Causal Discovery and Knowledge Injection for Contestable Neural Networks

Fabrizio Russo [a;*] **and Francesca Toni**[a]

[a]Imperial College London, UK

Abstract. Neural networks have proven to be effective at solving machine learning tasks but it is unclear whether they learn any relevant causal relationships, while their black-box nature makes it difficult for modellers to understand and debug them. We propose a novel method overcoming these issues by allowing a two-way interaction whereby neural-network-empowered machines can expose the underpinning learnt causal graphs and humans can *contest* the machines by modifying the causal graphs before re-injecting them into the machines. The learnt models are guaranteed to conform to the graphs and adhere to expert knowledge, some of which can also be given up-front. By building a window into the model behaviour and enabling knowledge injection, our method allows practitioners to debug networks based on the causal structure discovered from the data and underpinning the predictions. Experiments with real and synthetic tabular data show that our method improves predictive performance up to 2.4x while producing parsimonious networks, up to 7x smaller in the input layer, compared to SOTA regularised networks.

1 Introduction

Neural Networks (NNs) have proven to be a very effective Machine Learning (ML) model for solving a wide range of problems [9]. However, it is unclear whether NNs are able to encode the Data Generating Process (DGP) and the causal structure that governs it, as shown by their weakness to data perturbation and adversarial attacks [27, 10]. This poses a problem when deploying these models for high-stakes decisions, like granting credit or parole to individuals [26]. Knowledge injection advocates the integration of human knowledge, distilled and represented in various forms, with data-driven models, including NNs (see [30] for an overview). In particular, it supports complementing the data collected with external knowledge usually difficult to capture in the data alone. It has been shown to lead, amongst others, to less data need, improved generalisation [3] and interpretability [25]. In this paper, we propose a novel methodology for rendering NNs *contestable* by means of knowledge injection coupled with causal discovery.

In our methodology, knowledge injection relates to the causal structure extracted using, as a starting point, the CASTLE (CAusal STructure LEarning) [16] model, which has been shown to produce NNs better able to generalize to unseen data. For high-stakes decisions this is not enough: humans need to be in control, validate the models' recommendations and be able to challenge them. Thus, our knowledge injection methodology allows humans to inspect and modify learnt causal graphs iteratively while a disagreement remains between humans and the progressively refined model. This process can be seen as a form of contestation by the human, challenging models' outputs while still learning from data.

The need for algorithmic systems to be contestable has been advocated in AI ethics frameworks such as the ones from OECD[1] and ACM[2] as well as in regulations like GDPR.[3] Contestable AI has been brought to the attention of AI practitioners (see §2), however algorithmic methods for contestability are lacking. We contribute to this landscape by enabling the interaction of Subject Matter Experts (SMEs) with NNs, aided by causal graphs which can in turn be injected into the models. The causal graphs discovered are shown to SMEs, who can cut edges deemed anti-causal or explore the most influential effects using a threshold parameter.

Specifically, we make the following contributions:

- We propose a first algorithm to *inject*, into feed-forward NNs, expert knowledge in the form of causal graphs. Following the injection, the NNs are guaranteed to use only the direct relationships specified in the graphs, hence adhering to the knowledge captured therein. This is a key component towards contestability, provided by our second algorithm.
- We propose a second algorithm for *human-AI collaboration* on the causal discovery task with NNs. This algorithm can take the human feedback on the computed causal graph underpinning the model predictions at any point of the learning process, as an iterative refinement step for model *debugging* and understanding.
- We apply our algorithms to real and synthetic tabular data within regression and classification tasks, showing that injecting knowledge in the form of a graph can improve predictive performance up to 2.4x while making the models significantly more parsimonious (up to 85% reduction in number of weights of the input layer). Thus, contesting a NN through its computed causal graph can increase model understanding without hurting performance.

2 Related Work

Von Rueden et al. [30] propose a taxonomy for *informed ML*, categorizing the literature on knowledge injection along three main dimensions: knowledge source, representation and integration. These capture the *what* and *how* of knowledge injection. Another aspect analysed is the *why*. Our work broadly fits in the taxonomy, under *Expert Knowledge* represented by *Human Feedback* and integrated through the *Learning Algorithm*. We contribute two novel elements to the *why* and the *how* of knowledge injection: motivated by the

* Corresponding Author. Email: fabrizio@imperial.ac.uk.

[1] Principle 1.3: https://oecd.ai/en/dashboards/ai-principles/P7

[2] Principle 7: https://www.acm.org/binaries/content/assets/public-policy/final-joint-ai-statement-update.pdf

[3] Article 22(3): https://gdpr-text.com/read/article-22/

need to support *contestability*, we go beyond common incentives for expert knowledge integration into ML, such as the use of less data for training, prediction performance boosting and improved interpretability [30] and develop a novel methodology for *knowledge injection empowered by causal discovery*.

In our approach, knowledge injection facilitates contestability by allowing experts to incorporate feedback into models, closing a loop that begins with showing model results to stakeholders. Contestability could be seen as a prerogative of the *data subject*, the receiving end of an algorithmic decision [1], who would contest the model output and, possibly, a rationale thereof. However, extending contestability to a wider range of stakeholders, including experts evaluating a decision system and professionals using it, has been recommended [13]. Existing forms of contestability range from structured interactions and model explanations [14] to normative reasoning and process modelling [28]. The modality of interaction between the decision system and the "contester" should take into account the nature of the latter [13]. A data subject contesting a decision (e.g. not being granted credit) will often need layman explanations, while we focus on technical experts, providing them with detailed information in the form of causal graphs to understand and challenge model behaviour.

Our method allows technical experts to contest NNs at any point of the training, entrusting them to *debug* the model while validating the relationships that it is leveraging for its predictions. Human-in-the-loop (HITL) training usually involves humans in data processing or annotation [31]. Also, HITL debugging has been proposed to improve NNs used for Natural Language Processing (NLP) tasks (see [18] for a survey). In particular, [17] proposes *FIND*, a method to disable "spurious" filters in a Convolutional NN after showing a selection of filters to practitioners in the form of word-clouds. Instead of word-clouds, we employ causal graphs and we allow experts to disable spurious connections between features used in the NN. Our approach can be seen as a form of HITL contestability and debugging method guided by causal discovery.

Within the deep learning literature, in particular in the vision and NLP domains, inductive biases [11] and other strategies, ranging from architecture design to weights initialisation [3], have been proposed to enhance NNs through domain knowledge. Efforts in this field have been mainly towards tweaking the loss function or the hyper-parameters to make the NNs capture known characteristic of the modelling task [3]. Some works in this space have proposed injection of causal knowledge. Geiger et al. [7] propose *Interchange Intervention Training* (IIT) to induce NNs used in both computer vision and NLP tasks to have the same counterfactual behaviour of a given causal model. Zhang et al. [32] propose *deep CAusal Manipulation Augmented model* (CAMA), a method that uses inductive biases to make NNs robust to known manipulations of the input space, within computer vision tasks. In both [32] and [7] the whole causal model is given upfront and the induction/injection works by data augmentation. Our method instead (i) focuses on the causal structure only, without making assumptions on the causal model underpinning the DGP; (ii) modifies the weights of the model and (iii) involves humans in the discovery of causal relationships from the data used for a predictive task. Overall, none of the methods in the literature, to the best of our knowledge, allow for contesting discovered causal knowledge and injecting it back into NN in the form of causal graphs.

Beyond providing contestability and integration of expert knowledge, our method also helps in the causal discovery task (see [8] for an overview of causal discovery methods). Meek [19] is among the first to introduce background knowledge into the causal discovery task, however human input has been advocated since the inception of formal causal models [22]. Our experiments show the benefits of injecting partial causal knowledge, without hurting and generally improving, predictive performance in the downstream task. Constantinou et al. [5] have investigated the impact of ten different ways to incorporate prior knowledge on causal discovery for four causal discovery methods. Chowdhury et al. [4] have instead analysed the impact of prior knowledge for one specific method: NOTEARS [33]. This is of particular interest to this paper because the method we leverage on, CASTLE [16], includes NOTEARS' *acyclicity* formulation in its loss function, to induce the discovered graph to be a DAG. Our experiments show that contesting models by injecting knowledge in causal form helps causal discovery in smaller data settings, thus confirming the conclusions drawn by both [5] and [4], but in a novel setting.

