The Non-Linear Representation Dilemma: Is Causal Abstraction Enough for Mechanistic Interpretability?

Denis Sutter,[ETH] **Julian Minder,**[EPFL] **Thomas Hofmann,**[ETH] **Tiago Pimentel** [ETH]

[ETH]ETH Zürich   [EPFL]EPFL

densutter@ethz.ch, julian.minder@epfl.ch, {thomas.hofmann, tiago.pimentel}@inf.ethz.ch

 densutter/non-linear-representation-dilemma

Abstract

The concept of **causal abstraction** got recently popularised to demystify the opaque decision-making processes of machine learning models; in short, a neural network can be **abstracted** as a higher-level algorithm if there exists a function which allows us to map between them. Notably, most interpretability papers implement these maps as linear functions, motivated by the **linear representation hypothesis**: the idea that features are encoded linearly in a model's representations. However, this linearity constraint is not required by the definition of causal abstraction. In this work, we critically examine the concept of causal abstraction by considering arbitrarily powerful alignment maps. In particular, we prove that under reasonable assumptions, *any* neural network can be mapped to *any* algorithm, rendering this unrestricted notion of causal abstraction trivial and uninformative. We complement these theoretical findings with empirical evidence, demonstrating that it is possible to perfectly map models to algorithms even when these models are incapable of solving the actual task; e.g., on an experiment using randomly initialised language models, our alignment maps reach 100% interchange-intervention accuracy on the indirect object identification task. This raises the **non-linear representation dilemma**: if we lift the linearity constraint imposed to alignment maps in causal abstraction analyses, we are left with no principled way to balance the inherent trade-off between these maps' complexity and accuracy. Together, these results suggest an answer to *our* title's question: causal abstraction is *not* enough for mechanistic interpretability, as it becomes vacuous without assumptions about how models encode information. Studying the connection between this information-encoding assumption and causal abstraction should lead to exciting future work.

1   Introduction

The increasing popularity of machine learning (ML) models has led to a surge in their deployment across various industries. However, the lack of interpretability in these models raises significant concerns, particularly in high-stakes applications where understanding the decision-making process is crucial (Goodman and Flaxman, 2017; Tonekaboni et al., 2019; Zhu et al., 2020; Gao and Guan, 2023). Unsurprisingly, this opacity has motivated a multitude of research on mechanistic (or causal) interpretability, which tries to analyse and understand the hidden algorithms that underlie these models (Olah et al., 2020; Elhage et al., 2021; Mueller et al., 2024; Ferrando et al., 2024; Sharkey et al., 2025).

A promising approach to address this challenge is **causal abstraction** (Beckers and Halpern, 2019; Geiger et al., 2024a), which tries to map the behaviour of a model to a higher-level (and conceptually simpler) algorithm which solves the task. At the core of this concept is the idea that if an intervention is found to change a model's behaviour in a way that aligns with a specific algorithm, then that algorithm can be considered implemented by the model. Recent research, however, has raised considerable issues with this approach (e.g., Makelov et al., 2024; Mueller, 2024; Sun et al., 2025). Among those, Méloux et al. (2025) notes that a model's causal abstraction is not necessarily unique, showing that many algorithms can be aligned to the same neural network. Additionally, most work on causal
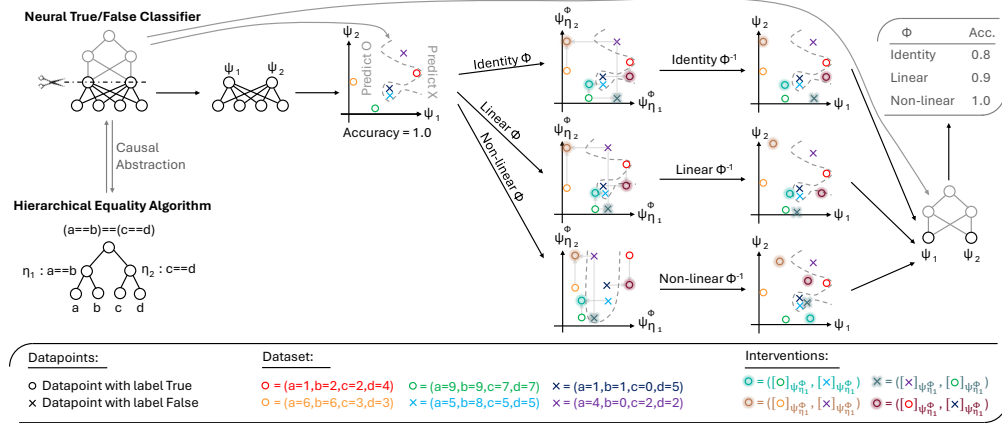
Figure 1: A visualisation of what happens when analysing causal abstractions with increasingly complex alignment maps ϕ. The more complex ϕ is, the higher the intervention accuracy—and, consequently, the stronger the algorithm–DNN alignment. In Theorem 1, we show that given arbitrarily complex alignment maps, we can always find a perfect alignment (under reasonable assumptions).

abstraction (Wu et al., 2023; Geiger et al., 2024b; Minder et al., 2025; Sun et al., 2025) implicitly assumes information is linearly encoded in models' representations, relying on the **linear representation hypothesis** (Alain and Bengio, 2016; Bolukbasi et al., 2016). Linearity, however, is not required by the definition of causal abstraction (Beckers and Halpern, 2019) and increasing evidence suggests that not all representations may be linearly encoded (White et al., 2021; Olah and Jermyn, 2024; Mueller, 2024; Csordás et al., 2024; Engels et al., 2025a,b; Kantamneni and Tegmark, 2025).

In this paper, we first prove that, once we drop the linearity constraint, *any* model can be perfectly mapped to *any* algorithm under relatively weak assumptions—e.g., hidden activation's input-injectivity and output-surjectivity, which we will define formally. This renders causal abstraction vacuous when used without constraints. If we restrict alignment maps to only consider, e.g., linear functions, this problem does not arise though. It follows that causal abstraction implicitly relies on strong assumptions about how features are encoded in deep neural networks (DNNs), and becomes trivial without such assumptions. This puts us at an impasse: we may want to rely on stronger notions of causal abstraction which may leverage non-linearly encoded information, but this may make our analyses vacuous; we call this the **non-linear representation dilemma** (schematised in Fig. 1).

To empirically validate our theoretical results, we reproduce the original distributed alignment search (DAS) experiments (Geiger et al., 2024b), but while leveraging more complex alignment maps. We find that key empirical patterns they observed—such as the first layer being easier to map to the tested algorithms in a hierarchical equality task—vanish when we use more powerful maps. Additionally, we find that we can achieve over 80% interchange intervention accuracy (IIA) using non-linear alignment maps in randomly initialised models. Extending our experiments to language models from the Pythia suite (Biderman et al., 2023), we show that near-perfect maps can be found for randomly initialised models in the indirect object identification (IOI) task (Wang et al., 2023); notably, as training progresses, the complexity of the alignment maps needed to achieve perfect IIA in this task decreases. Overall, our results show that causal abstraction, while promising in theory, suffers from a fundamental limitation: without *a priori* constraints on the used alignment maps, it becomes vacuous as a method for understanding neural networks.

2 Background

In this section, we formally define algorithms (§2.1) and deep neural networks (§2.2). These will then be used to define a causal abstraction (§3). First, we formalise a task as a function $\mathtt{T} : \mathcal{X} \to \mathcal{Y}$, where $\mathbf{x} \in \mathcal{X}$ represents a set of input features and $\mathbf{y} \in \mathcal{Y}$ denotes the corresponding output.

2.1 Algorithms

Given a task $\mathtt{T}$, we may hypothesise different ways it can be solved. We term each such hypothesis an algorithm[1] $\mathtt{A}$, which we represent as a deterministic causal model—a directed acyclic graph that

---

[1] "Algorithm" here need not match a formal definition as, e.g., the considered functions may be uncomputable.

implements a function $f_{\mathtt{A}} : \mathcal{X} \to \mathcal{Y}$.[2] These causal models have a set of nodes $\boldsymbol{\eta}_{\mathtt{all}}$ which can be decomposed into three disjoint sets: (i) input nodes $\boldsymbol{\eta}_{\mathbf{x}}$ representing elements in $\mathbf{x}$, (ii) output nodes $\boldsymbol{\eta}_{\mathbf{y}}$ representing elements in $\mathbf{y}$, and (iii) inner nodes $\boldsymbol{\eta}_{\mathtt{inner}}$ representing intermediate variables used in the computation of $f_{\mathtt{A}}$. As we focus on acyclic causal models, the edges in this graph induce a partial ordering on nodes $\boldsymbol{\eta}_{\mathbf{x}} \prec \boldsymbol{\eta}_{\mathtt{inner}} \prec \boldsymbol{\eta}_{\mathbf{y}}$. Let v_η denote the value held by node η, and let $\mathbf{v}_{\boldsymbol{\eta}}$ denote values taken by the set of nodes $\boldsymbol{\eta}$. The set of incoming edges to a node η represent a direct causal relationship between that node and its parents $\mathtt{par}_{\mathtt{A}}(\eta)$, denoted: $v_\eta = f_{\mathtt{A}}^\eta(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta)})$. We can compute algorithm $f_{\mathtt{A}}$ by iteratively solving the value of its nodes while respecting their partial ordering:

$$\mathbf{v}_{\boldsymbol{\eta}_{\mathbf{x}}} \stackrel{\text{def}}{=} \mathbf{x}, \qquad \underset{\eta \in \boldsymbol{\eta}_{\mathtt{inner}} \cup \boldsymbol{\eta}_{\mathbf{y}}}{\forall} v_\eta = f_{\mathtt{A}}^\eta(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta)}), \qquad f_{\mathtt{A}}(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{v}_{\boldsymbol{\eta}_{\mathbf{y}}} \qquad (1)$$

where we define $\mathbf{v}_{\boldsymbol{\eta}_{\mathbf{x}}}$ to be the input value $\mathbf{x}$, and take the value of the output nodes $\mathbf{v}_{\boldsymbol{\eta}_{\mathbf{y}}}$ as our algorithm's output. Importantly, for an algorithm $\mathtt{A}$ to represent a task $\mathtt{T}$, its output under "normal" operation must be $f_{\mathtt{A}}(\mathbf{x}) = \mathtt{T}(\mathbf{x})$. For an example of a task and related algorithms, see App. I.1.

These causal models, however, allow us to go beyond "normal" operations and investigate the behaviour of our algorithm under counterfactual settings. We can, for instance, investigate what its behaviour would be if we enforce a node η''s value to be a constant $v_{\eta'} = c$, which we write as:

$$\mathbf{v}_{\boldsymbol{\eta}_{\mathbf{x}}} = \mathbf{x}, \qquad v_{\eta'} = c, \qquad \underset{\eta \in (\boldsymbol{\eta}_{\mathtt{inner}} \cup \boldsymbol{\eta}_{\mathbf{y}}) \setminus \{\eta'\}}{\forall} v_\eta = f_{\mathtt{A}}^\eta(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta)}), \qquad f_{\mathtt{A}}(\mathbf{x}, (v_{\eta'} \leftarrow c)) = \mathbf{v}_{\boldsymbol{\eta}_{\mathbf{y}}} \quad (2)$$

Now, let $f_{\mathtt{A}}^{:\eta}(\mathbf{x})$ represent a function which runs our algorithm with input $\mathbf{x}$ until it reaches node η, outputting its value v_η. We can use such interventions to investigate the behaviour of algorithm $\mathtt{A}$ under input $\mathbf{x}$, when node η' is forced to assume the value it would have under $\mathbf{x}'$ as: $f_{\mathtt{A}}(\mathbf{x}, (v_{\eta'} \leftarrow f_{\mathtt{A}}^{:\eta}(\mathbf{x}')))$. Now, let $\mathbf{I}_{\mathtt{A}}$ be a multi-node intervention, e.g., $\mathbf{I}_{\mathtt{A}} = (\mathbf{v}_{\eta'} \leftarrow \mathbf{c}')$ where $\boldsymbol{\eta}' = [\eta', \eta'']$ and $\mathbf{c}' = [f_{\mathtt{A}}^{:\eta'}(\mathbf{x}'), c]$. We can observe how our model operates under those interventions by running $f_{\mathtt{A}}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})$. See App. B for a pseudo-code implementation.

2.2 Deep Neural Networks

Deep neural networks (DNNs) are the driving force behind recent advances in ML and can be defined as a sequence of functions $f_{\mathtt{N}}^\ell : \mathcal{H}_{\boldsymbol{\psi}_\ell} \to \mathcal{H}_{\boldsymbol{\psi}_{\ell+1}}$, where $\boldsymbol{\psi}_\ell$ denotes the set of neurons in layer ℓ and $\mathcal{H}_{\boldsymbol{\psi}_\ell}$ is the corresponding vector space. A DNN $\mathtt{N}$ with L layers can be specified as follows:

$$\mathbf{h}_{\boldsymbol{\psi}_1} = f_{\mathtt{N}}^0(\mathbf{x}), \qquad \mathbf{h}_{\boldsymbol{\psi}_{\ell+1}} = f_{\mathtt{N}}^\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell}), \qquad p_{\mathtt{N}}(\mathbf{y} \mid \mathbf{x}) = f_{\mathtt{N}}^L(\mathbf{h}_{\boldsymbol{\psi}_L}) \qquad (3)$$

where $\mathbf{h}_{\boldsymbol{\psi}_\ell}$ denotes the vector of activations for neurons $\boldsymbol{\psi}_\ell$. We focus on DNNs with real-valued neurons and probabilistic outputs, so that $\mathcal{H}_{\boldsymbol{\psi}_0} = \mathcal{X}$, $\mathcal{H}_{\boldsymbol{\psi}_\ell} = \mathbb{R}^{|\boldsymbol{\psi}_\ell|}$ and $\mathcal{H}_{\boldsymbol{\psi}_{L+1}} = \Delta^{|\mathcal{Y}|-1}$. We define $f_{\mathtt{N}}^{\boldsymbol{\psi}'}(\mathbf{x}')$ as the function that computes the activations of the subset of neurons $\boldsymbol{\psi}'$ when the network is evaluated on input $\mathbf{x}'$. In particular, $f_{\mathtt{N}}^{\boldsymbol{\psi}_\ell}(\mathbf{x})$ returns the activations at layer ℓ for input $\mathbf{x}$. Thus, the standard computation of the DNN corresponds to evaluating $p_{\mathtt{N}}(\mathbf{y} \mid \mathbf{x}) = f_{\mathtt{N}}^{\boldsymbol{\psi}_{L+1}}(\mathbf{x})$. This formulation allows us to instantiate common architectures, such as multi-layer perceptrons (MLPs) or transformers, by specifying the form of each $f_{\mathtt{N}}^\ell$ and the structure of the neuron sets $\boldsymbol{\psi}_\ell$. The parameters of these models are typically optimised to minimise the cross-entropy loss.

Notably, similarly to the algorithms above, a DNN's architecture induces a partial ordering on its neurons, respecting the order in which they are computed $\boldsymbol{\psi}_0 \prec \boldsymbol{\psi}_\ell \prec \boldsymbol{\psi}_L$. We can thus analogously define a **DNN intervention** as follows: given a set of neurons $\boldsymbol{\psi}$ in the network and a corresponding set of values $\mathbf{c}_{\boldsymbol{\psi}}$, we denote the intervention by $f_{\mathtt{N}}^{\boldsymbol{\psi}_{L+1}}(\mathbf{x}, (\mathbf{h}_{\boldsymbol{\psi}} \leftarrow \mathbf{c}_{\boldsymbol{\psi}}))$. This notation means that, during the forward computation of the DNN, the activations of the neurons in $\boldsymbol{\psi}$ are fixed to the specified values $\mathbf{c}_{\boldsymbol{\psi}}$, while the rest of the network operates as usual.

3 Causal Abstraction

To define causal abstraction, we will base ourselves on the definitions in Beckers and Halpern (2019) and Geiger et al. (2024a). Let an **abstraction map** be defined as $\tau : \mathcal{H} \to \mathcal{N}$, where $\mathcal{H}$ and $\mathcal{N}$ are, respectively, the Cartesian products of the hidden state-spaces in a neural network $\mathtt{N}$ (i.e., $\boldsymbol{\psi}_{\mathtt{int}} \stackrel{\text{def}}{=} \boldsymbol{\psi}_{1:}$), and the node value-spaces in an algorithm $\mathtt{A}$ (i.e., $\boldsymbol{\eta}_{\mathtt{int}} \stackrel{\text{def}}{=} \boldsymbol{\eta}_{\mathtt{inner}} \cup \boldsymbol{\eta}_{\mathbf{y}}$), both excluding the inputs $\boldsymbol{\psi}_0$ and $\boldsymbol{\eta}_{\mathbf{x}}$. In words, an abstraction map translates the inner states of a neural network into an algorithms' inner states. Now, consider the DNN intervention $\mathbf{I}_{\mathtt{N}} = \mathbf{h}_{\boldsymbol{\psi}} \leftarrow \mathbf{c}_{\boldsymbol{\psi}}$ and the algorithm

---

[2]Geiger et al. (2024a) also considers cyclic deterministic causal models and Beckers and Halpern (2019) considers cyclic and stochastic causal models. We leave the expansion of our work to such models for future work.

intervention $\mathbf{I}_{\mathtt{A}} = \mathbf{v}_{\boldsymbol{\eta}} \leftarrow \mathbf{c}_{\boldsymbol{\eta}}$. Further, let $\mathcal{H}_{\mathbf{h}_{\boldsymbol{\psi}}=\mathbf{c}_{\boldsymbol{\psi}}}$ be the set of states in a DNN for which $\mathbf{h}_{\boldsymbol{\psi}} = \mathbf{c}_{\boldsymbol{\psi}}$ holds, and equivalently for $\mathcal{N}_{\mathbf{v}_{\boldsymbol{\eta}}=\mathbf{c}_{\boldsymbol{\eta}}}$. Under abstraction map τ, we can define an intervention map as:

$$\omega_\tau(\mathbf{h}_{\boldsymbol{\psi}} \leftarrow \mathbf{c}_{\boldsymbol{\psi}}) = \left\{ \begin{array}{ll} \mathbf{v}_{\boldsymbol{\eta}} \leftarrow \mathbf{c}_{\boldsymbol{\eta}} & \texttt{if } \mathcal{N}_{\mathbf{v}_{\boldsymbol{\eta}}=\mathbf{c}_{\boldsymbol{\eta}}} = \{\tau(\mathbf{h}) \mid \mathbf{h} \in \mathcal{H}_{\mathbf{h}_{\boldsymbol{\psi}}=\mathbf{c}_{\boldsymbol{\psi}}}\} \\ \texttt{undefined} & \texttt{else} \end{array} \right. \quad (4)$$

Intuitively, ω_τ maps a DNN intervention $\mathbf{I}_{\mathtt{N}}$ to an algorithmic one $\mathbf{I}_{\mathtt{A}}$ if the sets of states they induce on $\mathtt{N}$ and $\mathtt{A}$, respectively, are the same. Further, let $\mathcal{I}_{\mathtt{A}}$ be a set of interventions $\mathbf{I}_{\mathtt{A}}$ which can be performed on algorithm $\mathtt{A}$. We can use ω_τ to derive a set of equivalent DNN interventions as:

$$\mathcal{I}_{\mathtt{N}} = \{\mathbf{h}_{\boldsymbol{\psi}} \leftarrow \mathbf{c}_{\boldsymbol{\psi}} \mid \mathbf{v}_{\boldsymbol{\eta}} \leftarrow \mathbf{c}_{\boldsymbol{\eta}} \in \mathcal{I}_{\mathtt{A}}, \omega_\tau(\mathbf{h}_{\boldsymbol{\psi}} \leftarrow \mathbf{c}_{\boldsymbol{\psi}}) = \mathbf{v}_{\boldsymbol{\eta}} \leftarrow \mathbf{c}_{\boldsymbol{\eta}}\} \quad (5)$$

Given these definitions, we now put forward a first notion of causal abstraction.

Definition 1 (**from Beckers and Halpern, 2019**). *An algorithm $\mathtt{A}$ is a $\boldsymbol{\tau}$-abstraction of a neural network $\mathtt{N}$ iff: τ is surjective; $\mathcal{I}_{\mathtt{A}} = \omega_\tau(\mathcal{I}_{\mathtt{N}})$;*[3] *and there exists a surjective $\tau_{\boldsymbol{\eta_x}}$ such that:*

$$\forall_{\substack{\mathbf{x} \in \mathcal{X} \\ \mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}}} : \tau(f_{\mathtt{N}}^{\boldsymbol{\psi}_{\texttt{int}}}(\mathbf{x}, \mathbf{I}_{\mathtt{N}})) = f_{\mathtt{A}}^{:\boldsymbol{\eta}_{\texttt{int}}}(\tau_{\boldsymbol{\eta_x}}(\mathbf{x}), \mathbf{I}_{\mathtt{A}}) \quad \texttt{where } \mathbf{I}_{\mathtt{A}} = \omega_\tau(\mathbf{I}_{\mathtt{N}}) \quad (6)$$

In words, the first condition in this definition enforces that all states in an algorithm are needed to abstract the DNN, while the second and third enforce that interventions in the algorithm have the same effect as interventions in the DNN. We further say that $\mathtt{A}$ is a **strong τ-abstraction** of $\mathtt{N}$ if it is a τ-abstraction and $\mathcal{I}_{\mathtt{A}}$ is maximal, meaning that any intervention is allowed on algorithm $\mathtt{A}$. However, while strong τ-abstractions give us a notion of equivalence between algorithms and DNNs, the maps τ may be highly entangled and provide little intuition about the DNN's behaviour. To ensure algorithmic information is disentangled in the DNN, we say τ is a **constructive abstraction map** if there exists a partition of $\mathtt{N}$'s neurons $\{\boldsymbol{\psi}_{\boldsymbol{\eta}} \mid \boldsymbol{\eta} \in \boldsymbol{\eta}_{\texttt{int}}\} \cup \{\boldsymbol{\psi}_{\perp}\}$—where $\boldsymbol{\psi}_{\boldsymbol{\eta}}$ are non-empty—and there exist maps $\tau_{\boldsymbol{\eta}}$ such that τ is equivalent to the block-wise application of $\tau_{\boldsymbol{\eta}}(\mathbf{h}_{\boldsymbol{\psi}_{\boldsymbol{\eta}}})$. In other words, constructive abstraction maps compute the value $v_{\boldsymbol{\eta}}$ of each node $\boldsymbol{\eta} \in \boldsymbol{\eta}_{\texttt{int}}$ in $\mathtt{A}$ using non-overlapping sets of neurons $\boldsymbol{\psi}_{\boldsymbol{\eta}}$ from $\mathtt{N}$, with set $\boldsymbol{\psi}_{\perp}$ being left unused. We now define a second notion of causal abstraction.

Definition 2 (**from Beckers and Halpern, 2019**). *An algorithm $\mathtt{A}$ is a **constructive abstraction** of a neural network $\mathtt{N}$ iff there exists an τ: for which $\mathtt{A}$ is a strong τ-abstraction of $\mathtt{N}$; and τ is constructive.*

As we will deal with algorithm–DNN pairs which share the same input and output spaces, we will impose an additional constraint on τ—one that is not present in Beckers and Halpern's (2019) definition. Namely, we restrict: $\boldsymbol{\psi}_{\boldsymbol{\eta_x}}$ to be the neurons in layer zero and $\tau_{\boldsymbol{\eta_x}}$ to be the identity; and $\boldsymbol{\psi}_{\boldsymbol{\eta_y}}$ to be the neurons in layer $L + 1$ and $\tau_{\boldsymbol{\eta_y}}$ to be the argmax operation.[4]

3.1 Information Encoding in Neural Networks

The definition of constructive abstraction above maps non-overlapping sets of neurons in $\mathtt{N}$, i.e., $\boldsymbol{\psi}_{\boldsymbol{\eta}}$, to nodes in $\mathtt{A}$, i.e., $\boldsymbol{\eta}$. However, much research in ML interpretability highlights that concept information is not always neuron-aligned and that neurons are often polysemantic (Olah et al., 2017, 2020; Arora et al., 2018; Elhage et al., 2022). In fact, there is a large debate about how information is encoded in DNNs. We highlight what we see as the three most prominent hypotheses here.

