Estimating the probabilities of causation via deep monotonic twin networks

Athanasios Vlontzos\textsuperscript{1,*}, Bernhard Kainz\textsuperscript{1,2}, Ciarán M. Gilligan-Lee\textsuperscript{3}
\textsuperscript{1} BioMedIA, Imperial College London
\textsuperscript{2} FAU Erlangen-Nuremberg, \textsuperscript{3} Spotify & University College London

Abstract

There has been much recent work using machine learning to answer causal queries. Most focus on interventional queries, such as the conditional average treatment effect. However, as noted by Pearl, interventional queries only form part of a larger hierarchy of causal queries, with counterfactuals sitting at the top. Despite this, our community has not fully succeeded in adapting machine learning tools to answer counterfactual queries. This work addresses this challenge by showing how to implement twin network counterfactual inference—an alternative to abduction, action, & prediction counterfactual inference—with deep learning to estimate counterfactual queries. We show how the graphical nature of twin networks makes them particularly amenable to deep learning, yielding simple neural network architectures that, when trained, are capable of counterfactual inference. Importantly, we show how to enforce known identifiability constraints during training, ensuring the answer to each counterfactual query is uniquely determined. We demonstrate our approach by using it to accurately estimate the probabilities of causation—important counterfactual queries that quantify the degree to which one event was a necessary or sufficient cause of another—on both synthetic and real data.

1 Introduction

The ability to answer causal questions is crucial in science, medicine, economics, and beyond (Gilligan-Lee 2020). This is because causal inference—unlike statistical inference from observational data—allows one to understand the impact of interventions and answer counterfactual queries. Recently there has been much interest in using machine learning tools to estimate causal inference queries (Schwab, Lindhardt, and Karlen 2018; Alaa, Weiss, and Van Der Schaar 2020; Shi, Blei, and Veitch 2019; Pawlowski, Castro, and Glocker 2020; Shalit, Johansson, and Sontag 2016; Perov et al. 2020; Dhir and Lee 2020; Lee et al. 2019). However, most of these focus on interventional queries, such as the conditional average and individual treatment effects. As noted by Pearl (Pearl 2009), interventional queries only form part of a larger causal hierarchy. Counterfactuals sit at the top of this hierarchy, subsuming interventions, and allow one to attribute truly causal explanations to data.

Counterfactual queries establish if certain outcomes would have occurred had some precondition been different. Given evidence $E = e$, counterfactual inference allows one to compute the probability a different outcome $E = e'$ would have occurred—counter-to-the-fact $E = e'$—had some intervention taken place. The crucial difference between counterfactual and interventional queries is that the evidence the counterfactual query is “counter-to” can contain the variables we wish to intervene on or predict. An example counterfactual query is “Given I currently have a headache, would I not, had I taken medicine?”. An interventional query is “What impact would medicine have on my headache?”. The counterfactual query explicitly uses the evidence a headache is present, and asks whether medicine would have changed this fact. The interventional query asks what effect medicine would have on a headache for a given individual, but does not make use of the fact that a headache is currently present. Counterfactual inference has been applied to difficult problems in high profile sectors such as medicine (Richens, Lee, and Johni 2020; Oberst and Sontag 2019), legal analysis (Chockler and Halpern 2004; Lagrado, Gerstenberg, and Zultan 2013), fairness (Kusner et al. 2017; Kilbertus et al. 2017), explainability (Galhotra, Pradhan, and Salimi 2021), planning in reinforcement learning (Forny, Pearl, and Bareinboim 2017; Buesing et al. 2018), and advertising (Ang L. 2019; Bottou et al. 2013).

Despite the importance of counterfactual inference, our community has not fully explored adapting machine learning to compute answers to counterfactual queries. The standard approach to answering counterfactual queries was introduced by Pearl (Pearl 2009), and is known as abduction-action-prediction. First, abduction corresponds to updating the model’s distributions under observed evidence $E$. Second, the desired intervention, or action, is applied. Last, the updated distributions are used to make predictions in the intervened model. A practical limitation is that the abduction step requires large computational resources. Indeed, even if we start with a Markovian model in which background variables are mutually independent, conditioning on evidence, as in abduction, normally destroys this independence and makes it necessary to carry over a full description of the joint distribution over the background variables (Pearl 2009). Balke and Pearl (Balke and Pearl 1994) introduced a method to address this difficulty. Their method...
reduces estimating counterfactuals to performing Bayesian inference on a larger causal model, known as a twin network, where the factual and counterfactual worlds are jointly graphically represented. Twin networks have indeed been shown empirically to be more computationally efficient than abduction-action-prediction in cases where the full causal model was known by (Graham, Lee, and Perov 2019). However, despite their potential importance, twin networks have not been widely investigated—in particular from a machine learning perspective. Combining twin networks with modern machine learning is crucial if we want to perform counterfactual inference using large-scale, real-world data.

This work directly addresses this challenge by, for the first time, combining twin networks with deep neural networks to estimate counterfactual distributions. In Section 3.1, we demonstrate that the graphical nature of twin networks makes them particularly amenable to deep learning, yielding simple neural network architectures that, when trained, are capable of counterfactual inference. Importantly, we show how to enforce known identifiability constraints during the learning process, ensuring each counterfactual distribution is uniquely determined from data. Without the ability to enforce these constraints, one could have two models that make the same observational and interventional predictions but give different counterfactual predictions. As one only has access to observational and interventional data, there would be no way of knowing which model is “correct”. Consequently, certain “non-intuitive” counterfactual distributions could be generated (Oberst and Sontag 2019).

We concretely demonstrate our approach by using it to estimate the probabilities of causation—while enforcing required identifiability constraints—from both synthetic and real data. These important counterfactual queries (Tian and Pearl 2000) provide fine-grained information about whether one event was a necessary or sufficient cause of another. The main contributions of this work are:

1. The first time twin networks have been combined with neural networks to compute counterfactual distributions.
2. Imposing known identifiability constraints during training, ensuring counterfactuals are unique while still making use of the expressive power of neural networks.
3. Experiments on synthetic and real data showing method accurately estimates probabilities of causation

2 Preliminaries

2.1 Structural causal models

We provide a brief overview of the Structural Causal Models (SCM) framework. See Chapter 7 of (Pearl 2009) for a more in-depth discussion. For an up-to-date, self-contained review of counterfactuals, see (Bareinboim et al. 2020).