3 Preliminaries

Modelling and Causal Set-up. Let $X_1, \ldots, X_d$ be the set of *input features* and Y the *target feature* within a regression or classification setting. Each feature X_k, for $k \in \{1, \ldots, d\}$, takes values in $\mathcal{X}_k \subseteq \mathbb{R}$ while Y takes values in $\mathcal{Y} \subseteq \mathbb{R}$ for regression and $\mathcal{Y} \subset \mathbb{Z}$ for classification. Let $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d$. We denote with $f_Y : \mathcal{X} \rightarrow \mathcal{Y}$ a function that maps assignments of values $[x_1, \ldots, x_d] \in \mathcal{X}$ to the input features onto a value $y \in \mathcal{Y}$ for the target feature Y. In practice, f_Y is learnt from a training dataset $\mathcal{D} = \{(x_{jk}, y_j) | j \in \{1, \ldots, N\}, k \in \{1, \ldots, d\}, x_j \in \mathcal{X}, y_j \in \mathcal{Y}\}$ drawn from a joint distribution of values for input and target features.

As in [23], a *causal structure* over input and target features is represented by a graph $\mathcal{G}$, a pair $\langle V, E \rangle$ with $V = \{X_1, \ldots, X_d, Y\}$ the set of nodes and $E \subseteq V \times V$ the set of edges of $\mathcal{G}$. We define a *full* DAG as a causal graph that is directed, acyclic and captures all applicable causal relationships amongst *all* features. For full DAGs, if an edge between two features is present (resp. absent), then the features are (resp. are not) in a causal relationship.

Given the scarcity of full DAGs for real-world applications, we also use what we call *partial* causal graphs, of the form $\mathcal{G}_p = \langle V_p, E_p \rangle$ with nodes $V_p \subseteq V$ and edges $E_p \subseteq V_p \times V_p$. Intuitively, if $i, k \in V_p$ but $(i, k) \notin E_p$ and $(k, i) \notin E_p$, then node i is definitely not causally related to node k; if $(i, k) \in E_p$ and $(k, i) \notin E_p$, then i *can* be a cause of k, but not vice versa. Hence, if a directed edge is present but the one between the same nodes with opposite direction is absent, then the latter relation is deemed as anti-causal. If $i, k \in V \setminus V_p$, then both (i, k) and $(k, i) \in E_p$, indicating the lack of knowledge about any causal relation among nodes i and k. Thus, if the graph is *complete* [23], i.e. with exactly $V_p = V$ and $E_p = V \times V$, then we know that all features could be causally related but not in which direction: NNs are used to aid the decision on the direction. Overall, our partial graphs compactly represent sets of constraints, and differ from full DAGs giving instead the causal structure among all observed features.

CASTLE [16]. Our proposed algorithms build upon the architecture of [16], whose schematic is provided in Fig. 1. CASTLE operates with a feed-forward NN combining $d+1$ sub-networks, each with d input neurons, amounting to all input and target features minus one: each sub-network masks a distinct element amongst $X_1, \ldots, X_d, Y$. The output layer of a sub-network with feature F masked in the input layer, has F as output neuron. Thus, each sub-network is responsible for reconstructing one feature, without using that feature. All sub-networks have $M + 1$ layers, and differ only in the input and output layers, i.e. layers $2, \ldots, M$ are shared. Hence, during training, the hidden layers are optimised to achieve the best performance for all

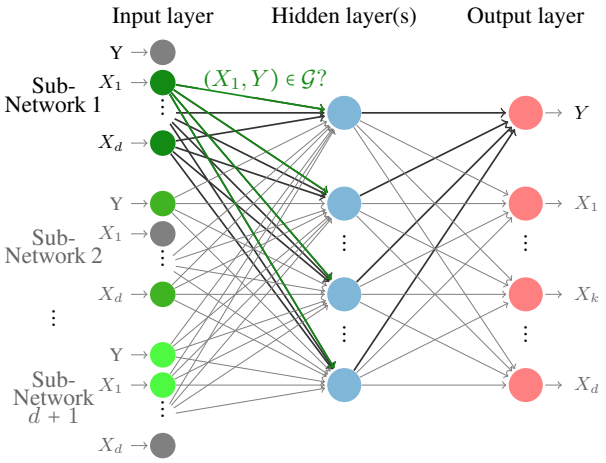


Figure 1: Joint Neural Network Structure. Darker arrows refer to the highlighted Sub-Network 1, predicting Y (Output layer), while having Y masked (grayed out in Input layer). Green arrows represent the weights that we consider masking when injecting causal knowledge answering questions like: "is X_1 a parent of Y?".

sub-networks at the same time, thus encoding the structure of the interactions among all features [16].

We refer to the feed-forward NN, with all its sub-networks, as the *joint NN*. We refer to weights for layer $l \in \{1, \dots, M\}$ as Θ_l, where $\Theta_l^{i,j}$ is the weight from neuron i in layer l to neuron j in layer $l+1$. $\Theta_1^{i,j,k}$ stands for the weight from input neuron i in layer 1 to neuron j in the first hidden layer of the k-th sub-network.

The joint NN carries out both the prediction of the target feature and the reconstruction of the input features. To train the NN, like [16], we use back-propagation and stochastic gradient descent applied to a loss function that includes a causal discovery element borrowed from NOTEARS [33]. More precisely, the loss is formed by two modules: the prediction loss and the *DAG loss*. The former is Mean Squared Error (MSE) or cross-entropy loss for regression and classification, respectively. The DAG loss is from [33] and can itself be broken down into three components: the reconstruction loss, MSE for each sub-network's output, assuming they are continuous; the *acyclicity* loss, a term that is 0 when $\mathbf{W}$ (described next) represents a DAG (from Theorem 1 of [33]); and finally an L_1 loss to induce sparsity in the weights' matrix.

$\mathbf{W}$ is a square hollow matrix of order $d+1$ holding a non-negative weight for each feature, including the target, from and towards all the others. The (i,k)-th entry of $\mathbf{W}$, for any $i, k \in \{1, \dots, d+1\}$, results from the square root of the sum of squared input layer weights across the hidden neurons. We denote the entries of $\mathbf{W}$ as w_{ik}. Formally, for h the number of hidden neurons in the first hidden layer:

$$w_{ik} = \sqrt{\sum_{j=1}^{h} \left(\Theta_1^{i,j,k}\right)^2} \tag{1}$$

Given the standardized input data, w_{ik} represent the magnitude of the effect that each feature i has on k. However, as discussed previously, $\mathbf{W}$ assumes causal connotations due to the *acyclicity* part of the loss function, which induces $\mathbf{W}$ to represent a DAG.

4 Methodology

In this section we first introduce our algorithm to inject causal knowledge in the form of a graph into feed-forward NNs (Alg. 1). The ability to make the NN respect external assumptions about the structure of the data, as afforded by Alg. 1, represents a key step in making NNs contestable. The contesting process is then provided in Alg. 2, which enables practitioners to challenge the recommendations of the NN, based on the causal graph discovered while computing them. Alg. 2 uses Alg. 1, to inject practitioners' feedback into NNs.

The Graph Injection Algorithm. Alg. 1 takes three main inputs: a training dataset $\mathcal{D}$, a joint NN with weights Θ_t, which can be randomly initialised or already fitted on $\mathcal{D}$, and a causal graph $\mathcal{G}$. The input graph $\mathcal{G}$ can take two forms: a full DAG, or a partial graph. In practice, having a full DAG is rare, hence we allow for $\mathcal{G}$ to be a partial graph (see §3), representing hard causal constraints while allowing the discovery of additional causal relations.

The output of Alg. 1 amounts to a masked joint NN with weights $\Theta_{\mathcal{G}}$, which only uses the relationships contemplated in $\mathcal{G}$: we call this an *Injected* NN. The injected NN resulting from Alg. 1 is fitted, or tuned if the input NN has already been fitted, to use only the relationships that were deemed causal by including them in $\mathcal{G}$. The injection is achieved through the masking of the weights of the input layer without a counterpart in the input DAG, namely $(i,k) \notin E$ means that X_i cannot cause X_k, hence $\Theta_1^{i,j,k} = 0 \ \forall j \in \{1, \dots, h\}$ (see line 4 of Alg. 1) which results in $w_{ik} = 0$. The `update` function represents a back-propagation pass and will iteratively refine the NN's weights to be effective in the prediction and reconstruction tasks without using spurious anti-causal relationships. The final weights $\Theta_{\mathcal{G}}$ of the injected NN result from progressive changes, for a number of steps that is at most T, if a *patience* threshold $T_s < T$ of loss improvement on the validation set is not reached.

