Neuron Dependency Graphs: A Causal Abstraction of Neural Networks

Yaojie Hu [1] **Jin Tian** [1]

Abstract

We discover that neural networks exhibit approximate logical dependencies among neurons, and we introduce Neuron Dependency Graphs (NDG) that extract and present them as directed graphs. In an NDG, each node corresponds to the boolean activation value of a neuron, and each edge models an approximate logical implication from one node to another. We show that the logical dependencies extracted from the training dataset generalize well to the test set. In addition to providing symbolic explanations to the neural network's internal structure, NDGs can represent a Structural Causal Model. We empirically show that an NDG is a causal abstraction of the corresponding neural network that "unfolds" the same way under causal interventions using the theory by Geiger et al. (2021a). Code is available at https://github.com/phimachine/ndg.

1. Introduction

Neural networks are "black-box" statistical models that do not provide explanations (Benítez et al., 1997; Samek et al., 2017) despite ability to accurately model ambiguous patterns (LeCun et al., 2015; Alom et al., 2019). Deep learning is a different AI paradigm compared to the traditional symbolic AI (Haugeland, 1989), which may be the fundamental cause to its "black-box" nature. Neural networks can consist of billions of low-level parameters that are interdependent as an entangled system, while meaningful concepts are usually high-level, disentangled, causal factors (Bengio, 2017). Traditional symbolic AI may not have exact equations to define ambiguous patterns such as "dog" or computations such as "annotate an image", but they do enjoy better explainability due to recursive computation of well-understood rules on interpretable states of information. Unfortunately, traditional symbolic tools such as first order logic are not central to the analysis and explanation of neural networks, because neural networks are considered not intrinsically symbolic or logical by default (Rudin, 2019).

We introduce Neuron Dependency Graphs (NDGs), a tool to extract and present approximate logical dependencies between pairs of neurons in a selected layer of a neural network. Once extracted, an NDG represents an internal logical structure of the neural network. By inspecting the logical structure, we discover a new general phenomenon that, for trained models, approximate logical dependencies exist commonly for a wide family of neural network architectures, datasets, and selected layers, even between two independently trained models. As a post-hoc explanation method (Das & Rad, 2020; Marcinkevičs & Vogt, 2020), NDGs do not require any architectural changes to the neural network explained. The graph is discovered from data and requires no a priori expert modeling.

Although an NDG is extracted from a neural network, it performs computation based on logical deduction, as opposed to tensor algebra of neural networks. To confirm that the extracted NDG is an accurate explanation of the internal structure of the neural network, we turn to the counterfactual theories of causation (Pearl, 2009; 2001; Morgan & Winship, 2015) and the theories of causal abstraction (Geiger et al., 2021a; Beckers & Halpern, 2019). The key of causal abstraction is to identify a function mapping between the two causal processes that ties the two together in terms of variables' values and intervention results. We show that an NDG can represent a causal abstraction of the neural network, which assures the faithfulness of the explanation even when counterfactual interventions are encountered.

To summarize, the main contributions of the paper are:

1. We introduce Neuron Dependency Graph as a new post-hoc explainable AI (XAI) method that provides logic-based, faithful explanations about the internal logical structure of a neural network.
2. We use NDGs to reveal a new general phenomenon that approximate logical dependency exists among neurons and can generalize to unseen data as a meaningful statistical model. The existence of neuron dependency occurs commonly for different datasets and architectures, even between independently trained models.

[1]Department of Computer Science, Iowa State University, United States. Correspondence to: Yaojie Hu <jhu@iastate.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

3. We show an NDG is a causal abstraction of the neural network. Specifically, we prove theoretically that an NDG defines a Structural Causal Model (SCM) that represents a causal abstraction of the neural network under an alignment condition (Thm. 3.11). Experimental results empirically support the alignment condition.

2. Related Work

Explainable artificial intelligence (XAI) Explainable artificial intelligence aims to make artificial intelligence models understandable to human beings (Adadi & Berrada, 2018; Das & Rad, 2020). Methods to extract explanations include gradients (Simonyan et al., 2014; Sundararajan et al., 2017; Selvaraju et al., 2017) and perturbations (Ribeiro et al., 2016; Lundberg & Lee, 2017; Shrikumar et al., 2017). Explanations may be in the form of feature importance values (Lundberg & Lee, 2017), heatmap (Simonyan et al., 2014; Selvaraju et al., 2017), cropped images (Ribeiro et al., 2016), or prototype examples (Ming et al., 2019). In terms of limitations, the correctness of these explanations still require human judgments as there is no objective metric such as accuracy for explanations, and the explanations are for human interpretation, not downstream computation.

NDGs extract explanations from contingency tables and present them in the form of logical formula. Accuracy is defined and measures the correctness of neuron dependencies as a statistical model, an objective metric to assess model qualities as opposed to, for example, potentially misleading visual assessments (Adebayo et al., 2018). Neuron dependencies can be used for downstream deductive reasoning. Related to our work as logic-based XAI methods, local explanations via necessity and sufficiency (Watson et al., 2021) produce sets of explanatory factors based on necessity and sufficiency. Logic explained networks (Ciravegna et al., 2021) produce first order logic explanations based on human understandable predicates provided as inputs. Compositional logical concepts (Mu & Andreas, 2020) may be approximated given overlapping pixels. ExplaiNN (Fischer et al., 2021) finds rules about subsets of binarized activations between hidden layers using the Minimum Description Lenght (MDL) principle.

Neural-symbolic reasoning NS reasoning aims to integrate deep learning and symbolic reasoning (d'Avila Garcez et al., 2019). It can be carried through many-valued logic (Serafini & d'Avila Garcez, 2016) or real-valued logic (Riegel et al., 2020). First-order or propositional logic formula can be learned to combine symbolic reasoning and fuzzy pattern cognition (Dong et al., 2019; Shi et al., 2019).

Causal Abstractions Causal abstraction methods often require human expert knowledge to establish the high-level causal model used for explanation (Geiger et al., 2021a;b). Related, counterfactuals can be used for data augmentation (Liu et al., 2021) and causal structures can be extracted from language models (Friedman et al., 2021).

3. Methods

For notations, an uppercase letter such as A denotes a variable (node). A lower case letter such as a denotes a value, e.g. $A = a$. A bold letter such as $\mathbf{A}$ or $\mathbf{a}$ denotes a vector of variables or values.

3.1. Neuron Dependency Graphs

We aim to investigate logical dependencies between neurons in a neural network given a dataset of input-label $(\mathbf{x}, y)$ sampled from a k-class classification task distribution $P(\mathbf{x}, y)$, with $y \in \mathbb{N}_k$, and $\hat{y} = f(\mathbf{x}) \in \mathbb{N}_k, \mathbb{N}_i = \{1, 2, ...i\}$. Input $\mathbf{x}$ is a vector value. We use $\mathbf{n} = \mathbf{N}(\mathbf{x}) = f_{\mathbf{N}}(\mathbf{x})$ to denote the value of neurons $\mathbf{N}$ computed by the neural network f given input $\mathbf{x}$ before the activation function.

Definition 3.1 (Strict Neuron Dependency)**.** Let D be the support of $P(\mathbf{x}, y)$. Let ρ be an activation function with Boolean values for two neurons N, M with real values. We say there exists a strict neuron dependency $\rho(N) \rightarrow_s \rho(M)$ if $\forall(\mathbf{x}, y) \in D : \rho(N(\mathbf{x})) \implies \rho(M(\mathbf{x}))$.

We also consider strict neuron dependency on the negation of activation function ρ denoted as $\overline{\rho}$, where $\overline{\rho}(N(\mathbf{x})) = \neg\rho(N(\mathbf{x}))$. For example, we say $\overline{\rho}(N) \rightarrow_s \rho(M)$ if $\forall(\mathbf{x}, y) \in D : \neg\rho(N(\mathbf{x})) \implies \rho(M(\mathbf{x}))$. Note that $\overline{\overline{\rho}} = \rho$.

Any neuron N can represent an attribute of input $\mathbf{x}$, and we can select $\{\mathbf{x} \mid (\mathbf{x}, y) \in D, \rho(N(\mathbf{x})) = True\}$ to be a natural cluster of all inputs that share the same attribute. This is related to the derivation operator of formal concept analysis, and we can further define a formal context lattice (Ganter & Wille, 2012; Birkhoff, 1940) based on strict neuron dependency, although we will not explore this line of theory further in this paper.

Note that strict neuron dependency may not hold in practice. The definition is strict for two reasons. 1) Typical activation functions such as ReLU and sigmoid have ranged real outputs instead of binary outputs. 2) Due to sampling error and inaccuracy, the neurons may not model strict logical dependency perfectly.

We relax the definition to use it in practice.

Definition 3.2 (Neuron Dependency)**.** Let binarized neuron $N' = \mathbb{1}(\phi(N))$, where $\mathbb{1}(c)$ is a binarization function $\mathbb{1} : \mathbb{R} \rightarrow \{\textit{True, False}\}$, and ϕ is an activation function $\phi : \mathbb{R} \rightarrow \mathbb{R}$. For binarized neurons N', M' corresponding to neurons N, M, we say there is a neuron dependency $N' \rightarrow M'$

given a threshold parameter α with $\alpha > 1$ if

$$l(N'; M') \equiv \frac{P(N' \wedge \neg M')}{P(N')P(\neg M')} \leq \frac{1}{\alpha}. \tag{1}$$

Conceptually, $N' \rightarrow_s M'$ when $N' \wedge \neg M'$ is *False* for the entire support. We relax the strictness by requiring $P(N' \wedge \neg M')$ to be close to zero in comparison with $P(N')P(\neg M')$, which is the probability of $N' \wedge \neg M'$ if the events N' and $\neg M'$ are probabilistically independent.

Similar to strict neuron dependency, we will consider neuron dependency on negated binarized neurons denoted as $\overline{N}'$. For example, we say $\overline{N}' \rightarrow \overline{M}'$ if $l(\neg N'; \neg M') \leq 1/\alpha$ by Eq. (1). For common activation functions ϕ, the binarization function $\mathbb{1}$ may be chosen based on their natural thresholds. For sigmoid : $\mathbb{1}(c) =$ *True* if $c > 0.5$; *False* otherwise. For ReLU, $\mathbb{1}(c) =$ *True* if $c > 0$; *False* otherwise.

Neuron dependency has some desirable properties, making it a good relaxation for logical implication. First of all, neuron dependency approximates strict neuron dependency and therefore can be considered as a logical rule as shown in the following.

Proposition 3.3 (Neuron dependency is a generalization of strict neuron dependency)**.** *Let function $\mathbb{1} \circ \phi$ be the binary activation function used in Def. 3.1. Strict neuron dependency is logically equivalent to neuron dependency when $\alpha = \infty$.*

Proof. When $\alpha = \infty$, we have $N' \rightarrow M' \iff P(N' \wedge \neg M') = 0 \iff N' \rightarrow_s M'$. □

Given the proposition, we set $1 < \alpha < \infty$ to be a large enough value in practice and neuron dependency $N' \rightarrow M'$ will become an approximation of the strict neuron dependency $N' \rightarrow_s M'$.

Notably, by Eq. (1), we have $N' \rightarrow M'$ if and only if $\overline{M}' \rightarrow \overline{N}'$, which satisfies the contraposition rule in logic. Therefore, neuron dependencies always occur in contraposition pairs. In addition, Eq. (1) makes sure that if two binarized neurons N', M' are independent variables probabilistically, then there is no neuron dependency $N' \rightarrow M'$. We also note that $N' \rightarrow M'$ does not imply or rule out the converse $M' \rightarrow N'$ given the definition in Eq. (1), which is consistent with logic. Lastly, neuron dependency $\rightarrow$ is reflexive. However, unlike strict neuron dependency, neuron dependency is not transitive: any method to approximate logical implication is likely not transitive when recursively applied due to a loss of information.

We can see every neuron dependency as a small distributed statistical model. We will show empirically that neuron dependency relationships commonly exist between neurons in a well-trained neural network. We represent these relationships with a directed graph over binarized neurons defined as follows.

Definition 3.4 (Neuron Dependency Graph (NDG))**.** Let $\mathbf{N}$ be a selected hidden layer of neurons $\mathbf{N} = \{N_i \mid i \in \mathbb{N}_h\}$. We define a Neuron Dependency Graph $NDG = (\mathbf{V}, \mathbf{E})$ to be a directed graph over nodes in $\mathbf{V} = \mathbf{B} \cup \mathbf{O}$ and edges $\mathbf{E}$, where

$$\mathbf{B} = \{N'_j \mid N_j \in \mathbf{N}\} \cup \{\overline{N}'_j \mid N_j \in \mathbf{N}\} \tag{2}$$

$$\mathbf{O} = \{O_i \mid i \in \mathbb{N}_k\} \cup \{\overline{O}_i \mid i \in \mathbb{N}_k\} \tag{3}$$

$$\mathbf{E} = \{(U, V) \mid U \rightarrow V, (U, V) \in \mathbf{V} \times \mathbf{V}\} \tag{4}$$

where nodes in $\mathbf{B}$ represent binarized neurons $N'_j = \mathbb{1}(\phi(N_j))$ and their negation, and nodes in $\mathbf{O}$ represent binarized output neurons and their negation with values $o_i =$ *True* if $f(\mathbf{x}) = i$ and $o_i =$ *False* otherwise.