Definition 1 (Structural Causal Model). A structural causal model (SCM) specifies a set of latent variables \( U = \{u_1, \ldots, u_n\} \) distributed as \( P(U) \), a set of observable variables \( \{v_1, \ldots, v_m\} \), a directed acyclic graph (DAG), called the causal structure of the model, whose nodes are the variables \( U \cup V \), a collection of functions \( F = \{f_1, \ldots, f_n\} \), such that \( v_i = f_i(PA_i, u_i) \), for \( i = 1, \ldots, n \), where \( PA \) denotes the parent observed nodes of an observed variable.

The collection of functions \( F \) and the distribution over latent variables induces a distribution over observable variables: \( P(V = v) := \sum_{\{u_i\} f_i(PA_i, u_i) = v_i} P(u_i) \). An example structure is depicted in Figure 1a.

Definition 2 (Submodel). Let \( M \) be a structural causal model, \( X \) a subset of observed variables with realisation \( x \). A submodel \( M_x \) is the causal model with the same latent and observed variables as \( M \), but with functions replaced with \( f_x = \{f_i | v_i \notin X \} \cup \{f_j(PA_j, u_j) := x_j | v_j \in X\} \).

Definition 3 (do-operator). Let \( M \) be a structural causal model, \( X \) a set of observed variables. The effect of action \( do(X = x) \) on \( M \) is given by the submodel \( M_x \).

The do-operator forces variables to take certain values, regardless of the original causal mechanism. Graphically, \( do(X = x) \) means deleting edges incoming to \( X \) and setting \( X = x \). Probabilities involving \( do(x) \) are normal probabilities in submodel \( M_x \); \( P(Y = y | do(X = x)) = P_{M_x}(y) \).

2.2 Abduction-action-prediction counterfactual inference

Definition 4 (Counterfactual). The counterfactual sentence “\( Y \) would be \( y \) in situation \( U = u \), had \( X \) been \( x \)”, denoted \( Y_x(u) = y \), mean \( Y = y \) in submodel \( M_x \) for \( U = u \).

The latent distribution \( P(U) \) allows one to define probabilities of counterfactual queries, \( P(Y_y = y) = \sum_{u \in Y_x(u) = y} P_{M_x}(u) \). For \( x \neq x' \) one can also define joint counterfactual probabilities, \( P(Y_x = y, Y_{x'} = y') = \sum_{u \in Y_x(u) = y, \& Y_{x'}(u) = y'} P_{M_x}(u) \). Moreover, one can define a counterfactual distribution given seemingly contradictory evidence. Given a set of observed evidence variables \( E \), consider the probability \( P(Y_x = y' | E = e) \). Despite the fact that this query may involve interventions that contradict the evidence, it is well-defined, as the intervention specifies a new submodel. Indeed, \( P(Y_x = y' | E = e) \) is given by \( P_{M_x}(u) \).

Theorem 1 (Theorem 7.1.7 in (Pearl 2009)). Given a structural causal model \( M \) with latent distribution \( P(U) \) and evidence \( e \), the conditional probability \( P(Y_x | e) \) is evaluated as follows: 1) Abduction: Infer the posterior of the latent variables with evidence \( e \) to obtain \( P(U | e) \). 2) Action: Apply \( do(x) \) to obtain submodel \( M_x \). 2) Prediction: Compute the probability of \( Y \) in the submodel \( M_x \) with \( P(U | e) \).
2.3 Twin network counterfactual inference

In (Balke and Pearl 1994), Balke & Pearl introduced twin networks, an alternative approach to computing the probabilities of counterfactual queries that has been shown empirically in (Graham, Lee, and Perov 2019) to offer computational savings relative to abduction-action-prediction. A twin network consists of two interlinked networks, one representing the real world and the other the counterfactual world being queried. Constructing a twin network given a structural causal model and using it to compute a counterfactual query is as follows: First, one duplicates the given causal model, denoting nodes in the duplicated model via superscript “*”. Let $V = \{v_1, \ldots, v_n\}$ be observable nodes in the causal model and $V^* = \{v_1^*, \ldots, v_n^*\}$ the duplication of these. Then, for every node $v_i^*$ in the duplicated, or “counterfactual,” model, its latent parent $u_i^*$ is replaced with the original latent parent $u_i$ in the original, or “factual,” model, such that the original latent variables are now a parent of two nodes, $v_i$ and $v_i^*$. The two graphs are linked only by common latent parents, but share the same node structure and generating mechanisms. To compute a general counterfactual query $P(Y = y \mid E = e, \text{do}(X = x))$, one modifies the structure of the counterfactual network by dropping arrows from parents of $X^*$ and setting them to value $X^* = x$. Then, in the twin network with this modified structure, one computes the following probability $P(Y^* = y \mid E = e, X^* = x)$ via standard inference techniques, where $E$ are factual nodes.

To illustrate this concretely, consider the causal model with causal structure depicted in Figure 1a where variables $X, Y$ are binary. The counterfactual statement to be computed is $P(Y_{X=0} = 0 \mid Y = 1)$. The twin network approach to this problem first constructs the linked factual and counterfactual networks depicted in Figure 15. The intervention $\text{do}(X^* = 0)$ is then performed in the counterfactual network. That is, all arrows from the parents of $X^*$ are removed and $X^*$ is set to the value 0—graphically depicted in Figure 1d. The above counterfactual query is reduced to the following conditional probability in the twin network: $P(Y^* = 0 \mid Y = 1)$, which can be computed using Bayesian inference techniques.