Definition 3. *The **privileged bases hypothesis** (Elhage et al., 2023) argues that neurons form privileged bases to encode information in a neural network.*

Most evidence in favour of this hypothesis comes from indirect evidence: i.e., the presence of neuron-aligned outlier features or activations in DNNs (Kovaleva et al., 2021; Elhage et al., 2023; He et al., 2024; Sun et al., 2024). Going back to 2015, Karpathy (2015) already showed that a single neuron in a language model could carry meaningful information. Importantly, this hypothesis is consistent with the notion of constructive abstraction above, as it argues each node η's information should be encoded in separate, non-overlapping sets of neurons. Several researchers, however, question the special status of neurons assumed by this hypothesis, assuming instead that information is encoded in linear subspaces of the representation space, of which neurons are only a special case.

Definition 4. *The **linear representation hypothesis** (Alain and Bengio, 2016) argues that information is encoded in linear subspaces of a neural network.*

---

[3] We overload function ω_τ here, with $\omega_\tau(\mathcal{I}_{\mathtt{N}})$ simply applying ω_τ elementwise to the interventions in set $\mathcal{I}_{\mathtt{N}}$.

[4] We note that this implies algorithm $\mathtt{A}$ and network $\mathtt{N}$ must have the same outputs on the input set $\mathcal{X}$.

A large literature has developed, backed by the linear representation hypothesis, including: concept erasure methods (Ravfogel et al., 2020, 2022), probing methodologies (Elazar et al., 2021; Ravfogel et al., 2021; Lasri et al., 2022), and work on disentangling activations (Yun et al., 2021; Elhage et al., 2022; Huben et al., 2024; Templeton et al., 2024). Some, however, still question this idea that all information must be encoded linearly in DNNs: as neural networks implement non-linear functions, there is no a priori reason for why information should be linearly encoded in them (Conneau et al., 2018; Hewitt and Liang, 2019; Pimentel et al., 2020b,a, 2022). Further, recent research presents strong evidence that some concepts are indeed non-linearly encoded in DNNs (White et al., 2021; Pimentel et al., 2022; Olah and Jermyn, 2024; Csordás et al., 2024; Engels et al., 2025a,b; Kantamneni and Tegmark, 2025).

Definition 5. *The **non-linear representation hypothesis** (Pimentel et al., 2020b) argues that information may be encoded in arbitrary non-linear subspaces of a neural network.*

3.2 Distributed Causal Abstractions

Following the discussion above, the definition of constructive abstraction may be too strict, as it assumes τ must decompose across neurons—and thus that node information is encoded in non-overlapping neurons. With this in mind, Geiger et al. (2024a,b) proposed the notion of distributed interventions: they expose the subspaces where node information is encoded in a DNN $\mathbb{N}$ by applying a bijective function to its hidden states; this function's output is then itself a constructive abstraction of algorithm $\mathbb{A}$. Here, we make this notion a bit more formal.

We define τ as a **distributed abstraction map** if the following two conditions hold. First, there exists a bijective function ϕ—termed here an **alignment map**—that maps the inner neurons ψ_{int} of $\mathbb{N}$ block-wise to an equal-sized set of **latent variables** ψ_{int}^{ϕ}, in a manner that respects the partial ordering of computations in the network. Specifically, for each layer ℓ, there exists a bijection $\phi_\ell : \mathbb{R}^{|\psi_\ell|} \to \mathbb{R}^{|\psi_\ell|}$ on its neurons such that ϕ is defined as the concatenation of these layer-wise bijections. Similarly to the neurons' activation $\mathbf{h}_\psi$, we will denote latent variables as $\mathbf{h}_{\psi^\phi} = \phi(\mathbf{h}_\psi)$. Second, there exists a partition $\{\psi_\eta^\phi \mid \eta \in \boldsymbol{\eta}_{\text{int}}\} \cup \{\psi_\perp^\phi\}$ of the resulting latent variables ψ_{int}^ϕ— where ψ_η^ϕ are non-empty—and a set of maps τ_η such that τ is equivalent to the block-wise application of $\tau_\eta(\mathbf{h}_{\psi_\eta^\phi})$. In words, a distributed abstraction map computes the value v_η of each node $\eta \in \boldsymbol{\eta}_{\text{int}}$ in $\mathbb{A}$ using non-overlapping partitions of latent variables ψ_η^ϕ from $\mathbb{N}$, with partition $\psi_\perp^\phi$ remaining unused.

Given an alignment map ϕ, we can perform distributed interventions: $\mathbf{h}_{\psi_\eta^\phi} \leftarrow \mathbf{c}_{\psi_\eta^\phi}$. These interventions are performed by first mapping the hidden state $\mathbf{h}_\psi$ to the latent variables $\mathbf{h}_{\psi^\phi} = \phi(\mathbf{h}_\psi)$, intervening on a subset ψ_η^ϕ by replacing $\mathbf{h}_{\psi_\eta^\phi}$ with desired values $\mathbf{c}_{\psi_\eta^\phi}$, and then mapping these intervened latent variables back to the original neuron base via $\mathbf{h}'_\psi = \phi^{-1}(\mathbf{h}'_{\psi^\phi})$. Thus, interventions are applied in the latent space defined by ϕ, generalising privileged-bases interventions to arbitrary (possibly non-linear) subspaces. We are now in a position to define distributed abstractions.

Definition 6. *An algorithm $\mathbb{A}$ is a **distributed abstraction** of a neural network $\mathbb{N}$ iff there exists an τ: for which $\mathbb{A}$ is a strong τ-abstraction of $\mathbb{N}$; and τ is a distributed abstraction map.*

Finally, we note that the set of all possible interventions $\mathcal{I}_{\mathbb{A}}$ and $\mathcal{I}_{\mathbb{N}}$ may be hard to analyse in practice. Geiger et al. (2024b) thus restrict their analyses to what we term here **input-restricted interventions**: the set of interventions which are themselves producible by a set of other input-restricted interventions. In other words, we restrict interventions $\mathbf{h}_{\psi'} \leftarrow \mathbf{c}_{\psi'}$ (where $\psi' \subseteq \psi_{\text{int}}$ or $\psi' \subseteq \psi_{\text{int}}^\phi$) and $\mathbf{v}_{\eta'} \leftarrow \mathbf{c}_{\eta'}$ to $\mathbf{c}_{\psi'}$ and $\mathbf{c}_{\eta'}$ which are a product of other input-restricted interventions, e.g., $\mathbf{c}_{\psi'} = f_{\mathbb{N}}^{:\psi'}(\mathbf{x}')$ or $\mathbf{c}_{\eta'} = f_{\mathbb{A}}^{:\eta}(\mathbf{x}', (\mathbf{v}_{\eta''} \leftarrow f_{\mathbb{A}}^{:\eta}(\mathbf{x}'')))$. This leads to the definition of **input-restricted τ-abstraction**: a weakened notion of strong τ-abstraction, where intervention sets are restricted to input-restricted interventions. Finally, we define an analogous version of distributed abstraction, which is input-restricted; this is the notion typically used in practice by machine learning practitioners.

Definition 7 (**inspired by Geiger et al., 2024b**). *An algorithm $\mathbb{A}$ is an **input-restricted distributed abstraction** of a neural network $\mathbb{N}$ iff there exists an τ: for which $\mathbb{A}$ is an input-restricted τ-abstraction of $\mathbb{N}$; and τ is a distributed abstraction map.*

A visual representation of how these causal abstraction definitions are related is given in App. D as Fig. 6. Finally, we further introduce **input-restricted $\mathcal{V}$-abstractions**: input-restricted distributed abstractions for which we restrict alignment maps ϕ to be in a specific variational family $\mathcal{V}$. The case of linear alignment maps, will be particular important here—as it relates to the linear representation hypothesis—and we will thus explicitly label it as **input-restricted linear abstraction**.

3.3 Finding Distributed Abstractions

How do we evaluate if an algorithm is an input-restricted distributed abstraction of a DNN? Geiger et al. (2024b) proposes an efficient method to answer this, called distributed alignment search (DAS). Before applying DAS, one must assume a partitioning $\{\psi_\eta^\phi \mid \eta \in \eta_{\texttt{int}}\} \cup \{\psi_\perp^\phi\}$ which remains fixed during the method's application; we term $|\psi_\eta^\phi|$ the **intervention size**. The principle behind DAS is then to leverage the constraint on $\tau_{\eta_\mathbf{y}}$, which is fixed as: $\mathbf{v}_{\eta_\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} p_{\texttt{N}}(\mathbf{y} \mid \mathbf{x})$ since $p_{\texttt{N}}(\mathbf{y} \mid \mathbf{x}) = \mathbf{h}_{\psi_{L+1}}$. Given this constraint, we can initialise a parametrised function ϕ, which we train to predict this equality under possible interventions; this is done via gradient descent, minimising the cross-entropy between the DNN and the algorithm. Specifically, we first select a set of nodes to be intervened $\overline{\eta} \in \mathcal{P}(\eta_{\texttt{inner}})$, where $\mathcal{P}$ is a function that takes the powerset of a set, along with corresponding counterfactual inputs $\mathbf{x}_\eta \in \mathcal{X}$ for each $\eta \in \overline{\eta}$ and a base input $\mathbf{x}_\emptyset \in \mathcal{X}$. We then define the following two interventions:

$$\mathbf{I}_{\texttt{N}} = \left[\mathbf{h}_{\psi_\eta^\phi}\right]_{\eta\in\overline{\eta}} \leftarrow \left[f_{\texttt{N}}^{\psi_\eta^\phi}(\mathbf{x}_\eta)\right]_{\eta\in\overline{\eta}} \quad \text{and} \quad \mathbf{I}_{\texttt{A}} = \left[\mathbf{v}_\eta\right]_{\eta\in\overline{\eta}} \leftarrow \left[f_{\texttt{A}}^{:\eta}(\mathbf{x}_\eta)\right]_{\eta\in\overline{\eta}} \tag{7}$$

Finally, we run our algorithm under base input $\mathbf{x}_\emptyset$ and intervention $\mathbf{I}_{\texttt{A}}$ to get a ground truth output: $\mathbf{y} = f_{\texttt{A}}(\mathbf{x}_\emptyset, \mathbf{I}_{\texttt{A}})$. Repeating this process N times, we build a dataset $\mathcal{D} = \{(\mathbf{x}_\emptyset^{(n)}, \mathbf{I}_{\texttt{N}}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ on which we can train the alignment map ϕ such that the DNN matches the algorithm:

$$\mathcal{L} = -\sum_{(\mathbf{x}_\emptyset^{(n)}, \mathbf{I}_{\texttt{N}}^{(n)}, \mathbf{y}^{(n)})\in\mathcal{D}} \log p_{\texttt{N}}^\phi(\mathbf{y}^{(n)} \mid \mathbf{x}_\emptyset^{(n)}, \mathbf{I}_{\texttt{N}}^{(n)}), \quad \text{where} \quad p_{\texttt{N}}^\phi(\mathbf{y} \mid \mathbf{x}_\emptyset, \mathbf{I}_{\texttt{N}}) = f_{\texttt{N}}(\mathbf{x}_\emptyset, \mathbf{I}_{\texttt{N}}) \tag{8}$$

Notably, DAS mostly ignores how function τ is constructed, relying solely on the assumed definition of $\tau_{\eta_\mathbf{y}}$. Finding a low-loss alignment map ϕ is then assumed as sufficient evidence that $\texttt{A}$ is an input-restricted distributed abstraction of $\texttt{N}$.

4 Unbounded Abstractions are Vacuous

In this section, we provide our main theorem: that under reasonable assumptions, *any* algorithm $\texttt{A}$ can be shown to be an input-restricted distributed abstraction of *any* DNN $\texttt{N}$, making this notion of causal abstraction vacuous. To show that, we need a few assumptions (for their formal definition, see App. F). Our first assumption (Assump. 1) is that we have a **countable input-space** $\mathcal{X}$. While this may not hold in general, it holds for common applications such as language modelling (where the input-space is the countably infinite set of finite strings) or computer vision (where the input-space is a countable union of pixels, which can assume a finite set of values). The second assumption (Assump. 2) is that DNNs are **input-injective in all layers**: i.e., $f_{\texttt{N}}^{\psi_\ell}$ is injective for all layers. This guarantees that no information about a DNN's input $\mathbf{x}$ is lost when computing the hidden states $\mathbf{h}_{\psi_\ell}$. This assumption is also present in prior work (e.g., Pimentel et al., 2020b) and we show in App. G— assuming real-valued weights and activations—that this is almost surely true for transformers at initialisation.[5] Due to floating point precision and neural collapse (Papyan et al., 2020), it is likely not to hold fully in practice; however, it still seems to be well-approximated in many empirical settings (Morris et al., 2023, and App. H). The third assumption (Assump. 3) is **strict output-surjectivity in all layers**. This assumption guarantees that in each layer there is at least one choice of $\mathbf{h}_{\psi_\ell}$ that will produce the desired output. Notably, this assumption may not hold in theory, due to issues like the softmax-bottleneck (Yang et al., 2018). In practice, however, even with large vocabulary sizes, it seems that almost all outputs can still be produced by language models (Grivas et al., 2022) which is sufficient for these DNNs to be abstracted by many algorithms. Our fourth assumption (Assump. 4) is that the **algorithm $\texttt{A}$ and DNN $\texttt{N}$ have matchable partial-orderings**, meaning that there is a partitioning of neurons in $\texttt{N}$ which would match the partial-ordering of nodes in $\texttt{A}$; this is likely to be the case for most reasonable algorithms given the size of state-of-the-art deep neural networks. Finally, our last assumption (Assump. 5) is that the **DNN $\texttt{N}$ solves the given task** $\texttt{T}$. We believe this assumption to be reasonable, as it would be impractical in practice to evaluate a neural network that does not perform the task correctly.[6] Given these assumptions, we can now present our main theorem.

Theorem 1. *Given any algorithm $\texttt{A}$ and any neural network $\texttt{N}$ such that Assumps. 1 to 5 hold, we can show that $\texttt{A}$ is an input-restricted distributed abstraction of $\texttt{N}$.*

Proof. We refer to App. F for the proof. $\qquad\square$

---

[5]Also see Nikolaou et al. (2025), who show almost sure injectivity holds for transformers throughout training.

[6]If the model does not solve the task, perfect IIA is impossible since non-intervened inputs yield incorrect outputs. Thus, assuming the model solves the task is necessary. In practice, however, even when the DNN is imperfect, an alignment map could produce correct outputs for all intervened inputs, achieving near-perfect IIA scores.

5 Experimental Setup

Building on the previous section's proof that alignment maps between DNNs and algorithms always exist, we now demonstrate their practical learnability and how increasingly complex alignment maps reveal various causal abstractions for different tasks, even on DNNs that do not solve them.

Alignment Maps. To assess how complexity impacts causal-abstraction analyses, we explore three ways to parameterise ϕ. First, we will consider the simplest **identity maps**: $\phi^{\texttt{id}}(\mathbf{h}) = \mathbf{h}$. This is the least expressive ϕ we consider, and if we find that A abstracts N under this map, we can say that A is a constructive abstraction of N; further, this map implicitly assumes the privileged bases hypothesis. For $\phi^{\texttt{id}}$, we greedily search for the optimal partition $\{\boldsymbol{\psi}_\eta^\phi \mid \eta \in \boldsymbol{\eta}_{\texttt{int}}\}$ (instead of keeping it fixed) by iteratively adding neurons to them. For all $\boldsymbol{\eta}_{\texttt{inner}}$ simultaneously, one neuron is added at a time for each $\boldsymbol{\psi}_\eta^\phi$, up to a maximum allowed intervention size; these neurons are chosen to minimise the loss in eq. (8). Second, we will consider **linear maps**: $\phi^{\texttt{lin}}(\mathbf{h}) = \mathbf{W}_{\texttt{orth}}\mathbf{h}$, where $\mathbf{W}_{\texttt{orth}} \in \mathbb{R}^{d_\ell \times d_\ell}$ is an orthogonal matrix. This is the type of alignment map originally considered by Geiger et al. (2024b),[7] and implicitly assumes the linear representation hypothesis, evaluating input-restricted linear abstractions. Finally, we consider **non-linear maps**: $\phi^{\texttt{nonlin}}(\mathbf{h}) = \texttt{revnet}[L_{\texttt{rn}}, d_{\texttt{rn}}](\mathbf{h})$, where $\texttt{revnet}[L_{\texttt{rn}}, d_{\texttt{rn}}]$ is a reversible residual network (RevNet; Gomez et al., 2017) with $L_{\texttt{rn}}$ layers and hidden size $d_{\texttt{rn}}$. We can modulate the complexity of this final map by increasing $L_{\texttt{rn}}$ and $d_{\texttt{rn}}$, assuming the non-linear representation hypothesis. We note that all three maps are bijective and easily invertible.

Evaluation Metric. We evaluate the effectiveness of an alignment map ϕ using the **interchange intervention accuracy** (IIA) metric proposed by Geiger et al. (2024b). For a held out test set $\mathcal{D}_{\texttt{test}}$ with the same structure as the training set $\mathcal{D}$ defined in §3.3, we compute the accuracy of our model (i.e., $\arg\max_{\mathbf{y}' \in \mathcal{Y}} p_{\mathtt{N}}^\phi(\mathbf{y}' \mid \mathbf{x}_\emptyset^{(n)}, \mathbf{I}_{\mathtt{N}}^{(n)})$) when predicting the intervened $\mathbf{y}^{(n)} = f_{\mathtt{A}}(\mathbf{x}_\emptyset^{(n)}, \mathbf{I}_{\mathtt{A}}^{(n)})$. We compare this to the DNN's accuracy on the test set $\mathcal{D}_{\texttt{test}}$ without interventions.

5.1 Tasks, Algorithms, and DNNs.

Hierarchical equality task (Geiger et al. (2024b)). We will showcase our results primarily on this task. Let $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3 \circ \mathbf{x}_4$ be a 16-dimensional vector, and $\mathbf{x}_1$ to $\mathbf{x}_4$ each be 4-dimensional vectors, where $\circ$ represents vector concatenation. Further, let $\mathcal{X} = [-.5, .5]^{16}$. This task consists of evaluating: $\mathbf{y} = (\mathbf{x}_1 == \mathbf{x}_2) == (\mathbf{x}_3 == \mathbf{x}_4)$. As our DNN N, we investigate a 3-layer multi-layer perceptron (MLP) with hidden size 16, trained to perform this task; we describe this DNN, and its training procedure in more detail in App. I.1. Finally, we explore three algorithms for this task. The `both equality relations` algorithm first computes the two equalities ($v_{\eta_1} = (\mathbf{x}_1 == \mathbf{x}_2)$ and $v_{\eta_2} = (\mathbf{x}_3 == \mathbf{x}_4)$) separately; it then determines whether they are equivalent as a second step. The `left equality relation` algorithm first computes the left equality ($v_{\eta_1} = (\mathbf{x}_1 == \mathbf{x}_2)$), and then determines in a single step if this is equivalent to ($\mathbf{x}_3 == \mathbf{x}_4$). Finally, the `identity of first argument` algorithm assumes we copy the first input to a node ($v_{\eta_1} = \mathbf{x}_1$) and then compute the output directly. These three algorithms are more rigorously defined in App. I.1.

Indirect object identification (IOI) task. In a second set of experiments, we explore this task, inspired by Wang et al. (2023) and using the dataset of Muhia (2022). This task is more realistic and relies on larger (language) models. Here, inputs $\mathbf{x} \in \mathcal{X}$ are strings where two people are first introduced, and later one of them assumes the role of subject (S), giving or saying something to the other, the indirect object (IO). The task is then to predict the first token of the IO, with the output set $\mathcal{Y}$ containing the first token of each person's name. E.g., $\mathbf{x} =$ "`Friends Juana and Kristi found a mango at the bar. Kristi gave it to`" and $\mathbf{y} =$ "Juana". As our DNN, we use models from the Pythia suite (Biderman et al., 2023) across different sizes (from 31M to 410M parameters) and training stages. We evaluate the `ABAB-ABBA` algorithm where, given two names A and B, an inner node v_{η_1} captures if the sentence structure is ABAB (e.g., "A and B ... A gave to B") or ABBA (e.g., "A and B ... B gave to A"), and the algorithm outputs prediction B if v_{η_1} is ABAB and A otherwise. This algorithm is more rigorously defined in App. I.2.

6 Experiments and Results

We now proceed with our empirical study, applying alignment maps of varying complexity on both "toy" and real neural networks to evaluate their effects on the causal abstraction method DAS.

---

[7]We note that, while Geiger et al. (2024b) describe the used ϕ as a rotation, their pyvene (Wu et al., 2024a) implementation uses orthogonal matrices. This, however, makes no difference in the power of the alignment map.
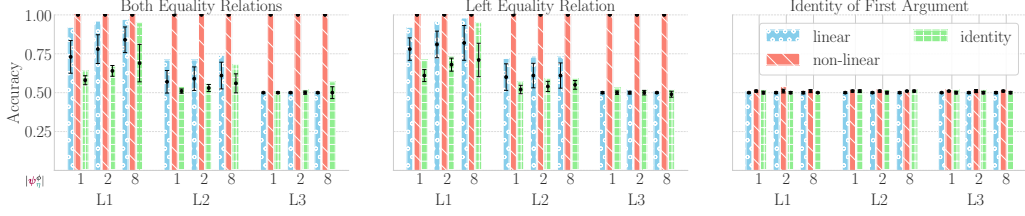
Figure 2: IIA in the hierarchical equality task for causal abstractions trained with different alignment maps ϕ. The figure shows results for all three analysed algorithms for this task. The bars represent the max IIA across 10 runs with different random seeds. The black lines represent mean IIA with 95% confidence intervals. The $|\psi_\eta^\phi|$ denotes the intervention size per node. Without interventions, all DNNs reach almost perfect accuracy (>0.99). The used $\phi^{\mathtt{nonlin}}$ uses $L_{\mathrm{rn}} = 10$ and $d_{\mathrm{rn}} = 16$.
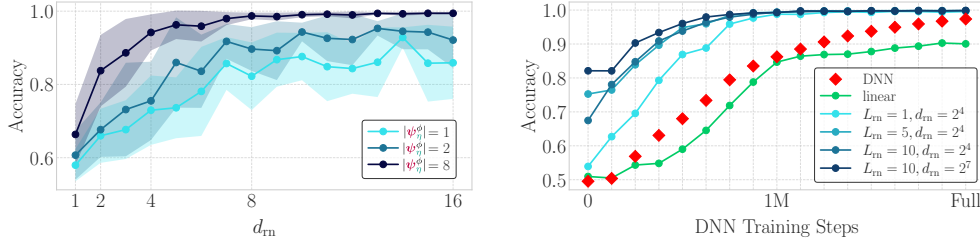
Figure 3: IIA of alignment between the `both equality relations` algorithm and an MLP, with interventions at layer 1. Left: Mean IIA over 5 seeds using $\phi^{\mathtt{nonlin}}$ ($L_{\mathrm{rn}} = 1$) on the trained DNN. Performance improves with larger hidden dimension d_{rn} and intervention size $|\psi_\eta^\phi|$. Right: Maximum IIA across 5 seeds using $\phi^{\mathtt{lin}}$ and $\phi^{\mathtt{nonlin}}$ with $|\psi_\eta^\phi| = 8$. Complex alignment maps achieve high IIA even with randomly initialised DNNs, while simpler maps gradually improve as training progresses.