Computationally, the main difference of Alg. 1 from CASTLE's training loop is that our algorithm focuses the training on the weights for the accepted edges, i.e. the causal relationships. Another way of understanding this process is as a "selective" causal dropout method. Intuitively, the original CASTLE methodology regularises the underlying NN so that the fitted model uses causal parents *more than* children and siblings for its predictions. Instead of only *preferring* the use of parents, our Alg. 1 *enforces* it: we reconstruct each feature

Algorithm 1 Inject Causal Graph

Input: Training Data $\mathcal{D}$; NN with M layers, h neurons in the first hidden layer, trained for t steps with final weights Θ_t; max number of steps T; patience $T_s < T$; causal graph $\mathcal{G} = \langle V, E \rangle$
Function: `inject_graph`$(\mathcal{D}, \Theta_t, T, T_s, \mathcal{G})$:

1: **for** $i \in \{1, \dots, d+1\}$, **for** $k \in \{1, \dots, d+1\}$ **do**,
2: **for** $j \in \{1, \dots, h\}$ **do**
3: **if** $i, k \in V \ \ \& \ \ (i,k) \notin E$ **then**
4: $\Theta_{1,t+1}^{i,j,k} \leftarrow 0$ ▷ mask anti-causal relations
5: **else**
6: $\mathcal{L}_{\text{best}}, t_s \leftarrow \infty, 0$
7: **while** $t < T \ \ \& \ \ t_s < T_s$ **do**
8: $\Theta_{1,t+1}^{i,j,k} \leftarrow$ `update`$(\Theta_{1,t}^{i,j,k}, \mathcal{D})$ ▷ causal relations
9: $\Theta_{m,t+1} \leftarrow$ `update`$(\Theta_{m,t}, \mathcal{D}) \ \forall m \in \{1, \dots, M\}$
10: $t \leftarrow t+1$
11: **if** $\mathcal{L}_t < \mathcal{L}_{\text{best}}$ **then**
12: $\mathcal{L}_{\text{best}}, t_s \leftarrow \mathcal{L}_t, 0$
13: **else**
14: $t_s \leftarrow t_s + 1$
15: $\Theta_{\mathcal{G}} \leftarrow \Theta_t$
16: **return** $\Theta_{\mathcal{G}}$

Output: Injected NN with weights $\Theta_{\mathcal{G}}$

Algorithm 2 Contest Computed Causal Graph

Input: Training Data $\mathcal{D}$; NN with M layers, h neurons in the first hidden layer, trained for t steps with final weights Θ_t; number of total training steps T; patience $T_s < T$; *Expert Knowledge*
Function: `contest_graph`$(\mathcal{D}, \Theta_t, T, T_s,$ *Expert Knowledge*$)$:

contested, $\tau \leftarrow$ *True,* 0
while *contested* = *True* **do**
  $\mathcal{G} \leftarrow g_\tau(\mathbf{W}_{\Theta_t})$
  $\mathcal{G}_r, \tau \leftarrow$ `revise_graph`$(\mathcal{G},$ *Expert Knowledge*$)$
  if $\mathcal{G}_r \neq \mathcal{G}$ **then**
    $\Theta_t \leftarrow$`inject_graph`$(\mathcal{D}, \Theta_t, T, T_s, \mathcal{G}_r)$
  else
    contested $\leftarrow$ *False*
$\Theta_\mathcal{G} \leftarrow \Theta_t$
return $\mathcal{G}, \Theta_\mathcal{G}$

Output: Refined DAG $\mathcal{G}$ and Injected NN with weights $\Theta_\mathcal{G}$

and carry out the target prediction using *only* each feature's parents. With this restriction, we aim at avoiding the use of a feature's children and/or siblings that may have unstable relationships with the parent feature being predicted: a change in a children/siblings will not necessarily change the feature, while a change in a parent of the feature will. Effectively, we encode the answers to causal questions into our masking scheme. An example of such questions is in Fig. 1: *does the edge from X_1 to Y belong to our agreed causal structure $\mathcal{G}$?* If not, we set the corresponding weights to 0 and prevent the effect of X_1 on Y. Note that we mask inputs and therefore direct effects while leave indirect effects to be captured by the hidden layers.

The Contesting Algorithm. Alg. 1 enables the injection of a graph representing the structural relationships among all or a subset of the features used by a NN to predict a target. With Alg. 2 we expose, in the form of a graph, the relationships that the NN has found in the data so that practitioners can critique and contest the output. We then use Alg. 1 to close the "contestation loop" and incorporate the human feedback into the NN. Alg. 2 has the same inputs as Alg. 1, apart from the input graph $\mathcal{G}$ which is replaced by what we call *Expert Knowledge*. This represents knowledge external to the data, and provided by SMEs that have prior experience with the modelling task. To leverage this external knowledge, we need to engage with the SMEs and we do so by means of the causal graph underpinning CASTLE's predictions. After calculating the adjacency matrix $\mathbf{W}_\Theta$ from the joint NN's weights Θ using Eq. 1, we transform it into a DAG $\mathcal{G} = g_\tau(\mathbf{W}_\Theta)$ using the following equations:

$$E_\tau(\mathbf{W}_\Theta) = \{(i,k) | w_{ik} > w_{ki} \wedge w_{ik} > \tau\} \tag{2}$$

$$g_\tau(\mathbf{W}_\Theta) = (V, E_\tau(\mathbf{W}_\Theta)) \tag{3}$$

Eq. 2 is a simple "edge creation function", applied to the adjacency matrix to produce the edges of a DAG with nodes $V = \{X_1, \ldots, X_d, Y\}$ as per Eq. 3. In general, the threshold $\tau \geq 0$ is meant to cut out the uninfluential relationships in the data. Thus, setting $\tau = 0$ results in treating all identified relationships, as represented by the elements w_{ik} of $\mathbf{W}_\Theta$, as influential.

Having extracted a DAG $\mathcal{G}$ from the joint NN, using Eq. 1 to 3 with $\tau = 0$, we present it to the SMEs for them to assess it, through the `revise_DAG` function. The output of this revision by the SMEs can be in regard to the threshold τ, to cut out more or less of the least influential effects, and/or in regard to specific edges in the computed DAG $\mathcal{G}$, resulting in the graph $\mathcal{G}_r$. As visible from the pseudo-code in Alg. 2, we propose an iterative contestation process, that outputs a revised causal DAG and a NN adhering to it, when the SMEs agree with the computed DAG, given the constraints they imposed.

SMEs are given the possibility to contest some or all of the relationships in the the DAG learnt from data and previous incorporated feedback, at any point of the learning process. This aims at assisting practitioners in validating and rectifying the causal relations discovered from the data, effectively debugging the NN based on its weights' structure. In the next section, we provide a case study on real data demonstrating how a practitioner can use our contesting algorithm to build more principled and predictive NNs.

5 Empirical Evaluation

We carried out two sets of experiments, on real and synthetic data, to assess the benefits of our proposed methods. Firstly, we present a case study on real data (§5.1), exemplifying the benefits that our methodology provides to modelling tasks in high-stakes decisions, when practitioners need to validate the relationships that the model is leveraging for its recommendations. Additionally, we provide experiments with synthetic data (§5.2) which, in line with [4, 5], show that prior knowledge helps the causal discovery for low data regimes, further motivating the importance of causal knowledge injection. Details of the implementation, including code, are in Appendix A.

5.1 Case Study with Real Data

We use real financial data from four publicly available datasets (see details in Appendix A.1): two classification and two regression tasks. For the classifications, we use the Adult Income dataset [15], useful for affordability checks in the lending business to predict whether a loan applicant's income is greater than USD50K and the FICO HELOC data [6] for *credit risk* assessment, to predict whether an applicant is likely to repay a loan. The two regression tasks are instead about predicting house prices: we use the Boston [12] and California [21] Housing datasets. We report analysis for three scenarios using Alg. 2: reconstructing a *full* DAG via threshold τ optimisation, with no prior causal knowledge, to then inject it into our NNs (§5.1.1); injecting partial a priori causal knowledge expressing basic common sense assumptions for the Adult dataset (§5.1.2), and showcasing how practitioners can contest the DAG computed in §5.1.1 using the assumptions adopted in §5.1.2 (§5.1.3). In the absence of a true DAG for these datasets, our quantitative metric is predictive performance, namely MSE for regression and Area Under the Curve (AUC) for classification. All results are reported with observed significance levels for a two-tailed two-sample t-test (details of the testing procedure is reported in Appendix A.1.1). Our objective is demonstrating that predictive performance is not necessarily impacted by knowledge injection which can, in turn, help building more transparent and validated models.