2.4 Distinction between twin networks and abduction-action-prediction

As was discussed in the introduction, abduction-action-prediction counterfactual inference requires large computational resources. Twin networks were specifically designed to address this difficulty (Pearl 2009; Balke and Pearl 1994). Indeed, consider the following passage from Pearl’s “Causality” (Pearl 2009, Section 7.1.4, page 214):

“The advantages of delegating this computation [abduction] to inference in a Bayesian network [i.e., a twin network] are that the distribution need not be explicated, conditional independencies can be exploited, and local computation methods can be employed”

This suggests that the computational resources required for counterfactual inference using twin networks can be less than in abduction-action-prediction. This was put to the test by (Graham, Lee, and Perov 2019) and shown empirically to be correct, with their abstract stating “twin networks are faster and less memory intensive by orders of magnitude than standard [abduction-action-prediction] counterfactual inference”

A key difference is that in a twin network, inference can be conducted in parallel rather than in the serial nature of abduction-action-prediction. For instance, sampling in twin networks is faster than in the abduction-action-prediction, as twin networks propagate samples simultaneously through the factual and counterfactual graphs—rather than needing to update, store and resample as in abduction-action-prediction. Thus, full counterfactual inference in a twin network can take up to no more than the amount of time sampling takes, while in abduction-action-prediction one incurs the additional cost of reusing samples and evaluating function values in the new mutilated graph. This is potentially advantageous for very large graphs, or for graphs with complex latent distributions that are expensive to sample.

Our contribution: Despite their importance for counterfactual inference, twin networks have not been widely studied—particularly from a machine learning perspective. Combining twin networks with machine learning is crucial if we want to perform counterfactual inference with large-scale, real-world data. We seek to directly tackle this challenge by, for the first time, combining twin networks with neural networks to estimate counterfactual distributions.

2.5 (Non-) Identifiability of counterfactuals

One difficulty that must be overcome in order to use machine learning to correctly perform counterfactual inference is ensuring that the counterfactual query can be identified from the data at hand. Indeed, two SCMs that have the same conditional and interventional distributions can disagree about the probabilities of certain counterfactual queries (Pearl 2009). Hence, without additional domain knowledge about the form of the functions underlying a given causal model, there is no matter-of-fact about which counterfactual probability is the “correct” one. This can result in a model learning “non-intuitive” counterfactuals that appear to conflict with observed interventions (Oberst and Sontag 2019, Section 3.1). We provide a concrete example illustrating this phenomenon in Appendix A.5. Thankfully, in certain fields, like epidemiology, identifiability has been shown to follow from reasonable, agreed-upon domain knowledge.

In epidemiology, causal models with the structure of Figure 1c, where $X$ is the presence of a risk factor and $Y$ is the presence of a disease, are studied. Here, domain knowledge can be applied to ensure counterfactuals are unique given data. Indeed, it is generally believed that risk factors always increase the likelihood of a disease being present (Tian and Pearl 2000)—referred to as “no-prevention”, that no individual in the population can be helped by exposure to the risk factor (Pearl 1999). Hence, if one observes a disease, but not the risk factor, then, in that context, if we had intervened to give that individual the risk factor, the likelihood of them not having the disease must be zero—as having the risk factor can only increase the likelihood of a disease. That is, for
that if one can generate counterfactual distributions, one then interpreting it as a twin network on which inference learned by the neural network satisfies identifiability importantly, we will discuss how to ensure the function space be the case that in some domains full identifiability is not be the case that in some domains full identifiability is not required. All that may be needed is to enforce functional constraints that only rule out certain kinds of “non-intuitive” counterfactuals, rather than ensuring every counterfactual query is uniquely defined from observational data alone.

**Takeaway for machine learning and counterfactual inference** In general, for given counterfactuals to be uniquely identified, one requires constraints on the functions constituting the data generating mechanisms. Hence, when employing machine learning techniques to learn an SCM—on which counterfactual inference is to be performed—one must ensure the required identifiability constraints are imposed during the training phase. In Epidemiological models with binary variables, monotonicity is the required constraint. But in other areas, domain knowledge is needed to determine relevant and reasonable constraints. Lastly, it may be the case that in some domains full identifiability is not required. All that may be needed is to enforce functional constraints that only rule out certain kinds of “non-intuitive” counterfactuals, rather than ensuring every counterfactual query is uniquely defined (Oberst and Sontag 2019). This is known as “partial” identifiability. Regardless, in either type of identifiability, functional constraints must be imposed during model training to ensure any resulting counterfactual inference will be trustworthy.

## 3 Methods

### 3.1 Deep twin networks

We now present our method combining twin networks with neural networks to estimate counterfactual distributions. Importantly, we will discuss how to ensure the function space learned by the neural network satisfies identifiability constraints. Our approach has two stages, training the neural network such that it satisfies identifiability constraints, then interpreting it as a twin network on which inference is performed to estimate counterfactual distributions. Note that if one can generate counterfactual distributions, one can also generate observational and interventional distributions

For clarity, we confine our explanations to the causal structure from Figure 1a where X, Y are categorical variables with X ∈ {1, ..., N} and Y ∈ {1, ..., M}, and Z can be categorical or numerical. Note there can be many Z. Our method can be extended to multiple causes and a single output straightforwardly. To generalise this approach to an arbitrary causal structure, one applies our method to each parent-child structure recursively in the topological specified by the direction of the arrows in the causal structure.

**Training deep twin networks**: To determine the architecture of our neural network, we start with the causal structure of the SCM we wish to learn, and consider the graphical structure of its twin network representation. Our neural network architecture then exactly follows this graphical structure. This is graphically illustrated for the case of binary X, Y from Figure 1a with twin network in Figure 1c. In the case of binary X, Y, the neural network has two heads, one for the outcome under the factual treatment and the other for the outcome under the counterfactual treatment. Furthermore two shared—but independent of one another—base representations, one corresponding to a representation of the observed confounders, Z, and the other to the latent noise term on the outcome, UY, are employed. For multiple treatments we have N neural network heads, each corresponding to the categories of X. To interpret this as a twin network for given evidence X, Y, Z and desired intervention X∗, we marginalise out the heads indexed by the elements of {1, ..., N}/X, X∗. To train this neural network, we require two things: 1) a label for head Y∗, and 2) a way to learn the distribution of the latent noise term UY.