Hierarchical equality task, main results.[8]   Fig. 2 presents IIA results across different alignment maps ϕ for all three algorithms. As expected, the identity map $\phi^{\mathtt{id}}$ generally results in the worst performance. Using linear alignments ($\phi^{\mathtt{lin}}$), we observe patterns consistent with Geiger et al. (2024b): IIA for `both equality relations` and `left equality relation` decreases substantially in the third layer, indicating information becomes difficult to manipulate using linear transformations at deeper layers. With the non-linear alignment ($\phi^{\mathtt{nonlin}}$), this layer-dependent degradation vanishes, yielding near-optimal IIA across all layers. Consequently, while assuming linear representations seems to enable us to identify the location of certain variables in our DNN, many of these insights fail to generalise when more powerful non-linear alignment maps are employed. The `identity of first argument` algorithm's IIA consistently hovers around 50% for $\phi^{\mathtt{id}}$, $\phi^{\mathtt{lin}}$ and $\phi^{\mathtt{nonlin}}$. Additional experiments (App. H) suggest this is caused by insufficient capacity of the used `revnet` model, as the identity of $\mathbf{x}_1$ seems to be encoded in the model's hidden states.

Hierarchical equality task, exploring $\phi^{\mathtt{nonlin}}$'s complexity.   Fig. 3 (left) illustrates how varying the hidden size d_{rn} and intervention size $|\psi_\eta^\phi|$ affects IIA with the `both equality relations` algorithm on layer 1 of our MLP. Fig. 3 (right) shows IIA evolution as alignment complexity increases throughout the MLP's training (evaluated on its layer 1). Remarkably, even with randomly initialised DNNs, we achieve over 80% IIA using the most complex alignment map. As training progresses, simpler alignment maps gradually attain higher IIA values. Additional results in App. I.1.3 extend these findings to other MLP layers, intervention sizes, and algorithms, consistently revealing similar patterns that reinforce our conclusion about the impact of alignment map complexity on IIA dynamics.

Indirect object identification task, main results.   Fig. 4 (left) presents the results of trying to find causal abstractions between the `ABAB-ABBA` algorithm and Pythia language models, exploring how model size affects alignment capabilities. Notably, despite only the larger models (160M and 410M parameters) successfully learning the IOI task, we can align the algorithm to models of all sizes—including the 31M and 70M parameter models that fail to learn the task. Further, and somewhat surprisingly, this alignment is perfect even for randomly initialised models across all sizes; smaller fully trained models (31M, 70M), though, show slightly reduced alignment accuracy. This reduction

---

[8]As an additional task similar to hierarchical equality, we also explore the **distributive law task** in App. I.3.
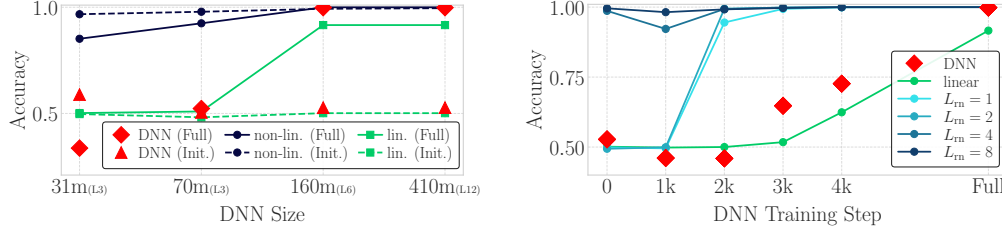
Figure 4: IIA of alignment between `ABAB-ABBA` algorithm and Pythia language models. Left: IIA across model sizes at initialisation (Init.) or after full training (Full), with intervention at the middle layer. Right: IIA with increasingly complex alignment maps during *Pythia-410m*'s training. Results show complex alignment maps yield near-perfect IIA. All $\phi^{\texttt{nonlin}}$ use $d_{\texttt{rn}} = 64$.

may stem from these smaller models saturating late in training (Godey et al., 2024), becoming highly anisotropic and making it harder for $\phi^{\texttt{nonlin}}$ to access the information needed to match the algorithm.

Indirect object identification task, exploring $\phi^{\texttt{nonlin}}$'s complexity. Fig. 4 (right) illustrates the interplay between model training progression and algorithmic alignment for the Pythia with 410M parameters. Notably, while this model begins to acquire task proficiency only around training step 3000 (as indicated by model accuracy), employing an 8-layer $\phi^{\texttt{nonlin}}$ as alignment map yields near-perfect IIA across all training steps, including for randomly initialised models. This pattern partially extends to a 4-layers $\phi^{\texttt{nonlin}}$ configuration; however, there is a noticeable dip in IIA at step 1000 for this configuration, which may be due to the model over-fitting to unigram statistics (Chang and Bergen, 2022; Belrose et al., 2024) at this point—thereby making context (and hidden states) be mostly ignored when producing model outputs. Interestingly, as training advances, even less complex alignment maps (1- and 2-layer $\phi^{\texttt{nonlin}}$) eventually attain perfect alignment. In contrast, linear maps only approximate perfect alignment in the fully trained model, following a similar trend to the DNN's performance.

7   Discussion

Our results show that when we lift the assumption of linear representations, sufficiently complex alignment maps can achieve near-perfect alignment across all models—regardless of their ability to solve the underlying task. This provides compelling evidence for the non-linear representation dilemma, suggesting causal alignment may be possible even when the model lacks task capability. We now discuss our results in the context of prior literature, with additional related work in App. C.

Causal Abstraction is not Enough. Causal abstraction (Geiger et al., 2024a) has gained traction as a theoretical framework for mechanistic interpretability, promising to overcome probing limitations by analysing DNN behaviour through interventions: if you intervene on a DNN's representations and its behaviour changes in a predictable way, you have identified how the DNN "truly" encodes that feature (Elazar et al., 2021; Ravfogel et al., 2021; Lasri et al., 2022). Recent critiques of causal abstraction (e.g., Mueller, 2024) highlight practical shortcomings, including the non-uniqueness of identified algorithms (Méloux et al., 2025) and the risk of "interpretability illusions" (Makelov et al., 2024). Despite counterarguments to some of these critiques (Wu et al., 2024b; Jørgensen et al., 2025), concerns have emerged that methods based on causal abstraction may introduce new information rather than accurately reflect the behaviour of the DNN (Wu et al., 2023; Sun et al., 2025); as an example, causal abstraction methods applied to random models sometimes yield above-chance performance (Geiger et al., 2024b; Arora et al., 2024). By examining the implications of assuming arbitrary complex ways in which features may be encoded in a DNN, we show that nearly any neural network can be aligned to any algorithm. Together, our results thus suggest that the shift in interpretability research to causal abstractions does not, by itself, resolve the core challenge of understanding how representations are encoded. Additionally, we note that early causal abstraction methods (Geiger et al., 2021) implicitly rely on the privileged bases hypothesis, while recent advancements (Geiger et al., 2024b) rely on the linear representation hypothesis instead.

Balancing the Accuracy vs. Complexity of ϕ. Diagnostic probing was a previously popular method for interpretability research (Alain and Bengio, 2016), where a **probe** was applied to the hidden representations of a DNN and trained to predict a specific variable. Notably, the architecture chosen for this probe implicitly reflected assumptions about representation encoding, and the absence of a universally accepted model for representation encoding precluded a theoretically founded

choice of probe architecture (Belinkov, 2022). The debate regarding the trade-off between probing complexity and accuracy (Hewitt and Liang, 2019; Pimentel et al., 2020b,a; Voita and Titov, 2020) underscores the risk of complex probes merely memorising variable-specific relations, instead of revealing which information the DNN "truly" encodes and uses. In this paper, we revive this debate by showing a clear analogue in causal abstraction methodologies: the effect of ϕ's complexity on IIA. Unfortunately, this debate was never solved by the probing literature, and solutions ranged from: controlling for the probe's memorisation capacity (Hewitt and Liang, 2019),[9] explicitly measuring a probe's complexity accuracy trade-off (Pimentel et al., 2020a), training minimum description length probes (Voita and Titov, 2020), or leveraging unsupervised probes (Burns et al., 2023).[10]

The Role of Generalisation. We now highlight that Theorem 1 provides an existence proof for a perfect abstraction map (thus guaranteeing perfect IIA) between a DNN and an algorithm. This existence proof, however, leverages complex interactions between the intervened hidden states and the DNN's structure, requiring perfect information about both and thus representing a form of extreme overfitting. Crucially, this theorem offers no guarantees regarding the learnability of the alignment map ϕ from limited data or its generalisation to unseen inputs. This gap between theoretical existence and practical learnability becomes evident in practise. For instance, in an additional experiment on the IOI task (in App. I.2.3), we show that when training and test sets contain disjoint sets of names, the learned alignment map fails to generalise, resulting in low IIA on the test set. This suggests that generalisation should play a crucial role in causal abstraction analysis, as the ability to learn abstraction maps that transfer beyond training data seems fundamental to interpreting a model, distinguishing a genuine understanding about its inner workings from mere training pattern memorisation.

Investigating Representation Encoding in DNNs. How neural networks encode variables/concepts is a long-standing question in interpretability, with three main hypotheses standing out: the privileged bases, linear representation, and non-linear representation hypotheses (see §3.1). One way to try to distinguish between these hypotheses is with causal abstraction analyses, but what can we learn about these hypotheses if our methods themselves rely on them as assumptions? One solution could be to compare results using ϕ with different architectures. Our Fig. 4 (right), for instance, shows that while $\phi^{\texttt{nonlin}}$ achieves consistently near-perfect results throughout model training, $\phi^{\texttt{lin}}$ accompanies the actual DNN's performance more closely. Intuitively, we may thus be inclined to support the linear representation hypothesis here. We (the authors), however, cannot make this intuition formal to justify why we believe this is the case. Furthermore, $\phi^{\texttt{lin}}$ still manages to sometimes achieve IIA higher than the DNN's accuracy, implying it may also "learn the task". We expect future work will propose novel methodologies to analyse information encoding and try to answer these questions.

8   Conclusion

This paper critically examines causal abstraction in machine learning, when no assumptions are imposed on how representations are encoded. We show that, under mild conditions, any algorithm can be perfectly aligned with any DNN, leading to the non-linear representation dilemma. Empirical validation through experiments on the hierarchical equality and the indirect object identification tasks corroborate our theoretical insights, demonstrating near perfect IIA even in randomly initialised DNNs. So, *what should you do if you want to perform a causal analysis of your DNN?* We believe that it must be decided on a case-by-case basis. If you have reason to believe the linear representation hypothesis holds for the features you wish to extract, constraining ϕ to linear functions may be advised. If you do not, however, you may face the non-linear representation dilemma, and be forced to investigate some kind of trade-off between ϕ's accuracy and complexity.

Limitations. Our proof that any algorithm can be aligned with any DNN (Theorem 1) relies on a form of overfitting. Yet, our experiments show that the learned alignment maps ϕ generalise to unseen test data; studying the factors behind this generalisation would be valuable. Further, our theorem relies on two strong assumptions: input-injectivity (Assump. 2) and strict output-surjectivity (Assump. 3) in all layers. While we justify both, there are settings—related to, e.g., the softmax bottleneck—where they may fail; studying these failure modes could clarify our assumptions' limitations.

---

[9]Notably, this method was previously applied to causal abstraction analysis by Arora et al. (2024).

[10]The complexity—accuracy trade-off in probing arises mainly in supervised settings, where more complex probes can extract richer features from model representations. Unsupervised probing avoids this, lacking the supervision that enables such "gerrymandered" mappings.

Contributions

Denis Sutter led the project, implemented the base version of the DAS code, conducted the MLP experiments and derived the base proof of Theorem 1 as well as the proof of Theorem 2. Julian Minder implemented and ran the language model experiments, produced all plots, and helped refine the proof of Theorem 2. Thomas Hofmann provided guidance throughout the project. Tiago Pimentel supervised the project, giving initial intuitions for the proofs in both Theorem 1 and Theorem 2, refining the proof of Theorem 1, and defining the main notation in the paper, integrating feedback from Denis and Julian. All authors wrote the paper together.

Acknowledgments

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv*.

Aryaman Arora, Dan Jurafsky, and Christopher Potts. 2024. CausalGym: Benchmarking causal interpretability methods on linguistic tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14638–14663, Bangkok, Thailand. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495.

Sander Beckers and Joseph Y. Halpern. 2019. Abstracting causal models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2678–2685.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Nora Belrose, Quintin Pope, Lucia Quirke, Alex Mallen, and Xiaoli Fern. 2024. Neural networks learn statistics of increasing complexity. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in Neural Information Processing Systems*, volume 29.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

Tyler A. Chang and Benjamin K. Bergen. 2022. Word acquisition in neural language models. *Transactions of the Association for Computational Linguistics*, 10:1–16.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. PaLM: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. 2024. Recurrent neural networks learn to store and generate sequences using non-linear representations. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 248–262, Miami, Florida, US. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*.

Nelson Elhage, Robert Lasenby, and Christopher Olah. 2023. Privileged bases in the transformer residual stream. *Transformer Circuits Thread*, page 24.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. 2025a. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*.

Joshua Engels, Logan Riggs Smith, and Max Tegmark. 2025b. Decomposing the dark matter of sparse autoencoders. *Transactions on Machine Learning Research*.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *arXiv*.

Lei Gao and Ling Guan. 2023. Interpretability of machine learning: Recent advances and future prospects. *IEEE MultiMedia*, 30(4):105–118.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. 2024a. Causal abstraction: A theoretical foundation for mechanistic interpretability. *arXiv*.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA. Curran Associates Inc.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. 2022. Inducing causal structure for interpretable neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7324–7338. PMLR.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. 2024b. Finding alignments between interpretable causal variables and distributed neural representations. *arXiv*.

Nathan Godey, Éric Villemonte de la Clergerie, and Benoît Sagot. 2024. Why do small language models underperform? Studying language model saturation via the softmax bottleneck. In *First Conference on Language Modeling*.

Satvik Golechha and James Dao. 2024. Challenges in mechanistically interpreting model representations. *arXiv*.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. 2017. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Bryce Goodman and Seth Flaxman. 2017. European Union regulations on algorithmic decision making and a "right to explanation". *AI Magazine*, 38(3):50–57.

Andreas Grivas, Nikolay Bogoychev, and Adam Lopez. 2022. Low-rank softmax can have unargmaxable classes in theory but rarely in practice. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6738–6758, Dublin, Ireland. Association for Computational Linguistics.

Bobby He, Lorenzo Noci, Daniele Paliotta, Imanol Schlag, and Thomas Hofmann. 2024. Understanding and minimising outlier features in transformer training. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*.

Frederik Hytting Jørgensen, Luigi Gresele, and Sebastian Weichwald. 2025. What is causal about causal models and representations? *arXiv*.

Subhash Kantamneni and Max Tegmark. 2025. Language models use trigonometry to do addition. *arXiv*.

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. 2021. BERT busters: Outlier dimensions that disrupt transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405, Online. Association for Computational Linguistics.

Karim Lasri, Tiago Pimentel, Alessandro Lenci, Thierry Poibeau, and Ryan Cotterell. 2022. Probing for the usage of grammatical number. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8818–8831, Dublin, Ireland. Association for Computational Linguistics.

Aleksandar Makelov, Georg Lange, Atticus Geiger, and Neel Nanda. 2024. Is this the subspace you are looking for? An interpretability illusion for subspace activation patching. In *The Twelfth International Conference on Learning Representations*.

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. 2025. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? In *The Thirteenth International Conference on Learning Representations*.

Julian Minder, Kevin Du, Niklas Stoehr, Giovanni Monea, Chris Wendler, Robert West, and Ryan Cotterell. 2025. Controllable context sensitivity and the knob behind it. In *The Thirteenth International Conference on Learning Representations*.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

Aaron Mueller. 2024. Missed causes and ambiguous effects: Counterfactuals pose challenges for interpreting neural networks. *arXiv*.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. 2024. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability. *arXiv*.

Brian Muhia. 2022. ioi (revision 223da8b).

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Giorgos Nikolaou, Tommaso Mencattini, Donato Crisostomi, Andrea Santilli, Yannis Panagakis, and Emanuele Rodolá. 2025. Language models are injective and hence invertible. *Preprint*, arXiv:2510.15511.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*. Https://distill.pub/2020/circuits/zoom-in.

Chris Olah and Adam Jermyn. 2024. What is a linear representation? What is a multidimensional feature? *Transformer Circuits Thread*.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill*. Https://distill.pub/2017/feature-visualization.

Vardan Papyan, X. Y. Han, and David L. Donoho. 2020. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020a. Pareto probing: Trading off accuracy for complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3138–3153, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020b. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Niklas Stoehr, and Ryan Cotterell. 2022. The architectural bottleneck principle. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11459–11472, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

Shauli Ravfogel, Grusha Prasad, Tal Linzen, and Yoav Goldberg. 2021. Counterfactual interventions reveal the causal effect of relative clause representations on agreement prediction. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 194–209, Online. Association for Computational Linguistics.

Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. 2022. Linear adversarial concept erasure. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18400–18421. PMLR.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, Stephen Casper, Max Tegmark, William Saunders, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. 2025. Open problems in mechanistic interpretability. *arXiv*.

Jiuding Sun, Jing Huang, Sidharth Baskaran, Karel D'Oosterlinck, Christopher Potts, Michael Sklar, and Atticus Geiger. 2025. HyperDAS: Towards automating mechanistic interpretability with hypernetworks. *arXiv*.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. 2024. Massive activations in large language models. In *First Conference on Language Modeling*.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*.

Sana Tonekaboni, Shalmali Joshi, Melissa D McCradden, and Anna Goldenberg. 2019. What clinicians want: Contextualizing explainable machine learning for clinical end use. *arXiv*.

Oskar van der Wal, Pietro Lesci, Max Müller-Eberstein, Naomi Saphra, Hailey Schoelkopf, Willem Zuidema, and Stella Biderman. 2025. PolyPythias: Stability and outliers across fifty language model pre-training runs. In *The Thirteenth International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Jennifer C. White, Tiago Pimentel, Naomi Saphra, and Ryan Cotterell. 2021. A non-linear structural probe. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 132–138, Online. Association for Computational Linguistics.

Zhengxuan Wu, Atticus Geiger, Aryaman Arora, Jing Huang, Zheng Wang, Noah Goodman, Christopher Manning, and Christopher Potts. 2024a. pyvene: A library for understanding and improving PyTorch models via interventions. In *Proceedings of the 2024 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations), pages 158–165, Mexico City, Mexico. Association for Computational Linguistics.

Zhengxuan Wu, Atticus Geiger, Jing Huang, Aryaman Arora, Thomas Icard, Christopher Potts, and Noah D. Goodman. 2024b. A reply to Makelov et al. (2023)'s "interpretability illusion" arguments. *arXiv*.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah Goodman. 2023. Interpretability at scale: Identifying causal mechanisms in Alpaca. In *Advances in Neural Information Processing Systems*, volume 36, pages 78205–78226. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Zeyu Yun, Yubei Chen, Bruno Olshausen, and Yann LeCun. 2021. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 1–10, Online.

Wenwu Zhu, Xin Wang, and Wen Gao. 2020. Multimedia intelligence: When multimedia meets artificial intelligence. *IEEE Transactions on Multimedia*, 22(7):1823–1835.

A   Reproducibility

We provide the code to reproduce our experiments in https://github.com/densutter/non-linear-representation-dilemma. Refer to the `README.md` for instructions.

B   Pseudo-code for Running an Intervention on an Algorithm

In Fig. 5, we present pseudo-code that demonstrates how algorithms execute under our intervention framework.

```
1    def f(A):
2        def f_A(x, I_A=None):
3            v_{η_x} = x
4            for η in A.topological_sort(η_inner ∪ η_y):
5                if I_A and η in I_A:
6                    v_η = I_A[η]
7                else:
8                    v_η = f_A^η(v_{par_A(η)})
9            return v_{η_y}
10       return f_A
```

Figure 5: Pseudo-code implementation of an algorithm with interventions, where interventions I_A are specified as a Python dictionary mapping nodes to their intervened values.

C   Additional Related Work

The concept of causal abstraction was ported to deep neural networks by Geiger et al. (2024a), providing a generalised framework for understanding how neural networks can be abstracted to higher-level algorithms. Early work by Geiger et al. (2021) explored direct interventions on neuron-aligned activations, laying the groundwork for more sophisticated approaches. Building on this, Geiger et al. (2024b) introduced distributed alignment search (DAS), which uses an alignment map to align distributed representations in neural networks with causal graphs. Several improvements to DAS have been proposed: Sun et al. (2025) developed HyperDAS, which automates the search for node information using hypernetworks, while Wu et al. (2023) introduced Boundless DAS, which automatically determines intervention size through gradient descent, scaling to larger models.

However, recent work has raised important critiques of causal alignment methods. Méloux et al. (2025) demonstrated that multiple algorithms can be causally aligned with the same neural network, and conversely, a single algorithm can align with different network subspaces. Mueller (2024) identified fundamental limitations in counterfactual theories, showing they may miss certain causes and that causal dependencies in neural networks are not necessarily transitive. Makelov et al. (2024) showed that subspace interventions such as those used in DAS can lead to "interpretability illusions"—cases where manipulating a subspace changes the behaviour of the model through activating parallel pathways, rather than directly controlling the target feature. In their response, Wu et al. (2024b) argued these illusions may be artefacts of specific evaluation approaches rather than fundamental flaws, and that they depend on the definition of causality being used, a point also made by Jørgensen et al. (2025).

Recent work has also raised significant challenges to the linear representation hypothesis. White et al. (2021) demonstrated that syntactic structure in language models is encoded non-linearly, showing that kernelised structural probes outperform linear ones while maintaining parameter count. Similarly, Csordás et al. (2024) found that recurrent neural networks use fundamentally non-linear representations for sequence tasks. Engels et al. (2025a) provided concrete examples of non-linear feature representations in language models, such as days of the week being encoded on a circular manifold. While Golechha and Dao (2024) argued that some language modelling behaviours may be represented linearly due to next-token prediction and LayerNorm folding, Mueller et al. (2024) advocated for exploring non-linear mediators to uncover more sophisticated abstractions. Additional evidence comes from Kantamneni and Tegmark (2025), who found that language models represent numbers on a helical manifold. Olah and Jermyn (2024) offered an important clarification: the linear representation hypothesis is not about dimensionality but rather about features behaving mathematically linearly through addition and scaling, allowing for multidimensional features with constrained geometry. This represents a relaxation of the strongest form of the hypothesis.

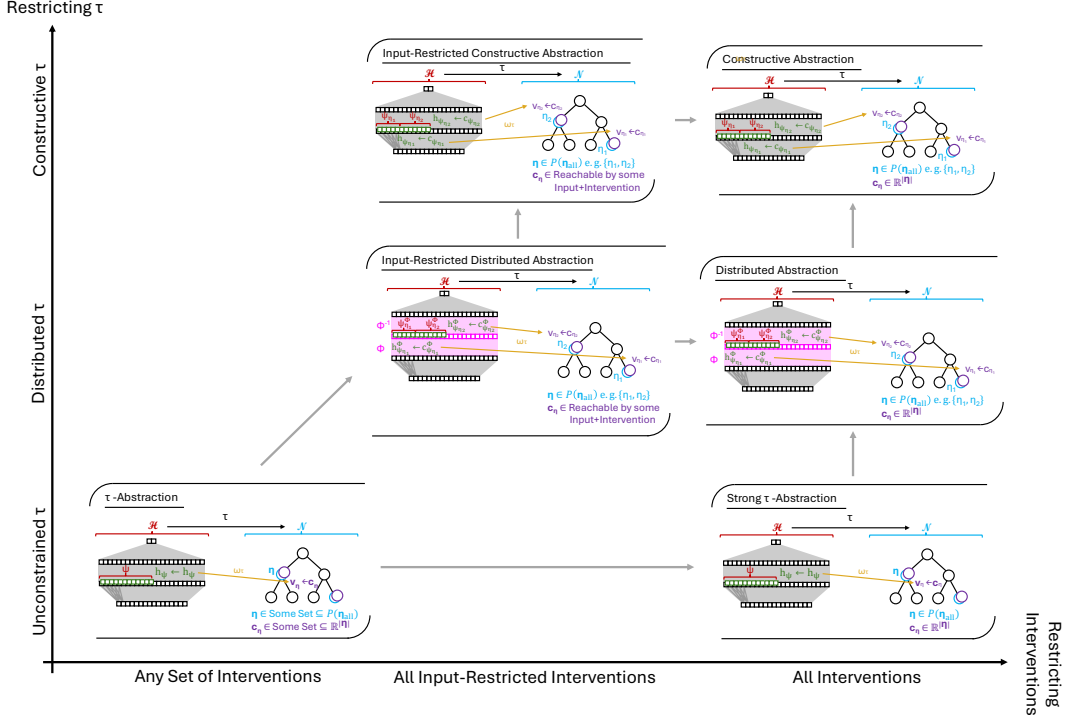# D Schematic of the Relation Between Notions of Causal Abstraction

Figure 6: A schematic of the definitions of causal abstraction in §3. The axes represent an increase in how restricted the notion of causal abstraction is based on: y-axis, constraints placed on τ; and x-axis, constraints placed on the set of allowed interventions. Grey arrows symbolise a superset→subset relationship: if an A-N pair fulfils the conditions in the subset, it also fulfils them in the superset.