Construction of an NDG from a dataset. To construct an NDG from a dataset T with input-label tuples $(\mathbf{x}, y) \in T$, we use the sample proportion $\hat{P}$ as an estimate of P for every $(U, V) \in \mathbf{V} \times \mathbf{V}$. To address the situation when there are not enough samples to reliably estimate $\hat{P}$ to establish or reject an edge, we add condition $\frac{|T|\hat{P}(U)\hat{P}(\neg V)}{\alpha} > 1$ to the construction of edges (see Appendix B for an explanation):

$$\mathbf{E} = \{(U, V) \mid \frac{\hat{P}(U \wedge \neg V)}{\hat{P}(U)\hat{P}(\neg V)} \leq \frac{1}{\alpha}, \frac{|T|\hat{P}(U)\hat{P}(\neg V)}{\alpha} > 1, (U, V) \in \mathbf{V} \times \mathbf{V}\}. \tag{5}$$

The complexity of neuron dependency graph construction is $\mathcal{O}(|T||\mathbf{N}|^2)$. When iterating over the dataset to construct the graph, the graph will converge after a large number (e.g. 10^5) of randomly sampled data points and early stopping can be applied, making the complexity effectively $\mathcal{O}(|\mathbf{N}|^2)$.

Our experimental results (see Sec. 4) show that neuron dependencies commonly exist for various types of datasets, model architectures, and selected layers. In addition to providing descriptions, neuron dependencies can be treated as strict logical rules to perform deductive reasoning. If we intervene on the neural network based on the deductions, the neural network will have aligned predictions, demonstrating that neuron dependency graphs are accurate representations of the internal logical structure of neural networks. Formally, we show a NDG can represent a causal abstraction of the neural network under an alignment condition (Thm. 3.11), as discussed next.

3.2. Causal Abstractions

Neural networks can be interpreted as a causal model (Chattopadhyay et al., 2019; Geiger et al., 2021a). To further show NDGs provide a symbolic representation of the internal structure of neural networks, we will first provide an

interpretation of NDGs as a structural causal model, then establish a condition under which a NDG is a causal abstraction of its corresponding neural network which will be empirically validated.

Definition 3.5 (Structural Causal Model (SCM) (Pearl, 2009))**.** A Structural Causal Model (SCM) is a 4-tuple $\langle \mathbf{V}, \mathbf{U}, \mathcal{F}, P \rangle$, where $\mathbf{V} = \{V_i \mid i \in \mathbb{N}_n\}$ are endogenous variables and $\mathbf{U} = \{U_i \mid i \in \mathbb{N}_m\}$ are exogenous variables. Structural equations $\mathcal{F} = \{f_i \mid i \in \mathbb{N}_n\}$ are functions that determine $\mathbf{V}$ with $v_i = f_i(\mathbf{pa}_i, \mathbf{u}_i)$, where $\mathbf{Pa}_i \subseteq \mathbf{V}$ and $\mathbf{U}_i \subseteq \mathbf{U}$. $P(\mathbf{u})$ is a distribution over $\mathbf{u}$.

An intervention $\mathbf{V}^* \leftarrow \mathbf{v}^*$ results in a new causal model with changed structural equations, where $\mathbf{V}^* \subseteq \mathbf{V}$. For every variable $V_i \notin \mathbf{V}^*$, $V_i = f_i$, and for every variable $V_j \in \mathbf{V}^*$, the structural equation is changed to $V_j = v_j^*$ for corresponding $v_j^* \in \mathbf{v}^*$.

Causal abstraction (τ-abstraction) is an ordered relationship that maps (SCM_L, I_L), a low-level causal model SCM_L and allowed interventions I_L, to a high-level (SCM_H, I_H). Causal abstraction occurs when the two causal processes "unfold" the same way (Geiger et al., 2021a). Every assignment to the low-level SCM_L variables has a corresponding assignment to the high-level SCM_H variables given τ. Every low-level intervention i_L has a corresponding high-level intervention i_H given $i_H = \omega_\tau(i_L)$. After applying the interventions i_L and $\omega_\tau(i_L)$ respectively, the counterfactuals (consequences of the interventions) also need to be "aligned" given correspondence τ, formally $\tau(i_L(SCM_L)) = \omega_\tau(i_L)(SCM_H)$.

Definition 3.6 (τ-abstraction (Beckers & Halpern, 2019; Geiger et al., 2021a))**.** Let I_L to be a set of interventions $\mathbf{U}_L \leftarrow \mathbf{u}_L$ on the low-level $SCM_L = \langle \mathbf{V}_L, \mathbf{U}_L, \mathcal{F}_L, P_L \rangle$. Similarly, let I_H be interventions on high-level $SCM_H = \langle \mathbf{V}_H, \mathbf{U}_H, \mathcal{F}_H, P_H \rangle$. Let τ be a partial function $\tau : \mathcal{D}(\mathbf{V}_L) \rightarrow \mathcal{D}(\mathbf{V}_H)$ where $\mathcal{D}$ maps a variable to its possible values. Let $\omega_\tau : I_L \rightarrow I_H$ be

$$\omega_\tau(\mathbf{V}_L^* \leftarrow \mathbf{v}_L^*) = \mathbf{V}_H^* \leftarrow \tau(\mathbf{v}_L^*) \tag{6}$$

where $\mathbf{V}_L^* \subseteq \mathbf{V}_L$, $\mathbf{V}_H^* \subseteq \mathbf{V}_H$, $\mathbf{v}_L^* \in \mathcal{D}(\mathbf{V}_L^*)$. We say (SCM_H, I_H) is a τ-abstraction of (SCM_L, I_L) if τ and ω_τ are surjective and

$$\forall i_L \in I_L : \tau(i_L(SCM_L)) = \omega_\tau(i_L)(SCM_H). \tag{7}$$

The surjective conditions ensure that SCM_L is more detailed than SCM_H (Beckers & Halpern, 2019).

Following (Chattopadhyay et al., 2019; Geiger et al., 2021a), we define an SCM given a neural network.

Definition 3.7 (The Structural Causal Model representing a neural network)**.** Given a neural network f with a layer of neurons $\mathbf{N}$ and input $\mathbf{X}$, let $\mathbf{O}$ be the set of binarized output neurons and their negation defined in Def. 3.4. We define an SCM $SCM_{NN} = \langle \mathbf{N} \cup \mathbf{X} \cup \mathbf{O}, \emptyset, \mathcal{F}, \emptyset \rangle$ as follows. Every neuron variable $N \in \mathbf{N}$ has parents $\mathbf{Pa}_N = \mathbf{X}$. Every output variable $O \in \mathbf{O}$ has parents $\mathbf{Pa}_O = \mathbf{N}$. Input $\mathbf{X}$ is a root variable. For neurons $\mathbf{N}$, the structural equation $\mathcal{F}_\mathbf{N}$ is given by $\mathbf{n} = f_\mathbf{N}(\mathbf{x})$, that is the value of neurons $\mathbf{N}$ computed by the neural network f given input $\mathbf{x}$. For outputs $\mathbf{O}$, the structural equation $\mathcal{F}_\mathbf{O}$ is given by $\mathbf{o} = f_\mathbf{O}(\mathbf{n})$, that is the network output values computed by the neural network given the neurons' values $\mathbf{n}$. Exogenous variables and their probability distributions are omitted because all variables are deterministic given any input.

We note that SCM_{NN} in Def. 3.7 is a faithful representation of the underlying neural network f except that SCM_{NN} selects one layer of neurons in f for analysis, and the layer has the original real-valued outputs $\mathbf{n}$.

Definition 3.8 (The Structural Causal Model representing a neuron dependency graph)**.** Given a Neuron Dependency Graph $NDG = (\mathbf{V}, \mathbf{E}) = (\mathbf{B} \cup \mathbf{O}, \mathbf{E})$ for neural network f with input $\mathbf{X}$, we define a SCM $SCM_{NDG} = \langle \mathbf{B} \cup \mathbf{O} \cup \mathbf{X}, \emptyset, \mathcal{F}, \emptyset \rangle$ with $\mathbf{X}$ being a root variable as follows. For $U \in \mathbf{B} \cup \{\overline{O}_j \mid j \in \mathbb{N}_k\}$, the structural equation $\mathcal{F}_U$ is

$$\begin{aligned} u =& \mathcal{F}_U\big(pa_{NDG}(U) \cup \mathbf{x} \cup \{\overline{u}\}, \emptyset\big) \\ =& \bigvee_{p \in pa_{NDG}(U)} p \vee \big(f'_U(\mathbf{x}) \wedge \neg\overline{u}\big) \end{aligned} \tag{8}$$

For $O_i \in \{O_j \mid j \in \mathbb{N}_k\}$, the structural equation $\mathcal{F}_{O_i}$ is

$$\begin{aligned} o_i =& \mathcal{F}_{O_i}\big(pa_{NDG}(O_i) \cup \mathbf{x} \cup \{\overline{o}_j \mid j \in \mathbb{N}_k\}, \emptyset\big) \\ =& \bigvee_{p \in pa_{NDG}(O_i)} p \vee \big(f'_{O_i}(\mathbf{x}) \wedge \neg\overline{o}_i\big) \vee \big(\bigwedge_{j \neq i, j \in \mathbb{N}_k} \overline{o}_j\big) \end{aligned} \tag{9}$$

$pa_{NDG}(V) \subseteq \mathbf{b} \cup \mathbf{o}$ is value of the parent nodes of V on NDG, and $f'_V(\mathbf{x})$ is value of V computed by $\mathbb{1} \circ \phi \circ f$ or its negation given input $\mathbf{X} = \mathbf{x}$. Exogenous variables and their probability distributions are omitted due to determinism.

Every clause in the structural causal equations of SCM_{NDG} applies logical deduction by treating neuron dependencies as strict neuron dependencies (logical implication rules). The clause $\bigvee_{p \in pa_{NDG}(U)}$ applies modus ponens such that a node U is true if any NDG parent is true. The clause $f'_U(\mathbf{x}) \wedge \neg\overline{u}$ allows a node U to be true when none of its parents are, if U is predicted true by the neural network. Without the term $\wedge\neg\overline{u}$ in $f'_U(\mathbf{x}) \wedge \neg\overline{u}$, variable U will be true if $f'_U(\mathbf{x})$ is, which does not allow causal effect to propagate and change the value of U, so we incorporate it. Lastly, the clause $\bigwedge_{j \neq i, j \in \mathbb{N}_k} \overline{o}_j$ derives from the property that all classification classes are mutually exclusive, such that if all other classes j are ruled out, class i can be deduced true. Structural equations $\mathcal{F}$ make use of the internal logical structures presented by NDG, and as we shall show in Thm. 3.11,

together they can prove that corresponding interventions on SCM_{NDG} and SCM_{NN} cause corresponding counterfactual effects.

We note that SCM_{NDG} using binarized neurons does not disqualify it being a causal abstraction of neural networks that have continuous activations. In fact, an abstraction must lose information and a higher-level representation often has a smaller domain. Also note that SCM_{NDG} is a cyclic SCM.

We define the correspondence τ from SCM_{NN} to SCM_{NDG} based on how the binarized neuron nodes in the NDG are constructed from neuron outputs with $\mathbf{n}' = \mathbb{1}(\phi(\mathbf{n}))$.