For 1), we must ask what the expected value of Y∗ is, for fixed covariates Z, under a change in input X∗. This corresponds to E(Y∗|X∗, Z). Given the correspondence between twin networks and the original SCM outlined in Section 2.3—this corresponds to E(Y|do(X), Z), which is the expected value of Y under an intervention on X for fixed Z. There are many approaches to estimating this quantity in the literature (Shalit, Johansson, and Sontag 2016; Alaa, Weissz, and Van Der Schaar 2017; Johansson, Shalit, and Sontag 2016; Shi, Bie, and Veitch 2019). We follow (Schwab, Linhardt, and Karlen 2018)—who utilise propensity score matching—due to their methods simplicity and empirical high performance. In addition to specifying the causal structure, the following standard assumptions are needed to correctly estimate E(Y|do(X), Z) (Schwab, Linhardt, and Karlen 2018):

1) **Overlap**: every unit has non-zero probability of receiving all treatments given their observed covariates.

Propensity score matching proceeds as follows. Given observed data of the form {X, Y, Z}, one computes the propensity score P(X | Z) for each datapoint and then marginalises over it by integrating the data distribution P(X, Y, Z) into the expected value, which is used as the counterfactual outcome under X′ for the original datapoint. This preprocessing is performed on the minibatch level to improve variance (Schwab, Linhardt, and Karlen 2018). This provides a label for Y∗.
For 2), consider the following. Formally, the causal structure Figure 1a has $Y = f(X, Z, U_Y)$ with $U_Y \sim q(U_Y)$ for some q. Without loss of generality (Goudet et al. 2018), one can rewrite this as $Y = f(X, Z, g(U_Y'))$ with $U_Y' \sim \mathcal{E}$ and $U_Y = g(U_Y')$, where $\mathcal{E}$ is some easy-to-sample-from distribution, such as a Gaussian or Uniform. Hence we have reduced learning $q(U_Y)$ to learning function $g$, whose input corresponds to samples from a Gaussian or Uniform.

Taken together, this provides a method to train our deep twin network. A summary is provided in Algorithm 1

**Enforcing identifiability constraints:** There are a few approaches to ensure that the function space learned by a neural network satisfies specified identifiability constraints. Recall from Section 2.5 that such constraints correspond to limits on the type of input-output pairs consistent with the function. That is, for a given set of inputs there are outputs that the learned function should not generate. One approach is to specify a loss function penalising the network for outputs that violate the constraints, as done in (Sill and Abu-Mostafa 1997). Alternatively, given the input-output pairs that are ruled-out, counterexample-guided learning (Sivaraman et al. 2020) can be employed, to ensure the trained network does not produce any of these outputs when given the corresponding input. Lastly, for monotonicity, a recent method uses “look-up tables” (Gupta et al. 2016) to enforce monotonicity, and has been implemented in TensorFlow Lattice.

Algorithm 1 summarises this for the case when identifiability constraints are specified by a loss function.

**Algorithm 1:** Training a deep twin network

**Input:** X: Treatment, Z: Confounders; X*: Counterfactual Treatment; Y: Outcome; C: DAG of causal structure; I: loss imposing identifiability constraint

**Output:** F: trained deep twin network

1. Set F's architecture to match twin network representation of C, as in Figure 2
2. To obtain label for counterfactual head, first estimate $P(Z | X)$
3. Then, for $x, z$ find $z' = \text{with closest } P(z' | x) \text{ to } P(z | x)$
4. Set $y' \leftarrow y(z', x')$, yielding training dataset $D := \{(X, X', Z, Y, Y')\}$
5. for $x, x', z; y, y' \in D$ and $u_y \sim N(0, 1)$ do
6. $y', y^* = F(x, x', u_y, z)$
7. Train F by minimizing $\text{MSE}(y, y') + \text{MSE}(y^*, y^*) + I(D)$
8. end for
9. **Estimating counterfactual distributions:** With the trained model in hand, we can now use it to perform counterfactual inference in the second step. The advantage of our neural network architecture matching the Twin Network structure is that performing Bayesian inference on the neural network explicitly equates to performing counterfactual inference. In the case of figure 1a, there are two counterfactual queries one can ask: (1) $P(Y_{X=x'} = y' | X = x, Y = y, Z = z)$, (2) $P(Y_{X=x} = y, Y_{X=x'} = y' | Z = z)$, where any of $x, y, z$ can be the empty set. Recall from Section 2.5 that in a twin network (1) corresponds to $P(Y^* = y' | X = x, Y = y, X^* = x')$, and (2) corresponds to $P(Y = y, Y^* = y' | X = x, Y^* = x')$. Any method of Bayesian inference can then be employed to compute these probabilities, such as Importance Sampling, Rejection Sampling, or variational methods. The case of (1) is presented in Algorithm 2

**Algorithm 2:** Counterfactual Inference

**Input:** X: Treatment, U_Y: Noise; Z: Confounders; X*: Counterfactual Treatment; Y: Outcome; Y*: Counterfactual Outcome; F: Trained deep twin network; Q: desired counterfactual query (in this example, $Y_{X=x'} = y' | X = x, Y = y, Z = z$)

**Output:** P(Q): Estimated distribution of Q

1. Convert P(Q) to twin network distribution: $P(Y_{X=x'} = y' | X = x, Y = y, Z = z) \rightarrow P(Y^* = y' | X = x, Y = y, X^* = x')$
2. Compute $P(Y^* = y' | X = x, Y = y, X^* = x')$
3. for $x, x', z \in D_{test}$ do
4. for $u_y \sim N(0, 1), N \in \mathbb{N}$ do
5. Sample $(\tilde{y}, \tilde{y}^*) = F(x, x', u_y, z)$ such that $\tilde{y} = y$, for example using Rejection sampling, Importance sampling, etc.
6. Frequency of these samples for which $\tilde{y}^* = y'$ yields P(Q)
7. end for
8. end for

### 3.2 Using deep monotonic twin networks to estimate the probabilities of causation

We wish to demonstrate the method from section 2.2 in a concrete setting with known identifiability constraint. As we saw in Section 2.5, one of the only causal models where full identifiability has been proven is Figure 1a with binary X, Y, and Y is monotonic in X. In Section 2.5 we saw monotonicity and ignorability imply all counterfactuals in this case are uniquely determined from observational data. Here, important counterfactuals are the **probabilities of causation**. These yield fine-grained information about whether one event caused of another. Recently these have been applied in medical diagnosis (Richens, Lee, and Johr 2020) to determine if a patient’s symptoms would not have occurred had it not been for a specific disease. Here, the proposition binary variable W is true will is denoted W = 1, while its negation, W = 0, denotes the proposition W is false.