*5.1.1 **No a priori Knowledge***

In this first scenario we assume no a priori causal knowledge, and use Alg. 2 to construct potential DAGs by optimising the choice of threshold τ, to then inject them into the NNs and measure the predictive performance with and without injection. Within the g_τ function from Eq. 3, used at the beginning of Alg. 2, we tried between 10 and 15 different thresholds τ for each dataset and chose the "best" DAG through the evaluation of the change in predictive performance. For each dataset, we selected the DAG with lowest MSE/highest (AUC) and, as tie breaker, the lowest number of edges in the computed DAG. Details of this evaluation are in Appendix A.1.2.

Table 1: Experiments with real data. We report mean MSE or AUC (std) for regression and classification, respectively, across different sample sizes of the training data (N) and 5-fold nested cross validation, best results in bold. Observed significance levels against CASTLE baseline are reported with the following intervals: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1. NA indicates a data size bigger than the full dataset. We also detail the number of features/nodes $|V|$ and the number of edges $|E|$ in the injected DAG (for our method) and in the graph drawn from the extracted adjacency matrix (for CASTLE). *Injected* columns refers to §5.1.1, *Partial* to §5.1.2 and *Refined* to §5.1.3.

	CLASSIFICATION (Metric: AUC)						**REGRESSION (Metric: MSE)**			
	Adult ($\|V\|$ = 14)				**HELOC** ($\|V\|$ = 23)		**California** ($\|V\|$ = 8)		**Boston** ($\|V\|$ = 14)	
Data (N)	**CASTLE** $\|E\|$ = 210	***Injected*** $\|E\|$ = 46	***Partial*** $\|E\|$ = 116	***Refined*** $\|E\|$ = 30	**CASTLE** $\|E\|$ = 552	***Injected*** $\|E\|$ = 85	**CASTLE** $\|E\|$ = 72	***Injected*** $\|E\|$ = 31	**CASTLE** $\|E\|$ = 182	***Injected*** $\|E\|$ = 48
100	0.67 (0.03)	**0.69 .**	0.66	**0.69 .**	**0.75 (0.02)**	0.74	7.05 (12.81)	**2.94 *****	112.04 (91.06)	**86.17 *****
500	0.72 (0.04)	**0.74 ***	0.71	**0.74 ***	**0.79 (0.01)**	0.78 ***	2.33 (1.39)	**2.25**	21.95 (6.84)	**20.45 ***
1000	0.75 (0.03)	**0.76**	0.74	**0.76**	0.78 (0.01)	0.78	2.96 (4.12)	**1.68 ****	NA	NA
2000	0.74 (0.03)	**0.77 *****	**0.76 ***	**0.77 *****	**0.79 (0.01)**	0.78 ***	3.86 (3.68)	**1.71 *****	NA	NA
5000	0.75 (0.03)	**0.79 *****	**0.76**	**0.79 *****	0.79 (0.01)	0.79	4.91 (7.41)	**1.51 *****	NA	NA
10000	0.75 (0.02)	**0.85 *****	**0.76 .**	**0.85 *****	**0.80 (0.01)**	0.79 ***	1.74 (1.70)	**1.16 ***	NA	NA
20000	0.76 (0.02)	**0.86 *****	**0.77 .**	**0.86 *****	NA	NA	**0.66 (0.08)**	1.02 **	NA	NA

Within Alg. 2, we make the contesting process terminate with the DAGs chosen through this automatic procedure. We aimed at checking whether fixing a plausible, though not humanly validated, DAG can lead to better predictive performance and whether the level of improvement depends on data size.

As reported in Table 1, the predictive performance of the injected NNs can be up to 2.4x better than CASTLE (California with N=100). AUC for the Adult dataset is consistently above CASTLE (up to 13% and significantly so for most of the sample sizes) while using only ~20% of the relationships that the "unconstrained" CASTLE network uses.[4] Similarly, for the Boston data, injection reduces edges by ~75% while significantly improving performance. For the California dataset, the MSE is significantly better for all sample sizes but the biggest and $N = 500$ where there is not significant difference in the means. However, the reduction of computed DAG's edges is ~40%. Finally, for the HELOC dataset the AUC for the injected NN is not significantly lower than CASTLE in half the cases, and by a maximum of 1%, but with a much sparser NN: the amount of first layer's weights is only 15% of the unconstrained NN. This stark reduction will have included some useful relationships that, with some refinement, could be reintroduced to improve performance. We note that, by *minimality* [23] or *parsimony* [29], when the performance stays equal, a modeller should prefer the sparser, more parsimonious model. Overall, simulating the use of Alg. 2 without prior knowledge produced parsimonious NNs with an average 75% less connections in the input layer. Moreover, predictive performance generally improves significantly or, in the worst case, worsens by at most 1%, with no clear distinction by sample size.

Note that, for this first scenario, our strategy for choosing the DAGs to inject is purely mechanic. The adopted strategy is meant to gauge impact on predictive performance without a qualitative assessment of the validity of domain specific causal assumptions, a task that SMEs should carry out. We envisage this strategy as a useful starting point in the absence of causal knowledge defined a priori. However, in real life applications, we intend the use of Alg. 2 by a panel of SMEs, iteratively assessing intermediate outputs to refine the NN in light of previous experiments, leveraging their experience and knowledge of the modelling task, while learning more about it.

Next, we introduce two experiments providing examples of how Alg. 2 can work in a human-AI collaboration setting, with SMEs constraining and contesting DAGs computed by NN. We run these experiments only on the Adult dataset, as it contains some features, notably *race, sex, age* and *native-county*, that lend themselves to the construction of common sense causal assumption by lay users e.g., sex cannot be caused by age. This way we avoid formulating domain specific assumptions while providing a tangible example application.

[4] This reduction matches the reduction in NNs' weights at the input layer.

5.1.2 *Partial a priori Knowledge*

In this experiment we build a very simple, yet intuitive, partial input graph $\mathcal{G}_p$ for the Adult dataset to constrain the NN to respect the following assumptions (reflected in the adjacency matrix in Fig. 2):

- *race, sex, age, native-county* cannot be caused by any feature, i.e. these features cannot have incoming edges. As a result, columns 2 to 5 of the adjacency matrix in Fig. 2 are blanked ($w_{ik} = 0$);
- *occupation* and *hours-per-week* cannot cause *fnlwgt (demographics index), education, education-num, relationship* and *marital-status*; the respective cells in Fig. 2 are blanked;
- the target ($income > USD50K$) cannot cause any feature and *capital-gain, capital-loss* can only affect the target; rows 1, 13 and 14 in Fig. 2 are blanked.

Note that some of these assumptions could easily be confuted, e.g., by arguing that the target *can* cause features, such as *capital-gain* and *loss*. We adopt these assumptions only to illustrate the effects on the adjacency matrix and to simulate a scenario whereby the modeller is testing whether the algorithm finds relationships in the data that help the prediction task. As in [23], we believe that the opportunity to extract and enforce such assumptions, or simply talk about them, has the potential to make models more transparent, robust and representative of causal mechanisms of the world.

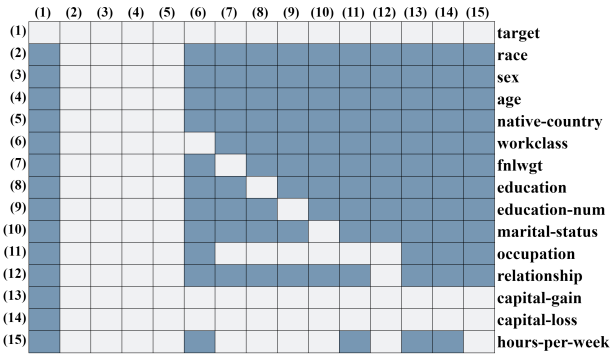


Figure 2: Input graph $\mathcal{G}_p$, as partial causal knowledge for the Adult dataset, in the form of an adjacency matrix $\mathbf{W}$. Blue represents edges; missing edges in white (hard constraints).

