$, $\mathbf{a} = \{C, D, E, F\}$ and $\mathbf{a} = \{C, D, E, F, G\}$. In other words, there are multiple structural solutions for the FT, when the FT has a path with two subsequent gates of the same type. In such cases, LIFT learns the solution with minimum-sized gates. The input sets are generated in increasing size, and LIFT will stop proposing input sets when the minimum correct input set is found. In this example, the smallest set has size two, namely $\{F, G\}$. Therefore, gate $\langle And, \{F, G\}, T\rangle$ is added to $\mathbf{F}$ (as shown in Fig. 3). In case of multiple correct sets of the same size (which can arise when there are more variables in the dataset than needed in the FT) the set with the highest PAMH-score is selected (line 19 in Alg. 1). The algorithm can be easily modified for another design decision, as argued later in Sect. 6.

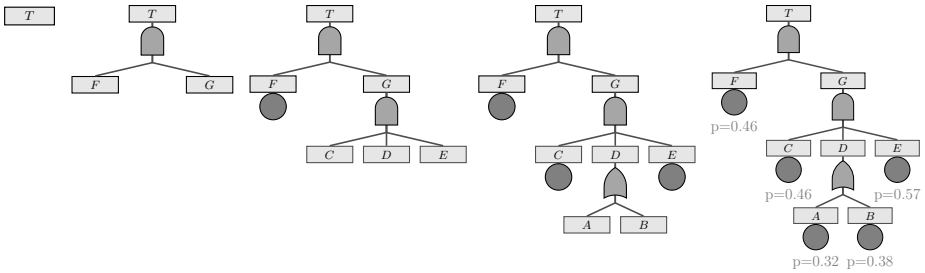


Fig. 3: Applying LIFT in Example 2 on the dataset shown in Table 1.

In the next iteration, for both F and G again sets of intermediate events are tried. The search space is now $2^5 - 5 - 1 = 26$, because $|\mathbf{V} \setminus IE(\mathbf{F})| = 5$. There is no correct gate $\langle Or/And, \mathbf{a}, F\rangle$ for intermediate event F. Therefore, a basic

event is added as input for F (line 26). For intermediate event G, one correct and significant AND gate $\langle And, \{C, D, E\}, G\rangle$ is found and added to $\mathbf{F}$. Similar iterations are done for C, D and E followed by A and B, as shown in Fig. 3.

When the dataset contains information on system states which are always measured in a fixed time horizon (i.e. *discrete* time), one can easily derive stochastic measures such as failure probabilities using standard probability laws. The statistical probability that an event $E \in \mathbf{D}$ occurs is simply $P(E = 1) = \frac{\#\text{ records where E=1}}{\text{total }\#\text{ records}}$; all basic events are annotated with these probabilities.

5 Evaluation

The algorithm is evaluated following the approach shown in Fig. 4. A number of fault trees $\mathbf{F}_{gt}$ are generated as ground truth; from each of these FTs, a dataset is synthesised randomly, including adding noise and superfluous variables (both of these processes of synthesis are described below). LIFT takes this dataset and a given significance value α as input, and learns another FT $\mathbf{F}$, which can then be compared to the ground truth. We say that a learnt FT is "correct" if it is *structurally equivalent* to the ground-truth FT, i.e., syntactically (and not only semantically) equivalent, where only the order of the inputs to any gate may differ. We require that the learnt FT recovers the *exact gates* as in the ground truth, since these gates may model concrete system components, for which the correct causes of failure should be learnt. Our evaluation is thus stronger than an isomorphism check for the FTs.

Furthermore, we assess how noise and superfluous variables in the dataset influence the ratio of correctly learnt FTs.

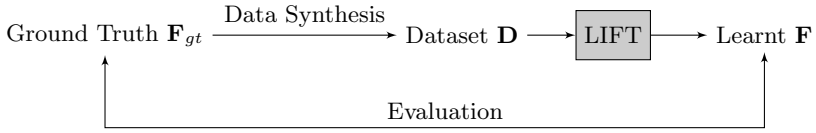


Fig. 4: Evaluation approach: Randomly generate a dataset from an FT, apply LIFT to that dataset and compare the learnt FT with the ground truth.

Generating all FTs of a certain size As ground truth, we generate *all possible* FTs over a fixed number (here, 8) of intermediate events, with no probabilities annotated on basic events. We only generate trees and leave DAGs (with shared variables) as future work. To mimic a manually constructed FT where readability is important, we set a minimum of 2 intermediate events and a maximum of 5 intermediate events as input to a gate, and thus obtain 76 different FTs.