Definition 3.9 (τ-mapping from SCM_{NN} to SCM_{NDG})**.** Given a $SCM_{NN} = \langle \mathbf{N} \cup \mathbf{X} \cup \mathbf{O}, \emptyset, \mathcal{F}, \emptyset \rangle$ with value $(\mathbf{n}, \mathbf{x}, \mathbf{o})$, define τ that maps SCM_{NN} to $SCM_{NDG} = \langle \mathbf{B} \cup \mathbf{X} \cup \mathbf{O}, \emptyset, \mathcal{F}, \emptyset \rangle$

$$\begin{aligned}(\mathbf{B}, \mathbf{X}_{NDG}, \mathbf{O}_{NDG}) &= \tau(\mathbf{n}, \mathbf{x}_{NN}, \mathbf{o}_{NN}) \\ &= \big((\mathbb{1}(\phi(\mathbf{n})), \neg\mathbb{1}(\phi(\mathbf{n}))), \mathbf{x}_{NN}, \mathbf{o}_{NN}\big)\end{aligned} \tag{10}$$

We want to design sets of interventions I_{NN} and I_{NDG} such that (SCM_{NN}, I_{NN}) is a causal abstraction of (SCM_{NDG}, I_{NDG}), and there exists an intervention $i_{NN,c} \in I_{NN}$ for every counterfactual class $c \in \mathbb{N}_k$ such that $i_{NN,c}$ causes the original neural network's prediction to change to c. In order for the intervention $i_{NN,c}$ on the neural network to emulate the activation pattern of an in-distribution input of class c, we want to set all descendant nodes of O_c on the NDG to be True, as they represent the necessary conditions of class c. Moreover, we want to set the ancestors of O_c on the NDG to be True, as they represent the sufficient conditions and result in a stronger intervention. If such an intervention i_{NN} exists and maps to $\omega_\tau(i_{NN})$, the descendants of the intervention $\omega_\tau(i_{NN})$ on the SCM_{NDG} will be True in the counterfactual given the structural equations. However, their corresponding neurons do not change value on SCM_{NN}, because no neuron is a parent of another neuron in the structural equation, causing node values to not correspond given Eq. (7). The binarized neurons directly intervened on by i_{NDG} must be the only binarized neurons in $\mathbf{B}$ that will change values as a result of the intervention.

On the NDG, define $desc(\mathbf{W})$ as the union of $\mathbf{W}$ and their descendant variables for any $\mathbf{W} \subseteq \mathbf{V}$. Define $ance(\mathbf{W})$ as the union of $\mathbf{W}$ and their ancestors. Define $\mathbf{Q}(\mathbf{W}) = \mathbf{B} \cap \big(desc(ance(\mathbf{W}))\big)$.

Definition 3.10 (Allowed interventions for NDGs and corresponding neural networks)**.** We select the set of allowed interventions I_{NDG} on SCM_{NDG}

$$I_{NDG} = \{i_{NDG,c} \mid c \in \mathbb{N}_k\} \tag{11}$$

$$i_{NDG,c} = \{Q \leftarrow \textit{True}, \overline{Q} \leftarrow \textit{False} \,|\, Q \in \mathbf{Q}(\{O_c\})\} \tag{12}$$

where O_c is an output node.

Define the set I_{NN} of allowed interventions on SCM_{NN}

$$I_{NN} = \{i_{NN,c} \,|\, c \in \mathbb{N}_k\} \tag{13}$$

$$\begin{aligned}i_{NN,c} = &\{N_j \leftarrow T_\phi \,|\, N'_j \in \mathbf{Q}(\{O_c\})\} \cup \\ &\{N_j \leftarrow F_\phi \,|\, \overline{N}'_j \in \mathbf{Q}(\{O_c\})\}\end{aligned} \tag{14}$$

where N'_j and $\overline{N}'_j$ are binarized neuron nodes for neuron $N_j \in \mathbf{N}$. T_ϕ and F_ϕ are two numbers such that $\mathbb{1}(\phi(T_\phi)) = \textit{True}$ and $\mathbb{1}(\phi(F_\phi)) = \textit{False}$.

We check if after intervention with i_{NN} and $\omega_\tau(i_{NN})$, the counterfactual values of SCM_{NN} and SCM_{NDG} correspond by Eq. (7). We know that the intervened nodes in both SCMs correspond by the construction of the intervention. Assume that every neuron dependency is satisfied as strict neuron dependency before the intervention, we prove that on SCM_{NDG}, only output variables $\mathbf{O}$ change values besides those directly intervened due to closure property of *desc* operation. On SCM_{NN}, only the neural network output changes besides those directly intervened. The unchanged nodes satisfy mapping τ given how binarized neuron nodes are constructed. The input variables are equal. Therefore, the only condition needed for Eq. (7) to be true is that the output nodes of both SCMs to correspond by τ.

Theorem 3.11 (The alignment condition for a Neuron Dependency Graph to be τ-abstraction of its Neural Network)**.** *Assume that all neuron dependencies $\rightarrow$ are strict $\rightarrow_s$. (SCM_{NDG}, I_{NDG}) is a τ-abstraction of (SCM_{NN}, I_{NN}) if the following* alignment condition *holds:*

$$\forall \mathbf{x} \in D, c \in \mathbb{N}_k : c = f^{i_{NN,c}}(\mathbf{x}), \mathbf{o}_c = \mathbf{o}_{NDG}^{i_{NDG},c}(\mathbf{x}) \tag{15}$$

where D is the support of the distribution $P(\mathbf{x})$, $f^{i_{NN,c}}(\mathbf{x})$ is the neural network output after intervention $i_{NN,c}$, $\mathbf{o}_{NDG}^{i_{NDG},c}(\mathbf{x})$ is the value of SCM_{NDG} nodes $\mathbf{O}_{NDG}$ after intervention $i_{NDG,c}$, and $\mathbf{o}_c = \{O_c = \textit{True}, \overline{O}_c = \textit{False}\} \cup \{O_d = \textit{False}, \overline{O}_d = \textit{True} \mid d \neq c, d \in \mathbb{N}_k\}$.

The alignment condition states that, given our prior definitions, if neuron dependencies are strict and if the values of output nodes of SCM_{NN} and SCM_{NDG} always correspond by τ after interventions $i_{NN,c}$ and $i_{NDG,c}$, then the values of the rest of the nodes will always correspond as well, and causal abstraction can be proven theoretically. A proof is in Appendix A.

Alternative formulations of causal abstraction may be derived based on the alignment condition, but they may not hold up in practice (see examples in Appendix H). The challenge is: Will the interventions on neural networks cause the expected outputs in practice? That is, will $c = f^{i_{NN,c}}(\mathbf{x})$? Even if NDGs are accurate representations of neural networks' internal logic, they may fail to model the causal

effect in practice due to noise in the dataset, inaccuracies of the neural network, and the threshold-based fuzziness of neuron dependency condition Eq. (1).

To address the situation, we use interchange intervention (Geiger et al., 2021a) to measure if there is a high percentage of dataset where $c = f^{i_{NN,c}}(\mathbf{x})$ holds. For an input-label tuple $(\mathbf{x}, y)$ such that the model predicts correctly $f(\mathbf{x}) = y$, we select a counterfactual class $c \neq y$, find the intervention $i_{NN,c}$ in Eq. (14), apply the intervention $i_{NN,c}$ to the original neuron activation $\mathbf{n}$, and see if the neural network changes prediction from y to c. The interchange intervention procedure is given in Algorithm 1, where we define $\phi^{-1}(\textit{True}) = T_\phi$, $\phi^{-1}(\textit{False}) = F_\phi$. As a preview of the results in Section 4.5, we find that a high percentage of data points are aligned, which empirically validates NDGs as a causal abstraction of neural networks.

Algorithm 1 Interchange intervention with a NDG

Input: A neural network f. An input-output tuple (x, y) where $f(x) = y$, from the dataset D. The NDG for f and D representing SCM_{NDG}.
Output: Whether $c = f^{i_{NN,c}}(\mathbf{x})$ holds
Compute $\mathbf{n} = \mathbf{N}(x)$, let $\mathbf{m} = \mathbf{n}$
Uniformly sample an alternative class $c \neq y$
Find $i_{NDG,c}$ given Eq. (12)
for $Q \leftarrow q \in i_{NDG,c}$ **do**
  if Q is N_i' **then**
    Set $m_i = \phi^{-1}(q)$
  end if
end for
return $c == f^{\mathbf{N} \leftarrow \mathbf{m}}(x)$

4. Experiments and Results

We extract neuron dependency graphs on a diverse set of datasets and architectures to demonstrate the generality of our method. Table 1 lists the datasets and architectures (LeCun et al., 1998; Socher et al., 2013; Conneau et al., 2017; Reimers & Gurevych, 2019; Welinder et al., 2010; Lu et al., 2021; Sanh et al., 2019; Dosovitskiy et al., 2021; He et al., 2021; Zhou et al., 2019; Feng et al., 2020; Liu et al., 2019; Devlin et al., 2018). For the datasets with only the training set and the test set, we leave 10% of the training set for validation.

Because last layers in the neural network tend to capture high-level abstract concepts (Zeiler et al., 2011; Bengio et al., 2013), and because we want the model to change prediction if interchange intervention is applied to the layer's output, we choose the neurons from one of the last fully connected feedforward layers (closer to the output) that is a graph cut of the neural network data flow. If not for interchange intervention, the choice of layer is not restrictive because neuron dependencies exist commonly, as shown in experiments. Threshold α is manually selected to improve interchange intervention accuracy.

Table 1. Experiment settings for each dataset and architecture. The training set size is reported. Threshold parameter α is used in Eq. (1) to extract the neuron dependency graphs. (NL: natural language. PL: programming language.)

dataset	size	task	classes	input	architecture	α
MNIST	54000	image classif.	10	image	CNN	100
MNIST even	54000	image classif.	2	image	CNN	100
SST2	6920	sentiment classif.	2	NL	DistilBERT	100
AllNLI	847863	NL inference	3	NL	DistilRoBERTa	20
CUB200	5395	image classif.	200	image	TransFG (ViT)	3
Devign	21854	defect detection	2	PL	CodeBERT	20

4.1. Empirical qualities of Neuron Dependency Graphs

Before confirming the correctness of NDGs as a distributed statistical model, we first gain intuitions about the empirical qualities of the graph. We extract NDGs on all datasets. We present a plot of the NDG extracted on MNIST dataset in Figure 1. NDG plots for other datasets are included in Appendix J. Graph statistics of NDGs are listed in Table 2.

Neuron dependencies exist commonly. We observe that neuron dependencies exist commonly. No nodes are isolated for three of the datasets. When isolated nodes exist, they are at most 5% of the total nodes (Devign). Neurons may have multiple neuron dependencies with one another, as the average degree ranges from 6.2 (MNIST) to 307.2 (Devign). For all datasets, most neurons have some undirected path with one another when binarized as NDG nodes, suggesting that most of the nodes are semantically related. In Appendix C, we extract NDGs from other layers such as a feedforward layer in an attention block, and neuron dependencies exist commonly, too.

Most Neurons are (approximately) necessary conditions of some predicted class for many-class classifications. For many-class classifications, we observe that most neurons (negation or not) are necessary conditions of some predicted class (not negated). Precisely, for most neurons N, there exists a predicted class T such that $T \rightarrow N'$ or $T \rightarrow \overline{N}'$. In Figure 1, nodes of the predicted classes are in red and are labeled with corresponding MNIST digits.

When there are many target classes to be predicted and neurons are approximately logically related to the classes, we observe that the neuron dependency graphs will be short and wide. This is because a neuron tends to have probability $1/2$ in order to maximize the expected entropy, and for many-class classification, a target class has probability lower than $1/2$ and, therefore, a neuron can only be a necessary condition of the target class if it is logically related to it.

Table 2. Statistics for neuron dependency graphs extracted on each dataset. Mutual neuron dependencies are removed for all plots and tables in this paper. An "equivalent" node has a mutual neuron dependency with another node. A "constant" node has a sample proportion greater than 99.99% or less than 0.01%. A "contradiction" occurs when both a node and its negation appear as descendants of another node. After removing mutual neuron dependencies, no NDG has cycles.

	neurons	nodes	edges	isolated	equivalent	constants	degree	height	contradictions	components
MNIST	32	84	524	0	0	0	6.2	4	0	1
MNIST even	32	68	686	2	4	1	10.1	8	0	4
SST2	768	1540	29132	48	1448	11	18.9	6	0	50
AllNLI	768	1542	202250	0	124	0	131.2	87	1	1
CUB200	768	1936	195998	0	0	0	101.2	3	1	1
Devign	768	1540	473058	74	1452	34	307.2	121	1	75

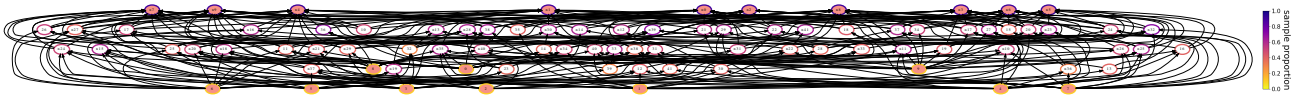


Figure 1. A plot of a neuron dependency graph extracted from the MNIST dataset with a convolutional neural network. For all NDG plots in this paper: Node border color indicates sample proportion; An output node O_i is filled in red and labeled $i-1$ (MNIST digits equal to labels); A binarized neuron N_i' is labeled $i+k-1$; Letter "n" marks negation (e.g. n0 is $\overline{O}_1$, n11 is $\overline{N}_2'$); Edges point upwards; Transitive reduction is performed; Isolated nodes and mutual neuron dependencies are removed.