E DNN Definitions

E.1 MLP

A multi-layer perceptron (MLP) consists of a sequence of linear transformations interleaved with non-linear activation functions.

Submodule 1. *We can define a **multi-layer perceptron (**`mlp`*) by choosing:*

$$\mathbf{h}_{\boldsymbol{\psi}_1} = f_{\text{N}}^0(\mathbf{x}) = \mathbf{W}_0\, \mathbf{x}, \qquad\qquad \mathbf{h}_{\boldsymbol{\psi}_{\ell+1}} = f_{\text{N}}^{\ell}(\mathbf{h}_{\boldsymbol{\psi}_\ell}) = \mathbf{W}_\ell\, (\sigma(\mathbf{h}_{\boldsymbol{\psi}_\ell})) + \mathbf{b}_\ell \qquad (9)$$

where $\mathbf{W}_0 \in \mathbb{R}^{|\boldsymbol{\psi}_1| \times |\mathbf{x}|}$, $\mathbf{W}_\ell \in \mathbb{R}^{|\boldsymbol{\psi}_{\ell+1}| \times |\boldsymbol{\psi}_\ell|}$, and $\mathbf{b}_\ell \in \mathbb{R}^{|\boldsymbol{\psi}_{\ell+1}|}$ are trainable parameters, and σ is a non-linearity like ReLU. For this model, $\mathcal{H}_\ell = \mathbb{R}^{|\boldsymbol{\psi}_\ell|}$ for $0 < \ell < L$ and $\mathcal{H}_L = \mathbb{R}^{|\mathbf{y}|}$.

In this work, we focus on MLPs used for classification tasks, whose final layer includes a softmax transformation.

DNN 1. *A **classification multi-layer perceptron** (MLP) is defined like Submodule 1 but with a softmax on the last layer:*

$$p_{\text{N}}(\mathbf{y} \mid \mathbf{x}) = f_{\text{N}}^L(\mathbf{h}_{\boldsymbol{\psi}_L}) = \texttt{softmax}(\mathbf{W}_L\, \mathbf{h}_{\boldsymbol{\psi}_L}) \qquad (10)$$

where $\mathbf{W}_L \in \mathbb{R}^{|\mathbf{y}| \times |\boldsymbol{\psi}_L|}$ is a trainable parameter.

E.2 Transformer Language Model

In this section, we provide a definition of decoder-only autoregressive language models (Radford et al., 2019; Vaswani et al., 2017). While many variations of transformer architectures have been developed, we focus on the original GPT-2 architecture (Radford et al., 2019). We highlight that the Pythia models explored in our experiments are slightly different from the original GPT-2 and use parallel attention (Chowdhery et al., 2023); however, we do not expect this change to strongly affect our results. We now define the different submodules that compose a transformer.

Submodule 2. *The **Embedding** layer in a transformer maps input tokens to vectors:*

$$f_{\mathrm{N}}^0(\mathbf{x}) = \mathbf{e_x}, \qquad \text{where } \mathbf{e_x} \in \mathbb{R}^{|\mathbf{x}| \times |\boldsymbol{\psi}_1|} \tag{11}$$

In this equation, $\mathbf{e} \in \mathbb{R}^{|\mathcal{X}| \times |\boldsymbol{\psi}_1|}$ *is a learned parameter matrix and* $\mathbf{x}$ *indexes into its rows.*[11]

Submodule 3. *Multi-Head Self-Attention with H heads is defined as:*

$$\mathtt{attn}(\mathbf{h}) = \mathtt{concat}(\boldsymbol{\eta}_1(\mathbf{h}), \dots, \boldsymbol{\eta}_H(\mathbf{h}))\mathbf{W}^O \tag{12}$$

where each head operates in dimension $d_{\boldsymbol{\eta}} \ll |\boldsymbol{\psi}_\ell|$ *and computes:*

$$\boldsymbol{\eta}_i(\mathbf{h}) = \mathtt{softmax}\left(\frac{(\mathbf{h}\mathbf{W}_i^Q)(\mathbf{h}\mathbf{W}_i^K)^\top}{\sqrt{d_k}}\right)\mathbf{h}\mathbf{W}_i^V \tag{13}$$

with learned parameters $\mathbf{W}^O \in \mathbb{R}^{Hd_{\boldsymbol{\eta}} \times |\boldsymbol{\psi}_\ell|}$, $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{|\boldsymbol{\psi}_\ell| \times d_{\boldsymbol{\eta}}}$.

Submodule 4. *Layer Normalization applies per-feature normalization:*

$$\mathtt{LN}(\mathbf{h}) = \gamma \odot \frac{\mathbf{h} - \mu}{\sigma} + \beta \tag{14}$$

where μ and σ are the mean and standard deviation across all features for a single input, and γ, β are learned parameters.

Using these submodules, we define a transformer block.

Submodule 5. *A **Transformer Block** chains together attention and MLP layers with residual connections:*

$$\mathbf{h}' = \mathbf{h} + \mathtt{attn}(\mathtt{LN}(\mathbf{h})), \qquad f_{\mathrm{N}}^\ell(\mathbf{h}) = \mathbf{h}' + \mathtt{mlp}(\mathtt{LN}(\mathbf{h}')) \tag{15}$$

where $\mathtt{mlp}$ *is applied to each token activations separatly as defined in Submodule 1.*

Finally, we define the complete transformer language model.

DNN 2. *A **transformer language model** consists of an embedding layer, transformer blocks, and an output layer:*

$$\mathbf{h}_{\boldsymbol{\psi}_1} = f_{\mathrm{N}}^0(\mathbf{x}) = \mathbf{e_x} \tag{16a}$$

$$\mathbf{h}_{\boldsymbol{\psi}_{\ell+1}} = f_{\mathrm{N}}^\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell}) \tag{16b}$$

$$p_{\mathrm{N}}(\mathbf{y} \mid \mathbf{x}) = f_{\mathrm{N}}^L(\mathbf{h}_{\boldsymbol{\psi}_L}) = \mathtt{softmax}(\mathtt{LN}(\mathbf{h}_{\boldsymbol{\psi}_L})_{-1}\mathbf{W}_L) \tag{16c}$$

where $\mathtt{LN}(\mathbf{h}_{\boldsymbol{\psi}_L})_{-1}$ *selects the final token's position after layernorm is applied. For this model,* $\mathcal{H}_{\boldsymbol{\psi}_\ell} = \mathbb{R}^{|\mathbf{x}| \times |\boldsymbol{\psi}_\ell|}$ *for* $1 \leq \ell \leq L$ *and* $\mathcal{H}_{\boldsymbol{\psi}_{L+1}} = \Delta^{|\mathcal{Y}|-1}$.

F   Proof of Theorem 1

In this section, we prove our main theorem. For notational simplicity, we write in this section:

$$\underbrace{f_{\mathrm{N}}^{:\ell} \stackrel{\text{def}}{=} f_{\mathrm{N}}^{\boldsymbol{\psi}_\ell} = f_{\mathrm{N}}^{\ell-1} \circ \cdots \circ f_{\mathrm{N}}^0}_{\text{Run DNN up to layer } \ell}, \qquad \text{and} \qquad \underbrace{f_{\mathrm{N}}^{\ell:} \stackrel{\text{def}}{=} f_{\mathrm{N}}^L \circ \cdots \circ f_{\mathrm{N}}^\ell}_{\text{Run DNN from layer } \ell} \tag{17}$$

We start by formally stating our assumptions.

Assumption 1 (Countable input-space). *We assume that the space of inputs (i.e., $\mathcal{X}$) is countable.*

Assumption 2 (Input-injectivity in all layers). *We assume that $f_{\mathrm{N}}^{:\ell}$ is injective for all layers.*

Assumption 3 (Strict output-surjectivity in all layers). *We assume that the composition of $f_{\mathrm{N}}^{\ell:}$ and $\tau_{\boldsymbol{\eta}_{\mathbf{y}}}$ is strictly surjective for all layers (we define strict surjectivity in Definition 10).*

Assumption 4 (Algorithm and DNN have matchable partial-orderings). *We assume that there exists a partitioning $\{\boldsymbol{\psi}_\eta \mid \eta \in \boldsymbol{\eta}_{\mathtt{int}}\} \cup \{\boldsymbol{\psi}_\perp\}$ of N's neurons $\boldsymbol{\psi}_{\mathtt{int}}$—where $\boldsymbol{\psi}_\eta$ are single neurons—which respects the partial-ordering of algorithm A, i.e., $\eta \prec \eta' \implies \boldsymbol{\psi}_\eta \prec \boldsymbol{\psi}_{\eta'}$. Further, for each layer at least one neuron is left unused in this partitioning, i.e., $\boldsymbol{\psi}_\perp \cap \boldsymbol{\psi}_\ell \neq \emptyset$.*

---

[11] We ignore positional embeddings here for simplicity, as they do not affect our proofs of injectivity in App. G. Note that, in that section, we show injectivity on an entire layer's activations $\mathbf{h}_{\boldsymbol{\psi}_\ell}$. Position embeddings would be needed to show injectivity on a single position, e.g., the last token's position $(\mathbf{h}_{\boldsymbol{\psi}_\ell})_{-1}$; a property which we conjecture should also hold. Further, we note that position embeddings are used in our experiments.

Assumption 5 (DNN solves the task). *We assume that for any input* $\mathbf{x} \in \mathcal{X}$, *the neural network solves the task correctly, satisfying* $\mathtt{T}(\mathbf{x}) = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\, p_{\mathtt{N}}(\mathbf{y} \mid \mathbf{x})$.

We provide a longer discussion about why we think these assumptions are reasonable in App. F.1. For convenience, we also put a self-contained version of Definition 7 (input-restricted distributed abstraction) in App. F.2. Now, we restate our theorem and present its proof.

Theorem 1. *Given any algorithm* $\mathtt{A}$ *and any neural network* $\mathtt{N}$ *such that Assumps. 1 to 5 hold, we can show that* $\mathtt{A}$ *is an input-restricted distributed abstraction of* $\mathtt{N}$.

Proof. To show that an algorithm $\mathtt{A}$ is an input-restricted distributed abstraction of a neural network $\mathtt{N}$, we must show (according to Definition 7) that there exists a τ for which: $\mathtt{A}$ is an input-restricted τ-abstraction of $\mathtt{N}$; and τ is a distributed abstraction map. For τ to be a distributed abstraction map, we need a partition of hidden variables which allows us to independently compute it per node. Further, we need the partitioned hidden variables $\boldsymbol{\psi}^{\phi}_{\mathtt{int}}$ to be the output of an alignment map ϕ which is layer-wise decomposable. We thus have:

$$\underbrace{\mathbf{h}_{\boldsymbol{\psi}^{\phi}} = \phi(\mathbf{h}) = [\phi_{\ell}(\mathbf{h}_{\boldsymbol{\psi}_{\ell}})]^{L+1}_{\ell=0}}_{\text{Align DNN}}, \quad \underbrace{\boldsymbol{\Psi}^{\phi} = \{\boldsymbol{\psi}^{\phi}_{\eta} \mid \eta \in \boldsymbol{\eta}_{\mathtt{int}}\} \cup \{\boldsymbol{\psi}^{\phi}_{\perp}\}}_{\text{Partition hidden variables}}, \quad \underbrace{\tau(\mathbf{h}) = [\tau_{\eta}(\mathbf{h}_{\boldsymbol{\psi}^{\phi}_{\eta}})]_{\eta \in \boldsymbol{\eta}_{\mathtt{int}}}}_{\text{Compute abstraction}} \quad (18)$$

Therefore, to define a distributed abstraction map τ, we must define the following three terms: (i) a set of layer-wise alignment maps $\{\phi_{\ell}\}^{L}_{\ell=1}$ (note that the alignment maps ϕ_0 and ϕ_{L+1} are fixed by definition); (ii) a partition of hidden variables $\boldsymbol{\Psi}^{\phi}$; and (iii) a set of per-node functions $\{\tau_{\eta}\}_{\eta \in \boldsymbol{\eta}_{\mathtt{int}}}$. To prove this theorem, then, we must show that there exists a way to define these terms while ensuring that $\mathtt{A}$ is an input-restricted τ-abstraction of $\mathtt{N}$.

We now note that—given Assump. 4 and independently of our choice of alignment map ϕ—there exists at least one partition $\boldsymbol{\Psi}^{\phi}$ of the hidden variables $\boldsymbol{\psi}^{\phi}_{\mathtt{int}}$ in $\mathtt{N}$ for which:

$$\underbrace{\forall_{\eta,\eta' \in \boldsymbol{\eta}_{\mathtt{int}}} : \eta \prec \eta' \implies \boldsymbol{\psi}^{\phi}_{\eta} \prec \boldsymbol{\psi}^{\phi}_{\eta'}}_{\text{respect partial ordering}}, \quad \underbrace{\forall_{\eta \in \boldsymbol{\eta}_{\mathtt{int}}} : |\boldsymbol{\psi}^{\phi}_{\eta}| = 1}_{\text{1 neuron per partition}}, \quad \underbrace{\forall_{0 < \ell \leq L} : \boldsymbol{\psi}^{\phi}_{\ell} \not\subseteq \bigcup_{\eta \in \boldsymbol{\eta}_{\mathtt{int}}} \boldsymbol{\psi}^{\phi}_{\eta}}_{\text{no layer is fully occupied}} \quad (19)$$

where we define $\boldsymbol{\psi}^{\phi}_{\ell}$ as the latent variables given when applying ϕ_{ℓ} on $\boldsymbol{\psi}_{\ell}$. To facilitate our proof, we choose one such partition $\boldsymbol{\Psi}^{\phi}$ which we will keep fixed independently of our choice of alignment map ϕ. Given partition $\boldsymbol{\Psi}^{\phi}$, we can assign each node η to a specific layer ℓ, as $\boldsymbol{\psi}^{\phi}_{\eta}$ contains a single hidden variable and therefore trivially belongs to a single layer. We therefore can define $\boldsymbol{\eta}_{\ell}$ as all nodes associated with layer ℓ:

$$\eta \in \boldsymbol{\eta}_{\ell} \Leftrightarrow \boldsymbol{\psi}^{\phi}_{\eta} \subseteq \boldsymbol{\psi}^{\phi}_{\ell} \tag{20}$$

We now consider the application of interventions on $\mathtt{N}$ as layer-wise on $\boldsymbol{\psi}^{\phi}_{\eta} \subseteq \boldsymbol{\psi}^{\phi}_{\ell}$ for $\eta \in \boldsymbol{\eta}_{\ell}$. Let us therefore define $\mathcal{I}^{\ell}_{\mathtt{N}}$ as the set of all interventions on $\boldsymbol{\psi}^{\phi}_{\eta}$ for $\eta \in \boldsymbol{\eta}_{\ell}$, where we note that $\mathcal{I}^{\ell}_{\mathtt{N}}$ also includes an empty intervention (i.e., no intervention). For notational convenience, we will write the set of all interventions up to layer ℓ as $\mathcal{I}^{:\ell}_{\mathtt{N}}$, and the set of all nodes associated with those layers as $\boldsymbol{\eta}_{:\ell}$:

$$\mathcal{I}^{:\ell}_{\mathtt{N}} = \mathcal{I}^{0}_{\mathtt{N}} \times \mathcal{I}^{1}_{\mathtt{N}} \times \cdots \times \mathcal{I}^{\ell}_{\mathtt{N}}, \qquad \boldsymbol{\eta}_{:\ell} = \boldsymbol{\eta}_0 \cup \boldsymbol{\eta}_1 \cup \cdots \cup \boldsymbol{\eta}_{\ell} \tag{21}$$

where $\times$ denotes a Cartesian product. We analogously define $\mathcal{I}^{\ell}_{\mathtt{A}}$ and $\mathcal{I}^{:\ell}_{\mathtt{A}}$.

Finally, we get to an induction proof that will complete this theorem. We will iteratively construct abstraction and alignment maps for each layer such that it holds that:

$$\underset{\substack{\mathbf{I}_{\mathtt{N}} \in \mathcal{I}^{:\ell-1}_{\mathtt{N}} \\ \mathbf{x} \in \mathcal{X}}}{\forall} \underset{\eta \in \boldsymbol{\eta}_{\ell}}{\forall} : \underbrace{\tau_{\eta}(\mathbf{h}'_{\boldsymbol{\psi}^{\phi}_{\eta}}) = f^{\eta}_{\mathtt{A}}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})}_{\text{tested condition}}, \qquad \texttt{where } \underbrace{\mathbf{h}'_{\boldsymbol{\psi}^{\phi}_{\ell}} = \phi_{\ell}(f^{:\ell}_{\mathtt{N}}(\mathbf{x}, \mathbf{I}_{\mathtt{N}}))}_{\text{pre-int. hidden variable, as } \mathbf{I}_{\mathtt{N}} \in \mathcal{I}^{:\ell-1}_{\mathtt{N}}}, \texttt{ and } \mathbf{I}_{\mathtt{A}} = \omega_{\tau}(\mathbf{I}_{\mathtt{N}}) \quad \textcircled{1}$$

where int. stands for intervention. Note that if this holds for all layers, we have proven that $\mathtt{A}$ is an input-restricted τ-abstraction of $\mathtt{N}$, as we can perfectly reconstruct the behaviour of algorithm $\mathtt{A}$ from $\mathtt{N}$'s states under any intervention.[12] Also note, however, that our definition of abstraction map restricts $\tau_{\boldsymbol{\eta}_{\mathbf{y}}}(\mathbf{h}_{\boldsymbol{\psi}_{L+1}}) = \mathrm{argmax}\,\mathbf{h}_{\boldsymbol{\psi}_{L+1}}$, so special care must be taken to guarantee that this last identity will

---

[12]The attentive reader may note condition $\textcircled{1}$ only guarantees we can reconstruct the behaviour of algorithm $\mathtt{A}$ from pre-intervention hidden variables. Lemma 1 shows the same holds for post-intervention hidden variables.

be preserved. We thus also require an additional condition to hold at each step:

$$\underset{\substack{\mathbf{I}_{\mathtt{N}}\in\mathcal{I}_{\mathtt{N}}^{:\ell}\\ \mathbf{x}\in\mathcal{X}}}{\forall} : \underbrace{\tau_{\boldsymbol{\eta}_{\mathbf{y}}}(f_{\mathtt{N}}^{\ell:}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell}})) = f_{\mathtt{A}}(\mathbf{x},\mathbf{I}_{\mathtt{A}}),}_{\texttt{tested condition}} \qquad \texttt{where } \underbrace{\mathbf{h}'_{\boldsymbol{\psi}_{\ell}} = f_{\mathtt{N}}^{:\ell}(\mathbf{x},\mathbf{I}_{\mathtt{N}}), \texttt{ and } \mathbf{I}_{\mathtt{A}} = \omega_{\tau}(\mathbf{I}_{\mathtt{N}})}_{\texttt{post-int. neurons, as } \mathbf{I}_{\mathtt{N}}\in\mathcal{I}_{\mathtt{N}}^{:\ell}} \qquad ②$$

Finally, for convenience, we add a third condition to our inductive proof which will make the other two conditions easier to guarantee:

$$\underset{\eta\in\boldsymbol{\eta}_{:\ell-1}}{\forall} \exists g_{\eta} \underset{\substack{\mathbf{I}_{\mathtt{N}}\in\mathcal{I}_{\mathtt{N}}^{:\ell-1}\\ \mathbf{x}\in\mathcal{X}}}{\forall} : \underbrace{g_{\eta}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell}^{\phi}\cap\boldsymbol{\psi}_{\perp}^{\phi}}) = f_{\mathtt{A}}^{\eta}(\mathbf{x},\mathbf{I}_{\mathtt{A}}),}_{\texttt{tested condition}} \texttt{ where } \underbrace{\mathbf{h}'_{\boldsymbol{\psi}_{\ell}^{\phi}} = \phi_{\ell}(f_{\mathtt{N}}^{:\ell}(\mathbf{x},\mathbf{I}_{\mathtt{N}})), \texttt{and } \mathbf{I}_{\mathtt{A}} = \omega_{\tau}(\mathbf{I}_{\mathtt{N}})}_{\texttt{pre-int. hidden variable, as } \mathbf{I}_{\mathtt{N}}\in\mathcal{I}_{\mathtt{N}}^{:\ell-1}} \quad ③$$

This condition guarantees that information about previous nodes (i.e., $\eta \in \boldsymbol{\eta}_{:\ell-1}$) is preserved in each layer's non-intervened neurons (i.e., $\boldsymbol{\psi}_{\ell}^{\phi} \cap \boldsymbol{\psi}_{\perp}^{\phi}$). This final condition will be useful to guarantee conditions ① and ② are preserved in future layers.

Statement. Conditions ①, ②, and ③ hold for all layers ℓ in a DNN.

Base Case ($\ell = 0$). For layer $\ell = 0$, we have $\boldsymbol{\eta}_{\ell} = \boldsymbol{\eta}_{\mathbf{x}}$. We also have both ϕ_{ℓ} and τ_{η} as the identity function. Further, we consider $\mathcal{I}^{:-1} = \{\emptyset\}$ and $\mathcal{I}^{:0} = \{\emptyset\}$—where symbol $\emptyset$ here denotes an empty intervention—and we consider $f_{\mathtt{N}}^{:0}$ to be the identity on $\mathbf{x}$. (Note that layer $f_{\mathtt{N}}^{0}$ is not applied in $f_{\mathtt{N}}^{:0}$.) Now, it is easy to prove our base case:

- ① follows trivially, as $f_{\mathtt{A}}^{\boldsymbol{\eta}_{\mathbf{x}}}(\mathbf{x},\mathbf{I}_{\mathtt{A}}) = \mathbf{x}$ and $f_{\mathtt{N}}^{:0}(\mathbf{x},\mathbf{I}_{\mathtt{N}}) = \mathbf{x}$.

- ② follows from Assump. 5.

- ③ follows trivially given $\boldsymbol{\eta}_{:-1}$ is an empty set.

Induction Step (given $\ell - 1$, then ℓ). Now, due to the inductive hypothesis, we assume that ①, ② and ③ hold for layer $(\ell-1)$. Given this, we must now prove that these conditions also hold for layer ℓ. We will consider two cases: $\boldsymbol{\eta}_{\ell}$ is either empty or not. Before doing so, however, we note that ① and ③ hold for layer $(\ell-1)$'s pre-intervention hidden variables. In Lemmas 1 and 2, we show that the same applies for the post-intervention hidden variables.

Let's consider the **case where $\boldsymbol{\eta}_{\ell}$ is empty**.