1. **Probability of necessity:**

$P(Y_{X=0} = 0 | X = 1, Y = 1, Z) = \text{The probability of necessity is the probability event Y would not have occurred without event X occurring, given that X, Y did in fact occur in context Z.}$

2. **Probability of sufficiency:**

$P(Y_{X=1} = 1 | X = 0, Y = 0, Z) = \text{The probability of sufficiency is the probability that in a situation where X, Y were absent, intervening to make X occur would have led to Y occurring in context Z.}$

3. **Probability of necessity & sufficiency:**

$P(Y_{X=0} = 0, Y_{X=1} = 1 | Z) = \text{The probability of necessity & sufficiency quantifies the sufficiency and necessity of event X to produce event Y in context Z. As discussed in section 2.2, joint counterfactual probabilities are well-defined.}$

Algorithm 1 in Appendix describes how to compute these.

### 4 Related work

Machine learning to estimate interventional queries: Recently there has been much interest in using machine
learning to estimate interventional conditional distributions. These were aimed at learning conditional average treatment effects: $E(Y_{X=1} \mid Z) - E(Y_{X=0} \mid Z)$. Examples include PerfectMatch (Schwab, Linhardt, and Karrer 2018), DragonNet (Shi, Blei, and Veitch 2019), PropensityDropout (Alaa, Weisz, and Van Der Schaar 2017), Treatment-Agnostic Representation Networks (TARNET) (Johansson, Sontag 2016), Balancing Neural Networks (Johansson, Shalit, and Sontag 2016). Each work utilised neural network architectures similar to the one depicted in Figure 2 with slight modifications based on the specific approach. Other machine learning approaches to estimating interventional queries made use of GANs, such as GANITE (Yoon, Jordon, and Van Der Schaar 2016) and CausalGAN (Kocaoglu et al. 2018), Gaussian Processes (Witty et al. 2020), Alaa and van der Schaar (2017), Variational Autoencoders (Louizos et al. 2017), and representation learning (Zhang, Bellot, and Schaar 2020) [Assaad et al. 2021, Yao et al. 2018].

While our architecture shares similarities with these, there is one main difference. By interpreting our architecture as a twin network in the sense of (Balke and Pearl 1994) as discussed in Section 5.1—and explicitly including an input for the latent noise term $U_Y$, we can elevate our network from relative fixed to one distribution—showing robust estimation. In Figures 1,2 of the Appendix we also show examples in unconfounded and confounded causal models when ground truth and candidate distributions are the same. In Figures 1,2 of the Appendix we also show examples in unconfounded and confounded cases when ground truth and candidate distributions are the same. In Figures 1,2 of the Appendix we also show examples in unconfounded and confounded cases when ground truth and candidate distributions are the same.

### Table 1: Results of Synthetic experiments. P(N): Probability of Necessity; P(S): Probability of Sufficiency; P(N&S): Probability of Necessity and Sufficiency. Our model achieves highly accurate estimations of the probabilities of causation on synthetic data.

| Method                  | $U_y$     | P(N)       | P(S)       | P(N&S)      |
|-------------------------|-----------|------------|------------|-------------|
| Synth Ground Truth      | Uniform   | 0.5        | 0.5        | 0.3333      |
| Synth Twin Net          | Uniform   | 0.50214 ± 0.00387 | 0.50046 ± 0.00631 | 0.33449 ± 0.00401 |
| Synth w/ Conf Ground Truth | Gaussian   | 0.54706    | 0.35512    | 0.27443     |
| Synth w/ Conf Twin Net  | Gaussian   | 0.54563 ± 0.00276 | 0.35177 ± 0.00144 | 0.27207 ± 0.00125 |

5 Experiments

To validate deep twin networks we estimate the probabilities of causation on 1) synthetically generated data where the ground truth is known, and 2) two real datasets. The identifiability constraint enforced was monotonicity. For inference we used rejection sampling. Our architecture and training are as in Section 5.1 implementations are in the Appendix. Code is shared as part of the supplement.

5.1 Synthetic data experiments

We test on both unconfounded and confounded causal models, with causal structure Figure 1a and Figure 1c respectively. Due to space constraints we mathematically analyze the generating functions in the Appendix. The functions remain monotonic in $X$. Given these, we construct synthetic datasets of 200,000 points split into training and testing under an 80 − 20 split. The samples $U_y$ were drawn from either a uniform or a Gaussian distribution, depending on the experiment. Confounders $Z$ were taken from a uniform distribution. We opt for a high number of samples such that we do not bias our analysis due to small sample sizes. In the real world experiments the dataset sizes are smaller. Results for a trained twin network are in Table 1. We accurately estimate all Probabilities of Causation in both unconfounded and confounded cases when ground truth and candidate distributions are the same. In Figures 1,2 of the Appendix we also show performance of (a) unconfounded and (b) confounded cases as ground truth distribution of $U_Y$ in synthetic generating functions changes, but candidate training distributions remain fixed to one distribution—showing robust estimation.
5.2 Real world data experiments

**Kenyan Water Real World Dataset:** To test deep twin networks in real scenarios we explore the Kenyan Water task from [Cuellar and Kennedy 2020](#). The task is to understand whether protecting water springs in Kenya by installing pipes and concrete containers reduces childhood diarrhoea, given confounders. First, monotonicity is a reasonable assumption here as protecting a spring is not expected to increase the bacterial concentration and hence increase the incidence of diarrhoea. [Cuellar and Kennedy 2020](#) reported a low value for Probability of Necessity here—suggesting that children who developed diarrhoea after being exposed to a high concentration of bacteria in their drinking water would have contracted the disease regardless. However, as there is no ground truth here, further studies reproducing this result with alternate methods are required to gain confidence in [Cuellar and Kennedy 2020](#)’s result. We follow the same data processing as in [Cuellar and Kennedy 2020](#), detailed in the Appendix. Our findings in Table 2 agree with [Cuellar and Kennedy 2020](#) on Probability of Necessity. Moreover, unlike [Cuellar and Kennedy 2020](#), we can also compute Probability of Sufficiency and Probability of Necessity and Sufficiency. We can thus offer a more comprehensive understanding of the role protecting water springs plays in childhood disease. Our results show that exposure to water-based bacteria is not a necessary condition to exhibit diarrhoea and it is neither a sufficient, nor a necessary-and-sufficient condition. This provides further evidence that protecting water springs has little effect on the development of diarrhoea in children in these populations, indicating the source of the disease is not related to water. Note that in both [Cuellar and Kennedy 2020](#) and here the median child is outside the test set. This explains the difference in Probability of Necessity on the test set when compared to the median child.