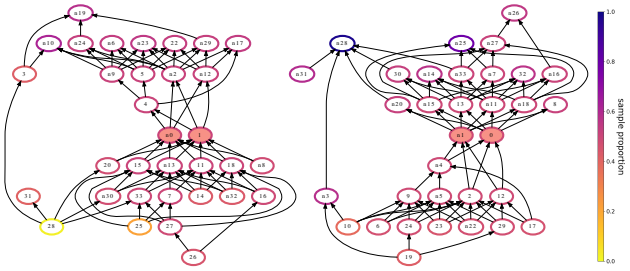


Figure 2. A plot of a Neuron Dependency Graph for MNIST dataset when the label is the parity of the digit for binary classification (MNIST even). The two disconnected components in the figure are contrapositive.

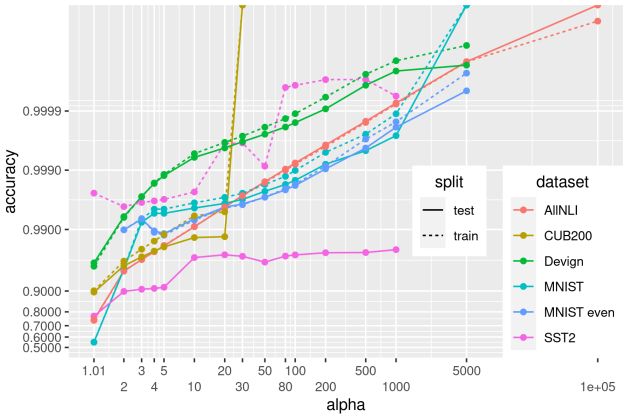


Figure 3. Average training and test accuracy for non-mutual-dependency edges versus various alpha thresholds. The x-axis is log transformed and the y-axis is logistically transformed.

Neurons evenly spread out for every class and cause the graph to be wide, and for dataset such as CUB200 with many targets, the graph can be very wide (Table 2).

To further examine the relationship between the shape of the graph and the number of classes, we train a model to predict the parity of digits on the MNIST dataset with the same input images. The graph statistics are listed in Table 2 and the plot is included in Figure 2. We do see that the shape of the graph changes as the number of classes change, despite the same input images. Specifically, the height of the graph increases and the nodes for the predicated class move to the middle of the graph in terms of depth. This result corroborates with our previous analysis.

Neuron dependencies exist between different layers in the same model. We may select multiple layers in the same model to construct a neuron dependency graph. We see that neuron dependencies exist between neurons in different layers, and the edges generalize well to unseen data. Details of this experiment are provided in Appendix C.

4.2. A neuron dependency is a meaningful statistical model that generalizes to unseen data

A neuron dependency graph can be seen as a collection of small, distributed statistical models represented by neu-

Table 3. The number of edges, average training and test accuracy for graph extracted from random models with real inputs and trained model with random inputs.

	original			random model			random input		
	edges	train	test	edges	train	test	edges	train	test
MNIST	524	99.90	99.85	0	-	-	22	99.94	99.89
MNIST even	686	99.83	99.82	0	-	-	316	99.95	99.95
SST2	29132	100.0	97.35	0	-	-	464024	99.92	99.92
AllNLI	202250	99.58	99.58	2686	99.82	48.04	149116	99.68	99.55
CUB200	195998	97.87	97.15	237286	96.43	50.15	748	95.76	91.80
Devign	473058	99.97	99.96	4200	99.64	53.01	26786	99.75	99.99

Table 4. The number of inter-model neuron dependency edges and the average accuracy when two models independently trained on the same dataset are given the same inputs, and when the same model is given two different inputs.

	different models same input			same model different inputs		
	edges	train acc	test acc	edges	train acc	test acc
MNIST	0	-	-	0	-	-
MNIST even	27	99.71	99.75	0	-	-
SST2	35354	100.00	92.29	0	-	-
AllNLI	138406	99.48	99.43	0	-	-
CUB200	0	-	-	0	-	-
Devign	75597	99.90	99.92	0	-	-

ron dependency edges of the graph. For every neuron dependency edge $A \to B$, we report if it holds true for the training set and the test set, hence we define accuracy to be $\hat{P}(A \wedge B)/\hat{P}(A)$. The average training and test accuracy versus alpha thresholds is plotted in Figure 3.

Figure 3 shows that training and test accuracy generally increases as α increases, suggesting that higher α extracts better edges in terms of accuracy. We see that the test accuracy closely tracks the training accuracy for all datasets except SST2. For SST2, the training set reaches 99.99% accuracy and the test accuracy reaches 95% for a subset of α values. We conclude that neuron dependency is a meaningful statistical model that can generalize to unseen data. In Appendix D, we plot the number of edges versus α. In Appendix E, we plot the outliers in terms of accuracy.

4.3. Neuron dependencies emerge from trained models

We investigate possible underlying cause for neuron dependencies to emerge from a pair of model and dataset. We extract NDGs by feeding random inputs to every model presented in Table 2. We also extract NDGs by feeding real inputs to untrained models with the same architectures. The number of edges and accuracy are reported in Table 3.

From Table 3, we see that random models that are not trained have neuron dependency edges extracted from the training set. However, the edges do not generalize to the test set, with accuracy around 50%. Trained models with random inputs have neuron dependency edges that generalize to unseen random inputs. We believe that neurons in trained models are structured internally to have related computations which cause them to be logically related even when the inputs are random. However, the graphs extracted from random inputs are different. Further investigations about the relationship between the graphs extracted from random datasets and real datasets are in Appendix F.

4.4. Neuron dependencies exist between models independently trained on the same dataset

We are interested in inter-model neuron dependencies because models learning from the same distribution may learn similar latent concepts, which may serve as a basis for simple languages to emerge between models. We train two models with the same architecture and different seeds on the same dataset. Let U, V be two binarized neurons from different networks, and let X be the input variable. We check if inter-model neuron dependency exists as $U(X) \to V(X)$. The number of inter-model edges with accuracies are presented in Table 4, column "different models same input".

From Table 4 , we see that inter-model neuron dependencies exist for some datasets, and when they do exist, they can generalize to unseen data. This suggests that the two models, although independently trained, could learn to capture logically related latent concepts, and the respective neurons activate at similar times.

As an additional sanity check experiment, we extract neuron dependency between two copies of the same model with different inputs from the same underlying distribution. Formally, let U, V be two binarized neurons from the same network, and let X_1, X_2 be two input variables created by splitting the dataset into two disjoint halves. We check if inter-mdoel neuron dependency exists as $U(X_1) \to V(X_2)$. We report the number of edges with accuracies in Table 4 column "same model different inputs".

We see from Table 4 "same model different inputs" that inter-model edges do not exist, although two models are the same. This is expected: assuming infinite samples, neuron dependency $U(X_1) \to V(X_2)$ does not exist because X_1 and X_2 are i.i.d variables, and functions of i.i.d variables are i.i.d. Given finite samples, neuron dependencies may be falsely extracted as a result of sampling error, but the sanity check experiment results show this did not occur for our experiment settings.

Table 5. After interchange intervention on the neural network, the percentage of aligned predictions $f^{i_{NN,c}}(\mathbf{x}) = c$, unchanged predictions $f^{i_{NN,c}}(\mathbf{x}) = y$, and unaligned predictions $f^{i_{NN,c}}(\mathbf{x}) \notin \{y, c\}$. Contradiction is when $N \leftarrow T_\phi$, $N \leftarrow F_\phi \in i_{NN,c}$ for some neuron N, and N is not intervened when it occurs. The average contradictions per input is reported. Interchange intervention empirically validates NDGs as a causal abstraction of neural networks.

	aligned		unchanged		unaligned		model accuracy		contradictions	
	train	**test**	train	test	train	test	train	test	train	test
MNIST	**93.41**	**95.21**	5.53	3.78	1.05	1.01	99.48	98.93	0.00	0.00
MNIST even	**100.00**	**100.00**	0.00	0.00	0.00	0.00	99.28	99.17	0.00	0.00
SST2	**100.00**	**100.00**	0.00	0.00	0.00	0.00	100.00	90.00	0.00	0.00
AllNLI	**100.00**	**100.00**	0.00	0.00	0.00	0.00	76.28	72.05	14.02	14.12
CUB200	**89.38**	**88.39**	2.26	1.77	8.36	9.84	92.63	87.79	0.01	0.02
Devign	**100.00**	**100.00**	0.00	0.00	0.00	0.00	78.78	60.86	102.12	102.04

4.5. Interchange intervention empirically validates NDGs as a causal abstraction of neural networks

Neuron dependency graphs reason with logical deduction by treating the neuron's activation values as boolean predicates, as defined by the structural equations Eq. (8). We want to show when logical deduction is performed on a NDG given a counterfactual label, the consequence of the deduction is meaningful to the original neural network as activations, and the network would predict the same counterfactual label in agreement. Formally, we conduct interchange intervention (Geiger et al., 2021a) experiments to measure the percentage of data points for which $c = f^{i_{NN,c}}(\mathbf{x})$ in alignment condition (15) holds true for a selected counterfactual label c (the alignment percentage). We follow Algorithm 1 for interchange intervention with the additional step to remove any contradiction from the intervention, because we cannot assume that all neuron dependencies are satisfied strictly. For ReLU, we select $T_\phi = 1$, $F_\phi = 0$. For sigmoid, $T_\phi = \infty$, $F_\phi = -\infty$. The results of interchange intervention are presented in Table 5.

From Table 5, we observe that the model predicts the counterfactual class c after intervention $f^{i_{NN,c}}(\mathbf{x}) = c$ for at least 88% of data points on all datasets for both the training set and the test set, with MNIST even, SST2, AllNLI, and Devign reaching 100% alignment percentage. We do observe from Table 5 that contradictions occur when reasoning with neuron dependency graphs, because neuron dependency is an approximation of logical implication. Moreover, converting binary values to the activation output incurs noise, because, for example, sigmoid activation never reaches 0 or 1. However, the two factors do not prevent successful interchange intervention results.

We conclude that the high alignment percentage empirically supports the alignment condition and our theory that a neuron dependency graph is a causal abstraction of its neural network. In addition, neuron dependency can approximate logical implication. Logical deductions using neuron dependencies are successful, despite contradictions caused by approximation errors. Conversion from binary deduction results back to neuron activations with ϕ^{-1} is meaningful as the neural network predicts the counterfactual label selected in alignment. We have performed interchange intervention experiments with alternative definitions of allowed interventions and results are presented in Appendix H.

5. Conclusion and Discussions

Despite outstanding performance and prevalent use of neural networks, we are still learning its fundamental properties. Neuron dependency graphs reveal one such fundamental property that there exist many approximate logical implication relations among neurons. In addition to providing explanation about the internal structure of the neural network, a neuron dependency graph is a computational model that can perform downstream reasoning such as causal intervention.

Combining neural networks and neuron dependency graphs, we have a method to automatically discover symbolic rules from datasets, with deep learning as a medium. The ability to discover symbolic rules from datasets may be necessary to produce artificial general intelligence with human-like knowledge and languages.

Acknowledgements

We thank the reviewers for their valuable feedback. Jin Tian was partially supported by ONR grant N000141712140. This work was done in part while Jin Tian was visiting the Simons Institute for the Theory of Computing.

References

Adadi, A. and Berrada, M. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., and Asari, V. K. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8 (3):292, 2019.

Beckers, S. and Halpern, J. Y. Abstracting causal models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 2678–2685, 2019.

Bengio, Y. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Benítez, J. M., Castro, J. L., and Requena, I. Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164, 1997.

Birkhoff, G. *Lattice theory*, volume 25. American Mathematical Soc., 1940.

Chattopadhyay, A., Manupriya, P., Sarkar, A., and Balasubramanian, V. N. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pp. 981–990. PMLR, 2019.

Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M., and Melacci, S. Logic explained networks. *arXiv preprint arXiv:2108.05149*, 2021.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. Supervised learning of universal sentence representations from natural language inference data. In Palmer, M., Hwa, R., and Riedel, S. (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 670–680. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1070. URL https://doi.org/10.18653/v1/d17-1070.

Das, A. and Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

d'Avila Garcez, A. S., Gori, M., Lamb, L. C., Serafini, L., Spranger, M., and Tran, S. N. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019. URL https://collegepublications.co.uk/ifcolog/?00033.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. Neural logic machines. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=B1xY-hRctX.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., and Zhou, M. Codebert: A pre-trained model for programming and natural languages. In Cohn, T., He, Y., and Liu, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 1536–1547. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.139. URL https://doi.org/10.18653/v1/2020.findings-emnlp.139.

Fischer, J., Oláh, A., and Vreeken, J. What's in the box? exploring the inner life of neural networks with robust rules. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3352–3362. PMLR, 2021. URL http://proceedings.mlr.press/v139/fischer21b.html.

Friedman, S. E., Magnusson, I. H., and Schmer-Galunder, S. M. Extracting qualitative causal structure with transformer-based nlp. *arXiv preprint arXiv:2108.13304*, 2021.