We report the AUC of the NN injected with the graph in Fig. 2 in Table 1, ***Partial*** column. The "causal constraints" result in performances generally not significantly different from CASTLE, but with a computed DAG that has about a half the edges of the unconstrained NN ($|E|$ = 116 vs $|E|$ = 210). On the whole, we obtain a sparser NN, adherent to common-sensical causal knowledge following our assumptions, whose recommendations are therefore arguably more understandable and trustworthy, and whose performance is comparable to an unconstrained NN.

5.1.3 *Contesting a Computed DAG*

Our last experiment on real data aims at showcasing the process of contesting the causal structure computed and used by the NN, as afforded by Alg. 2. The experiment starts, as the experiment in §5.1.1, with no prior knowledge about the problem. We adopt the same threshold optimisation strategy detailed in Appendix A.1.2 and chose $\tau = 0.08$. Hence, in the first few runs of Alg. 2 the output of the `revise_graph` function would only change the threshold τ. Having chosen $\tau = 0.08$, the DAG is extracted and shown to the SMEs, as in Fig. 3. In this DAG, the target (Income>50K) is deemed to be causing a few features including sex and age (see purple arrows).

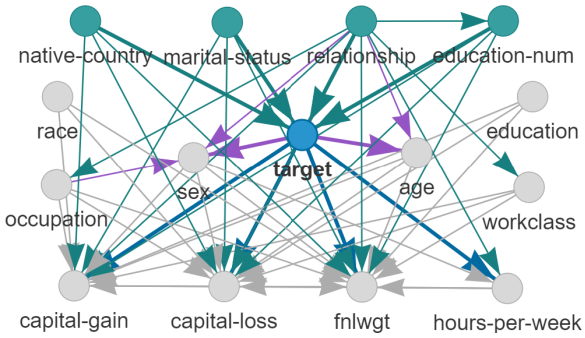


Figure 3: Example of computed DAG for Adult dataset. Cyan nodes at the top are computed causes for the target ("Income>50K"), edges coming out of the target are in blue while in purple are the edges into nodes that cannot be caused (as per basic assumptions in Fig. 2).

This is counter-intuitive from a common sense, let alone causal, perspective. Thus, the contestation now addresses specific edges in the computed DAG, and `revise_graph` produces a DAG $\mathcal{G}_r$ that differs from $\mathcal{G}$ in Fig. 3 by the purple edges. $\mathcal{G}_r$ is then injected back into the NN producing the results in Table 1, ***Refined*** column: the NN injected with the DAG refined by means of contesting is not only more intuitive and adhering to common sense, but presents the same predictive performance as the NN using non-sensical relationships and significantly better predictive performance than CASTLE. Ultimately, our *Refined* NN is 7x smaller in the input layer, adheres to common sense, and yet it is up to 13% more predictive than an unconstrained NN.

5.2 *Experiments with Synthetic Data*

To confirm the results from our case study on real data, we investigate the effectiveness of our proposed method on synthetic data. Our simulations compare scenario (i) in §5.1.1, where no prior causal knowledge is available, to scenario (ii) in §5.1.2, where practitioners do have a set of a priori assumptions. The comparison of scenarios (i) and (ii) can be seen, in the setting with synthetic data serving as a proxy for domain experts, as amounting to the scenario in §5.1.3, where contesting an initial computed DAG corresponds to providing a priori knowledge in scenario (ii); the only difference lies in the starting point. We chose to test these scenarios also because of the easier simulation. The experiments aim at answering the following questions: (Q1) Does knowledge injection by our algorithms improve predictive performance? (Q2) Can we reconstruct a DAG, known to be underpinning the DGP, using Alg. 2? (Q3) How well can Alg. 2 fill the gaps of an input graph contributing only partial knowledge? (Q4) How does knowledge injection performance change in different data size regimes? (Q5) How resilient are our algorithms to noise?

Using Alg. 2, we represent the *Expert Knowledge* with an input graph $\mathcal{G}$ encapsulating a priori partial causal knowledge among a subset of the features fed to the NN. In the experiments shown in Fig. 4 we inject a 20% random sample of the total amount of edges in the true DAG; experiments with 10% and 50% of DAG edges injected are reported in Appendix A.2.1. Once the known edges are selected, the entries of the edges representing the opposite direction in the adjacency matrix $\mathbf{W}$ are set to 0, e.g. $X_i \rightarrow X_j$ is selected as known, then $(i, j) \in E$ and $w_{ji} = 0$.

Synthetic Data Generation. We generate synthetic data adhering to a series of randomly generated DAGs of different sizes, using the methodology of [16].[5] The generated synthetic DAGs and data vary across three main dimensions: number of nodes in $\mathcal{G}$ ($|V| \in \{10, 20, 50\}$), number of edges ($|E| = |V| * e$, where $e \in \{1, 2, 5\}$), and data size ($N = |V| * s$, where $s \in \{50, 100, 200, 300, 500\}$). In the remainder, we refer to s as *proportional sample size*.

Evaluation Metrics. We use average and Standard Deviation (Std) of *Mean Squared Error (MSE)* for the evaluation of *predictive performance* for questions Q1, Q4 and Q5. For the evaluation of *reconstruction accuracy* for questions Q2, Q3 and Q5 we use the distribution of the percentage of edges that match those in the true DAG. We run each scenario, involving one of the combinations of $|V|$, e and s, 10 times and report the distribution of results as a boxplot in Fig. 4, in comparison with the baselines detailed next. As in the experiments with real data, differences in means are tested with a t-test.

Baselines. CASTLE's predictive performance has been compared to the main NNs' regularisation methods in the literature [16], providing the best performance for both classification and regression tasks on both synthetic and real data. Hence, for prediction, our baseline is a well-performing regularised NN: CASTLE. However, CASTLE alone cannot be used as a baseline for reconstruction accuracy but the adjacency matrix extracted from CASTLE using Eq. 1 provides a useful starting point. Our baseline is thus built by: using CASTLE's weights Θ to compute $\mathbf{W}_\Theta$ according to Eq. 1 and then Eq. 3 to derive a DAG $\mathcal{G} = g_\tau(\mathbf{W}_\Theta)$. We call this method CASTLE+.

The differences between CASTLE+ and our Alg. 1 are as follows. For CASTLE+ we let CASTLE run unconstrained and only after training we extract an adjacency matrix (Eq. 1) and transform it into a DAG (Eq. 3 with an appropriate choice of threshold τ). Instead, our method for injecting the DAG involves feeding a graph into the NN so that a mask is applied and only the non-masked weights are optimised. For an illustration of the procedure as to how we inject causal knowledge through masking, refer to §3. Note that CASTLE+ amounts to using Alg. 1 with a complete graph $\mathcal{G}_p$. In the experiments we use the value of τ that produced the lowest number of mismatches, for each of CASTLE+ and our method.

[5] For details refer to Appendix B.1 of [16]. Note that we standardise all feature values in the generated data to mean 0 and std 1. Thus, following [24], our results can be regarded as conservative estimates of reconstruction accuracy.