**Twin Mortality Real World Dataset:** In our second real-world experiment we use the Twin mortality dataset [Louizos et al. 2017](#). Given confounders regarding the health of the mother and background of the parents, the goal is to understand the effect being born the heavier of the twins has on mortality one year after birth. Previous work addressed this with intervention queries. We use counterfactual queries—specifically the probabilities of causation. We follow [Louizos et al. 2017](#), [Yoon, Jordon, and Van Der Schaar 2018](#)’s preprocessing. As in [Louizos et al. 2017](#) we treat each twin as the counterfactual of their sibling—providing a ground truth reported in Table 2. Again, monotonicity is justified here as we do not expect increasing birth weight to lead to reduced mortality.

First, given birth weight and mortality evidence provided by one twin, we aim to estimate the expected counterfactual outcome had their weight been different. That is, compute $E(Mortality | Weight, Z)$, where $Z$ are observed confounders. We achieve a counterfactual AUC-ROC of 86% and F1 score of 83%. [Louizos et al. 2017](#) addressed this same question using only used interventional queries. That is, they computed $E(Mortality | Weight, Z)$. Table 2 shows our AUC outperformed theirs. Here, by explicitly conditioning on and using the fact that the observed twins had birth weight and mortality, we are able to update our knowledge about the latent noise term of the other twin. Our improved AUC score showed using this allowed more accurate estimation of the “hidden” twins outcome. This cleanly illustrate the difference between interventions and counterfactuals. To give a comparison to prior work, we computed the average treatment effect from our model, yielding $−2.34\% ± 0.019$ which matches [Louizos et al. 2017](#)—showing our model estimates interventions and counterfactuals.

Table 2 also reports our estimation of the Probabilities of Causation. Note that no previous work has computed these counterfactual distributions. Despite accurate estimation of the Probability of Sufficiency, and Necessity & Sufficiency, our model underestimates the Probability of Necessity. This can be explained by a large data imbalance regarding the mortality outcome—affecting the Probability of Necessity the most as mortality is the evidence conditioned here. Nevertheless, we correctly reproduce the relative sizes of the Probabilities of Causation, with Probability of Necessity an order of magnitude larger than the others.

### Table 2: Results of Kenyan Water (KW) & Twins Mortality (TM) with Twin Network (TN), P(N): Prob. of Necessity; P(S): Prob. of Sufficiency; P(N&S): Prob. of Necessity & Sufficiency; MC: median child. In KW we agree with & improve [Cuellar and Kennedy 2020](#). In TM we agree with & improve [Louizos et al. 2017](#).

| Method                  | P(N)   | P(S)   | P(N&S)  | AUC-ROC / F1 |
|-------------------------|--------|--------|---------|--------------|
| KW Median Child         | 0.12±0.01 | -      | -       | -            |
| KW TN Median Child      | 0.13598±0.049 | 0.09811±0.031 | 0.31778±0.012 | -            |
| KW TN Test Set          | 0.06273±0.020 | 0.03914±0.016 | 0.08521±0.034 | -            |
| Twin Mortality Ground Truth | 0.33372 | 0.01011 | 0.01353 | 0.83/- Louizos et al. 2017 |
| TM TN Test Set          | 0.12241±0.019 | 0.01401±0.003 | 0.01174±0.002 | 0.86/0.83    |

6 Conclusion

Previous work applying machine learning to causality was either limited to interventional queries, or used abduction-action-prediction methodology—for which identifiability was not enforced. We proposed a novel method combining twin networks, an alternate counterfactual methodology, with machine learning, showed how to enforce identifiability, and accurately estimated counterfactuals from real data.
References

Alaa, A. M.; and van der Schaar, M. 2017. Bayesian inference of individualized treatment effects using multi-task Gaussian processes. In Proceedings of the 31st International Conference on Neural Information Processing Systems, 3427–3435.

Alaa, A. M.; Weisz, M.; and Van Der Schaar, M. 2017. Deep counterfactual networks with propensity-dropout. arXiv preprint arXiv:1706.05966.

Ang Li, J. P. 2019. Unit Selection Based on Counterfactual Logic.

Assaad, S.; Zeng, S.; Tao, C.; Datta, S.; Mehta, N.; Henao, R.; Li, F.; and Duke, L. C. 2021. Counterfactual representation learning with balancing weights. In International Conference on Artificial Intelligence and Statistics, 1972–1980.

Balke, A.; and Pearl, J. 1994. Probabilistic evaluation of counterfactual queries. In AAAI.

Bareinboim, E.; Correa, J. D.; Ibeling, D.; and Icard, T. 2020. On Pearl’s Hierarchy and the Foundations of Causal Inference. Technical report, Columbia University, Stanford University.

Bottou, L.; Peters, J.; Quiñonero-Candela, J.; Charles, D. X.; Chickering, D. M.; Portugalí, E.; Ray, D.; Simard, P.; and Snelson, E. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. The Journal of Machine Learning Research, 14(1): 3207–3260.

Buesing, L.; Weber, T.; Zwols, Y.; Racaniere, S.; Guez, A.; Lespiau, J.-B.; and Heess, N. 2018. Woulda, coulda, shoulda: Counterfactually-guided policy search. arXiv preprint arXiv:1811.06272.

Chockler, H.; and Halpern, J. Y. 2004. Responsibility and blame: A structural-model approach. Journal of Artificial Intelligence Research, 22: 93–115.

Cuellar, M.; and Kennedy, E. H. 2020. A non-parametric projection-based estimator for the probability of causation, with application to water sanitation in Kenya. Journal of the Royal Statistical Society: Series A (Statistics in Society), 183(4): 1793–1818.

Dhir, A.; and Lee, C. M. 2020. Integrating overlapping datasets using bivariate causal discovery. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 3781–3790.