Ganter, B. and Wille, R. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

Geiger, A., Lu, H., Icard, T. F., and Potts, C. Causal abstractions of neural networks. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=RmuXDtjDhG.

Geiger, A., Wu, Z., Lu, H., Rozner, J., Kreiss, E., Icard, T., Goodman, N. D., and Potts, C. Inducing causal structure for interpretable neural networks. *arXiv preprint arXiv:2112.00826*, 2021b.

Haugeland, J. *Artificial intelligence: The very idea*. MIT press, 1989.

He, J., Chen, J.-N., Liu, S., Kortylewski, A., Yang, C., Bai, Y., Wang, C., and Yuille, A. Transfg: A transformer architecture for fine-grained recognition. *arXiv preprint arXiv:2103.07976*, 2021.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.

Liu, Q., Kusner, M., and Blunsom, P. Counterfactual data augmentation for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 187–197, 2021.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., GONG, M., Zhou, M., Duan, N., Sundaresan, N., Deng, S. K., Fu, S., and LIU, S. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=6lE4dQXaUcb.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.

Marcinkevičs, R. and Vogt, J. E. Interpretability and explainability: A machine learning zoo mini-tour. *arXiv preprint arXiv:2012.01805*, 2020.

Ming, Y., Xu, P., Qu, H., and Ren, L. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 903–913, 2019.

Morgan, S. L. and Winship, C. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

Mu, J. and Andreas, J. Compositional explanations of neurons. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c74956ffb38ba48ed6ce977af6727275-Abstract.html.

Pearl, J. Direct and indirect effects. In Breese, J. S. and Koller, D. (eds.), *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pp. 411–420. Morgan Kaufmann, 2001. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=126&proceeding_id=17.

Pearl, J. *Causality*. Cambridge university press, 2009.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Ribeiro, M. T., Singh, S., and Guestrin, C. "Why should I trust you?": Explaining the predictions of any classifier. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R. (eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1135–1144. ACM, 2016. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I. Y., Qian, H., Fagin, R., Barahona, F., Sharma, U., et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Samek, W., Wiegand, T., and Müller, K.-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. In *The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Serafini, L. and d'Avila Garcez, A. S. Logic tensor networks: Deep learning and logical reasoning from data

and knowledge. In Besold, T. R., Lamb, L. C., Serafini, L., and Tabor, W. (eds.), *Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), New York City, NY, USA, July 16-17, 2016*, volume 1768 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1768/NESY16_paper3.pdf.

Shi, S., Chen, H., Zhang, M., and Zhang, Y. Neural logic networks. *arXiv preprint arXiv:1910.08629*, 2019.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6034.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Watson, D. S., Gultchin, L., Taly, A., and Floridi, L. Local explanations via necessity and sufficiency: unifying theory and practice. In de Campos, C. P., Maathuis, M. H., and Quaeghebeur, E. (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1382–1392. AUAI Press, 2021. URL https://proceedings.mlr.press/v161/watson21a.html.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Zeiler, M. D., Taylor, G. W., and Fergus, R. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pp. 2018–2025. IEEE, 2011.

Zhou, Y., Liu, S., Siow, J. K., Du, X., and Liu, Y. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 10197–10207, 2019.

A. Proof for Theorem 3.11

Recall that $desc(\mathbf{W})$ denotes the union of $\mathbf{W}$ and the descendant nodes of $\mathbf{W}$ for some $\mathbf{W} \subseteq \mathbf{V}$ on $NDG = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \mathbf{B} \cup \mathbf{O}$, with binarized neurons $\mathbf{B} = \{N'_j \mid N_j \in \mathbf{N}\} \cup \{\overline{N}'_j \mid N_j \in \mathbf{N}\}$ and outputs $\mathbf{O}$.

Definition A.1 (Interventions closed under *desc* on $\mathbf{B}$)**.** Let i_{NDG} be an intervention on SCM_{NDG} where $i_{NDG} = \{C \leftarrow \mathit{True}, \overline{C} \leftarrow \mathit{False}) \mid C \in \mathbf{C}\}$, $\mathbf{C} \subseteq \mathbf{B}$, and there does not exist i where $N'_i \in \mathbf{C}$ and $\overline{N}'_j \in \mathbf{C}$. We say i_{NDG} is closed under *desc* operator if $desc(\mathbf{C}) \cap \mathbf{B} \subseteq \mathbf{C}$.

Note that every $i_{NDG,c}$ in allowed interventions set I_{NDG} in Eq. (12) is closed under *desc* operator on $\mathbf{B}$ because $\mathbf{Q}(\{O_c\}) = \mathbf{B} \cap \big(desc(ance(\{O_c\}))\big)$. We now prove Theorem 3.11.

Proof. We prove that τ is a surjective function. Let $(\mathbf{u}, \hat{\mathbf{y}}, \mathbf{x})$ be any element in the codomain $(\mathbf{u}, \mathbf{x}, \mathbf{o}) \in \mathcal{D}(\mathbf{U} \cup \mathbf{X} \cup \mathbf{O})$ for $SCM_{NDG} = \langle \mathbf{U} \cup \mathbf{X} \cup \mathbf{O}, \emptyset, \mathcal{F}, \emptyset \rangle$. Let $(\mathbf{u}, \mathbf{x}, \mathbf{o}) = \big((\mathbf{n}', \neg\mathbf{n}'), \mathbf{x}, \mathbf{o}\big)$. Consider an element $(\phi^{-1}(\mathbf{n}'), \mathbf{x}, \mathbf{o}) \in \mathcal{D}(\mathbf{N} \cup \mathbf{X} \cup \mathbf{O})$ for SCM_{NN}. Given τ defined in Eq. (10), we see that $\tau(\phi^{-1}(\mathbf{n}'), \mathbf{x}, \mathbf{o}) = \big(\big(\mathbb{1}(\phi(\phi^{-1}(\mathbf{n}'))), \neg\mathbb{1}(\phi(\phi^{-1}(\mathbf{n}')))\big), \mathbf{x}, \mathbf{o}\big) = (\mathbf{u}, \mathbf{x}, \mathbf{o})$, because $\mathbb{1}(\phi(\phi^{-1}(\mathbf{n}'))) = \mathbf{n}'$.

We prove that ω_τ is a surjective function. Let $i_{NDG,c}$ be any element in the codomain $i_{NDG,c} \in I_{NDG}$ defined in Eq. (12). Consider $i_{NN,c}$ defined in Eq. (14), and $i_{NN,c} \in I_{NN}$. Given ω_τ defined in Eq. (6), we see

$$\begin{aligned}\omega_\tau(i_{NN,c}) =& \{(N'_j, \overline{N}'_j) \leftarrow (\mathit{True}, \mathit{False}) \mid N'_j \in \mathbf{Q}(\{O_c\})\} \cup \\ & \{(N'_j, \overline{N}'_j) \leftarrow (\mathit{False}, \mathit{True}) \mid \overline{N}'_j \in \mathbf{Q}(\{O_c\})\} \\ =& \{Q \leftarrow \mathit{True}, \overline{Q} \leftarrow \mathit{False} \mid Q \in \mathbf{Q}(\{O_c\})\} \\ =& i_{NDG,c} \end{aligned} \tag{16}$$

We want to prove that $\forall i_{NN,c} \in I_{NN} : \tau(i_{NN,c}(SCM_{NN})) = \omega_\tau(i_{NN,c})(SCM_{NDG})$. Recall in the alignment condition Def 3.11, we assume that all neuron dependencies are satisfied as strict neuron dependencies before the intervention, and (15) holds, shown below

$$\forall \mathbf{x} \in D, c \in \mathbb{N}_k : c = f^{i_{NN,c}}(\mathbf{x}), \mathbf{o}_c = \mathbf{o}_{NDG}^{i_{NDG},c}(\mathbf{x})$$

Before intervention, let $\big((\mathbf{n}', \overline{\mathbf{n}}'), \mathbf{x}, \mathbf{o}\big)$ be the value of SCM_{NDG} and $\big(\mathbf{n}, \mathbf{x}, \mathbf{o}\big)$ be the value of SCM_{NN}. Let $i_{NN,c}$ be any intervention $i_{NN,c} \in I_{NN}$ for some class c.

For $\tau(i_{NN,c}(SCM_{NN}))$ on the left-hand-side, we find the counterfactual results of the intervention, $i_{NN,c}(SCM_{NN}) = (\mathbf{N}, \mathbf{X}, \mathbf{O}) = \big(\mathbf{n}^{i_{NN,c}}, \mathbf{x}, f_{\mathbf{O}}(\mathbf{n}^{i_{NN,c}})\big)$, where

$$n_j^{i_{NN,c}} = T_\phi \qquad , N'_j \in \mathbf{Q}(\{O_c\}) \tag{17}$$
$$n_k^{i_{NN,c}} = F_\phi \qquad , \overline{N}'_k \in \mathbf{Q}(\{O_c\}) \tag{18}$$
$$n_l^{i_{NN,c}} = f_{N_l}(\mathbf{x}) = n_l \qquad , N'_l, \overline{N}'_l \notin \mathbf{Q}(\{O_c\}) \tag{19}$$

Apply mapping τ, we have $\tau(\big(\mathbf{n}^{i_{NN,c}}, \mathbf{x}, f_{\mathbf{O}}(\mathbf{n}^{i_{NN,c}})\big)) = \big((\mathbb{1}(\phi(\mathbf{n}^{i_{NN,c}})), \neg\mathbb{1}(\phi(\mathbf{n}^{i_{NN,c}}))), \mathbf{x}, f_{\mathbf{O}}(\mathbf{n}^{i_{NN,c}})\big)$.

For $\omega_\tau(i_{NN,c})(SCM_{NDG})$ on the right-hand-side, we first note $\omega_\tau(i_{NN,c}) = i_{NDG,c}$ as shown above. Apply intervention $i_{NDG,c}$ and find the counterfactual $i_{NDG,c}(SCM_{NDG}) = ((\mathbf{N}', \overline{\mathbf{N}'}), \mathbf{X}, \mathbf{O}) = \big((\mathbf{n'}^{i_{NDG,c}}, \overline{\mathbf{n}}'^{i_{NDG,c}}), \mathbf{x}, \mathbf{o}_c\big)$,

$$n_j'^{i_{NDG,c}} = \mathit{True} \qquad , N'_j \in \mathbf{Q}(\{O_c\}) \tag{20}$$
$$\overline{n}_j'^{i_{NDG,c}} = \mathit{False} \qquad , N'_j \in \mathbf{Q}(\{O_c\}) \tag{21}$$
$$n_k'^{i_{NDG,c}} = \mathit{False} \qquad , \overline{N}'_k \in \mathbf{Q}(\{O_c\}) \tag{22}$$
$$\overline{n}_k'^{i_{NDG,c}} = \mathit{False} \qquad , \overline{N}'_k \in \mathbf{Q}(\{O_c\}) \tag{23}$$
$$n_l'^{i_{NDG,c}} = r'_l \qquad , N'_l, \overline{N}'_l \notin \mathbf{Q}(\{O_c\}) \tag{24}$$
$$\overline{n}_l'^{i_{NDG,c}} = \overline{r}'_l \qquad , N'_l, \overline{N}'_l \notin \mathbf{Q}(\{O_c\}) \tag{25}$$

Our goal is to prove that $n'_l = n_l'^{i_{NDG,c}}$ and $n'_l = \overline{n}_l'^{i_{NDG,c}}$ unchanged, i.e. $r'_l = \mathbb{1}(\phi(f_{N_l}(\mathbf{x}))) = \mathbb{1}(\phi(n_l))$ and $\overline{r}'_l = \neg r'_l$. Assume for the sake of contradiction that there exists $U \in \{N'_l, \overline{N}'_l\}$, $U, \overline{U} \notin \mathbf{Q}(\{O_c\})$ and U changes value from u to $u^{i_{NDG,c}}$, $u \neq u^{i_{NDG,c}}$.

Case 1 Assume that $u = \mathit{True}$, then $\overline{u} = \mathit{False}$ and $u^{i_{NDG,c}} = \mathit{False}$. Because $U \in \mathbf{B}$, we know $u = \bigvee_{p \in pa_{NDG}(U)} p \vee \big(f'_U(\mathbf{x}) \wedge \neg\overline{u}\big)$ given structural equation in Eq. (8). We know $f'_U(\mathbf{x})$ does not change, and if $\overline{u} = \overline{u}^{i_{NDG,c}} = \mathit{False}$, then $u^{i_{NDG,c}} = f'_U(\mathbf{x}) \wedge \neg\overline{u}^{i_{NDG,c}} = \mathit{True}$. We reach a contradiction if $\overline{u}$ does not change from *False* to *True*, so this case reduces to Case 2.