Generating a synthetic dataset from an FT Based on a generated FT, we mimic a real-life situation by randomly synthesising 1000-record datasets where basic events happen with a certain probability (and are not rare events). The generation process starts with valuating all basic events to either 0 or 1, with a

randomly chosen probability between 20% and 50% that each basic event is 1. These values are propagated through the gates in the FT up to the top event; dependent on the type of each gate, the gate's output event is assigned 0 or 1. Each iteration of this procedure results in one data record.

To assure that gates are correctly recognised and that every gate is at least once true, every combination of inputs for a gate occurs in at least c% of the rows in the dataset (i.e. in the case of 1000 rows and c=2, every combination occurs at least 20 times). We created datasets for both $c = 0.5\%$ and $c = 2\%$. We leave the task of discriminating between rare events and noise for future work.

Adding noise to the dataset In a real-life situation, having perfectly clean data is rare because of wrong measurements, sensor glitches or manual errors for example. To mimic noise, a number up to 5% of the rows in the dataset are added, each with 1-2 wrong (flipped) values.

Adding superfluous variables to the dataset Our algorithm should also create a correct fault tree when there are variables in the dataset which have *no causal effect* and should not be included in the learnt FT. We thus experiment with adding up to 4 non-causal system variables. The case of a causal superfluous variable is discussed in Sect. 6.

5.1 Results

An analysis is done on the influence of noise or superfluous variables in the dataset on the number of correct fault trees obtained by LIFT.

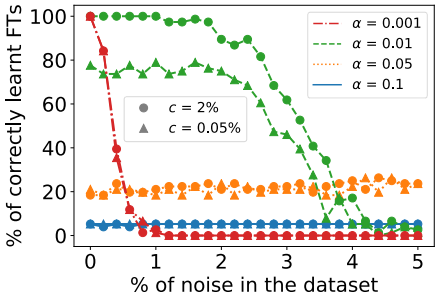


Fig. 5: Percentage of correctly learnt fault trees relative to the percentage of rows with noise in the dataset. All 76 different FTs with 8 intermediate events are generated. The dataset for each FT contains no non-causal variables, and 1000 records plus an extra percentage of noisy records.

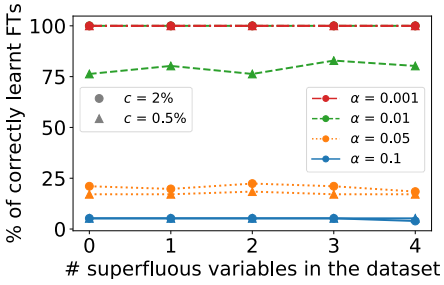


Fig. 6: Percentage of correctly learnt FTs relative to the number of non-causal variables in the dataset. All 76 different FTs with 8 intermediate events are generated. The dataset for each FT contains 1000 records, no noisy records, but up to 4 non-causal variables.

As we generated all ground-truth FTs with exactly 8 intermediate events, and 2-5 inputs for each gate (76 FTs in total), the basic datasets contain 8 columns (one for each intermediate event), and 1000 rows. Noisy rows are then added to the dataset, depending on the level of noise desired.

Figure 5 shows the percentage of correctly learnt fault trees relative to the percentage of rows with noise in the dataset. All learnt FTs are correct in the absence of noise and with the significance level $\alpha = 0.001$. However, this α is by nature incapable of correctly dealing with noise, since LIFT may not find a significant gate due to the noise. A higher α is less sensitive for noise, but does result in a lower number of correct FTs. A learnt FT may be incorrect when LIFT finds a significant gate with a smaller number of inputs than what should actually be the case. Therefore, the significance level should be chosen based on the amount of noise in the dataset. Furthermore, one can see that c naturally influences the number of correctly learnt FTs: the less rare the events are in the dataset, the more likely is LIFT to learn the correct FT.

Figure 6 shows the percentage of correctly learnt FTs relative to the number of non-causal random variables present in the dataset; one can see that these variables have little effect on the accuracy, showing that LIFT indeed finds only causal relationships.

5.2 Complexity

Time complexity LIFT exhaustively checks all input event combinations in order of their size, so in worst case there is one gate with all variables (except the top event) as input. This means that for all input sets, a stratum is created that loops over all r records and over k variables that are in that set. The number of different combinations of size k is $\binom{n}{k}$ where $n = |\mathbf{V}| - 1$. Therefore, the time complexity of these operations is $r \cdot \sum_{k=2}^{n} k \cdot \binom{n}{k}$. The PAMH-score of each stratum is calculated and compared with the significance level, which has a constant time complexity. This results in a time complexity of $O(nr2^n)$.