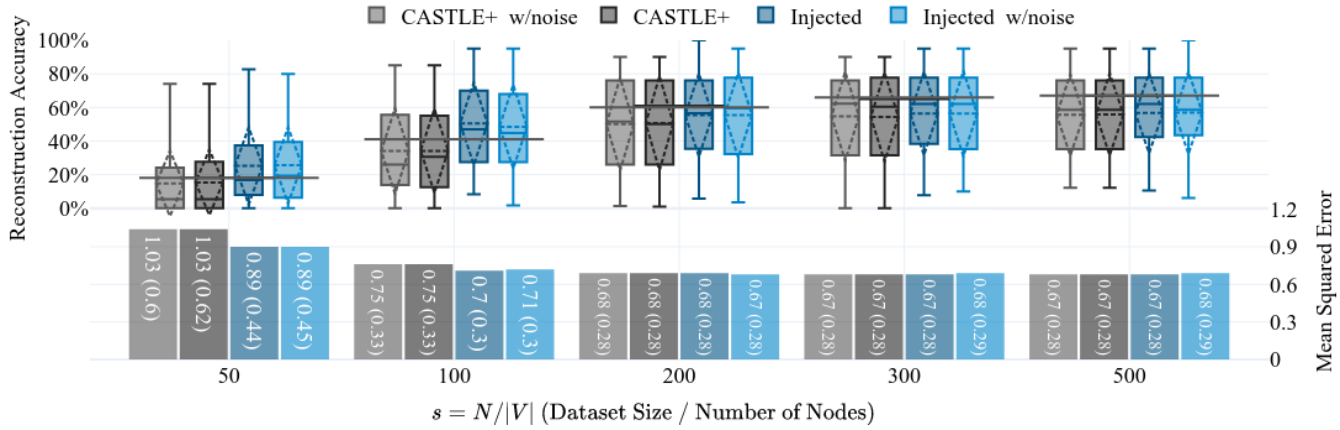


Figure 4: Reconstruction Accuracy and MSE when changing $s = N/|V| \in \{50, 100, 200, 300, 500\}$, the sample size N proportional to the number of nodes $|V| \in \{10, 20, 50\}$ in the causal DAG $\mathcal{G}$. Darker (grey or blue) colors refer to the no-noise scenario, whereas lighter colors refer to the scenario with noise. The values are an average over 10 runs for each combination of $|V|, s$ and $e = |E|/|V| \in \{1, 2, 5\}$. The boxplots (left y axis) show Min/Max/Median (solid lines) and Mean/Std (dashed lines) of the reconstruction accuracy. The bottom bars (right y axis) show the MSE (std). The solid horizontal lines spanning across each of the pairs of boxplots are the re-based value of the CASTLE+ mean to account for the advantage that our Injection methodology knows 20% of the edges. If the mean of *Injected* is above/below the level of the horizontal lines, the average increase in reconstruction accuracy is more/less than proportional to the amount injected.

Simple DGP. The results for this scenario are given in darker colors in Fig. 4, where we show two metrics: the predictive performance (MSE) with the bars at the bottom of the figure; and the distributions of reconstruction accuracy through the boxplots at the top of the figure. The results presented vary across one of the three dimensions mentioned earlier, namely the proportional sample size s. Analysis of the changes over the other two dimensions ($e = |E|/|V|$ and $|V|$) are left to Appendices A.2.2 and A.2.3. From Fig. 4 we can observe that the predictive performance improves with injection for small data regimes (up to 15% for $s < 200$) while it is not affected for bigger proportional sample sizes. However, none of the effects are statistically significant. Also for reconstruction accuracy the biggest gains are again observed in the low data scenario. The proportional gains in the number of correct edges is greater than the proportion of edges injected by up to 10%, as represented by the mean of the boxplots lying above the gray longitudinal lines spanning across them. The increases for $s = 50$ and $s = 100$ result to be statistically significant at the 5% level ($t(178) = 2.226, p = 0.027$ and $t(178) = 2.464, p = 0.015$, respectively). For the other s, no significant differences are observed. With these experiments we can answer questions Q1 through Q4: Alg. 1 can improve DAG reconstruction (Q2) as well as fill in gaps in partial causal knowledge (Q3), with no decrease in prediction performance (Q1), but only for low data regimes (Q4).

Noisy DGP. To assess the robustness of Alg. 1 to noise, aiming at answering question Q5, we add to the training data a number of features amounting to 20% of the number of nodes in the DAG used to generate the data. These additional "noisy" features are generated out of a standard normal distribution and have no links to the other features in the data. Results are again presented in Fig. 4 for ease of comparison with the no-noise scenario. As visible from the bottom bar charts, the MSE for the target feature Y stays effectively the same across the different proportional data sizes (no significant differences in mean). Also the reconstruction accuracy (top boxplots) appears not to be affected at all (again, no significant differences in mean). This is in line with the results presented in [16]. We can conclude that our algorithm is resilient to noise with regard to both reconstruction and predictive performance (Q5).

6 Conclusion

The proposed methods represent a principled approach to fitting neural networks (NNs): we leverage knowledge injection in the form of causal graphs to empower technical experts to contest NNs, based on the structural assumptions discovered from the data. We propose two algorithms to deliver *contestable* NNs: the first unlocks contestability by allowing networks to take feedback in the form of causal graph injection; the second uses computed causal graphs to elicit feedback from experts, so that they can contest the data-driven causal graph and inject their causal views into the NN. We apply our algorithms to real financial datasets demonstrating how they can yield very parsimonious, hence more interpretable and easier to debug, NNs, while either significantly improving or losing very little predictive performance. Finally, we demonstrate, through empirical results on synthetic data, that knowledge injection, as afforded by our method, generally improves causal discovery in low data regimes, despite noise. We used predictive performance to assess the viability of our method for prediction tasks and found that knowledge injection can produce similar or better performing NNs, but with the added confidence of being able to explore the relationships used in the predictions.

We envisage interesting lines of future work including: exploring the indirect causal effects that take place in the hidden layers of our injected NN; introducing Bayesian learning weight updates, e.g. as in [20], to improve our method's human-AI collaboration capabilities through uncertainty quantification. Further, we would like to equip our method with the capability to aggregate independent views of several experts into a consistent causal graph, similar to Alrajeh et al. [2]'s proposal for causal models, and to allow the enforcement of the presence of causal direction on top of the absence thereof. Our proposed method not only produces contestable NNs, but also improves their interpretability and, we believe, their trustworthiness. This is because SMEs are called to examine computed causal graph and provide feedback that the NN is guaranteed to respect: we will explore this angle in future work. Finally, we plan to explore further the human-in-the-loop debugging capabilities of our method, especially for high-stakes decision models, and conduct user studies on the propensity and efficacy of SMEs in understanding and contesting model outputs depending on presentation and interaction modalities.

Acknowledgements

We would like to thank Ruben Menke, Torgunn Ringsø, Antonio Rago, Francesco Leofante, Mark Somers and all the anonymous reviewers for the helpful feedback on earlier version of the paper. Russo was supported by UK Research and Innovation (grant number EP/S023356/1), in the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence (www.safeandtrustedai.org). Toni was partially funded by the ERC under the EU's Horizon 2020 research and innovation programme (grant agreement No. 101020934) and by J.P. Morgan and by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme.