Case 2 Assume that $u = \mathit{False}$, then $\overline{u} = \mathit{True}$ and $u^{i_{NDG,c}} = \mathit{True}$. In $u = \bigvee_{p \in pa_{NDG}(U)} p \vee \big(f'_U(\mathbf{x}) \wedge \neg\overline{u}\big)$, we have $f'_U(\mathbf{x}) = f'_U(\mathbf{x}) \wedge \neg\overline{u} = \mathit{False}$. Therefore, some parent $p = \mathit{False}$ changed to $p^{i_{NDG,c}} = \mathit{True}$.

Case 2.1 Assume $P \in \mathbf{Q}(\{O_c\})$, then by closure of $\mathbf{Q}(\{O_c\})$ under *desc* on $\mathbf{B}$, we know $U \in \mathbf{Q}(\{O_c\})$ and reach a contradiction.

Case 2.2 Assume that every ancestor A of U is in $\mathbf{B} \cup \{\overline{O}_j \mid j \in \mathbb{N}_k\} \setminus \mathbf{Q}(\{O_c\})$, then $a = \bigvee_{q \in pa_{NDG}(A)} q \vee \big(f'_A(\mathbf{x}) \wedge \neg\overline{a}\big) = \bigvee_{q \in pa_{NDG}(A)} q$. Due to finiteness of the graph, there must exist some ancestor E of U changing from $e = \mathit{False}$ to $e^{i_{NDG,c}} = \mathit{True}$, with no ancestor of E changing from False to True. We know $e = \bigvee_{p \in pa_{NDG}(E)} p \vee \big(f'_E(\mathbf{x}) \wedge \neg\overline{e}\big) = \mathit{False}$ and cannot change to $e^{i_{NDG,c}} = \mathit{True}$, so a contradiction is reached.

Case 2.3 Assume that some ancestor A of U is in $A \in \{O_i \mid i \in \mathbb{N}_k\}$, $a = \mathit{False}$, $a^{i_{NDG,c}} = \mathit{True}$. Due to finiteness of the graph, there must exist some $E = O_j$ that fits this condition and no ancestor of E fits this condition. Because of Case 2.1 and 2.2, we know E has no ancestor in $\mathbf{B} \cup \{\overline{O}_j \mid j \in \mathbb{N}_k\}$ that changes from False to True. Therefore, although E changes from False to True, it has no ancestor that changes from False to True. Given the structural equation of o_j, we know $o_j = \bigvee_{p \in pa_{NDG}(O_j)} p \vee (\bigwedge_{l \neq j, l \in \mathbb{N}_k} \overline{o}_l) \vee \left(f'_{O_j}(\mathbf{x}) \wedge \neg \overline{o}_j\right) = \mathit{False} \vee (\bigwedge_{l \neq j, l \in \mathbb{N}_k} \overline{o}_l) \vee \mathit{False}$, so $\bigwedge_{l \neq j, l \in \mathbb{N}_k} \overline{o}_l$ changes from False to True. Given the alignment condition, $\mathbf{o}_c = \mathbf{o}^{i_{NDG,c}}$, it must be that $j = c$. Because $E = O_c$ is an ancestor of U, we have $U \in \mathbf{Q}(\{O_c\})$ in contradiction.

We have proven that $n'_l = n_l'^{i_{NDG,c}}$ and $n'_l = \overline{n}_l'^{i_{NDG,c}}$ are unchanged after the intervention. We conclude that $\forall i_{NN,c} \in I_{NN} : \tau(i_{NN,c}(SCM_{NN})) = \omega_\tau(i_{NN,c})(SCM_{NDG})$. We conclude that (SCM_{NDG}, I_{NDG}) is a τ-abstraction of (SCM_{NN}, I_{NN}).

□

B. Explanations for the reliability criterion in the construction of NDGs

When constructing NDGs from dataset, we use Eq. (5) presented below:

$$\mathbf{E} = \{(U,V) \mid \frac{\hat{P}(U \wedge \neg V)}{\hat{P}(U)\hat{P}(\neg V)} \leq \frac{1}{\alpha}, \frac{|T|\hat{P}(U)\hat{P}(\neg V)}{\alpha} > 1, (U,V) \in \mathbf{V} \times \mathbf{V}\}$$

Condition $\frac{\hat{P}(U \wedge \neg V)}{\hat{P}(U)\hat{P}(\neg V)} \leq \frac{1}{\alpha}$ is an estimate of the neuron dependency Definition 3.2 based on the sample proportion. The reliability criterion $\frac{|T|\hat{P}(U)\hat{P}(\neg V)}{\alpha} > 1$ is to address the situation when there are not enough samples to reliably estimate $\hat{P}$ to establish or reject an edge, so that events $U \wedge \neg V$ needed to falsify $U \to V$ are expected to occur more than once.

We consider the scenario where neuron dependency does not exist but may be estimated to be true by the sample proportion without the reliability criterion. Let U, V be two binarized neurons where neuron dependency does not exist $U \not\to V$ where $\frac{P(U \wedge \neg V)}{P(U)P(\neg V)} > \frac{1}{\alpha}$. The expected frequency $\mathbb{E}[U \wedge \neg V]$ of events $U \wedge \neg V$ is $\mathbb{E}[U \wedge \neg V] = |T|P(U \wedge \neg V) > |T|P(U)P(\neg V)/\alpha$. If we do not have the reliability criterion, then $\mathbb{E}[U \wedge \neg V]$ may be arbitrarily close to zero. If due to small sample size, the frequency of $U \wedge \neg V$ is indeed zero, then $\frac{\hat{P}(U \wedge \neg V)}{\hat{P}(U)\hat{P}(\neg V)} = 0$, $\frac{\hat{P}(U \wedge \neg V)}{\hat{P}(U)\hat{P}(\neg V)} \leq \frac{1}{\alpha}$, even though $\frac{P(U \wedge \neg V)}{P(U)P(\neg V)} > \frac{1}{\alpha}$, so an edge from U to V will be constructed. Neuron dependency is estimated to exist given data, even though it does not exist. With the reliability criterion, we have $\mathbb{E}[U \wedge \neg V] > 1$. Events $U \wedge \neg V$ needed to falsify $U \to V$ are expected to occur more than once, not arbitrarily close to zero.

C. Inter-layer neuron dependency experiment details

We select multiple layers in the same model and observe that inter-layer neuron dependencies exist. For this experiment, we also choose layers from the attention block of a Transformer, which does not cut off the data flow of the neural network architecture. Notably, the neuron activation are selected from the first time step of a time series, following the convention of doing classification with Transformer encoder. The PyTorch program using Huggingface library (Wolf et al., 2019) to select layers is in Figure 4. We report the number of edges as well as the training and test accuracy in Table 6.

```
def inter_layer_selection(model, dataset):
    if "MNIST" == dataset:
        return [model.seq1.fc1,
                model.seq2.fc2]
    elif "MNIST even" == dataset:
        return [model.seq1.fc1,
                model.seq2.fc2]
    elif "SST2" == dataset:
        return [model.pre_classifier,
                model.distilbert.
                  transformer.layer[5].
                  ffn.lin1]
    elif "AllNLI" == dataset:
        return [model.classifier.dense,
                model.roberta.encoder.
                  layer[5].intermediate.
                  dense]
    elif "CUB200" == dataset:
        return [model.part_head[0],
                model.transformer.encoder.
                  part_layer.ffn.fc1]
    elif "Devign" == dataset:
        return [model.classifier.dense,
                model.roberta.encoder.
                  layer[11].intermediate.
                  dense]
```

Figure 4. The PyTorch function used to select the two layers in the architecture for inter-layer experiments. Huggingface models are used. For MNIST, the seq1 function is a convolutional neural network, and the seq2 function is a feedforward neural network. The other architectures are given in Table 1. The first layer selected for every dataset is used for all non-inter-layer experiments. The second layer selected for Transformer is located in the attention block, and, when applicable, we select the first timestep of the time series output for binarized neuron activations in the NDG following the common practice for classification with Transformers.

Table 6. The number of edges, average training and test accuracy when neurons are selected from two layers in the same model.

	layer 1			layer 2			inter-layer		
	edges	train	test	edges	train	test	edges	train	test
MNIST	4172	99.93	99.90	10648	99.95	99.92	4986	99.92	99.89
MNIST even	790	99.94	99.95	7002	99.99	99.99	1088	99.88	99.90
SST2	529394	100.00	99.62	1877060	99.98	99.63	997980	100.00	99.74
AllNLI	747115	99.52	99.50	2186033	99.58	99.55	1087438	99.58	99.56
CUB200	900258	97.89	97.16	902210	97.89	97.16	900402	97.89	97.16
Devign	2191519	99.94	99.94	7952771	99.91	99.91	3430234	99.94	99.93

From Table 6, we see that there are edges within each of the two layers selected, and there are edges between the two layers. All training and test accuracies are high. We conclude that inter-layer edges in a trained model exist commonly and generalize to unseen data.

D. Relationship between the number of NDG edges and alpha thresholds

The plot for the number of edges versus different α thresholds is presented in Figure 5.

We see that the number of edges, including mutual neuron dependencies between a pair of nodes, decreases as the α threshold increases, which is implied by how NDGs are constructed from data in Eq. (5). For lower α, there tend to be more mutual neuron dependencies.

E. Neuron dependency outliers in terms of accuracy

For neuron dependency graphs with settings in Table 1, we plot the quantiles and outliers of training and test accuracy in the box plots in Figure 6.

From Figure 6, we see that the edges extracted from training data generalize well to unseen data from the test set given the accuracy quantiles. Notably, for all datasets, 75% of the edges have test accuracy higher than 99.5%, and 50% of the edges have test accuracy higher than 99.8%. The box plot shows that there may be many outliers with low training and test accuracy. To improve the accuracy of edges, the outliers may be removed with cross validation, which we do not remove for our experiments.

F. Explorations about the relationship between graphs extracted from real and random data given a trained model

As a further investigation of Section 4.3, we raise questions about the relationship between the original graph G generated from real data and the graph generated from random inputs G', for a trained model. Intuitively speaking, is one graph contained in the other one? Given the inconsistencies

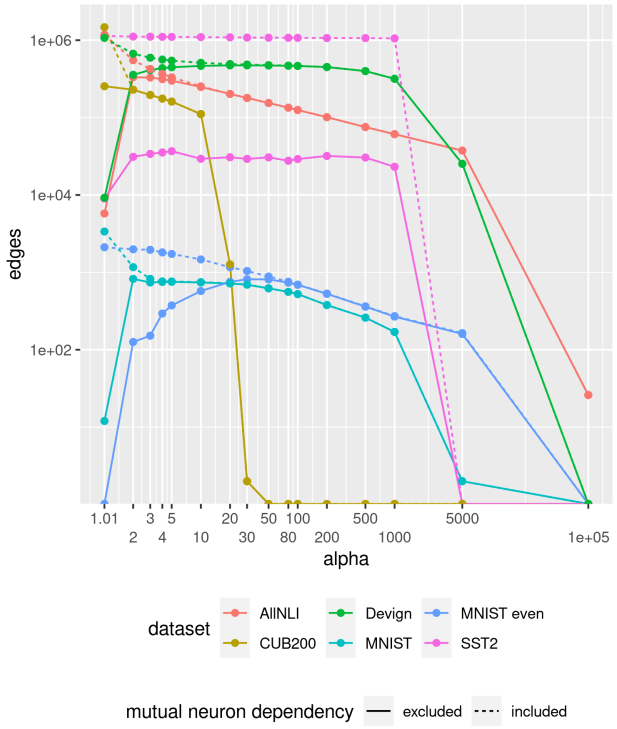
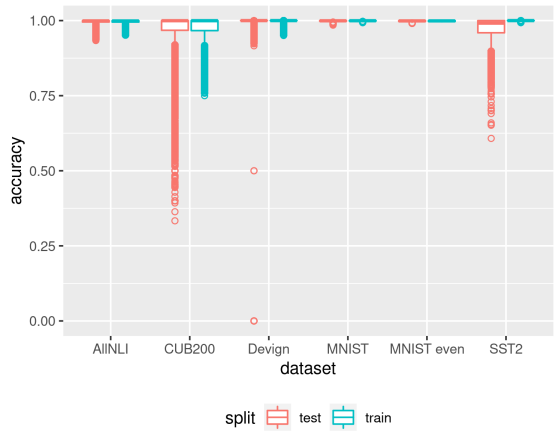


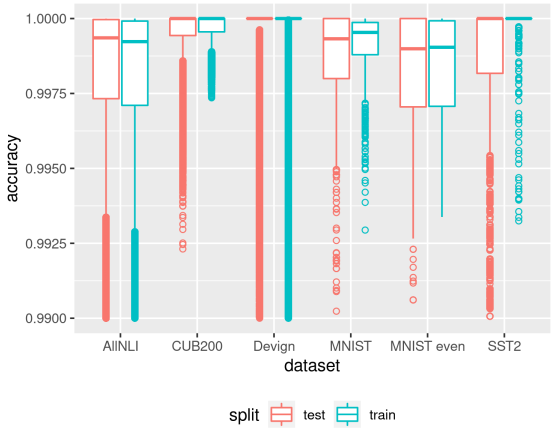
Figure 5. The number of edges in the neuron dependency graph for different alpha thresholds including and excluding mutual neuron dependencies. The x-axis is log transformed.

of the α threshold in different contexts, we cannot directly check if the set of edges of one graph is a subgraph of the other one. To address the problem, we alternatively present the average accuracy for the neuron dependency edges from G on random inputs and that of G' on real inputs in Table 7.