Forney, A.; Pearl, J.; and Bareinboim, E. 2017. Counterfactual data-fusion for online reinforcement learners. In International Conference on Machine Learning, 1156–1164. PMLR.

Galhotra, S.; Pradhan, R.; and Salimi, B. 2021. Explaining Black-Box Algorithms Using Probabilistic Contrastive Counterfactuals. arXiv preprint arXiv:2103.11972.

Gilligan-Lee, C. 2020. Cauing trouble. New Scientist, 246(3279): 32–35.

Goudet, O.; Kalainathan, D.; Caillon, P.; Guyon, I.; Lopez-Paz, D.; and Sebag, M. 2018. Learning functional causal models with generative neural networks. In Explainable and interpretable models in computer vision and machine learning, 39–80. Springer.

Graham, L.; Lee, C. M.; and Perov, Y. 2019. Copy, paste, infer: A robust analysis of twin networks for counterfactual inference. NeurIPS Causal ML workshop 2019, https://cpb-us-w2.wpmucdn.com/sites.coecis.cornell.edu/dist/a/238/files/2019/12/Id_65_final.pdf.

Gupta, M.; Cotter, A.; Pfeifer, J.; Voevodski, K.; Canini, K.; Mangylov, A.; Moczydlowski, W.; and Van Esbroeck, A. 2016. Monotonic calibrated interpolated look-up tables. The Journal of Machine Learning Research, 17(1): 3790–3836. https://www.tensorflow.org/lattice/overview. ????. Tensor-Flow Lattice.

Johansson, F.; Shalit, U.; and Sonntag, D. 2016. Learning representations for counterfactual inference. In International Conference on Machine Learning, 3020–3029.

Kilbertus, N.; Carulla, M. R.; Parascandolo, G.; Hardt, M.; Janzing, D.; and Schölkopf, B. 2017. Avoiding discrimination through causal reasoning. In Advances in Neural Information Processing Systems, 656–666.

Kocaoglu, M.; Snyder, C.; Dimakis, A. G.; and Vishwanath, S. 2018. CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training. In International Conference on Learning Representations.

Kremer, M.; Leino, J.; Miguel, E.; and Peterson, A. 2015. Replication data for: Spring Cleaning: Rural Water Impacts, Valuation, and Property Rights Institutions.

Kusner, M.; Loftus, J.; Russell, C.; and Silva, R. 2017. Counterfactual Fairness. Advances in Neural Information Processing Systems 30 (NIPS 2017) pre-proceedings, 30.

Lagnado, D. A.; Gerstenberg, T.; and Zultan, R. 2013. Causal responsibility and counterfactuals. Cognitive science, 37(6): 1036–1073.

Lee, C. M.; Hart, C.; Richens, J. G.; and Johri, S. 2019. Leveraging directed causal discovery to detect latent common causes. arXiv preprint arXiv:1910.10174.

Louizos, C.; Shalit, U.; Mooij, J.; Sontag, D.; Zemel, R.; and Welling, M. 2017. Causal effect inference with deep latent-variable models. In Proceedings of the 31st International Conference on Neural Information Processing Systems, 6449–6459.

Oberst, M.; and Sontag, D. 2019. Counterfactual off-policy evaluation with gumbel-max structural causal models. In International Conference on Machine Learning, 4881–4890. PMLR.

Pawlowski, N.; Castro, D. C.; and Glocker, B. 2020. Deep structural causal models for tractable counterfactual inference. arXiv preprint arXiv:2006.06485.

Pearl, J. 1999. Probabilities of causation: three counterfactual interpretations and their identification. Synthese, 121(1-2): 93–149.

Pearl, J. 2009. Causality (2nd edition). Cambridge University Press.

Perov, Y.; Graham, L.; Gourgoulias, K.; Richens, J.; Lee, C.; Baker, A.; and Johri, S. 2020. Multiverse: causal reasoning
using importance sampling in probabilistic programming. In *Symposium on advances in approximate bayesian inference*, 1–36. PMLR.

Richens, J. G.; Lee, C. M.; and Johri, S. 2020. Improving the accuracy of medical diagnosis with causal machine learning. *Nature communications*, 11(1): 1–9.

Schwab, P.; Linhardt, L.; and Karlen, W. 2018. Perfect match: A simple method for learning representations for counterfactual inference with neural networks. arXiv preprint arXiv:1810.00656.

Shalit, U.; Johansson, F. D.; and Sontag, D. 2016. Estimating individual treatment effect: generalization bounds and algorithms. arXiv preprint arXiv:1606.03976.

Shi, C.; Blei, D. M.; and Veitch, V. 2019. Adapting neural networks for the estimation of treatment effects. arXiv preprint arXiv:1906.02120.

Sill, J.; and Abu-Mostafa, Y. S. 1997. Monotonicity hints. *ICML*, 28(1): 287–313.

Sivaraman, A.; Farnadi, G.; Millstein, T.; and Broeck, G. V. d. 2020. Counterexample-Guided Learning of Monotonic Neural Networks. arXiv preprint arXiv:2006.08852.

Tian, J.; and Pearl, J. 2000. Probabilities of causation: Bounds and identification. *Annals of Mathematics and Artificial Intelligence*, 28(1): 287–313.

Witty, S.; Takatsu, K.; Jensen, D.; and Mansinghka, V. 2020. Causal inference using Gaussian processes with structured latent confounders. In *International Conference on Machine Learning*, 10313–10323. PMLR.

Yao, L.; Li, S.; Li, Y.; Huai, M.; Gao, J.; and Zhang, A. 2018. Representation learning for treatment effect estimation across overlapping representations for the estimation of individualized treatment effects. In *International Conference on Machine Learning*, 1005–1014. PMLR.