References

[1] Marco Almada, 'Human intervention in automated decision-making: Toward the construction of contestable systems', in *Proc. ICAIL*, (2019).
[2] Dalal Alrajeh, Hana Chockler, and Joseph Y Halpern, 'Combining experts' causal judgments', *Artificial Intelligence*, (2020).
[3] Andrea Borghesi, Federico Baldo, and Michela Milano, 'Improving deep learning models via constraint-based domain knowledge: a brief survey', *arXiv:2005.10691*, (2020).
[4] Jawad Chowdhury, Rezaur Rashid, and Gabriel Terejanu, 'Evaluation of Induced Expert Knowledge in Causal Structure Learning by NOTEARS', in *Proc. ICPRAM*, (2023).
[5] Anthony C Constantinou, Zhigao Guo, and Neville K Kitson, 'The impact of prior knowledge on causal structure learning', *arXiv:2102.00473*, (2021).
[6] FICO. Fico xml challenge found at community.fico.com/s/xml, 2017.
[7] Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts, 'Inducing causal structure for interpretable neural networks', in *Proc. ICML*, (2022).
[8] Clark Glymour, Kun Zhang, and Peter Spirtes, 'Review of causal discovery methods based on graphical models', *Frontiers in genetics*, 524, (2019).
[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.
[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', in *Proc. ICLR*, (2015).
[11] Anirudh Goyal and Yoshua Bengio, 'Inductive biases for deep learning of higher-level cognition', *Proceedings of the Royal Society A*, **478**(2266), (2022).
[12] David Harrison and Daniel Rubinfeld, 'Hedonic housing prices and the demand for clean air', *Journal of Environmental Economics and Management*, (1978).
[13] Clément Henin and Daniel Le Métayer, 'Beyond explainability: justifiability and contestability of algorithmic decision systems', *AI & SOCIETY*, 1–14, (2021).
[14] Daniel N Kluttz, Nitin Kohli, and Deirdre K Mulligan, 'Shaping our tools: Contestability as a means to promote responsible algorithmic decision making in the professions', in *Ethics of Data and Analytics*, 420–428, Auerbach Publications, (2022).
[15] Ron Kohavi et al., 'Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.', in *Proc. KDD*, (1996).
[16] Trent Kyono, Yao Zhang, and Mihaela van der Schaar, 'Castle: Regularization via auxiliary causal graph discovery', in *Proc. NeurIPS*, (2020).
[17] Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni, 'Find: Human-in-the-loop debugging deep text classifiers', in *Proc. EMNLP*, pp. 332–348, (2020).
[18] Piyawat Lertvittayakumjorn and Francesca Toni, 'Explanation-based human debugging of nlp models: A survey', *Transactions of the Association for Computational Linguistics*, **9**, 1508–1528, (2021).
[19] C Meek. Causal inference and causal explanation with background knowledge in uncertainty in artificial intelligence 11, 1995.
[20] Vikram Mullachery, Aniruddh Khera, and Amir Husain, 'Bayesian neural networks', *arXiv:1801.07710*, (2018).
[21] R Kelley Pace and Ronald Barry, 'Sparse spatial autoregressions', *Statistics & Probability Letters*, 291–297, (1997).
[22] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
[23] Judea Pearl, *Causality*, Cambridge University Press, 2 edn., 2009.
[24] Alexander Reisach, Christof Seiler, and Sebastian Weichwald, 'Beware of the simulated dag! causal discovery benchmarks may be easy to game', in *Proc. NeurIPS*, (2021).
[25] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke, 'Explainable machine learning for scientific insights and discoveries', *Ieee Access*, **8**, 42200–42216, (2020).
[26] Cynthia Rudin, 'Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead', *Nature Machine Intelligence*, 206–215, (2019).
[27] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus, 'Intriguing properties of neural networks', in *Proc. ICLR*, (2014).
[28] Andrea Aler Tubella, Andreas Theodorou, Virginia Dignum, and Loizos Michael, 'Contestable black boxes', in *Proc. RuleML+RR*, (2020).
[29] Joachim Vandekerckhove, Dora Matzke, and Eric-Jan Wagenmakers, 'Model comparison and the principle of parsimony', *The Oxford handbook of computational and mathematical psychology*, **300**, (2015).
[30] Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al., 'Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems', *IEEE Transactions on Knowledge and Data Engineering*, **35**(1), 614–633, (2021).
[31] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He, 'A survey of human-in-the-loop for machine learning', *Future Generation Computer Systems*, (2022).
[32] Cheng Zhang, Kun Zhang, and Yingzhen Li, 'A causal view on robustness of neural networks', in *Proc. NeurIPS*, (2020).
[33] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing, 'Dags with no tears: Continuous optimization for structure learning', in *Proc. NeurIPS*, (2018).

Appendices

A Further Details for Experiments

For all experiments[6] we choose 3 hidden layers ($M = 3$) of sizes $2 * |V|, 2/3 * |V|, 2 * |V|$ respectively, with ReLU activations. Experiments with smaller networks are provided for synthetic data in §A.2.4. All networks are initialized and seeded identically and use the Adam optimizer with a learning rate of 0.001 for a maximum of 1000 steps (T). A patience (T_s) of 50 steps on the loss on the validation set is used to stop training. Results for the synthetic dataset are then reported for 10 randomly generated DAGs. For the real data experiments, given the hyper-parameters optimisation of the threshold τ, we have used 5-fold nested cross validation and the results reported are the average over the resulting 25 runs.

A.1 Experiments with Real Data

Here we provide additional details for the experiments in §5.1: we start from the datasets to then cover the optimisation of the threshold τ, part of experiment (i) in §5.1.1.

We report details for the datasets in Table 2. For the regression tasks, with Boston [12] and California Housing [21], we used the data out of the box from scikit-learn[7]. For the classification tasks, with HELOC[8] [6] and Adult Income[9] [15], we used pre-processed data. Additionally, for Adult, we sampled 20000 observations and we balanced the proportion of positive and negative examples in target feature to 50-50 (from 25-75).

Table 2: Real-world dataset details. Type is Regression (R) for real-valued target or Classification (C) for binary target.

Dataset	Sample Size	Features	Type
Boston Housing (BH)	506	14	R
California Housing (CH)	16512	8	R
Home Equity Line Of Credit (HELOC)	7844	23	C
Adult Income (IN)	48842	14	C

A.1.1 Details on Statistical Tests

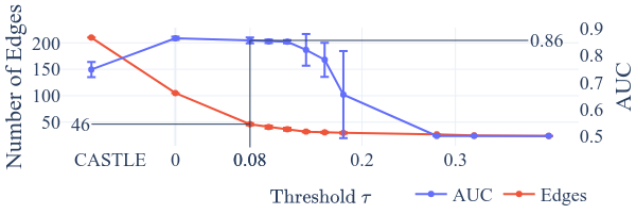
Here we report the details to reproduce the observed significance levels reported in Table 1. Table 3 reports the means and standard deviations for all the experiments. As mentioned in the main text, we used a two-tailed, two-sample t-test for difference in means. Each sample was made up of 25 runs within a 5-fold nested cross validation resulting in 48 degrees of freedom for the Student's t-distribution. The t-statistics and associated p-values are reported in Table 4. Note that for California and Boston the homoscedasticity (equal variance) assumption was not satisfied (after testing the difference in variance with an F-test) hence we used the heteroscedastic t-test.

[6] Our code is available at: https://github.com/briziorusso/causal-injection-FFNN/tree/main

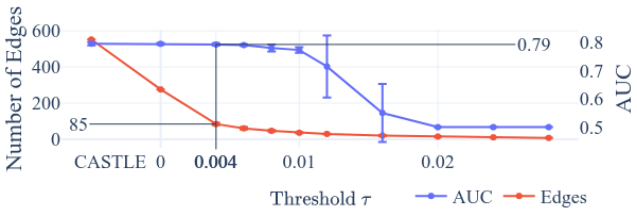
[7] https://scikit-learn.org/stable/datasets/toy_dataset.html

[8] Data and pre-processing mapping (keys.csv) is downloadable from our repo at https://github.com/briziorusso/causal-injection-FFNN/tree/main/data/fico/;

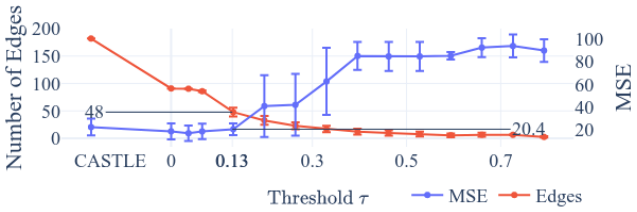
[9] Data and pre-processing mapping are from Penn Machine Learning Datasets repository at https://github.com/EpistasisLab/pmlb/blob/master/datasets/adult/



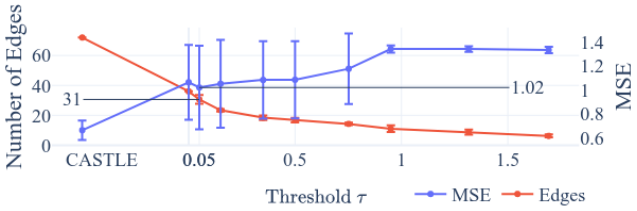
(a) Adult Income



(b) FICO HELOC



(c) Boston Housing



(d) California Housing

Figure 5: Threshold τ optimisation for experiment in §5.1.1. Here $\tau < 0$ corresponds to the application of "unconstrained" CASTLE [16]. Chosen thresholds are in bold on the x-axis. The number of edges and the predictive performance of the network injected with the DAG derived with the chosen τ are reported on the y-axes.

A.1.2 Threshold Optimisation.

Here we provide details on the optimisation of the threshold τ to chose a DAG without having to make qualitative causal judgments, as part of the experiment in §5.1.1. The results for all datasets are in Fig. 5. The optimisation runs through several thresholds and compares the number of edges (red) and the appropriate predictive performance metric (blue) when increasing the threshold τ, along the x axis, so that more and more edges are masked. As expected, the number of edges decreases monotonically for bigger thresholds while MSE/AUCs are the trends of interest.

As visible in Fig. 5a for the Adult dataset, $\tau = 0.08$ keeps the best AUC, while reducing the number of edges by more than 50% compared to $\tau = 0$. Increasing τ produces small gains in parsimony (lower $|E|$) but starts to reduce performance. The same applies to the FICO HELOC data in Fig.5b, where we select $\tau = 0.004$, before the AUC starts to decrease. For both datasets, valid alternative choices are $\tau = 0.1$ and 0.01, respectively, but we preferred a lesser worsening of performance for small gains in parsimony. For the Boston dataset in Fig.5c, we select $\tau = 0.13$ as the MSE thereafter increases significantly. Finally, California (Fig.5d) shows a different scenario:

injecting always hurts predictive performance. We chose $\tau = 0.05$ which has the lowest MSE. However, as visible in Table 1 in the main text, the worse MSE is not observed for injected NNs on smaller sample sizes.