Table 7 shows no row with all accuracy numbers greater than 95%, suggesting that the NDG extracted is conditioned on the input distribution. For the same trained model, a different input distribution leads to a different NDG. We note that for every dataset, one of the graphs tend to perform relatively well on both real and random input data. We manually label them with $G > G'$ if we expect edges from random data to hold true on real data, and $G < G'$ if we expect edges from real data to hold true on random data, which is similar to graph containment. We hypothesize that, compared to uniform noise, real dataset introduces patterns that can add edges, remove edges, or both. The exact condition that determines the relation between G and G' requires further research.



(a) The entire box plot.



(b) Cropped box plot with accuracy range 0.99 to 1.

Figure 6. Box plot for training and test accuracy for every neuron dependency edge in the neuron dependency graph.

Table 7. The average accuracy for NDG G extracted from real data and G' from random input data given a trained model, and the accuracy when the datasets are exchanged. The relation inferred is for hypothetical reference.

	G		G'		relation
	real	random	real	random	
MNIST	99.90	91.40	63.16	99.94	$G < G'$
MNIST even	99.83	94.82	98.19	99.95	$G > G'$
SST2	100.00	88.55	99.71	99.92	$G > G'$
AllNLI	99.58	97.19	94.69	99.68	$G < G'$
CUB200	97.87	99.51	49.80	95.76	$G < G'$
Devign	99.97	98.54	94.42	99.75	$G < G'$

G. Precision-recall plot of neuron dependency graph edges

For an neuron dependency edge $A \to B$, we define the precision to be $\hat{P}(A \wedge B)/\hat{P}(A)$, and we define the recall to be $\hat{P}(A \wedge B)/\hat{P}(B)$. We present the training set precision-recall plot from Figure 7 to Figure 12.

We see from the precision-recall plots that the precision of neuron dependency edges is high across all datasets. This is expected given our definition of neuron dependency in Eq. (1). Recall of neuron dependency edges may spread out between range 0 and 1. For CUB200 dataset in Figure 11, there is a cluster of points at the bottom of the plot with low recall, and this cluster do not exist in other plots. This is likely because there are 200 target classes in CUB200 dataset, far more than other datasets, and a binary neuron N' with $P(N') = 0.5$ will have a recall around 0.01 for edge $T \to N'$ for some target T. We draw the line for recall equal to 0.01 and see that the cluster overlap with the line.

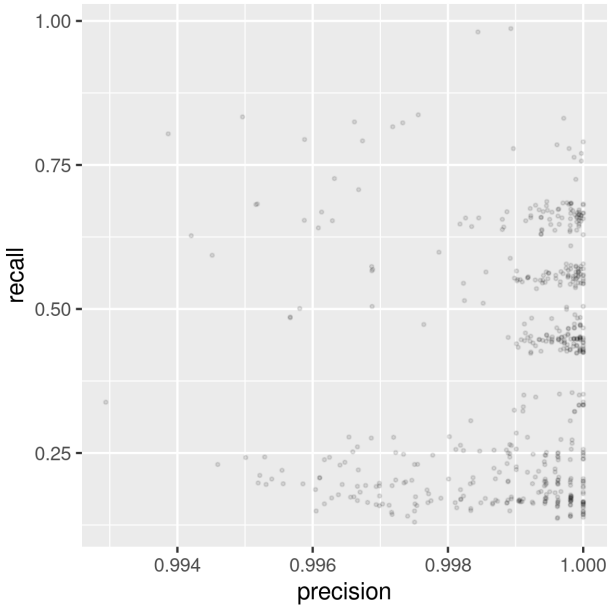


Figure 7. The precision-recall plot for MNIST dataset.

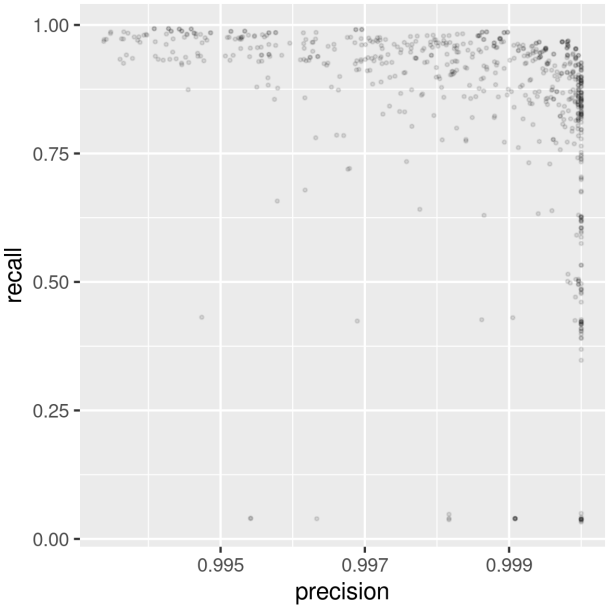


Figure 8. The precision-recall plot for MNIST even dataset.

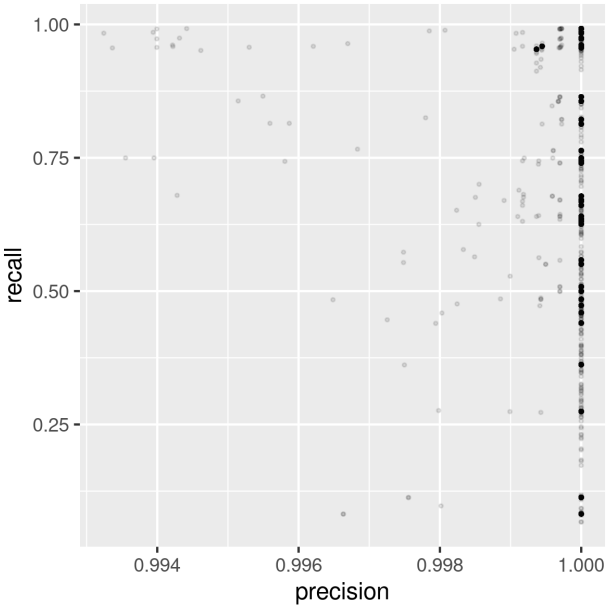


Figure 9. The precision-recall plot for SST2 dataset.

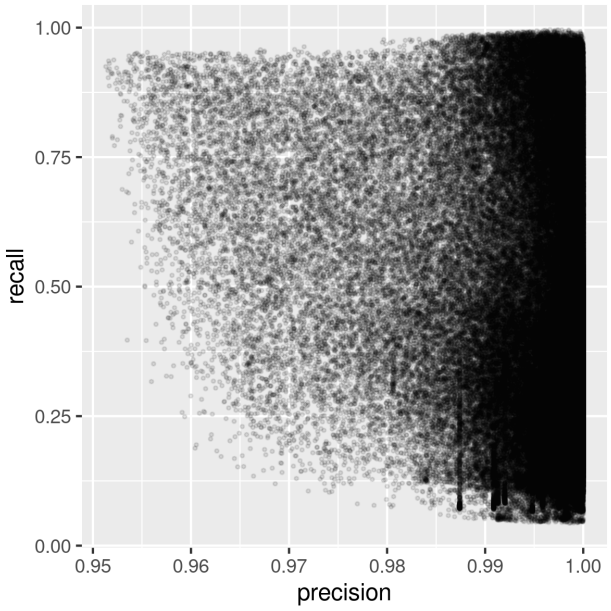


Figure 10. The precision-recall plot for AllNLI dataset.

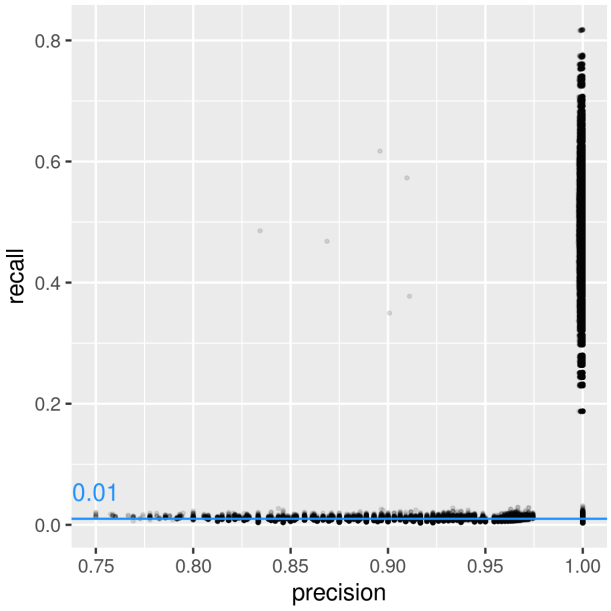


Figure 11. The precision-recall plot for CUB200 dataset. Line y=0.01 marks the expected recall if a binary neuron with 1/2 probability is logically related to one of the 200 classes. Note that there are a few edge data points around the center of the plot.

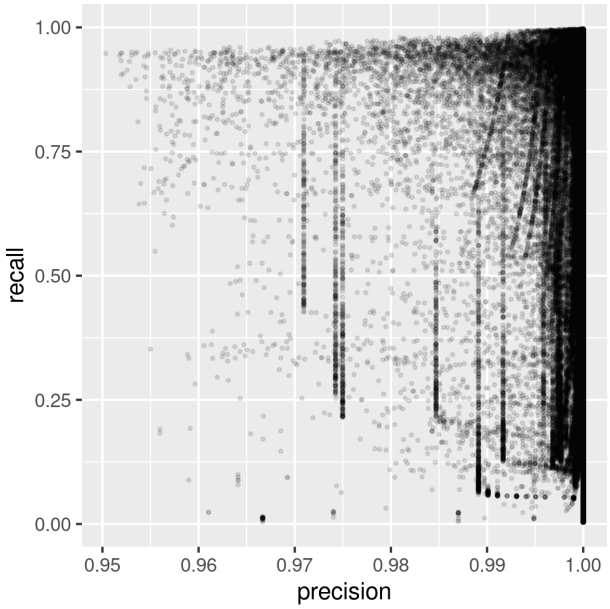


Figure 12. The precision-recall plot for Devign dataset.

H. Experiments with alternative definitions of allowed interventions

We provide alternative definitions of interventions as opposed to Eq. (12) and demonstrate the results if they are used in interchange intervention.

Definition H.1 (The first alternative definition of allowed interventions)**.** Given the original output class y and an alternative class c, with $c, y \in \mathbb{N}_k$ for k-class classification, we define the set of allowed interventions I_{NDG} on SCM_{NDG}

$$I_{NDG} = \{i_{NDG,c} \mid c = 0, 1, ...k-1\} \tag{26}$$

$$i_{NDG,c} = \{Q \leftarrow \textit{True}, \overline{Q} \leftarrow \textit{False} \mid Q \in \mathbf{B} \cap \big(\textit{desc}(\{O_c\}) \cup \textit{ance}(\{\overline{O}_y\})\big)\} \tag{27}$$

Define the set I_{NN} of allowed interventions on SCM_{NN}

$$I_{NN} = \{i_{NN,c} \mid c = 0, 1, ...k-1\} \tag{28}$$

$$i_{NN,c} = \{N_j \leftarrow T_\phi \mid N'_j \in \textit{desc}(\{O_c\}) \cup \textit{ance}(\{\overline{O}_y\})\} \cup \{N_j \leftarrow F_\phi \mid \overline{N}'_j \in \textit{desc}(\{O_c\}) \cup \textit{ance}(\{\overline{O}_y\})\} \tag{29}$$

Especially for many-class classifications with a wide graph, we believe that there are nodes implied by the original target O_y but are not descendants of the counterfactual target O_c. Setting these nodes to False helps improve the alignment percentage for many-class classification datasets. We present the interchange intervention experiment results in Table 8.