In this case, we can simply define ϕ_{ℓ} as the identity map. Further, given an empty $\boldsymbol{\eta}_{\ell}$, we know that there are no interventions in this layer, i.e., $\mathcal{I}_{\mathtt{N}}^{\ell} = \{\emptyset\}$, and, as such, we have that: $\mathcal{I}_{\mathtt{N}}^{:\ell} = \mathcal{I}_{\mathtt{N}}^{:\ell-1} \times \mathcal{I}_{\mathtt{N}}^{\ell} = \mathcal{I}_{\mathtt{N}}^{:\ell-1}$. We can now prove the induction step for this case.

- ① is true trivially, since $\boldsymbol{\eta}_{\ell}$ is empty.

- ② follows using the inductive hypothesis. Let $\mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}^{:\ell}$ and $\mathbf{x} \in \mathcal{X}$. Now, let $\mathbf{h}'_{\boldsymbol{\psi}_{\ell}} = f_{\mathtt{N}}^{:\ell}(\mathbf{x},\mathbf{I}_{\mathtt{N}})$, $\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}} = f_{\mathtt{N}}^{:\ell-1}(\mathbf{x},\mathbf{I}_{\mathtt{N}})$, and $\mathbf{I}_{\mathtt{A}} = \omega_{\tau}(\mathbf{I}_{\mathtt{N}})$. We can now show that:

$$\tau_{\boldsymbol{\eta}_{\mathbf{y}}}\left(f_{\mathtt{N}}^{\ell:}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell}})\right) = \tau_{\boldsymbol{\eta}_{\mathbf{y}}}\left(f_{\mathtt{N}}^{\ell:}\left(f_{\mathtt{N}}^{:\ell}(\mathbf{x},\mathbf{I}_{\mathtt{N}})\right)\right) \qquad \text{definition of } \mathbf{h}'_{\boldsymbol{\psi}_{\ell}} \quad (22\text{a})$$

$$= \tau_{\boldsymbol{\eta}_{\mathbf{y}}}\left(f_{\mathtt{N}}^{\ell:}\left(f_{\mathtt{N}}^{\ell-1}\left(f_{\mathtt{N}}^{:\ell-1}(\mathbf{x},\mathbf{I}_{\mathtt{N}})\right)\right)\right) \qquad \text{no intervention at layer } \ell \quad (22\text{b})$$

$$= \tau_{\boldsymbol{\eta}_{\mathbf{y}}}\left(f_{\mathtt{N}}^{\ell-1:}\left(f_{\mathtt{N}}^{:\ell-1}(\mathbf{x},\mathbf{I}_{\mathtt{N}})\right)\right) \qquad \text{definition of } f_{\mathtt{N}}^{\ell-1:} \quad (22\text{c})$$

$$= \tau_{\boldsymbol{\eta}_{\mathbf{y}}}\left(f_{\mathtt{N}}^{\ell-1:}\left(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}}\right)\right) \qquad \text{definition of } \mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}} \quad (22\text{d})$$

$$= f_{\mathtt{A}}(\mathbf{x},\mathbf{I}_{\mathtt{A}}) \qquad \text{inductive hypothesis on } ② \quad (22\text{e})$$

This shows ② holds for layer ℓ when $\boldsymbol{\eta}_{\ell}$ is empty.

- ③ follows using the inductive hypothesis. Let $\mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}^{:\ell-1}$, $\mathbf{x} \in \mathcal{X}$, and $\eta \in \boldsymbol{\eta}_{:\ell-1}$. Now, let $\mathbf{h}'_{\boldsymbol{\psi}_{\ell}^{\phi}} = \phi_{\ell}(f_{\mathtt{N}}^{:\ell}(\mathbf{x},\mathbf{I}_{\mathtt{N}}))$, $\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^{\phi}} = \phi_{\ell-1}(f_{\mathtt{N}}^{:\ell-1}(\mathbf{x},\mathbf{I}_{\mathtt{N}}))$, and $\mathbf{I}_{\mathtt{A}} = \omega_{\tau}(\mathbf{I}_{\mathtt{N}})$. Further, let

$g_\eta^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}) = f_{\mathtt{A}}^{:\eta}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})$; we know such function exists due to the inductive hypothesis on ① and ③, together with Lemmas 1 and 2. Finally, since $f_{\mathtt{N}}^{:\ell}$ is injective (by Assump. 2) and since $f_{\mathtt{N}}^{:\ell} = f_{\mathtt{N}}^{\ell-1} \circ f_{\mathtt{N}}^{:\ell-1}$, we know that each $\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}$ is mapped to a unique $\mathbf{h}'_{\boldsymbol{\psi}_{\ell}^\phi}$ in the next layer. We can thus define function $\lambda_{\mathtt{N}}^{\ell-1}$ on the domain formed by these hidden variables which, given a hidden variable $\mathbf{h}'_{\boldsymbol{\psi}_{\ell}^\phi}$ returns its "parent" $\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}$; in other words, $\lambda_{\mathtt{N}}^{\ell-1}$ is an partial inverse of $f_{\mathtt{N}}^{\ell-1}$ defined only on its image. Defining now function $g_\eta^\ell = g_\eta^{\ell-1} \circ \lambda_{\mathtt{N}}^{\ell-1}$, we can show:

$$g_\eta^\ell(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = g_\eta^\ell(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi}) \hspace{3cm} \eta_\ell \text{ is empty} \quad (23\mathrm{a})$$

$$= g_\eta^\ell\left(\phi_\ell\left(f_{\mathtt{N}}^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi})\right)\right) \hspace{1.5cm} \text{no intervention at layer } \ell \quad (23\mathrm{b})$$

$$= g_\eta^\ell\left(f_{\mathtt{N}}^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi})\right) \hspace{1.5cm} \phi_\ell \text{ is the identity function} \quad (23\mathrm{c})$$

$$= g_\eta^{\ell-1}\left(\lambda_{\mathtt{N}}^{\ell-1}\left(f_{\mathtt{N}}^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi})\right)\right) \hspace{1.5cm} \text{definition of } g^\ell \quad (23\mathrm{d})$$

$$= g_\eta^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}) \hspace{2.5cm} \text{definition of } \lambda_{\mathtt{N}}^{\ell-1} \quad (23\mathrm{e})$$

$$= f_{\mathtt{A}}^{:\eta}(\mathbf{x}, \mathbf{I}_{\mathtt{A}}) \hspace{2cm} \text{inductive hypothesis on } ① \text{ and } ③ \quad (23\mathrm{f})$$

Let's now consider the **case when η_ℓ is not empty**.

To show ①, ② and ③ for layer ℓ we need to find a suitable bijective ϕ_ℓ. We now show a careful way to construct this map which satisfies these conditions. To do so, we will again split this step of the proof into two parts. We will first take care of the case in which no interventions are applied to layer ℓ, guaranteeing that the model behaves correctly in those cases. In that case, we must handle the set of **input-restricted pre-intervention hidden states** in layer ℓ, which we define as:

$$\mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge \overset{\mathtt{def}}{=} \{f_{\mathtt{N}}^{\boldsymbol{\psi}_\ell}(\mathbf{x}, \mathbf{I}_{\mathtt{N}}) \mid \mathbf{x} \in \mathcal{X}, \underbrace{\mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}^{:\ell-1}}_{\texttt{pre-int., as we do not include } \mathcal{I}^\ell}\} \tag{24}$$

Notably, instead of defining the entire alignment map ϕ_ℓ at once, we will first define its behaviour only on those hidden states. We will denote this domain-restricted function as $\phi_\ell^\blacklozenge$. Given this function, we will be able to define a set of **input-restricted pre-intervention hidden variables** in layer ℓ as:

$$\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\blacklozenge \overset{\mathtt{def}}{=} \{\phi_\ell^\blacklozenge(\mathbf{h}_{\boldsymbol{\psi}_\ell}) \mid \mathbf{h}_{\boldsymbol{\psi}_\ell} \in \mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge\} \tag{25}$$

where $\blacklozenge$ represents the non-intervened hidden states and variables, and we will use $\maltese$ to represent the intervened instances. Note that $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\blacklozenge$ is the set of representations output by alignment map $\phi_\ell^\blacklozenge$.

The second case we will consider will then handle interventions on this layer, and will again guarantee that the model behaves as expected in those cases. We thus define the set of **input-restricted post-intervention hidden variables** as:

$$\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi} \overset{\mathtt{def}}{=} \{f_{\mathtt{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\mathtt{N}}) \mid \mathbf{x} \in \mathcal{X}, \underbrace{\mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}^{:\ell}}_{\texttt{post-int., as we include } \mathcal{I}^\ell}\} \tag{26}$$

Notably, what an intervention on layer ℓ does is re-combine the representations in $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\blacklozenge$. We can thus write $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}$ in terms of $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\blacklozenge$ as:

$$\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi} = \left(\underset{\eta \in \eta_\ell}{\times} \underbrace{\{\phi_\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell})_{\boldsymbol{\psi}_\eta^\phi} \mid \mathbf{h}_{\boldsymbol{\psi}_\ell} \in \mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge\}}_{\texttt{pre-int. h.v., projected to } \boldsymbol{\psi}_\eta^\phi}\right) \times \underbrace{\{\phi_\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell})_{\boldsymbol{\psi}_\perp^\phi \cap \boldsymbol{\psi}_\ell^\phi} \mid \mathbf{h}_{\boldsymbol{\psi}_\ell} \in \mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge\}}_{\texttt{pre-int. h.v., projected to } \boldsymbol{\psi}_\perp^\phi} \tag{27}$$

We further define the set of **input-restricted intervention-only hidden variables** as:

$$\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\maltese = \mathcal{H}_{\boldsymbol{\psi}_\ell^\phi} \setminus \mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\blacklozenge \tag{28}$$

By carefully defining the behaviour of ϕ_ℓ on this set, we can guarantee the conditions above to hold. In particular, we will define this part of the function via its inverse $\phi_\ell^{\clubsuit^{-1}}$, which maps these hidden variables back to hidden states. We therefore have ϕ_ℓ and its partial inverse defined as:

$$\phi_\ell(\mathbf{h}) = \begin{cases} \phi_\ell^{\blacklozenge}(\mathbf{h}), & \text{if } \mathbf{h} \in \mathcal{H}_{\psi_\ell}^{\blacklozenge} \\ \phi_\ell^{\clubsuit}(\mathbf{h}), & \text{if } \mathbf{h} \in \mathcal{H}_{\psi_\ell}^{\clubsuit} \end{cases} \qquad \phi_\ell^{-1}(\mathbf{h}) = \begin{cases} \phi_\ell^{\blacklozenge^{-1}}(\mathbf{h}), & \text{if } \mathbf{h} \in \mathcal{H}_{\psi_\ell^\phi}^{\blacklozenge} \\ \phi_\ell^{\clubsuit^{-1}}(\mathbf{h}), & \text{if } \mathbf{h} \in \mathcal{H}_{\psi_\ell^\phi}^{\clubsuit} \end{cases} \qquad (29)$$

We now define $\phi_\ell^{\blacklozenge}$.

Definition 8. *Partial map* $\phi_\ell^{\blacklozenge} : \mathcal{H}_{\psi_\ell}^{\blacklozenge} \to \mathbb{R}^{|\psi^\ell|}$ *is some fixed function that is injective on each dimension, i.e.,* $\forall_{i \in \{1, \ldots, |\psi_\ell^\phi|\}} \forall_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}_{\psi_\ell}^{\blacklozenge}} : \mathbf{h}_1 \neq \mathbf{h}_2 \Rightarrow \phi_\ell^{\blacklozenge}(\mathbf{h}_1)_i \neq \phi_\ell^{\blacklozenge}(\mathbf{h}_2)_i$.

Such a function exists, because $\mathcal{H}_{\psi_\ell}^{\blacklozenge}$ is countable (Lemma 4) and $\mathbb{R}$ is uncountable. Further, its partial inverse $\phi_\ell^{\blacklozenge^{-1}}$, defined on the image $\mathcal{H}_{\psi_\ell^\phi}^{\blacklozenge}$, exists because $\phi_\ell^{\blacklozenge}$ is injective.

We can now prove that conditions ① and ③ hold. We also prove that ② holds when $\mathbf{I}_N \in \mathcal{I}_N^{:\ell-1}$, i.e., when there is no intervention in layer ℓ.

- ① follows using the inductive hypothesis. Let $\mathbf{I}_N \in \mathcal{I}_N^{:\ell-1}$, $\mathbf{x} \in \mathcal{X}$, and $\eta \in \boldsymbol{\eta}_\ell$. Further, let $\mathbf{h}'_{\psi_\ell^\phi} = \phi_\ell(f_N^{:\ell}(\mathbf{x}, \mathbf{I}_N))$, $\mathbf{h}'_{\psi_{\ell-1}^\phi} = \phi_{\ell-1}(f_N^{:\ell-1}(\mathbf{x}, \mathbf{I}_N))$, and $\mathbf{I}_A = \omega_\tau(\mathbf{I}_N)$. Now, note that there exists a function g_η for which $v_\eta = g_\eta(\mathbf{v}_{\eta:\ell-1})$, as the parents of η are a subset of $\boldsymbol{\eta}_{:\ell-1}$. It now suffices to show that $\mathbf{h}'_{\psi_\eta^\phi}$ encodes information about $\mathbf{v}_{\eta:\ell-1} = [f_A^{:\eta}(\mathbf{x}, \mathbf{I}_A)]_{\eta \in \boldsymbol{\eta}:\ell-1}$. By the inductive hypothesis on ① and ③, together with Lemmas 1 and 2, we know that $\mathbf{h}'_{\psi_{\ell-1}^\phi}$ encodes information about $\mathbf{v}_{\eta:\ell-1}$; let $g_{\eta:\ell-1}$ be a function that extracts this information, i.e., $g_{\eta:\ell-1}(\mathbf{h}'_{\psi_{\ell-1}^\phi}) = \mathbf{v}_{\eta:\ell-1}$. Now, since $f_N^{:\ell}$ is injective, and $\phi_\ell^{\blacklozenge}$ is injective on each output dimension, we know that $\mathbf{h}'_{\psi_\ell} = [\phi_\ell^{\blacklozenge}(f_N^{\ell-1}(\phi_{\ell-1}^{-1}(\mathbf{h}'_{\psi_{\ell-1}^\phi})))]_{\psi_\eta^\phi}$ contains the same information as $\mathbf{h}'_{\psi_{\ell-1}^\phi}$. We can thus construct (partial) inverses $\lambda_N^{\ell-1}$, $\phi_{\ell,\psi_\eta^\phi}^{\blacklozenge^{-1}}$ and define τ_η as the composition $g_\eta \circ g_{:\eta-1} \circ \phi_{\ell-1} \circ \lambda_N^{\ell-1} \circ \phi_{\ell,\psi_\eta^\phi}^{\blacklozenge^{-1}}$, which concludes this step of the proof:

$$\tau_\eta([\mathbf{h}'_{\psi_\ell^\phi}]_{\psi_\eta^\phi}) = \tau_\eta([\phi_\ell^{\blacklozenge}(f_N^{\ell-1}(\phi_{\ell-1}^{-1}(\mathbf{h}'_{\psi_{\ell-1}^\phi})))]_{\psi_\eta^\phi}) \qquad \text{definition of } \mathbf{h}'_{\psi_\ell} \quad (30a)$$

$$= g_\eta(g_{:\eta-1}(\phi_{\ell-1}(\lambda_N^{\ell-1}(\phi_{\ell,\psi_\eta^\phi}^{\blacklozenge^{-1}}([\phi_\ell^{\blacklozenge}(f_N^{\ell-1}(\phi_{\ell-1}^{-1}(\mathbf{h}'_{\psi_{\ell-1}^\phi})))]_{\psi_\eta^\phi}))))) \qquad (30b)$$

$$= g_\eta(g_{:\eta-1}(\mathbf{h}'_{\psi_{\ell-1}^\phi})) \qquad (30c)$$

$$= f_A^{:\eta}(\mathbf{x}, \mathbf{I}_A) \qquad (30d)$$

- ② when $\mathbf{I}_N \in \mathcal{I}_N^{:\ell-1}$ follows using the inductive hypothesis. Let $\mathbf{I}_N \in \mathcal{I}_N^{:\ell-1}$ and $\mathbf{x} \in \mathcal{X}$. Now, let $\mathbf{h}'_{\psi_\ell} = f_N^{:\ell}(\mathbf{x}, \mathbf{I}_N)$, $\mathbf{h}'_{\psi_{\ell-1}} = f_N^{:\ell-1}(\mathbf{x}, \mathbf{I}_N)$, and $\mathbf{I}_A = \omega_\tau(\mathbf{I}_N)$. We can show that:

$$\tau_{\boldsymbol{\eta}_\mathbf{y}}\left(f_N^{\ell:}(\mathbf{h}'_{\psi_\ell})\right) = \tau_{\boldsymbol{\eta}_\mathbf{y}}\left(f_N^{\ell:}\left(f_N^{:\ell}(\mathbf{x}, \mathbf{I}_N)\right)\right) \qquad \text{definition of } \mathbf{h}'_{\psi_\ell} \quad (31a)$$

$$= \tau_{\boldsymbol{\eta}_\mathbf{y}}\left(f_N^{\ell:}\left(f_N^{\ell-1}\left(f_N^{:\ell-1}(\mathbf{x}, \mathbf{I}_N)\right)\right)\right) \qquad \text{no intervention at layer } \ell \quad (31b)$$

$$= \tau_{\boldsymbol{\eta}_\mathbf{y}}\left(f_N^{\ell-1:}\left(f_N^{:\ell-1}(\mathbf{x}, \mathbf{I}_N)\right)\right) \qquad \text{definition of } f_N^{\ell-1:} \quad (31c)$$

$$= \tau_{\boldsymbol{\eta}_\mathbf{y}}\left(f_N^{\ell-1:}\left(\mathbf{h}'_{\psi_{\ell-1}}\right)\right) \qquad \text{definition of } \mathbf{h}'_{\psi_{\ell-1}} \quad (31d)$$

$$= f_A(\mathbf{x}, \mathbf{I}_A) \qquad \text{inductive hypothesis on ②} \quad (31e)$$

This shows ② holds for layer ℓ when there is no intervention in layer ℓ.

- ③ follows using the inductive hypothesis. Let $\mathbf{I}_N \in \mathcal{I}_N^{:\ell-1}$, $\mathbf{x} \in \mathcal{X}$, and $\eta \in \boldsymbol{\eta}_{:\ell-1}$. Now, let $\mathbf{h}'_{\psi_\ell^\phi} = \phi_\ell(f_N^{:\ell}(\mathbf{x}, \mathbf{I}_N))$, $\mathbf{h}'_{\psi_{\ell-1}^\phi} = \phi_{\ell-1}(f_N^{:\ell-1}(\mathbf{x}, \mathbf{I}_N))$, and $\mathbf{I}_A = \omega_\tau(\mathbf{I}_N)$. Further, let

23

$g_\eta^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}) = f_{\mathtt{A}}^{:\eta}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})$; we know such function exists due to the inductive hypothesis on ① and ③ together with Lemmas 1 and 2. Finally, since $f_{\mathtt{N}}^{:\ell}$ and $\phi_\ell^\blacklozenge$ are injective and $\phi_{\ell-1}$ is bijective, we can define the partial inverse function $\gamma_{\mathtt{N}}^{\ell-1}$ of their composition $\phi_\ell^\blacklozenge \circ f_{\mathtt{N}}^\ell \circ \phi_{\ell-1}^{-1}$ (applied only to their image) which, given the hidden variable $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi}$ returns its "parent" $\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}$. Defining now a function $\widehat{g}_\eta^\ell = g_\eta^{\ell-1} \circ \gamma_{\mathtt{N}}^{\ell-1}$ and $g_\eta^\ell(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = \widehat{g}_\eta^\ell(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi})$—which exists, as $\phi_\ell^\blacklozenge$ is injective on each dimension—we can show:

$$g_\eta^\ell(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = \widehat{g}_\eta^\ell(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi}) \qquad\qquad \phi_\ell^\blacklozenge \text{ is injective on all dimensions} \quad (32\text{a})$$

$$= \widehat{g}_\eta^\ell\Big(\phi_\ell^\blacklozenge\big(f_{\mathtt{N}}^{\ell-1}(\phi_{\ell-1}^{-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}))\big)\Big) \qquad\qquad \text{no intervention at layer } \ell \quad (32\text{b})$$

$$= g_\eta^{\ell-1}\Big(\gamma_{\mathtt{N}}^{\ell-1}\big(\phi_\ell^\blacklozenge\big(f_{\mathtt{N}}^{\ell-1}(\phi_{\ell-1}^{-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}))\big)\big)\Big) \qquad\qquad \text{definition of } g^\ell \quad (32\text{c})$$

$$= g_\eta^{\ell-1}(\mathbf{h}'_{\boldsymbol{\psi}_{\ell-1}^\phi}) \qquad\qquad \text{definition of } \gamma_{\mathtt{N}}^{\ell-1} \quad (32\text{d})$$

$$= f_{\mathtt{A}}^{:\eta}(\mathbf{x}, \mathbf{I}_{\mathtt{A}}) \qquad\qquad \text{inductive hypothesis on ① and ③} \quad (32\text{e})$$

We have now proved ① and ③. We have also partially proved ② for cases where there is no intervention in layer ℓ.[13] We now finish our proof by considering cases where there is an intervention in this layer ℓ. In the second case, we need to handle intervention-only representations $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\clubsuit$. We will now define $\phi_\ell^{\clubsuit^{-1}}$ on this domain $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\clubsuit$ to fulfil ②.

Definition 9. *Partial map $\phi_\ell^{\clubsuit^{-1}} : \mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\clubsuit \to \mathbb{R}^{|\boldsymbol{\psi}^\ell|}$ is some fixed function such that it holds:*

1. *$\phi_\ell^{\clubsuit^{-1}}$ maps to the set $\mathcal{H}_{\boldsymbol{\psi}_\ell} \setminus \mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge$*

2. *$\phi_\ell^{\clubsuit^{-1}}$ is an injective map*

3. *Let $\mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}^{:\ell} \setminus \mathcal{I}_{\mathtt{N}}^{:\ell-1}$ and $\mathbf{x} \in \mathcal{X}$. Now, let $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} = f_{\mathtt{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\mathtt{N}})$, and $\mathbf{I}_{\mathtt{A}} = \omega_\tau(\mathbf{I}_{\mathtt{N}})$. We have that $f_{\mathtt{N}}^{\ell:}(\phi_\ell^{\clubsuit^{-1}}(\mathbf{h})) = f_{\mathtt{A}}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})$.*

Where the first two conditions ensure the necessary bijectivity of ϕ_ℓ and the last characteristic ensures ②. Now, let $\mathbf{x} \in \mathcal{X}$ be any input and $\mathbf{I}_{\mathtt{N}} \in \mathcal{I}_{\mathtt{N}}^{:\ell}$ any intervention. Further, let $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} = f_{\mathtt{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\mathtt{N}})$, and $\mathbf{I}_{\mathtt{A}} = \omega_\tau(\mathbf{I}_{\mathtt{N}})$. We now note that—given ① and ③, and Lemmas 1 and 2—the value $v_\eta = f_{\mathtt{A}}^{:\eta}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})$ for all nodes $\eta \in \boldsymbol{\eta}_{:\ell}$ are encoded in $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi}$. This is enough information to determine the algorithm's output $\mathbf{y}^\star = f_{\mathtt{A}}(\mathbf{x}, \mathbf{I}_{\mathtt{A}})$. Now, define a function $g^\star$ which maps an element $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} \in \mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\clubsuit$ to the output algorithm $\mathtt{A}$ expects. Further, by Lemma 6 there exists an uncountably infinite set of hidden states $\mathcal{H}_{\boldsymbol{\psi}_\ell}^{(\mathbf{y}^\star)}$ such that:

$$\forall \mathbf{h} \in \mathcal{H}_{\boldsymbol{\psi}_\ell}^{(\mathbf{y}^\star)} : \mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}} [f_{\mathtt{N}}^{\ell:}(\mathbf{h}_{\boldsymbol{\psi}_\ell})]_{\mathbf{y}'} \tag{33}$$

We define $\hat{\mathcal{H}}_{\boldsymbol{\psi}_\ell}^{(\mathbf{y}^\star)} = \mathcal{H}_{\boldsymbol{\psi}_\ell}^{(\mathbf{y}^\star)} \setminus \mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge$, which—as $\mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge$ is countable—is still uncountably infinite. We can now map any $\mathbf{h} \in \mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\clubsuit$ to an element in $\hat{\mathcal{H}}_{\boldsymbol{\psi}_\ell}^{(g^\star(\mathbf{h}))}$ fulfilling the third characteristic of Definition 9. That such a mapping exists adhering to the first and second characteristic of Definition 9 is ensured by the fact that $\hat{\mathcal{H}}_{\boldsymbol{\psi}_\ell}^{(\mathbf{y}^\star)}$ is uncountable and $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}^\clubsuit$ is countable (shown in Lemma 5). Further, as $\phi_\ell^{\clubsuit^{-1}}$ is injective, its partial inverse $\phi_\ell^\clubsuit$ on its image $\mathcal{H}_{\boldsymbol{\psi}_\ell}^\clubsuit$ exists.