Learning boolean formulae, closely related to learning static fault trees, from examples obtained by querying an oracle is exponential in the size of the vocabulary in the general case as well as for many restrictions [16]. More precisely, a static fault tree with only AND and OR gates can be seen as a monotone boolean function for which the Vapnik-Chervonenkis (VC) dimension is exponential in n [20]. So, a general *exact* FT learning algorithm cannot be more efficient than the VC dimension. Reaching better complexity, which could be useful for large datasets, is then only possible when an approximated FT is learnt, instead of an exact solution. Such a variant of Alg. 1 may apply a *greedy search-and-score* approach rather than our constraint-based approach with exhaustive search, as inspired by structure-learning algorithms for Bayesian networks. However, those algorithms may suffer from getting stuck in a local maximum, resulting in a lower reconstruction accuracy. Furthermore, the highest-scoring network structure is not necessarily the only viable hypothesis [21].

Space complexity The input for Algorithm 1 consists of dataset $\mathbf{D}$ with r records and n columns, top event T and significance level α. Therefore, the input space complexity is $\Theta(rn)$. If the generator of combinations is on-the-fly, its auxiliary memory complexity is $O(n^2)$.

6 Discussion

Interpretation of causality Currently, all intermediate events that should be in the fault tree have to be included in the dataset. However, obtaining a dataset containing all relevant variables may be impractical. One problem is the presence of hidden variables that influence measured variables but are not in the dataset themselves [18]. The other one is the selection bias: values of unmeasured variables may influence whether a unit is included in the dataset [21]. This can result in a learnt causal relationship between observed variables that does not correspond to the real causal relations. Drawing valid causal inferences using observational data is therefore not just a mechanistic procedure, but always depends on assumptions and justification that require domain knowledge [22]. We are aware of the critical assessment of causal claims based on observational data, but we think the learnt fault tree will still be valuable to give insights which possibly were unknown beforehand and facilitates further causal inference.

Algorithm variants We made certain design decisions for the basic LIFT algorithm in Alg. 1. Below, we present some of the many possible variants.

Multiple gate types In the case of multiple significant correct gates with the same number of inputs, the LIFT algorithm chooses the one with the highest PAMH-score. However, there may be cases where both an OR gate and an AND gate are correct. For example, in case of the dataset as shown in Table 3, an OR gate will be created when a very high significance level is chosen. However, two of these records may be noise, so with a lower significance level an AND gate will result in a correct gate as well. Selecting the gate type is then a matter of choice: one can argue to choose the OR gate as this matches exactly the dataset, or choose the AND gate since the interpretation of this gate is stricter than the OR gate. One can also argue that the algorithm need not make a decision at all and that it outputs multiple FTs. LIFT is easily modified for any design decision.

Table 3: Dataset where both an OR gate and an AND gate may be correct.

A	B	T	count
0	0	0	1
1	0	1	1
0	1	1	1
1	1	1	10,000

Multiple significant FTs When there are *causal* superfluous variables in the dataset, there may be cases of multiple correct sets of intermediate events of the same size, that can all serve as input to a statistically significant gate. While the basic LIFT algorithm chooses the input set with the highest PAMH-score, it is easily modified for a different design choice, such as returning all correct FTs.

The FT as a Directed Acyclic Graph (DAG) The basic LIFT algorithm learns trees, so the examples and evaluation presented in this paper all learn tree structures. However, in general FTs may share subtrees, meaning that an intermediate event can be the input of multiple gates, and therefore have a directed acyclic structure [1]. The LIFT algorithm can be modified to create DAGs by generating broader combinations $\mathbf{a}$ of intermediate events, outside the while loop at line 14 of Alg. 1: instead of a generator of all combinations of size ≥ 2 from $\mathbf{V} \setminus IE(\mathbf{F})$, one can instead have a generator of all combinations from $\mathbf{V} \setminus T$, with an extra check that the created graph $\mathbf{F}$ remains acyclic.

More efficient exploration of variable combinations Other features of the dataset (e.g., the graph of dependencies between variables), or even domain knowledge, may be used to reduce the number of combinations of variables to be tried by LIFT as inputs to gates.