Table 8. The percentage of aligned predictions $f^{i_{NN,c}}(\mathbf{x}) = c$, unchanged predictions $f^{i_{NN,c}}(\mathbf{x}) = y$, and unaligned changed predictions $f^{i_{NN,c}}(\mathbf{x}) \notin \{y, c\}$ for interchange intervention with the set of interventions replaced with Eq. (26) and Eq. (28). Contradiction occurs when a node and its negation both appear in the descendants. The contradiction column shows the average number of contradictions per interchange intervention. Although the percentage of aligned predictions is high, causal abstraction does not follow from the alignment condition given the alternative definitions of allowed interventions.

	aligned		unchanged		unaligned		model accuracy		contradictions	
	train	**test**	train	test	train	test	train	test	train	test
MNIST	**99.99**	**100.00**	0.00	0.00	0.01	0.00	99.57	98.93	17.23	17.25
MNIST even	**100.00**	**100.00**	0.00	0.00	0.00	0.00	99.27	99.17	0.00	0.00
SST2	**100.00**	**100.00**	0.00	0.00	0.00	0.00	100.00	90.00	0.00	0.00
AllNLI	**94.26**	**94.21**	0.00	0.00	5.74	5.79	75.84	72.05	98.97	98.63
CUB200	**89.24**	**90.48**	0.00	0.00	10.76	9.53	92.63	87.79	360.17	360.45
Devign	**100.00**	**100.00**	0.00	0.00	0.00	0.00	78.78	60.86	0.00	0.00

From Table 8, we see that the alignment percentage is improved for MNIST and CUB200 compared to Table 5, but the alignment percentage reduces for AllNLI dataset.

We do not use this alternative definition because the counterfactual of this intervention may cause binarized neuron nodes that are not directly intervened to change values. As a result, the alignment condition cannot derive causal abstraction given the alternative definition of interventions. If the theory of causal abstraction is not vital to the experiments, one may consider any set of meaningful reasoning rules using NDG for practical purposes.

We have another alternative definition of interventions.

Definition H.2 (The second alternative definition of allowed interventions)**.** Given the original output class y and an alternative class c, with $c, y \in \mathbb{N}_k$ for k-class classification, we define the set of allowed interventions I_{NDG} on SCM_{NDG}

$$I_{NDG} = \{i_{NDG,c} \mid c \in \mathbb{N}_k\} \tag{30}$$

$$i_{NDG,c} = \{Q \leftarrow \textit{True}, \overline{Q} \leftarrow \textit{False} \mid Q \in \mathbf{B} \cap \textit{desc}(\{O_c\})\} \tag{31}$$

Define the set I_{NN} of valid interventions on SCM_{NN}

$$I_{NN} = \{i_{NN,c} \mid c \in \mathbb{N}_k\} \tag{32}$$

$$i_{NN,c} = \{N_j \leftarrow T_\phi \mid N'_j \in \textit{desc}(\{O_c\})\} \cup \{N_j \leftarrow F_\phi \mid \overline{N}'_j \in \textit{desc}(\{O_c\})\} \tag{33}$$

Compared to the original definition in Eq. (12) and the first alternative definition above, the second alternative definition only intervene the descendants of O_c. Logically, we are only setting the necessary conditions to be true. Note that the proof of causal abstraction can be modified and apply to this definition. The interchange intervention experiment results are presented in Table 9.

Table 9. The percentage of aligned predictions $f^{i_{NN,c}}(\mathbf{x}) = c$, unchanged predictions $f^{i_{NN,c}}(\mathbf{x}) = y$, and unaligned changed predictions $f^{i_{NN,c}}(\mathbf{x}) \notin \{y, c\}$ for interchange intervention with the set of interventions replaced with Eq. (30) and Eq. (32). Contradiction occurs when a node and its negation both appear in the descendants. The contradiction column shows the average number of contradictions per interchange intervention. Although causal abstraction follows from the alignment condition given the alternative definitions of allowed interventions, the percentage of aligned predictions is sometimes low.

	aligned		unchanged		unaligned		model accuracy		contradictions	
	train	**test**	train	test	train	test	train	test	train	test
MNIST	**95.53**	**95.33**	3.43	3.65	1.05	1.02	99.57	98.93	0.00	0.00
MNIST even	**1.60**	**1.55**	98.40	98.45	0.00	0.00	99.27	99.17	0.00	0.00
SST2	**100.00**	**100.00**	0.00	0.00	0.00	0.00	100.00	90.00	0.00	0.00
AllNLI	**43.85**	**43.76**	40.82	40.17	15.33	16.07	75.84	72.05	0.00	0.00
CUB200	**89.24**	**89.14**	2.10	1.71	8.66	9.15	92.63	87.79	0.02	0.02
Devign	**56.14**	**58.44**	43.86	41.56	0.00	0.00	78.78	60.86	0.00	0.00

From Table 9, we see alignment percentage reduces significantly for MNIST even, AllNLI and Devign datasets compared to results obtained in Table 5. This is because for a neuron dependency graph for few-class classifications, there are often sufficient conditions for O_c that are *True* for

in-distribution data. However, if we do not set such sufficient conditions to be *True*, the constraints implied by the intervention are weak and the neural network will likely not change predictions to class c, and causal abstraction may not be empirically validated. For this reason, we do not use the second alternative definition in the main paper.

I. A training trick to reduce approximation error due to binarization for the sigmoid activation function

When converting the logical deduction performed on neuron dependency graph back to neuron activations values, an approximation error will occur, because activation functions such as sigmoid or ReLU almost never output boolean values 0 and 1. For sigmoid activation function, we share a trick that modifies the sigmoid activation function slightly during training, so that the sigmoid function outputs are close to 0 or 1. The PyTorch program for the training trick is presented in Figure 13.

```
class TrickFun(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input):
        return input * e

    @staticmethod
    def backward(ctx, grad_output):
        return grad_output
```

Figure 13. The PyTorch program for a training trick that causes the sigmoid function to output close to 0 or 1. The variable e is a hyperparameter set to be greater than 1 (e.g. $e = 10$).

For any neuron with sigmoid activation $\sigma(n)$, we first pass the neuron logits to the trick function t before passing it to sigmoid function, i.e. $\sigma(t(n))$. The trick function multiplies the neuron output with a constant ($e = 10$ for example) during the forward pass but does not change its gradients during back-propagation. After the model is trained, the sigmoid function $\sigma(t(n))$ output will be close to 0 or 1, with no discernible sacrifice of model performance.

J. Additional plots of neuron dependency graphs

We present plots of neuron dependency graphs for AllNLI and Devign datasets in addition to MNIST in Figure 1 and MNIST even in Figure 2. Neuron dependency graphs for SST2 and CUB200 datasets are extremely wide (see Table 2) and cannot be recognized when plotted. Please refer to our open-sourced code base to extract and explore the graphs.

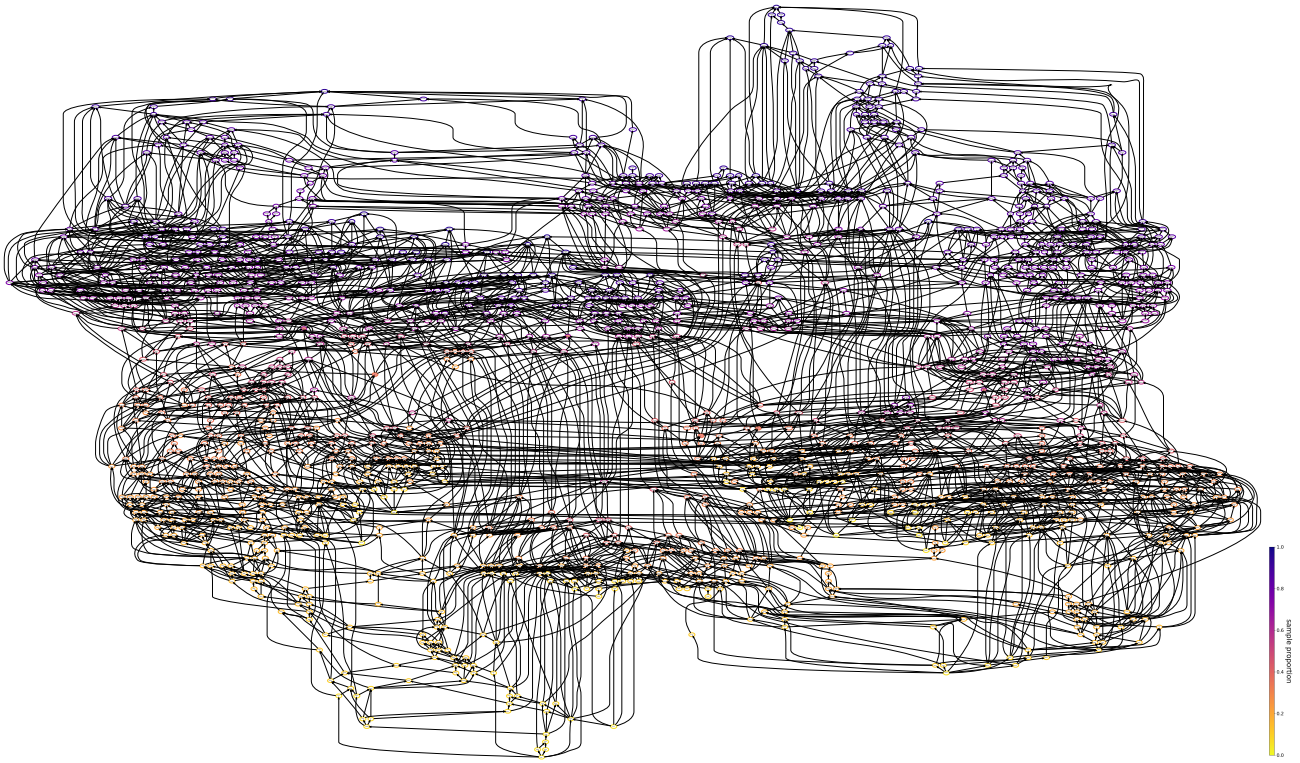


Figure 14. A neuron dependency graph for AllNLI dataset.

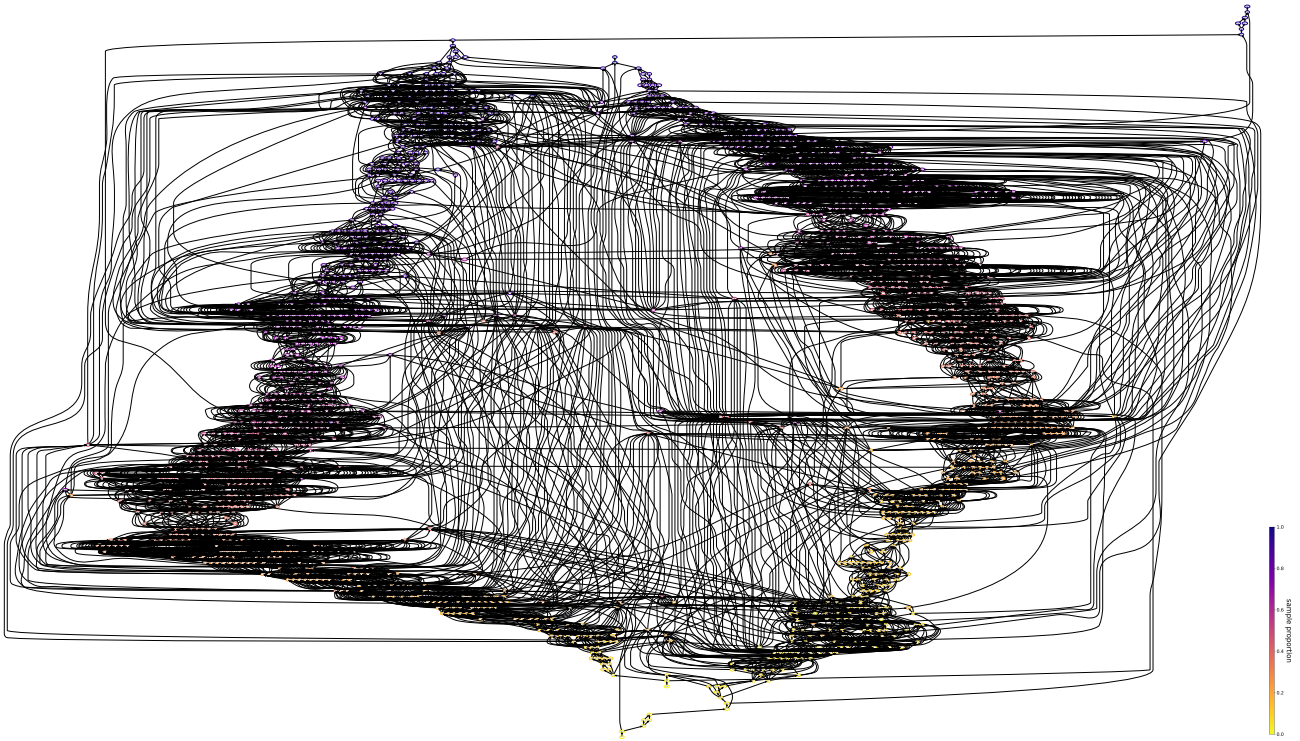


Figure 15. A neuron dependency graph for Devign dataset.