---

[13]This also proves the result for any input $\mathbf{x} \in \mathcal{X}$ and intervention $\mathbf{I}_{\mathtt{N}} \in \mathcal{I}^\ell$ where $\mathbf{h} \in \mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge$. By ③ at layer ℓ, there exists $\mathbf{I}'_{\mathtt{N}} \in \mathcal{I}^{\ell-1}$—constructed by applying only the interventions from $\mathbf{I}_{\mathtt{N}}$ on layers $< \ell$—such that $f_{\mathtt{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\mathtt{N}}) = f_{\mathtt{N}}^{:\ell}(\mathbf{x}, \mathbf{I}'_{\mathtt{N}})$. Since both encode the same values on $\boldsymbol{\eta}_{<\ell}$ according to ③, which fully determine the output of $f_{\mathtt{A}}$, and since ② holds for $(\mathbf{x}, \mathbf{I}'_{\mathtt{N}})$ at layer ℓ, it must also hold for $(\mathbf{x}, \mathbf{I}_{\mathtt{N}})$.

```
1    def extract_unused_rep(h, $\mathcal{H}^{\blacklozenge}_{\psi^{\phi}_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi^{\phi}_{\ell}}$):
2        rep=h
3        while rep∈ $\mathcal{H}^{\blacklozenge}_{\psi^{\phi}_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi^{\phi}_{\ell}}$:
4            rep=$\phi_{\ell}^{-1}$(rep)
5        return rep
```

Figure 7: Pseudo-code for `extract_unused_rep`. This function returns an unique element in $(\mathcal{H}^{\blacklozenge}_{\psi_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi_{\ell}}) \backslash (\mathcal{H}^{\blacklozenge}_{\psi^{\phi}_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi^{\phi}_{\ell}})$ for each $\mathbf{h} \in (\mathcal{H}^{\blacklozenge}_{\psi^{\phi}_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi^{\phi}_{\ell}}) \backslash (\mathcal{H}^{\blacklozenge}_{\psi_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi_{\ell}})$ ensuring bijectivity of $\phi'_{\ell}(\mathbf{h})$.

The attentive reader may have noticed that we defined ϕ_{ℓ} only over the domain $\mathcal{H}^{\blacklozenge}_{\psi_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi_{\ell}}$ instead over $\mathbb{R}^{|\psi_{\ell}|}$. We note that it is simple to extend ϕ_{ℓ} to an ϕ'_{ℓ} defined over $\mathbb{R}^{|\psi_{\ell}|}$. Let `id` be the identity function and `extract_unused_rep` be defined by the algorithm given in Fig. 7. A bijective function ϕ'_{ℓ} over $\mathbb{R}^{|\psi_{\ell}|}$ mapping to $\mathbb{R}^{|\psi_{\ell}|}$ can be defined as:

$$\phi'_{\ell}(\mathbf{h}) = \begin{cases} \phi_{\ell}(\mathbf{h}) & \text{if } \mathbf{h} \in \mathcal{H}^{\blacklozenge}_{\psi_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi_{\ell}} \\ \texttt{extract\_unused\_rep}(\mathbf{h}, \mathcal{H}^{\blacklozenge}_{\psi^{\phi}_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi^{\phi}_{\ell}}), & \text{if } \mathbf{h} \in (\mathcal{H}^{\blacklozenge}_{\psi^{\phi}_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi^{\phi}_{\ell}}) \backslash (\mathcal{H}^{\blacklozenge}_{\psi_{\ell}} \cup \mathcal{H}^{\clubsuit}_{\psi_{\ell}}) \\ \mathbf{h} & \text{else} \end{cases}$$

(34)

which completes the proof. □

F.1 Discussion about Assumptions

Assump. 1 (Countable input-space). While this assumption cannot be made on all neural networks like MLPs, it holds for models working on language and images. The set of all images with a specific resolution is finite, as it considers a finite number of pixels where each pixel has a finite number of channels (e.g. values for red, green and blue) and each channel is a number between 0 and 255. The set of all sequences in a language model is also countably infinite, as each set of sequences of some length is finite given finite tokens; so we have a set made out of the countable union of finite sets, which is still countable.

Assump. 2 (Input-injectivity in all layers). Neural network layers (e.g., MLP blocks) are not necessarily injective. The usage of learnable weights, activation functions like ReLU (Nair and Hinton, 2010) and information bottlenecks makes it possible to have a non-injective model. However, we prove in App. G that transformers, at least, are almost surely injective at initialisation on their inputs. Further, Nikolaou et al. (2025) recently published a proof—as well as empirical evidence—that transformers are almost surely injective in the hidden states of their last token, both at initialisation and after training. We also see in our empirical experiments in App. H that the MLPs we analyse are also, in practice, injective—or close enough to it that we observe no collisions in embedding space.

Assump. 3 (Strict output-surjectivity in all layers). Surjectivity can be defined on the output distribution $f_{\texttt{N}}^{\ell:} : \mathcal{H}_{\psi_{\ell}} \to \Delta^{|\mathcal{Y}|-1}$, but that is a rather strong assumption. For our proofs, we will rely on strict surjectivity on the classification space instead ($\tau_{\eta_{\mathbf{y}}} \circ f_{\texttt{N}}^{\ell:} : \mathcal{H}_{\psi_{\ell}} \to |\mathcal{Y}|$), such that every class can be predicted. However, surjectivity on the classification space still does not necessarily hold for DNNs. LLMs have problems like the softmax bottleneck (Yang et al., 2018), which can lead to a model having insufficient capacity to predict all possible tokens. Grivas et al. (2022) also evaluate and find this problem, but show that surjectivity on the tokens is still likely in practice, making this a reasonable assumption in LLM settings.

Assump. 4 (Algorithm and DNN have matchable partial-orderings). We assume this since, for a neural network $\texttt{N}$ to be abstracted by the algorithm $\texttt{A}$, we need it to have this minimal width and depth.

Assump. 5 (DNN solves the task). We assume this because, if a neural network does not solve the given task, it will also not be abstracted by an algorithm which implements it.

F.2 Detailed Version of Definition 7

Definition 7 can also be written without referring to previous definitions as following:

Alternative Definition 1 (Equivalent to Definition 7). *An algorithm* $\texttt{A}$ *is an* ***input-restricted distributed abstraction*** *of a neural network* $\texttt{N}$ *iff there exists an* τ, $\mathcal{I}_{\texttt{A}}$, *and* $\mathcal{I}_{\texttt{N}}$ *such that*

- τ *is a distributed abstraction map. I.e., there exists an alignment map ϕ, a latent-variable partition $\{\boldsymbol{\psi}_\eta^\phi \mid \eta \in \boldsymbol{\eta}_{\mathrm{int}}\} \cup \{\boldsymbol{\psi}_\perp^\phi\}$ of $\boldsymbol{\psi}_{\mathrm{int}}^\phi$ (with non-empty $\boldsymbol{\psi}_\eta^\phi$), and subabstraction maps $\{\tau_\eta \mid \eta \in \boldsymbol{\eta}_{\mathrm{int}}\}$ such that τ is equivalent to computing the value of each node block-wise with $\tau_\eta(\mathbf{h}_{\boldsymbol{\psi}_\eta^\phi})$. An alignment map ϕ is a bijective function that maps the inner neurons $\boldsymbol{\psi}_{\mathrm{int}}$ of N onto an equal-sized set of latent variables $\boldsymbol{\psi}_{\mathrm{int}}^\phi$, with ϕ respecting the network's computational order by being the combination of layer-wise bijections $\phi_\ell : \mathbb{R}^{|\boldsymbol{\psi}_\ell|} \to \mathbb{R}^{|\boldsymbol{\psi}_\ell|}$ applied to the neurons of each of the DNN's layers (ℓ);*

- $\mathcal{I}_{\mathrm{A}}$ *and $\mathcal{I}_{\mathrm{N}}$ are a maximal input-restricted intervention set. A maximal input-restricted intervention set is composed of all interventions produced from other input-restricted interventions, i.e., it is a set with $\mathbf{h}_{\boldsymbol{\psi}^\phi} \leftarrow \mathbf{c}_{\boldsymbol{\psi}^\phi}$ (where $\boldsymbol{\psi}^\phi \subseteq \boldsymbol{\psi}_{\mathrm{int}}^\phi$) or $\mathbf{v}_\eta \leftarrow \mathbf{c}_\eta$ (where $\eta \subseteq \boldsymbol{\eta}_{\mathrm{int}}$) where $\mathbf{c}_{\boldsymbol{\psi}^\phi}$ or $\mathbf{c}_\eta$ arise from valid input-restricted computations (e.g., $\mathbf{c}_{\boldsymbol{\psi}^\phi} = f_{\mathrm{N}}^{\boldsymbol{\psi}^\phi}(\mathbf{x})$ or $\mathbf{c}_\eta = f_{\mathrm{A}}^{:\eta}(\mathbf{x}, \mathbf{c}_\eta \leftarrow f_{\mathrm{A}}^{:\eta}(\mathbf{x})))$.*

- τ *is surjective;*

- $\mathcal{I}_{\mathrm{A}} = \omega_\tau(\mathcal{I}_{\mathrm{N}})$;

- *There exists a surjective $\tau_{\boldsymbol{\eta}_{\mathbf{x}}}$ such that*

$$\bigvee_{\substack{\mathbf{x} \in \mathcal{X} \\ \mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}}} : \tau(f_{\mathrm{N}}^{\boldsymbol{\psi}_{\mathrm{int}}^\phi}(\mathbf{x}, \mathbf{I}_{\mathrm{N}})) = f_{\mathrm{A}}^{:\boldsymbol{\eta}_{\mathrm{int}}}(\tau_{\boldsymbol{\eta}_{\mathbf{x}}}(\mathbf{x}), \mathbf{I}_{\mathrm{A}}) \quad \text{where } \mathbf{I}_{\mathrm{A}} = \omega_\tau(\mathbf{I}_{\mathrm{N}}) \tag{35}$$

F.3 Useful Definitions and Lemmas for Theorem 1

Definition 10. *We say the composition of a function $f : \mathbb{R}^d \to \Delta^{|\mathcal{Y}|-1}$ with argmax is **strictly surjective** if, for any output $\mathbf{y}^\star \in \mathcal{Y}$, there exists an input $\mathbf{h} \in \mathbb{R}^d$ for which f outputs $\mathbf{y}^\star$ no matter how ties are broken in the argmax. Formally:*

$$\forall \mathbf{y}^\star \in \mathcal{Y}, \exists \mathbf{h} \in \mathbb{R}^d, \forall \mathbf{y}' \in \mathcal{Y} \setminus \{\mathbf{y}^\star\} : [f(\mathbf{h})]_{\mathbf{y}^\star} > [f(\mathbf{h})]_{\mathbf{y}'} \tag{36}$$

Lemma 1. *Let N be a DNN and A be an algorithm. Further, let τ be a distributed abstraction map with partition $\boldsymbol{\Psi}$ and $\{\tau_\eta\}_{\eta \in \boldsymbol{\eta}_{\mathrm{int}}}$. If, for all $\eta \in \boldsymbol{\eta}_\ell$, τ_η satisfies the conditions in ① (defined in Theorem 1's proof) applied on layer ℓ's pre-intervention hidden variables, i.e., if:*

$$\bigvee_{\substack{\mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell-1} \\ \mathbf{x} \in \mathcal{X}}} : \underbrace{\tau_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi}) = f_{\mathrm{A}}^\eta(\mathbf{x}, \mathbf{I}_{\mathrm{A}})}_{\textit{tested condition}}, \quad \text{where } \underbrace{\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} = f_{\mathrm{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\mathrm{N}})}_{\textit{pre-int. hidden variable, as } \mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell-1}}, \text{ and } \mathbf{I}_{\mathrm{A}} = \omega_\tau(\mathbf{I}_{\mathrm{N}}) \tag{37}$$

where we note that $f_{\mathrm{N}}^{\boldsymbol{\psi}_\ell^\phi} = \phi_\ell \circ f_{\mathrm{N}}^{:\ell}$ when no intervention is applied to layer ℓ. Then τ_η also satisfies this condition when applied to layer ℓ's post-intervention hidden variables:

$$\bigvee_{\substack{\mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell} \\ \mathbf{x} \in \mathcal{X}}} : \underbrace{\tau_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi}) = f_{\mathrm{A}}^\eta(\mathbf{x}, \mathbf{I}_{\mathrm{A}})}_{\textit{tested condition}}, \quad \text{where } \underbrace{\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} = f_{\mathrm{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\mathrm{N}})}_{\textit{post-int. hidden variable, as } \mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell}}, \text{ and } \mathbf{I}_{\mathrm{A}} = \omega_\tau(\mathbf{I}_{\mathrm{N}}) \tag{38}$$

Proof. Let τ_η be the abstraction map of $\eta \in \boldsymbol{\eta}_\ell$. By assumption, condition ① holds for all pre-intervention hidden variables, i.e., hidden variables of the form $\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi} = [\phi_\ell(f_{\mathrm{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\mathrm{N}}))]_{\boldsymbol{\psi}_\eta^\phi}$. We can show the same function applies to post-intervention hidden variables, i.e., hidden variables of the form:

$$\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi} = \begin{cases} \mathbf{c}_{\boldsymbol{\psi}_\eta^\phi} & \text{if } \mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi} \leftarrow \mathbf{c}_{\boldsymbol{\psi}_\eta^\phi} \in \mathbf{I}_{\mathrm{N}} \\ \left[\phi_\ell(f_{\mathrm{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\mathrm{N}}))\right]_{\boldsymbol{\psi}_\eta^\phi} & \text{else} \end{cases} \tag{39}$$

Now let $\mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell}$ be any intervention and $\mathbf{x} \in \mathcal{X}$ be any input. Further, let $\mathbf{I}_{\mathrm{A}} = \omega_\tau(\mathbf{I}_{\mathrm{N}})$. If $\mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell-1}$, then the post-intervention hidden variable is identical to a pre-intervention one, and the conditions in ① still hold, i.e.,: $\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi} = [\phi_\ell(f_{\mathrm{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\mathrm{N}}))]_{\boldsymbol{\psi}_\eta^\phi}$ and τ_η is such that $\tau_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi}) = f_{\mathrm{A}}^\eta(\mathbf{x}, \mathbf{I}_{\mathrm{A}})$. If $\mathbf{I}_{\mathrm{N}} \notin \mathcal{I}_{\mathrm{N}}^{:\ell-1}$, for each node's hidden variables $\boldsymbol{\psi}_\eta^\phi$, we might or not intervene on it. If we do not intervene on node η, then we still have the case $\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi} = [\phi_\ell(f_{\mathrm{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\mathrm{N}}))]_{\boldsymbol{\psi}_\eta^\phi}$ and thus τ_η still gives us the correct solution, i.e., $\tau_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi}) = f_{\mathrm{A}}^\eta(\mathbf{x}, \mathbf{I}_{\mathrm{A}})$. If we intervene on η, then we know there exists an intervention of form

$\mathbf{h}_{\boldsymbol{\psi}_\eta^\phi} \leftarrow f_{\text{N}}^{:\ell}(\mathbf{x}', \mathbf{I}_{\text{N}}')$ in $\mathbf{I}_{\text{N}}$, for which $\mathbf{I}_{\text{N}}' \in \mathcal{I}_{\text{N}}^{:\ell-1}$, as our interventions are input-restricted. We also know (by eq. (4)) that there exists an equivalent intervention $v_\eta \leftarrow \tau_\eta(f_{\text{N}}^{:\ell}(\mathbf{x}', \mathbf{I}_{\text{N}}'))$ in $\mathbf{I}_{\text{A}}$. We thus have that $\tau_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\eta^\phi}) = f_{\text{A}}^\eta(\mathbf{x}, \mathbf{I}_{\text{A}})$. $\qquad\square$

Lemma 2. *Let* N *be a DNN and* A *be an algorithm. Further, let* τ *be a distributed abstraction map with partition* $\boldsymbol{\Psi}$. *If, for all* $\eta \in \boldsymbol{\eta}_{:\ell-1}$, *there exists a function* g_η *which satisfies the conditions in* ③ *(defined in Theorem 1's proof) applied on layer* ℓ*'s pre-intervention hidden variables, i.e., if:*

$$\underset{\substack{\mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell-1} \\ \mathbf{x} \in \mathcal{X}}}{\forall} : \underbrace{g_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = f_{\text{A}}^\eta(\mathbf{x}, \mathbf{I}_{\text{A}})}_{\text{tested condition}}, \qquad \text{where } \mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} = \underbrace{f_{\text{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\text{N}}), \text{ and } \mathbf{I}_{\text{A}} = \omega_\tau(\mathbf{I}_{\text{N}})}_{\text{pre-int. hidden variable, as } \mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell-1}} \qquad (40)$$

where we note that $f_{\text{N}}^{\boldsymbol{\psi}_\ell^\phi} = \phi_\ell \circ f_{\text{N}}^{:\ell}$ *when no intervention is applied to layer* ℓ. *Then* g_η *also satisfied this condition when applied to layer* ℓ*'s post-intervention hidden variables:*

$$\underset{\substack{\mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell} \\ \mathbf{x} \in \mathcal{X}}}{\forall} : \underbrace{g_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = f_{\text{A}}^\eta(\mathbf{x}, \mathbf{I}_{\text{A}})}_{\text{tested condition}}, \qquad \text{where } \mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi} = \underbrace{f_{\text{N}}^{\boldsymbol{\psi}_\ell^\phi}(\mathbf{x}, \mathbf{I}_{\text{N}}), \text{ and } \mathbf{I}_{\text{A}} = \omega_\tau(\mathbf{I}_{\text{N}})}_{\text{post-int. hidden variable, as } \mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell}} \qquad (41)$$

Proof. Let $\eta \in \boldsymbol{\eta}_{:\ell-1}$ and g_η be a function that satisfies condition ③ for it. Note that condition ③ holds for all pre-intervention hidden variables, i.e., hidden variables of the form $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} = [\phi_\ell(f_{\text{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\text{N}}))]_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}$. We can show the same function g_η also applies to post-intervention hidden variables, i.e., hidden variables of the form:

$$\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} = \begin{cases} \mathbf{c}_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} & \text{if } \mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} \leftarrow \mathbf{c}_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} \in \mathbf{I}_{\text{N}} \\ [\phi_\ell(f_{\text{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\text{N}}))]_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} & \text{else} \end{cases} \qquad (42)$$

Now, let $\mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell}$, $\mathbf{x} \in \mathcal{X}$. Further, let $\mathbf{I}_{\text{A}} = \omega_\tau(\mathbf{I}_{\text{N}})$. If $\mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell-1}$, then the post-intervention hidden variable is identical to a pre-intervention one, and the conditions in ③ still hold, i.e.,: $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} = [\phi_\ell(f_{\text{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\text{N}}))]_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}$ and g_η is such that $g_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = f_{\text{A}}^\eta(\mathbf{x}, \mathbf{I}_{\text{A}})$. If $\mathbf{I}_{\text{N}} \notin \mathcal{I}_{\text{N}}^{:\ell-1}$, it means that we intervene on at least one hidden variable in this layer $\boldsymbol{\psi}_\ell^\phi$. However, we never intervene on neurons in $\boldsymbol{\psi}_\perp^\phi$, meaning that for those we still have the case $\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi} = [\phi_\ell(f_{\text{N}}^{:\ell}(\mathbf{x}, \mathbf{I}_{\text{N}}))]_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}$ and thus the same function g_η still satisfies our condition $g_\eta(\mathbf{h}'_{\boldsymbol{\psi}_\ell^\phi \cap \boldsymbol{\psi}_\perp^\phi}) = f_{\text{A}}^\eta(\mathbf{x}, \mathbf{I}_{\text{A}})$. $\qquad\square$

Lemma 3. *Under Assump. 1 and given a fixed* ϕ, *the set of input-restricted interventions* $\mathcal{I}_{\text{N}}$ *is countable.*

Proof. This can be shown by induction. More specifically, we show that for any layer ℓ, the set of input-restricted interventions $\mathcal{I}_{\text{N}}^{:\ell}$ is countable for a specific ϕ.

Base Case ($\ell = 0$). The base case can be proved trivially, as $\mathcal{I}_{\text{N}}^{:0} = \{\emptyset\}$.

Induction step (ℓ given $\ell - 1$). By the induction hypothesis, $\mathcal{I}_{\text{N}}^{:\ell-1}$ is countable. Now, note that $\mathcal{I}_{\text{N}}^{:\ell}$ can be decomposed as:

$$\mathcal{I}_{\text{N}}^{:\ell} = \mathcal{I}_{\text{N}}^{:\ell-1} \times \mathcal{I}_{\text{N}}^\ell \qquad (43)$$

As the Cartesian product of two countable sets is itself countable, and as $\mathcal{I}_{\text{N}}^{:\ell-1}$ is countable by the inductive hypothesis, we only need to show that $\mathcal{I}_{\text{N}}^\ell$ is countable to complete our proof. This set $\mathcal{I}_{\text{N}}^\ell$ is defined as the set of all input-restricted interventions to layer ℓ. Given a set of neurons or hidden variables in this layer $\boldsymbol{\psi}'$, we are thus dealing with interventions of the form: $\mathbf{h}_{\boldsymbol{\psi}'} \leftarrow f_{\text{N}}^{\boldsymbol{\psi}'}(\mathbf{x}, \mathbf{I}_{\text{N}})$, where: (i) $\boldsymbol{\psi}' \subseteq \boldsymbol{\psi}_\ell$ or $\boldsymbol{\psi}' \subseteq \boldsymbol{\psi}_\ell^\phi$; (ii) $\mathbf{x} \in \mathcal{X}$; and (iii) $\mathbf{I}_{\text{N}} \in \mathcal{I}_{\text{N}}^{:\ell-1}$. The set of all input-restricted interventions in this layer is thus bounded in size by the Cartesian product: $\bigtimes_{\boldsymbol{\psi} \in \boldsymbol{\psi}_\ell^\phi} \mathcal{X} \times \mathcal{I}_{\text{N}}^{:\ell-1}$.