**APPENDIX**

**A Identifiability Example**

To concretely illustrate the epidemiological monotonicity constraint used for identifiability, consider the DAG of Figure 1(e) from the main text, with $X$, $Y$ binary, $U_Y$ a four-valued variable distributed as $q(U_Y)$, and the function relating them given by:

$$
Y = \begin{cases} 
X, & \text{if } U_Y = 0 \\
0, & \text{if } U_Y = 1 \\
1, & \text{if } U_Y = 2 \\
\neg X, & \text{if } U_Y = 3 
\end{cases}
$$

(1)

where $\neg X$ is the negation of $X$. Given $q(U_Y)$ and the conditional probabilities $P(Y \mid X)$ are: $P(Y = 0 \mid X = 0) = q(U_Y = 0) + q(U_Y = 1)$ and $P(Y = 0 \mid X = 1) = q(U_Y = 1) + q(U_Y = 3)$. As there are no confounders, these coincide with the interventional distributions $P(Y \mid do(X))$. Consider the following counterfactual query $P(Y_{X=1} = 1 \mid Y = 0, X = 0)$. It can be written as:

$$
P(Y_{X=1} = 0 \mid Y = 1, X = 0) = \frac{q(U_Y = 3)}{q(U_Y = 2) + q(U_Y = 3)},
$$

which follows by noting $Y = 1$ and $X = 0$ implies either $Y = \neg X$ or $Y = 1$, which happens with probability $q(U_Y = 2) + q(U_Y = 3)$, and in this context the only way $Y = 0$ when $X = 1$ is if $Y = \neg X$, which occurs with probability $q(U_Y = 3)$. Note that there is no way to write this counterfactual probability in terms of the conditional distributions $P(Y \mid X)$ alone. Moreover, one can have two models with the same functional form as Eq. 1 that yield the same conditional distributions, but give different predictions for the counterfactual query. An example is one model with distribution over $U_Y$ given by $\{q(U_Y)\}_0^3 = \{1/2, 1/6, 1/6, 1/6\}$, the other model with $\{q(U_Y)\}_0^3 = \{1/3, 1/3, 1/3, 0\}$. This shows there are counterfactuals that cannot be uniquely identified from data. Hence, one needs additional constraints on the model to ensure this.

**B Implementation Details**

**B.1 Data**

The synthetic data used in the experimental section was the creation of the authors, code to produce these datasets will be provided together with the rest of the codebase.

For the Twin Mortality data, two versions were used. First databases provided by (Louizos et al. 2017) were processed to remove NaN entries. No further processing was administered apart from propensity score matching. This constituted the completely real version of the Twin Mortality dataset. However, as both (Louizos et al. 2017) and (Yoon, Jordan, and Van Der Schaar 2018) process the their data to create a semi-synthetic task, in the spirit of proper comparison we used the data as processed and provided by (Yoon, Jordan, and Van Der Schaar 2018), with no additional processing.

**B.2 Unconfounded Synthetic example**

$$
Y = \begin{cases} 
X, & \text{if } U_Y = 0 \\
0, & \text{if } U_Y = 1 \\
1, & \text{if } U_Y = 2 
\end{cases}
$$

(3)

Hence

$$
P(N) = \frac{P(U_Y = 0)}{P(U_Y = 0) + P(U_Y = 2) + P(U_Y = 0)}
$$

(4)

$$
P(S) = \frac{P(U_Y = 0)}{P(U_Y = 1) + P(U_Y = 0)}
$$

(5)

$$
P(NS) = P(U_Y = 0)
$$

(6)
Figure 3: Predicted and ground truth Probability of Necessity as distribution of $U_Y$ varies in synthetic generating functions, but training distributions do not. Plots show robust estimation. (a) unconfounded case, (b) confounded case. Errors bars in both

B.3 Confounded Synthetic example

$$Y = \begin{cases} X \times Z, & \text{if } U_Y = 0 \\ 0, & \text{if } U_Y = 1 \\ 1, & \text{if } U_Y = 2 \end{cases}$$

where

$$X := U_x \oplus Z$$

Hence

$$P(N) = \frac{P(U_Y = 0)P(Z = 1)}{P(U_Y = 2) + P(U_Y = 0)P(Z = 1)}$$

$$P(S) = \frac{P(U_Y = 0)P(Z = 1)}{P(U_Y = 1) + P(U_Y = 0)}$$

$$P(NS) = P(U_Y = 0)P(Z = 1)$$

B.5 Algorithm

Below we show a pseudo-algorithm on how to use a trained deep twin network to compute the probabilities of necessity. This follows Algorithm 2 from the main text. Other probabilities of causation follow similarly. Note that sampling can happen by any Bayesian procedure such as rejection sampling or importance sampling for instance. We choose to showcase rejection sampling.

Algorithm 3: Probability of Necessity

**Inputs:** $X$: Treatment, $U_Y$: Noise, $Z$: Confounders, $X^*$: Counterfactual Treatment; $Y$: Outcome, $Y^*$: Counterfactual Outcome

**Output:** Probability of Necessity

1: idx-given-y-1 = where ($y' == y$)
2: idx-query-y-0 = where ($y'^* == y^*$)
3: idx-y-l-y-prime-0 = idx-given-y-1 \cap idx-query-y-0
4: Prob-of-Necessity = idx-y-l-y-prime-0 / idx-given-y-1

B.4 Architecture

For the implementation of twin networks we used the Tensorflow 2 framework, with the addition of TFLattice (https://www.tensorflow.org/lattice/overview), as it incorporates the ability to enforce monotonic relations in the network. Inputs to the network were passed through a PWL-Calibration layer to restrict their range, which incorporates a single fully connected layer, different number of units were used in the calibration layers, but most commonly 3 units were chosen. Monotonicity constraints were used as appropriate. The $Z - X$ connection layer was implemented with a single monotonic fully connected layer. The output stage used a lattice layer with 1 end unit. Throughout the architecture monotonicity was imposed as appropriate.

We trained all models for 500 – 2000 epochs, using an Adam optimizer, a MSE loss and a $1e-3$ learning rate. The hyperparameters for the associated experiments are detailed in the README file of the supplementary code.

B.6 Distribution shift plots

In Figures [3a] and [3b] we show how the probability of necessity varies as we vary $U_Y$.

B.7 Computational Resources

All experiments were conducted using a single Titan RTX NVIDIA GPU with 24GB of memory. Through all the experimental settings, training required up to 15 seconds per epoch while inference was performed in less than 2 seconds.

C Ethical Considerations

Causal inference is a tool that can have significant impact on society depending on its use. As such the authors are adamant that all uses of causal inference that could have negative societal impacts should be accompanied with the proper due diligence and fail-safes in order to minimize and even eliminate said negative impacts.