A.2 Parameter Study for Synthetic Data

In §5.2 we report results for a fixed number of edges injected (20%). Here we provide comparisons of performance for different percentages of edges injected (§A.2.1). Moreover, the results reported in Fig. 4 show variation wrt one of the three dimensions considered when generating the random DAGs and data, namely, proportional sample size ($s = N/|V|$). Here, we report additional results for the other two dimensions: number of nodes $|V|$ (§A.2.3), and proportion of edges over nodes $e = |E|/|V|$ (§A.2.2). Finally, we report a comparison for network size (§A.2.4). The results of this section corroborate the ones presented in §5.2.

A.2.1 Percentage of Known Edges.

In Fig. 6 we vary the proportion of edges injected (10%, 20% as in Fig.4, and 50%). As visible, injecting 50% of edges never pays off proportionally, i.e. the average reconstruction accuracy, although higher than CASTLE+, generates less of an increase than "rebased" CASTLE+ (solid horizontal lines). On the other hand, the reconstruction accuracy when injecting 10% of the DAGs is always more than proportional to the injected amount. Most of the gains are recorded for denser DAGs ($e \in \{2, 5\}$) when reconstruction is generally more difficult, as shown by the decreasing overall trend.

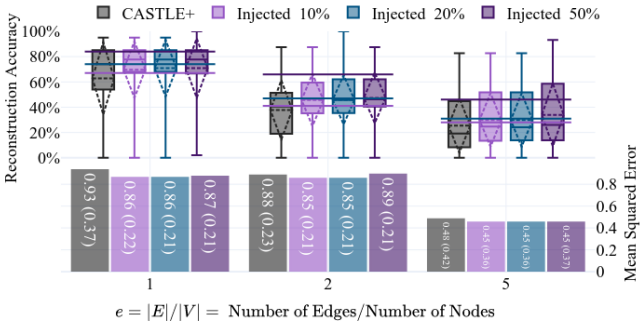


Figure 6: Reconstruction Accuracy and MSE when changing the proportion of edges over nodes (see §A.2.2). The amount of known edges injected is 10%, 20% (as in the paper, see Fig.4) and 50% (see §A.2.1).

A.2.2 Number of Edges in the DAG

The effect of changing the proportion of edges per node ($e = |E|/|V|$) is presented in Fig. 6 (where CASTLE+ vs Injected 20% is the scenario shown in Fig.4 in the main text). Results show that the sparser the DAG (the smaller e) the better the performance of our algorithm, achieving reconstruction accuracy averaging at around 75% for $e = 1$. For $e = 2$ the average drops to the average level across all proportional sample sizes s, while increasing e further, to 5, results in averages dropping to less then 40%, comparable to the effect of having only 50 observations per node ($s = 50$, Fig.4). Overall, the denser the DAG, the worse the performance of both CASTLE+ and our algorithms.

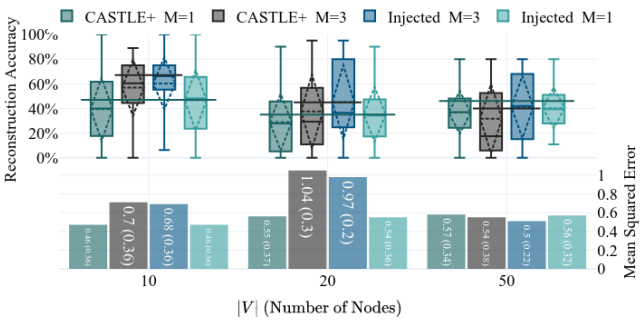


Figure 7: Reconstruction Accuracy and MSE when changing the number of nodes in the DAG underpinning the data (see §A.2.3) and the numbers of layers (M=3 in Fig.4, see §A.2.4).

A.2.3 Number of Nodes in the DAG

MSE and reconstruction accuracy results when changing $|V|$ are shown in Figure 7 ($M = 3$ correspond to the main scenario, as in Fig.4). We can observe that the performance varies significantly for increasing DAG sizes. For $|V| = 10$, both CASTLE+ and our method show better reconstruction accuracy than the average across s (in Fig. 4). For $|V| > 10$, however, we observe a significant drop in overall performance for CASTLE+, whereas our method suffers less from the increased size of the DAG.

A.2.4 Network Size

In Fig. 7, jointly with the analysis on the number of nodes, we show a comparison of 3-layers networks (used for the experiments in §5.2) with smaller networks of one single hidden layer and an amount of neurons of 3.2 times the number of input features (i.e. $M = 1, h = (d + 1) * 3.2$). Interestingly, as visible from the bottom bar charts, the MSE for the target variable Y generally improves with smaller networks, while the same change worsens reconstruction accuracy. Better prediction does not always couple with better causal discovery.

Table 3: MSE or AUC (std) for regression and classification, respectively, across different sample sizes of the training data (N).

	CLASSIFICATION (Metric: AUC)						REGRESSION (Metric: MSE)			
	Adult				HELOC		California		Boston	
Data	**CASTLE**	***Injected***	***Partial***	***Refined***	**CASTLE**	***Injected***	**CASTLE**	***Injected***	**CASTLE**	***Injected***
100	0.67 (0.03)	0.69 (0.04)	0.66 (0.02)	0.69 (0.04)	0.75 (0.02)	0.74 (0.04)	7.05 (12.81)	2.94 (2.63)	112.04 (91.06)	86.17 (13.75)
500	0.72 (0.04)	0.74 (0.02)	0.71 (0.02)	0.74 (0.02)	0.79 (0.01)	0.78 (0.01)	2.33 (1.39)	2.25 (1.07)	21.95 (6.84)	20.45 (5.12)
1000	0.75 (0.03)	0.76 (0.03)	0.74 (0.03)	0.76 (0.02)	0.78 (0.01)	0.78 (0.01)	2.96 (4.12)	1.68 (1.14)	NA	NA
2000	0.74 (0.03)	0.77 (0.01)	0.76 (0.03)	0.77 (0.02)	0.79 (0.01)	0.78 (0.01)	3.86 (3.68)	1.71 (0.57)	NA	NA
5000	0.75 (0.03)	0.79 (0.03)	0.76 (0.02)	0.79 (0.03)	0.79 (0.01)	0.79 (0.01)	4.91 (7.41)	1.51 (0.62)	NA	NA
10000	0.75 (0.02)	0.85 (0.01)	0.76 (0.02)	0.85 (0.01)	0.80 (0.01)	0.79 (0.01)	1.74 (1.70)	1.16 (0.31)	NA	NA
20000	0.76 (0.02)	0.86 (0.01)	0.77 (0.02)	0.86 (0.01)	NA	NA	0.66 (0.08)	1.02 (0.35)	NA	NA

Table 4: t-statistic (p-value) comparing each column of Table 3 against the respective CASTLE baseline for each dataset and sample size.

	Adult			HELOC	California	Boston
Data	***Injected***	***Partial***	***Refined***	***Injected***	***Injected***	***Injected***
100	2.000 (0.051)	1.387 (0.172)	2.000 (0.051)	1.118 (0.269)	1.571 (0.123)	1.405 (0.167)
500	2.236 (0.030)	1.118 (0.269)	2.236 (0.03)	3.536 (0.001)	0.228 (0.821)	0.878 (0.384)
1000	1.179 (0.224)	1.179 (0.244)	1.387 (0.172)	0.000 (1.000)	1.497 (0.141)	NA
2000	4.743 (0.000)	2.357 (0.023)	4.160 (0.000)	3.536 (0.001)	2.887 (0.006)	NA
5000	4.714 (0.000)	1.387 (0.172)	4.714 (0.000)	0.000 (1.000)	2.286 (0.027)	NA
10000	22.361 (0.000)	1.768 (0.083)	22.361 (0.000)	3.536 (0.001)	1.678 (0.100)	NA
20000	22.361 (0.000)	1.768 (0.083)	22.361 (0.000)	NA	5.014 (0.000)	NA