These three sets are countable, and thus so is $\mathcal{I}_{\text{N}}^\ell$. This concludes our proof. $\qquad\square$

Lemma 4. *Under Assump. 1 and given a fixed* ϕ, *the set of input-restricted pre-intervention hidden states in layer* ℓ, *i.e.,* $\mathcal{H}_{\boldsymbol{\psi}_\ell}^\blacklozenge$, *is countable.*

Proof. The set of input-restricted hidden states $\mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell}$ is formed by hidden states $\mathbf{h}_{\boldsymbol{\psi}_\ell} = f_{\mathrm{N}}^{\boldsymbol{\psi}_\ell}(\mathbf{x}, \mathbf{I}_{\mathrm{N}})$, which we can write as:

$$\mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell} = \{ f_{\mathrm{N}}^{\boldsymbol{\psi}_\ell}(\mathbf{x}, \mathbf{I}_{\mathrm{N}}) \mid \mathbf{x} \in \mathcal{X}, \mathbf{I}_{\mathrm{N}} \in \mathcal{I}_{\mathrm{N}}^{:\ell-1} \} \tag{44}$$

We thus have that the size of $\mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell}$ is bounded by the size of the Cartesian product $\mathcal{X} \times \mathcal{I}_{\mathrm{N}}^{:\ell}$. As both of these sets are countable (by Assump. 1 and Lemma 3, respectively), $\mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell}$ is also countable. This completes our proof. $\qquad\square$

Lemma 5. *Under Assump. 1 and given a fixed ϕ, the set of input-restricted intervention-only hidden variables in layer ℓ, i.e., $\mathcal{H}^{\clubsuit}_{\boldsymbol{\psi}_\ell^\phi}$, is countable.*

Proof. A similar proof to Lemma 4 applies here. In short, we have three relevant sets for this proof. First, the set of input-restricted pre-intervention hidden variables:

$$\mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell^\phi} = \{ \phi_\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell}) \mid \mathbf{h}_{\boldsymbol{\psi}_\ell} \in \mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell} \} \tag{45}$$

Second, we have the set of input-restricted post-intervention hidden variables:

$$\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi} = \left(\underset{\eta \in \eta_\ell}{\times} \underbrace{\{ \phi_\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell})_{\boldsymbol{\psi}_\eta^\phi} \mid \mathbf{h}_{\boldsymbol{\psi}_\ell} \in \mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell} \}}_{\texttt{Pre-int. h.v., projected to } \boldsymbol{\psi}_\eta^\phi} \right) \times \underbrace{\{ \phi_\ell(\mathbf{h}_{\boldsymbol{\psi}_\ell})_{\boldsymbol{\psi}_\perp^\phi \cap \boldsymbol{\psi}_\ell^\phi} \mid \mathbf{h}_{\boldsymbol{\psi}_\ell} \in \mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell} \}}_{\texttt{Pre-int. h.v., projected to } \boldsymbol{\psi}_\perp^\phi} \tag{46}$$

Both sets above are countable, since $\mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell}$ is countable (by Lemma 4), and η_ℓ is finite. Third, we have the set of input-restricted intervention-only hidden variables, defined as:

$$\mathcal{H}^{\clubsuit}_{\boldsymbol{\psi}_\ell^\phi} = \mathcal{H}_{\boldsymbol{\psi}_\ell^\phi} \setminus \mathcal{H}^{\blacklozenge}_{\boldsymbol{\psi}_\ell^\phi} \tag{47}$$

Since $\mathcal{H}_{\boldsymbol{\psi}_\ell^\phi}$ is countable, $\mathcal{H}^{\clubsuit}_{\boldsymbol{\psi}_\ell^\phi}$ is clearly also countable. This completes the proof. $\qquad\square$

Lemma 6. *Under Assump. 3 and given a target output $\mathbf{y}^\star \in \mathcal{Y}$, we know that there is an uncountably infinite set $\mathcal{H}^{(\mathbf{y}^\star)}_{\boldsymbol{\psi}_\ell}$ which predicts it, i.e.,:*

$$\mathbf{h} \in \mathcal{H}^{(\mathbf{y}^\star)}_{\boldsymbol{\psi}_\ell} \Leftrightarrow \mathbf{y}^\star = \underset{\mathbf{y}' \in \mathcal{Y}}{\operatorname{argmax}} \, [f_{\mathrm{N}}^{\ell:}(\mathbf{h})]_{\mathbf{y}'} \tag{48}$$

Proof. Under Assump. 3, we know that—for any target output $\mathbf{y}^\star \in \mathcal{Y}$—there is at least one hidden state $\mathbf{h}^\star \in \mathbb{R}^{|\boldsymbol{\psi}_\ell|}$ which predicts it, i.e.:

$$\mathbf{y}^\star = \underset{\mathbf{y}' \in \mathcal{Y}}{\operatorname{argmax}} \, [f_{\mathrm{N}}^{\ell:}(\mathbf{h}^\star)]_{\mathbf{y}'} \tag{49}$$

where we note that $f_{\mathrm{N}}^{\ell:}(\mathbf{h}^\star)$ outputs a probability distribution over $\mathcal{Y}$, i.e., $p_{\mathrm{N}}(\mathbf{y}' \mid \mathbf{h}^\star)$.

To show that we have an uncountably infinite set, let us first notice that

$$\forall \mathbf{h}' \in \mathbb{R}^{|\boldsymbol{\psi}_\ell|} : ||f_{\mathrm{N}}^{\ell:}(\mathbf{h}) - f_{\mathrm{N}}^{\ell:}(\mathbf{h}')||_2 < \frac{m_1 - m_2}{2} \Rightarrow \mathbf{y}^\star = \underset{\mathbf{y}' \in \mathcal{Y}}{\operatorname{argmax}} \, [f_{\mathrm{N}}^{\ell:}(\mathbf{h}')]_{\mathbf{y}'} \tag{50}$$

for m_1 be the max value of $f_{\mathrm{N}}^{\ell:}(\mathbf{h})$ and m_2 the second highest value of $f_{\mathrm{N}}^{\ell:}(\mathbf{h})$. $m_1 > m_2$ follows by the strict subjectivity mentioned in *Assump.* 3. Eq. (50) follows by the definition of the euclidean norm ($||.||_2$), argmax and $f_{\mathrm{N}}^{\ell:}(\mathbf{h}') \in \Delta^{|\boldsymbol{\psi}_L|-1}$ as m_1 has to be lowered at least $\frac{m_1 - m_2}{2}$ to increase m_2 by $\frac{m_1 - m_2}{2}$ for those two values to be the same. Increasing any other value in $f_{\mathrm{N}}^{\ell:}(\mathbf{h})$ would require m_1 being lowered more than $\frac{m_1 - m_2}{2}$ or any other value increased by more than $\frac{m_1 - m_2}{2}$. Now, given continuity of neural networks, we know that:

$$\forall \epsilon > 0, \exists \delta > 0, \forall \mathbf{h}' \in \mathbb{R}^{|\boldsymbol{\psi}_\ell|} : 0 < ||\mathbf{h} - \mathbf{h}'||_2 < \delta,$$
$$\Rightarrow ||f_{\mathrm{N}}^{\ell:}(\mathbf{h}) - f_{\mathrm{N}}^{\ell:}(\mathbf{h}')||_2 < \epsilon. \tag{51}$$

Therefore, we see that:

$$\exists \delta > 0, \forall \mathbf{h}' \in \mathbb{R}^{|\psi_\ell|} : 0 < ||\mathbf{h} - \mathbf{h}'||_2 < \delta,$$

$$\Rightarrow ||f_{\mathbb{N}}^{\ell:}(\mathbf{h}) - f_{\mathbb{N}}^{\ell:}(\mathbf{h}')||_2 < \frac{m_1 - m_2}{2} \tag{52a}$$

$$\Rightarrow \mathbf{y}^\star = \underset{\mathbf{y}' \in \mathcal{Y}}{\operatorname{argmax}} \, [f_{\mathbb{N}}^{\ell:}(\mathbf{h}')]_{\mathbf{y}'} \tag{52b}$$

We notice that $||\mathbf{h} - \mathbf{h}'||_2 < \delta$ for $\delta > 0$ denotes a continuous region in $\mathbb{R}^{|\psi_\ell|}$ which therefore includes uncountably infinite points. □

G   Transformers at Initialisation are Almost Surely Injective on each Layer

Theorem 2. *Transformers like DNN 2 with randomly independent initialised from a continuous distribution (riicd.) weights are almost surely injective at initialisation up to each layer $0 \le \ell < L$.*

Proof. To show injectivity up to a layer ℓ' in a transformer, it suffices to show that $f_{\mathbb{N}}^\ell$ is injective on any countable subset $\mathcal{H}$ of its domain for all layers ℓ ($0 \le \ell \le \ell'$). This suffices as we assume the set of inputs $\mathcal{X}$ is countable, and the composition of injective functions is injective. Let Θ be the random variable representing the transformer's weights. To show (almost sure) injectivity on layer ℓ for any fixed input set $\mathcal{H}$, we need that (because of Lemma 10):[14]

$$p_\Theta \left(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 \neq \mathbf{h}_2 : f_{\mathbb{N}}^\ell(\mathbf{h}_1) \neq f_{\mathbb{N}}^\ell(\mathbf{h}_2) \right) = 1 \tag{53}$$

Since the transformer operates over sequences of tokens, any element $\mathbf{h} \in \mathcal{H}$ has its first dimension indexing the sequence length. Let $|\mathbf{h}|$ denote the sequence length and $[\mathbf{h}]_t$ refer to the t-th element in $\mathbf{h}$. Let T be the set of token positions $T = \{1, \ldots, \min(|\mathbf{h}_1|, |\mathbf{h}_2|)\}$. For injectivity, it suffices to show that:

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : [f_{\mathbb{N}}^\ell(\mathbf{h}_1)]_t \neq [f_{\mathbb{N}}^\ell(\mathbf{h}_2)]_t) = 1 \tag{54}$$

Note that eq. (54) only ensures injectivity when $|\mathbf{h}_1| = |\mathbf{h}_2|$. However, this is sufficient because when $|\mathbf{h}_1| \neq |\mathbf{h}_2|$, eq. (53) follows trivially: since $|f_{\mathbb{N}}^\ell(\mathbf{h}_1)| \neq |f_{\mathbb{N}}^\ell(\mathbf{h}_2)|$, we immediately have $f_{\mathbb{N}}^\ell(\mathbf{h}_1) \neq f_{\mathbb{N}}^\ell(\mathbf{h}_2)$. When $|\mathbf{h}_1| = |\mathbf{h}_2|$, we can show that eq. (54) implies eq. (53) as follows: if $\mathbf{h}_1 \neq \mathbf{h}_2$, then there exists at least one token position $t' \in T$ where $[\mathbf{h}_1]_{t'} \neq [\mathbf{h}_2]_{t'}$. By eq. (54), this implies $[f_{\mathbb{N}}^\ell(\mathbf{h}_1)]_{t'} \neq [f_{\mathbb{N}}^\ell(\mathbf{h}_2)]_{t'}$ almost surely, and therefore $f_{\mathbb{N}}^\ell(\mathbf{h}_1) \neq f_{\mathbb{N}}^\ell(\mathbf{h}_2)$ almost surely.

We observe that a transformer's input set $\mathcal{X}$ consists of all sequences formed from a finite token vocabulary, which is countably infinite. Since transformers are deterministic functions, the input set $\mathcal{H}$ encountered at any sublayer is also countably infinite. Therefore, it suffices to prove eq. (54) for any fixed countably infinite input set $\mathcal{H}$.

We show that Eq. (54) holds for any fixed countably infinite subset $\mathcal{H}$ of the layer's domain. This is established for the embedding layer ($f_{\mathbb{N}}^\ell(\mathbf{h}) = \mathbf{e_h}$), the MLP layer ($f_{\mathbb{N}}^\ell(\mathbf{h}) = \mathbf{h} + \mathtt{mlp}(\mathtt{LN}(\mathbf{h}))$), and the attention layer ($f_{\mathbb{N}}^\ell(\mathbf{h}) = \mathbf{h} + \mathtt{attn}(\mathtt{LN}(\mathbf{h}))$) by Lemma 7, Lemma 8, and Lemma 9, respectively. □

The 3 theorems facilitating the proof above are:

Lemma 7. *Lets assume we have an embedding layer randomly independent initialized from a continuous distribution (riicd.) weights and any countably infinite input sets (in embeddings token indexes). We denote the set of random variables over the weights as Θ. We then can show for any fixed countably infinite input set $\mathcal{H}$ that this Layer is injective almost surely.*

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : [\mathbf{e_{h_1}}]_t \neq [\mathbf{e_{h_2}}]_t) = 1 \tag{55}$$

Proof. See App. G.2. □

Lemma 8. *Lets assume we have a sub-block consisting of an MLP with a residual connection and layer norm (i.e., $\mathbf{h} + (\mathtt{mlp}(\mathtt{LN}(\mathbf{h})))$ with riicd. weights. We can show that, for any fixed countably*

---

[14]We note that $p_Z(\forall z \in \mathcal{Z} : z)$, where $\mathcal{Z}$ is a set of events, is the same as $p_Z(\cap_{z \in \mathcal{Z}} \{z\})$ formally.

infinite input set $\mathcal{H}$, this layer is injective almost surely:

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : \tag{56}$$
$$[\mathbf{h}_1 + \mathtt{mlp}(\mathtt{LN}(\mathbf{h}_1))]_t \neq [\mathbf{h}_2 + \mathtt{mlp}(\mathtt{LN}(\mathbf{h}_2))]_t) = 1$$

Proof. See App. G.3. □

Lemma 9. *Lets assume we have a sub-block consisting of a self-attention with a residual connection and layer norm (i.e., $\mathbf{h} + \mathtt{attn}(\mathtt{LN}(\mathbf{h}))$) with riicd. weights. We can show that, for any fixed countably infinite input set $\mathcal{H}$, this layer is injective almost surely:*

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : \tag{57}$$
$$[\mathbf{h}_1 + \mathtt{attn}(\mathtt{LN}(\mathbf{h}_1))]_t \neq [\mathbf{h}_2 + \mathtt{attn}(\mathtt{LN}(\mathbf{h}_2))]_t) = 1$$

Proof. See App. G.4 □

G.1 Fundamental Lemmas

In this section we present some fundamental lemmas used to prove G.2 to G.4.

Lemma 10. *For a layers function $f_{\mathtt{N}}^\ell$ to be injective on its input set $\mathcal{H}$, it has to hold that:*

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H} : \mathbf{h}_1 \neq \mathbf{h}_2 \Rightarrow f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)) = 1 \tag{58}$$

This can equivalently be written as:

$$p_\Theta\left(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 \neq \mathbf{h}_2 : f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)\right) = 1 \tag{59}$$

Proof. We can derive eq. (59) from eq. (58):

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H} : \mathbf{h}_1 \neq \mathbf{h}_2 \Rightarrow f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2))$$

$$= p_\Theta\left(\bigcap_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}} \{\mathbf{h}_1 \neq \mathbf{h}_2 \Rightarrow f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)\}\right) \tag{60a}$$

$$= p_\Theta\left(\bigcap_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}} \{((\mathbf{h}_1 \neq \mathbf{h}_2) \wedge f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)) \vee (\mathbf{h}_1 = \mathbf{h}_2)\}\right) \tag{60b}$$

$$= p_\Theta\left(\bigcap_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 \neq \mathbf{h}_2} \{(\mathbf{h}_1 \neq \mathbf{h}_2) \wedge f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)\} \cap \underbrace{\bigcap_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 = \mathbf{h}_2} \{(\mathbf{h}_1 = \mathbf{h}_2)\}}_{\text{always true}}\right) \tag{60c}$$

$$= p_\Theta\left(\bigcap_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 \neq \mathbf{h}_2} \left\{\underbrace{(\mathbf{h}_1 \neq \mathbf{h}_2)}_{\text{always true}} \wedge f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)\right\}\right) \tag{60d}$$

$$= p_\Theta\left(\bigcap_{\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 \neq \mathbf{h}_2} \{f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)\}\right) \tag{60e}$$

$$= p_\Theta\left(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, \mathbf{h}_1 \neq \mathbf{h}_2 : f_{\mathtt{N}}^\ell(\mathbf{h}_1) \neq f_{\mathtt{N}}^\ell(\mathbf{h}_2)\right) \tag{60f}$$

□

Lemma 11. *If we have a countable set $\mathcal{Z}$ of almost sure events z, we know that their intersection is also almost surely. Formally:*

$$\left(\forall z \in \mathcal{Z} : p(z) = 1\right) \implies \left(p(\forall z \in \mathcal{Z} : z) = 1\right) \tag{61}$$

Proof. First, observe that

$$p\left(\bigcap_{z\in\mathcal{Z}}z\right) = 1 - p\left(\left(\bigcap_{z\in\mathcal{Z}}z\right)^c\right) = 1 - p\left(\bigcup_{z\in\mathcal{Z}}z^c\right) \tag{62}$$

where z^c is the complement of an event z. Since $p(z) = 1$ for all $z \in \mathcal{Z}$, it follows that $p(z^c) = 0$ for all $z \in \mathcal{Z}$ By the countable subadditivity of probability measures:

$$p\left(\bigcup_{z\in\mathcal{Z}}z^c\right) \leq \sum_{z\in\mathcal{Z}}p(z^c) = \sum_{z\in\mathcal{Z}}0 = 0 \tag{63}$$

Therefore,

$$p(\forall z \in \mathcal{Z} : z) = p\left(\bigcap_{z\in\mathcal{Z}}z\right) = 1 - 0 = 1\,\square \tag{64}$$

G.2 Proof of Lemma 7

In this section, we will prove Lemma 7 which states that the embedding layer is almost surely injective on countably infinite inputs.

Lemma 7. *Lets assume we have an embedding layer randomly independent initialized from a continuous distribution (riicd.) weights and any countably infinite input sets (in embeddings token indexes). We denote the set of random variables over the weights as Θ. We then can show for any fixed countably infinite input set $\mathcal{H}$ that this Layer is injective almost surely.*

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : [\mathbf{e_{h_1}}]_t \neq [\mathbf{e_{h_2}}]_t) = 1 \tag{55}$$

Proof. We can apply Lemma 11 three times (on $\mathbf{h}_1, \mathbf{h}_2$ and t) to show that eq. (55) is equivalent to, for any $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$ and $t \in T$ for which $[\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t$, it holding that:

$$p_\Theta([\mathbf{e_{h_1}}]_t \neq [\mathbf{e_{h_2}}]_t) = 1 \tag{65}$$

We further note that, by the definition of an embedding block (Submodule 2):

$$p_\Theta([\mathbf{e_{h_1}}]_t \neq [\mathbf{e_{h_2}}]_t) = 1 \quad \Leftrightarrow \quad p_\Theta(\mathbf{e}_{[\mathbf{h}_1]_t} \neq \mathbf{e}_{[\mathbf{h}_2]_t}) = 1 \tag{66}$$

We can thus apply the law of total probability by defining Θ' as all the random variables Θ except the one for the first element of $\mathbf{e}_{[\mathbf{h}_1]_t}$, i.e., except $[\mathbf{e}_{[\mathbf{h}_1]_t}]_1$,[15] and $\mathbf{e}'_{[\mathbf{h}_1]_t}$ as the embedding of $[\mathbf{h}_1]_t$ without the first element:

$$\int p_{\Theta\setminus\Theta'}(\mathbf{e}_{[\mathbf{h}_1]_t} \neq \mathbf{e}_{[\mathbf{h}_2]_t} \mid \mathbf{e}'_{[\mathbf{h}_1]_t}, \mathbf{e}_{[\mathbf{h}_2]_t})p_{\Theta'}(\mathbf{e}'_{[\mathbf{h}_1]_t} \cup \mathbf{e}_{[\mathbf{h}_2]_t})d(\mathbf{e}'_{[\mathbf{h}_1]_t} \cup \mathbf{e}_{[\mathbf{h}_2]_t}) = 1 \tag{67}$$

It therefore suffices to show that, for any $\mathbf{e}'_{[\mathbf{h}_1]_t}$ and $\mathbf{e}_{[\mathbf{h}_2]_t}$:

$$p_{\Theta\setminus\Theta'}(\mathbf{e}_{[\mathbf{h}_1]_t} \neq \mathbf{e}_{[\mathbf{h}_2]_t} \mid \mathbf{e}'_{[\mathbf{h}_1]_t}, \mathbf{e}_{[\mathbf{h}_2]_t}) = 1 \tag{68}$$

This holds trivially when any embedding dimension other than the first of $\mathbf{e}_{[\mathbf{h}_1]_t}$ and $\mathbf{e}_{[\mathbf{h}_2]_t}$ differs. When all dimensions except the first are equal, we apply:

$$p_{\Theta\setminus\Theta'}([\mathbf{e}_{[\mathbf{h}_1]_t}]_1 \neq [\mathbf{e}_{[\mathbf{h}_2]_t}]_1 \mid \mathbf{e}'_{[\mathbf{h}_1]_t}, \mathbf{e}_{[\mathbf{h}_2]_t}) = 1 \tag{69}$$
$$\Leftrightarrow \quad p_{\Theta\setminus\Theta'}([\mathbf{e}_{[\mathbf{h}_1]_t}]_1 = [\mathbf{e}_{[\mathbf{h}_2]_t}]_1 \mid \mathbf{e}'_{[\mathbf{h}_1]_t}, \mathbf{e}_{[\mathbf{h}_2]_t}\}) = 0$$

The right-hand side $[\mathbf{e}_{[\mathbf{h}_2]_t}]_1$ is a constant while the left-hand side $[\mathbf{e}_{[\mathbf{h}_1]_t}]_1$ is a random variable over a continuous region; this event has measure 0, resulting in probability 0. $\square$

G.3 Proof of Lemma 8

In this section, we will prove Lemma 8, which will show that the block consisting of an MLP, residual connection and layer norm is almost sure injective on its countably infinite inputs.