7 Conclusion

In this paper, we presented an algorithm to automatically learn a statistically significant fault tree from Boolean observational data, inspired by the construction algorithm for Causal Decision Trees. In absence of noise, all learnt FTs were found to be structurally equivalent to the ground truth when the significance level is 0.001. With up to 3% noise in the data, a significance level of 0.01 results in around 65% correct FTs. As a downside, the basic LIFT algorithm does an exhaustive search, and thus has exponential time complexity. It also cannot deal with hidden variables.

In future work, the algorithm can be extended to learn other elements of a fault tree, such as the XOR gate (true if and only if exactly one of its input events is true). Note that elements that need sequence information (such as the Priority-AND gate or the SPARE gate) cannot be implemented, since the required dataset format doesn't contain timing information. Learning fault trees from *timed observational data* is also a direction for future work. For this, learning Bayesian networks, closely related to FTs, may also be a competitive direction to take. Moreover, one may allow continuous data instead of only binary values, similar to the C4.5 algorithm for decision trees [23] that creates a binary expression for continuous values. This expression encodes the conditions under which a measurement results in a failure.

Acknowledgements This research was supported by the Dutch STW project SEQUOIA (grant 15474). The authors would like to thank Joost-Pieter Katoen and Djoerd Hiemstra for valuable feedback.

References

1. Ruijters, E., Stoelinga, M.: Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. Computer Science Review **15** (2015) 29–62
2. Murthy, S.K.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery **2**(4) (1998) 345–389
3. Tan, P., Steinbach, M., Kumar, V.: Introduction To Data Mining. Pearson Education (2006)
4. Li, J., Ma, S., Le, T., Liu, L., Liu, J.: Causal decision trees. IEEE Transactions on Knowledge and Data Engineering **29**(2) (2017) 257–271
5. Mantel, N., Haenszel, W.: Statistical aspects of the analysis of data from retrospective studies of disease. J. of the national cancer institute **22**(4) (1959) 719–748
6. Kabir, S.: An overview of fault tree analysis and its application in model based dependability analysis. Expert Systems with Applications **77** (2017) 114–135
7. Aizpurua, J.I., Muxika, E.: Model-based design of dependable systems: Limitations and evolution of analysis and verification approaches. International Journal on Advances in Security Volume 6, Number 1 & 2, 2013 (2013)
8. Sharvia, S., Kabir, S., Walker, M., Papadopoulos, Y.: Model-based dependability analysis: State-of-the-art, challenges, and future outlook. In: Software Quality Assurance. Elsevier (2016) 251–278
9. Madden, M.G., Nolan, P.J.: Generation of fault trees from simulated incipient fault case data. WIT Trans. Information and Communication Technologies **6** (1994)
10. Papadopoulos, Y., McDermid, J.: Safety-directed system monitoring using safety cases. PhD thesis, University of York (2000)
11. Li, S., Li, X.: Study on generation of fault trees from Altarica models. Procedia Engineering **80** (2014) 140–152
12. Bozzano, M., Villafiorita, A.: The FSAP/NuSMV-SA safety analysis platform. International Journal on Software Tools for Technology Transfer **9**(1) (2007) 5
13. Li, Y., Zhu, Y.a., Ma, C.y., Xu, M.: A method for constructing fault trees from AADL models. In: Int. Conf. on Autonomic and Trusted Computing, Springer (2011) 243–258
14. Leitner-Fischer, F., Leue, S.: Probabilistic fault tree synthesis using causality computation. Int. J. Critical Computer-Based Systems 30 **4**(2) (2013) 119–143
15. Li, J., Shi, J.: Knowledge discovery from observational data for process control using causal Bayesian networks. IIE transactions **39**(6) (2007) 681–690
16. Jha, S., Raman, V., Pinto, A., Sahai, T., Francis, M.: On learning sparse boolean formulae for explaining AI decisions. In: NASA Formal Methods Symposium, Springer (2017) 99–114
17. Chickering, D.M., Heckerman, D., Meek, C.: Large-sample learning of Bayesian networks is NP-hard. Journal of Machine Learning Research **5** (2004) 1287–1330
18. Kleinberg, S.: Why: A Guide to Finding and Using Causes. O'Reilly (2015)
19. Birch, M.: The detection of partial association, i: the 2×2 case. Journal of the Royal Statistical Society. Series B (Methodological) (1964) 313–324
20. Kearns, M., Li, M., Valiant, L.: Learning boolean formulas. Journal of the ACM (JACM) **41**(6) (1994) 1298–1328
21. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
22. Rohrer, J.M.: Thinking clearly about correlations and causation: Graphical causal models for observational data. (2017)
23. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (2014)