Lemma 8. *Lets assume we have a sub-block consisting of an MLP with a residual connection and layer norm (i.e., $\mathbf{h} + (\mathtt{mlp}(\mathtt{LN}(\mathbf{h})))$ with riicd. weights. We can show that, for any fixed countably*

---

[15]By this we refer to the first element of the embedding vector of $[\mathbf{h}_1]_t$.

infinite input set $\mathcal{H}$, *this layer is injective almost surely:*

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : \tag{56}$$
$$[\mathbf{h}_1 + \texttt{mlp}(\texttt{LN}(\mathbf{h}_1))]_t \neq [\mathbf{h}_2 + \texttt{mlp}(\texttt{LN}(\mathbf{h}_2))]_t) = 1$$

Proof. For notational convenience, let $\mathbf{m}_i = \texttt{mlp}(\texttt{LN}(\mathbf{h}_i))$. Given Lemma 11, it suffices to prove that for any $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$ and $t \in T$, where $[\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t$, we have:

$$p_\Theta\Big([\mathbf{h}_1 + \mathbf{m}_1]_t \neq [\mathbf{h}_2 + \mathbf{m}_2]_t\Big) = 1 \tag{70}$$

Without loss of generality, fix one such $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$ and $t \in T$. We can manipulate this probability distribution as:

$$p_\Theta\Big([\mathbf{h}_1 + \mathbf{m}_1]_t \neq [\mathbf{h}_2 + \mathbf{m}_2]_t\Big)$$
$$= p_\Theta\Big([\mathbf{h}_1]_t + [\mathbf{m}_1]_t \neq [\mathbf{h}_2]_t + [\mathbf{m}_2]_t\Big)$$
$$= p_\Theta\Big([\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big) p_\Theta\Big([\mathbf{h}_1]_t + [\mathbf{m}_1]_t \neq [\mathbf{h}_2]_t + [\mathbf{m}_2]_t \mid [\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big) \tag{71a}$$
$$+ p_\Theta\Big([\mathbf{m}_1]_t = [\mathbf{m}_2]_t)\Big) \underbrace{p_\Theta\Big([\mathbf{h}_1]_t + [\mathbf{m}_1]_t \neq [\mathbf{h}_2]_t + [\mathbf{m}_2]_t \mid [\mathbf{m}_1]_t = [\mathbf{m}_2]_t\Big)}_{=1,\text{ since } [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t}$$
$$= p_\Theta\Big([\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big) p_\Theta\Big([\mathbf{h}_1]_t + [\mathbf{m}_1]_t \neq [\mathbf{h}_2]_t + [\mathbf{m}_2]_t \mid [\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big) \tag{71b}$$
$$+ p_\Theta\Big([\mathbf{m}_1]_t = [\mathbf{m}_2]_t)\Big)$$

Therefore, it suffices to show that:

$$p_\Theta\Big([\mathbf{h}_1]_t + [\mathbf{m}_1]_t \neq [\mathbf{h}_2]_t + [\mathbf{m}_2]_t \mid [\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big) = 1 \tag{72}$$

since $p_\Theta\Big([\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t)\Big) + p_\Theta\Big([\mathbf{m}_1]_t = [\mathbf{m}_2]_t)\Big)$ is trivially 1. We now unfold the last layer of the MLP as $\mathbf{m}_i = \mathbf{W}_L(\mathbf{m}_i') + \mathbf{b}_L$, where $\mathbf{m}_i' = \sigma(f_{\texttt{N}_{\texttt{MLP}}}^{:L-1}(\texttt{LN}(\mathbf{h}_i)))$. We can rewrite eq. (72) as:

$$p_\Theta\Big([\mathbf{h}_1]_t + [\mathbf{W}_L\mathbf{m}_1' + \mathbf{b}_L]_t \neq [\mathbf{h}_2]_t + [\mathbf{W}_L\mathbf{m}_2' + \mathbf{b}_L]_t \mid [\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big)$$
$$= p_\Theta\Big([\mathbf{h}_1]_t + \mathbf{W}_L[\mathbf{m}_1']_t \neq [\mathbf{h}_2]_t + \mathbf{W}_L[\mathbf{m}_2']_t \mid [\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t\Big) \tag{73a}$$
$$\overset{(1)}{=} p_\Theta\Big([\mathbf{h}_1]_t + \mathbf{W}_L[\mathbf{m}_1']_t \neq [\mathbf{h}_2]_t + \mathbf{W}_L[\mathbf{m}_2']_t \mid [\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}\Big) \tag{73b}$$
$$\overset{(2)}{\geq} p_\Theta\Big([\mathbf{h}_1]_{[t,1]} + [\mathbf{W}_L\mathbf{m}_1']_{[t,1]} \neq [\mathbf{h}_2]_{[t,1]} + [\mathbf{W}_L\mathbf{m}_2']_{[t,1]} \mid [\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}\Big) \tag{73c}$$
$$= p_\Theta\Big([\mathbf{h}_1]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_{[t,j]} \neq [\mathbf{h}_2]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} \mid [\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}\Big)$$
$$\tag{73d}$$
$$\overset{(3)}{=} \int_{\Theta \setminus \Theta'} p_{\Theta'}\Big([\mathbf{h}_1]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_j \neq [\mathbf{h}_2]_{[t,1]} + \tag{73e}$$
$$\sum_{j=1}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} \mid [\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}, \theta\Big) p_{\Theta \setminus \Theta'}(\theta \mid [\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]})d\theta$$

where equality (1) holds since $[\mathbf{m}_1]_t \neq [\mathbf{m}_2]_t$ implies there exists some index i such that $[\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}$.[16] (2) holds because if the inequality is satisfied for a single component of the vector, it must also be satisfied for the entire vector. In (3), we define Θ' as the random variable responsible for the value of $[\mathbf{W}_L]_{[1,i]}$ and θ as a realisation of the random variables $\Theta \setminus \Theta'$. Therefore, to prove

---

[16]$[\mathbf{m}_1']_{[t,i]}$ represents a two dimensional indexing, referring to the i-th element of the representation of the t-th token.

eq. (72), it suffices to show:

$$p_{\Theta'}\left([\mathbf{h}_1]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_j \neq [\mathbf{h}_2]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} \mid [\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}, \theta\right)$$
$$= 1 \qquad (74)$$

For brevity, we omit repeating the conditions in the following probabilities as they remain unchanged to the previous equation:

$$p_{\Theta'}\left([\mathbf{h}_1]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_j \neq [\mathbf{h}_2]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} \mid \ldots\right) = 1 \qquad (75a)$$

$$\Leftrightarrow p_{\Theta'}\left([\mathbf{h}_1]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_j = [\mathbf{h}_2]_{[t,1]} + \sum_{j=1}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} \mid \ldots\right) = 0 \quad (75b)$$

$$\Leftrightarrow p_{\Theta'}\left([\mathbf{W}_L]_{[1,i]}[\mathbf{m}_1']_{[t,i]} - [\mathbf{W}_L]_{[1,i]}[\mathbf{m}_2']_{[t,i]} = [\mathbf{h}_2]_{[t,1]} - [\mathbf{h}_1]_{[t,1]}\right. \qquad (75c)$$
$$\left. + \sum_{j=1,j\neq i}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} - \sum_{j=1,j\neq i}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_{[t,j]} \mid \ldots\right) = 0$$

$$\Leftrightarrow p_{\Theta'}\left([\mathbf{W}_L]_{[1,i]} = \frac{1}{[\mathbf{m}_1']_{[t,i]} - [\mathbf{m}_2']_{[t,i]}}\left([\mathbf{h}_2]_{[t,1]} - [\mathbf{h}_1]_{[t,1]}\right.\right. \qquad (75d)$$
$$\left.\left. + \sum_{j=1,j\neq i}^{|[\mathbf{h}_2]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_2']_{[t,j]} - \sum_{j=1,j\neq i}^{|[\mathbf{h}_1]_t|}[\mathbf{W}_L]_{[1,j]}[\mathbf{m}_1']_{[t,j]}\right) \mid \ldots\right) = 0$$

where the last step follows from the condition $[\mathbf{m}_1']_{[t,i]} \neq [\mathbf{m}_2']_{[t,i]}$ which ensures the denominator is non-zero. Now, Eq. (75d) holds because the right-hand side is a constant (since its elements are fixed given the conditions of the probability) while the left-hand side is a random variable drawn from a continuous distribution (since the weights are riicd.). Therefore, the probability that this equality holds is zero, as the event has measure zero. $\qquad\square$

G.4   Proof of Lemma 9

In this Section, we prove Lemma 9, which establishes that the self-attention sub-block (consisting of attention, residual connection, and layer normalisation) is almost surely injective on countably infinite inputs. The proof structure parallels that of Lemma 8, so we highlight the key differences and necessary adaptations without repeating the full derivation.

Lemma 9. *Lets assume we have a sub-block consisting of a self-attention with a residual connection and layer norm (i.e., $\mathbf{h} + \texttt{attn}(\texttt{LN}(\mathbf{h}))$) with riicd. weights. We can show that, for any fixed countably infinite input set $\mathcal{H}$, this layer is injective almost surely:*

$$p_\Theta(\forall \mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}, t \in T, [\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t : \qquad (57)$$
$$[\mathbf{h}_1 + \texttt{attn}(\texttt{LN}(\mathbf{h}_1))]_t \neq [\mathbf{h}_2 + \texttt{attn}(\texttt{LN}(\mathbf{h}_2))]_t) = 1$$

Proof. We follow a proof strategy analogous to that of Lemma 8 in App. G.3. Following the same steps up to eq. (72), it suffices to show for this lemma that for any $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}$ and $t \in T$, where $[\mathbf{h}_1]_t \neq [\mathbf{h}_2]_t$, we have:

$$p_\Theta\left([\mathbf{h}_1]_t + [\texttt{attn}(\texttt{LN}(\mathbf{h}_1))]_t \neq [\mathbf{h}_2]_t + [\texttt{attn}(\texttt{LN}(\mathbf{h}_2))]_t \mid [\texttt{attn}(\texttt{LN}(\mathbf{h}_1))]_t \neq [\texttt{attn}(\texttt{LN}(\mathbf{h}_2))]_t\right)$$
$$= 1 \qquad (76)$$

We can write this according to the definition of an attention block (Submodule 1):

$$p_\Theta\left([\mathbf{h}_1]_t + (\mathbf{W}^O)^T[\mathbf{h}_1']_t \neq [\mathbf{h}_2]_t + (\mathbf{W}^O)^T[\mathbf{h}_2']_t \mid [\texttt{attn}(\texttt{LN}(\mathbf{h}_1))]_t \neq [\texttt{attn}(\texttt{LN}(\mathbf{h}_2))]_t\right) \quad (77)$$

where $\mathbf{h}_1'$ and $\mathbf{h}_2'$ are the hidden states after concatenation in the self-attention mechanism (see eq. (12)). The remainder of the proof follows the same approach as the proof of Lemma 8 in App. G.3, starting from eq. (73a). $\qquad\square$

	All Pairs	Same Output	Not Same Output	Same Variables	Not Same Variables
Input	$8.5e{-}2_{\pm1.1e-2}$	$8.5e{-}2_{\pm1.1e-2}$	$1.7e{-}1_{\pm1.9e-2}$	$8.5e{-}2_{\pm1.1e-2}$	$1.6e{-}1_{\pm1.9e-2}$
Layer 1	$5.7e{-}4_{\pm4.5e-4}$	$5.7e{-}4_{\pm4.5e-4}$	$2.4e{-}2_{\pm6.2e-3}$	$5.7e{-}4_{\pm4.5e-4}$	$1.1e{-}2_{\pm4.3e-3}$
Layer 2	$4.5e{-}4_{\pm3.8e-4}$	$4.5e{-}4_{\pm3.8e-4}$	$3.2e{-}2_{\pm1.2e-2}$	$4.5e{-}4_{\pm3.8e-4}$	$1.2e{-}2_{\pm6.7e-3}$
Layer 3	$3.3e{-}4_{\pm2.8e-4}$	$3.3e{-}4_{\pm2.8e-4}$	$8.9e{-}2_{\pm4.3e-2}$	$3.3e{-}4_{\pm2.8e-4}$	$1.5e{-}2_{\pm1.0e-2}$

Table 1: An approximation of the minimal Euclidean distance of the trained MLP model in the hierarchical equality task using 1,280,000 samples. The minimal Euclidean distance is computed between all samples to a randomly selected subset of 10,000 samples. We compute it over 10 different random seeds and present the mean and standard deviation (the number after $\pm$).

H MLP Injectivity in Hierarchical Equality Task

We see in Fig. 2 that the IIA remains low for the `identity of first argument` algorithm on a fully trained model even when using a $\phi^{\mathtt{nonlin}}$ alignment map (based on `revnet`). A reasonable assumption for why would be that the fully trained model does not fulfil some assumption required by our proof of Theorem 1 (any algorithm is an input-restricted distributed abstraction for any model) given in §4. In this section, we present follow-up experiments investigating the reason for this disagreement between our empirical results on the `identity of first argument` algorithm and the theoretical result of Theorem 1.

Let us first note that to prove Theorem 1 we rely on an existence proof: showing there exists a function ϕ which satisfies the conditions for a DNN to be abstracted by an algorithm. It says nothing, however, about this function being learnable in practice. Our experiments, however, measure IIA on an unseen test set—which requires $\phi^{\mathtt{nonlin}}$ to not only fit a training set, but generalise to new data. Therefore, following our proof of Theorem 1 we explore the IIA on the train set. However, on the normal training set (with 1,280,000 samples), we still do not get an IIA over 0.55. On the other hand, if we repeat the experiment with only 1,000 training samples, we see $\phi^{\mathtt{nonlin}}$ achieves an IIA of over 0.99 on the training set. Therefore, it is likely that the `revnet` used when defining $\phi^{\mathtt{nonlin}}$ does not have enough capacity to fit the overly complex function our proof describes.

To further analyse why the capacity of the used `revnet` is not sufficient, we analyse the injectivity of the evaluated MLP by investigating its hidden representations. We first evaluate 1,280,000 randomly sampled inputs and their hidden states, checking if they are all unique. In these 1,280,000 samples (and repeating this experiment with 10 different random seeds), no collisions were found, implying the evaluated MLP is (at least close to) injective.

We now examine the supposition that the model finds it more difficult to distinguish between hidden states that share the same values for the variables in `both equality relations` than between those that do not. To this end, we compute the minimal Euclidean distance between hidden states across the entire set of 1,280,000 samples to a randomly selected subset of 10,000 samples. Specifically, we measure the minimal pairwise Euclidean distance among: (i) all sample pairs, (ii) sample pairs sharing the same output, (iii) sample pairs sharing the same values for both equality variables in `both equality relations`, (iv) sample pairs that do not share the same output and (v) sample pairs that do not share the same values for both equality variables. The results are presented in Table 1. We observe that the minimal Euclidean distances are smaller for pairs sharing the same output or the same equality-variable values compared to pairs that do not. This suggests that, although injectivity is preserved, a RevNet likely will find it more challenging to separate hidden states that share variable values.

I Additional Experiment Details

In this section, we present additional details about our hierarchical equality task (in App. I.1) and indirect object identification (in App. I.2) experiments. We also present details and results on the distributive law task (in App. I.3).

I.1 Hierarchical Equality Task

Task 1 (from Geiger et al., 2024b). *The **hierarchical equality task** is defined as follows. Let $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3 \circ \mathbf{x}_4$ be a 16-dimensional vector, where each $\mathbf{x}_i \in \mathbb{R}^4$ for $i \in \{1,2,3,4\}$, and $\circ$ denotes vector concatenation. The input space is $\mathcal{X} = [-0.5, 0.5]^{16}$, and the output space is*

$\mathcal{Y} = \{\texttt{false}, \texttt{true}\}$. *The task function is:*

$$\texttt{T}(\mathbf{x}) = \big((\mathbf{x}_1 == \mathbf{x}_2) == (\mathbf{x}_3 == \mathbf{x}_4)\big), \tag{78}$$

where the equality $(\mathbf{x}_i == \mathbf{x}_j)$ holds if and only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are equal as vectors in $\mathbb{R}^4$.

I.1.1 Algorithms

We define the following three candidate algorithms in detail.

Alg 1. *The* `both equality relations alg.` *to solve Task 1 has* $\boldsymbol{\eta}_{\text{inner}} = \{\eta_{\mathbf{x}_1==\mathbf{x}_2}, \eta_{\mathbf{x}_3==\mathbf{x}_4}\}$ *and:*

$$f_{\mathtt{A}}^{\eta_{\mathbf{x}_1==\mathbf{x}_2}}\big(\mathbf{v}_{\text{par}_{\mathtt{A}}(\eta_{\mathbf{x}_1==\mathbf{x}_2})}\big) = (v_{\eta_{\mathbf{x}_1}} == v_{\eta_{\mathbf{x}_2}})$$

$$f_{\mathtt{A}}^{\eta_{\mathbf{x}_3==\mathbf{x}_4}}\big(\mathbf{v}_{\text{par}_{\mathtt{A}}(\eta_{\mathbf{x}_3==\mathbf{x}_4})}\big) = (v_{\eta_{\mathbf{x}_3}} == v_{\eta_{\mathbf{x}_4}})$$

$$f_{\mathtt{A}}^{\eta_y}\big(\mathbf{v}_{\text{par}_{\mathtt{A}}(\eta_y)}\big) = (v_{\eta_{\mathbf{x}_1==\mathbf{x}_2}} == v_{\eta_{\mathbf{x}_3==\mathbf{x}_4}})$$

Alg 2. *The* `left equality relation alg.` *to solve Task 1 has* $\boldsymbol{\eta}_{\text{inner}} = \{\eta_{\mathbf{x}_1==\mathbf{x}_2}\}$ *and:*

$$f_{\mathtt{A}}^{\eta_{\mathbf{x}_1==\mathbf{x}_2}}\big(\mathbf{v}_{\text{par}_{\mathtt{A}}(\eta_{\mathbf{x}_1==\mathbf{x}_2})}\big) = (v_{\eta_{\mathbf{x}_1}} == v_{\eta_{\mathbf{x}_2}})$$

$$f_{\mathtt{A}}^{\eta_y}\big(\mathbf{v}_{\text{par}_{\mathtt{A}}(\eta_y)}\big) = (v_{\eta_{\mathbf{x}_1==\mathbf{x}_2}} == (v_{\eta_{\mathbf{x}_3}} == v_{\eta_{\mathbf{x}_4}}))$$

Alg 3. *The* `identity of first argument alg.` *to solve Task 1 has* $\boldsymbol{\eta}_{\text{inner}} = \{\eta_{\mathbf{x}_1}\}$ *and:*

$$f_{\mathtt{A}}^{\eta_{\mathbf{x}_1}}\big(\text{par}_{\mathtt{A}}(\eta_{\mathbf{x}_1})\big) = \mathbf{x}_1$$

$$f_{\mathtt{A}}^{\eta_y}\big(\text{par}_{\mathtt{A}}(\eta_y)\big) = ((v_{\eta_{\mathbf{x}_1}} == \mathbf{x}_2) == (\mathbf{x}_3 == \mathbf{x}_4))$$

I.1.2 Training Details

For the hierarchical equality task, we use a 3-layer MLP with $|\boldsymbol{\psi}_1| = |\boldsymbol{\psi}_2| = |\boldsymbol{\psi}_3| = 16$. The model is trained using the Adam optimiser with learning rate 0.001 and cross-entropy loss. We use a batch size of 1024 and train on 1,048,576 samples, with 10,000 samples each for evaluation and testing. Training runs for a maximum of 20 epochs with early stopping after 3 epochs of no improvement.

For the training progression experiments, we use the same configuration but limit training to 2 epochs.

When training the alignment maps ϕ, we use a batch size of 6400 and train for up to 50 epochs with early stopping after 5 epochs of no improvement (using a threshold of 0.001 for the required change, compared to 0 for MLP training). We use the Adam optimiser with learning rate 0.001 and cross-entropy loss. To generate the datasets for DAS, for Alg. 1 we intervene with a probability of 1/3 on $\eta_{\mathbf{x}_1==\mathbf{x}_2}$, 1/3 on $\eta_{\mathbf{x}_3==\mathbf{x}_4}$, and 1/3 on both variables. The samples for the base and source inputs are generated such that $(\mathbf{x}_1 == \mathbf{x}_2)$ and $(\mathbf{x}_3 == \mathbf{x}_4)$ each hold 50% of the time. For Alg. 2 and Alg. 3 we intervene on $\eta_{\mathbf{x}_1==\mathbf{x}_2}$ and $\eta_{\mathbf{x}_1}$ for all samples, respectively. For each algorithm, we sample 1,280,000 interventions for training, 10,000 for evaluation, and 10,000 for testing.

I.1.3 Additional Results

We present results for the three candidate algorithms for the hierarchical equality task, analysing the effect of hidden size d_{rn} and intervention size $|\boldsymbol{\psi}_\eta^\phi|$ across all MLP layers.

For the `both equality relations` algorithm, Fig. 8a, 9a and 9b demonstrate that the hidden size experiment aligns with previously reported trends, while also showing how alignment maps ϕ of increasing complexity perform across training epochs, layers, and intervention sizes.

For the `left equality relation` algorithm, as shown in Fig. 8b, 9c and 9d, we observe similar patterns: increasing hidden size and intervention size improves performance, and alignment is generally more successful in later layers during early training.

For the `identity of first argument` algorithm, Fig. 8c, 9e and 9f reveal that, interestingly, some alignment is achieved—especially in layer 3—during the first half of training, but this effect diminishes in the second half.

Overall, these results demonstrate that the hidden size experiment is consistent with the findings reported in the main paper. They also show that it is easier, in untrained models, to find an alignment map for later layers, and that transient alignment can occur in specific layers and algorithms during the initial stages of training.
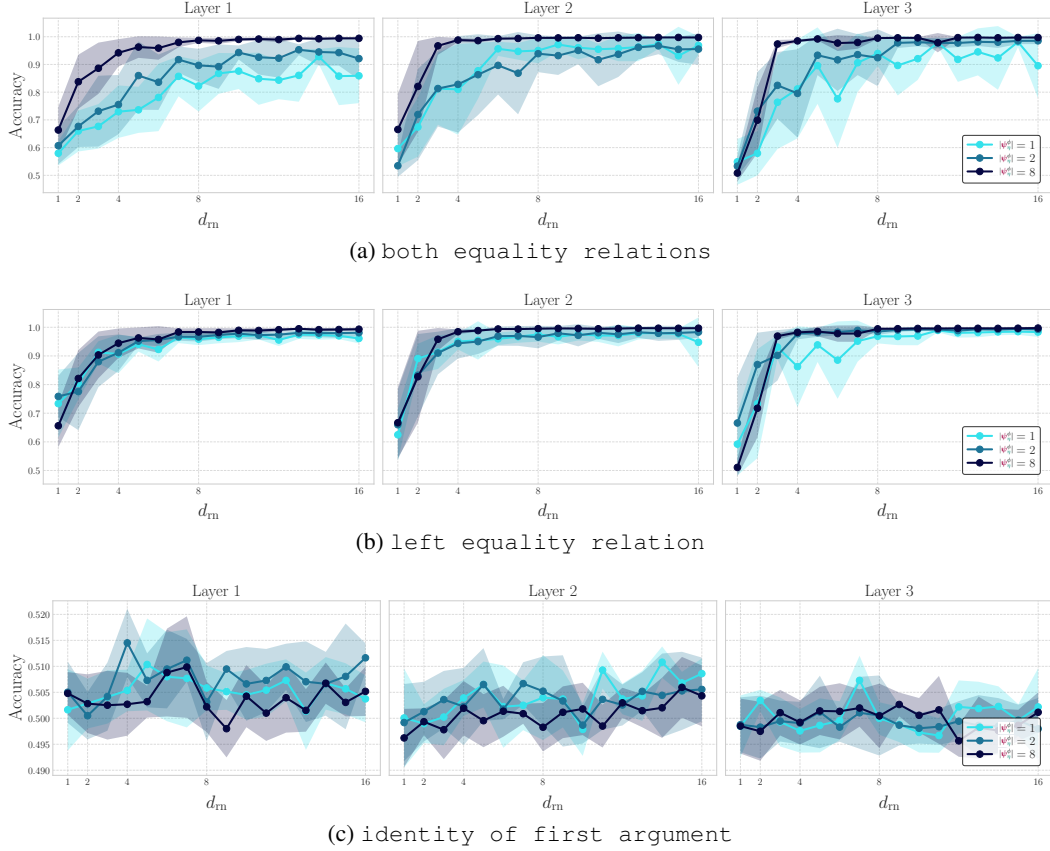
(a) `both equality relations`

(b) `left equality relation`

(c) `identity of first argument`

Figure 8: Mean IIA over 5 seeds using $\phi^{\texttt{nonlin}}$ ($L_{\text{rn}} = 1$) on the trained DNN. Performance improves with larger hidden dimension d_{rn} and intervention size $|\boldsymbol{\psi}_\eta^\phi|$. Each subplot corresponds to one of the three candidate algorithms for the hierarchical equality task, showing how the model's representational capacity influences performance.

I.2 Indirect Object Identification Task

Task 2. *The **Indirect Object Identification** (IOI) task involves predicting the indirect object in sentences with a specific structure. Each input $\mathbf{x} \in \mathcal{X}$ consists of a text where a subject (*S*) and an indirect object (*IO*) are introduced, followed by the* S *giving something to the* IO*. For example:*

"Friends Juana and Kristi found a mango at the bar. Kristi gave it to" $\Rightarrow$ *"Juana"*

*Here, "Juana" and "Kristi" are introduced, with "Kristi" (*S*) appearing again before giving something to "Juana" (*IO*). The output set $\mathcal{Y}$ consists of the first tokens of the two names:*

$$\mathcal{Y} = \{\texttt{first\_token(S)}, \texttt{first\_token(IO)}\} \tag{79}$$

I.2.1 Algorithm

For this task, we evaluate the **ABAB-ABBA** algorithm. Denoting the two names in the story as A and B, this algorithm determines whether the sentence follows an ABAB pattern (e.g., *"Friends Juana and Kristi found a mango at the bar. Juana gave it to Kristi"*) or an ABBA pattern (e.g., *"Friends Juana and Kristi found a mango at the bar. Kristi gave it to Juana"*). If the pattern is ABAB (where B is the indirect object IO), the algorithm predicts the first token of B. Conversely, for an ABBA pattern, it predicts the first token of A. In our experiments, we intervene on whether an input follows the ABAB pattern or not.

(a) Max IIA for `both equality relations` (b) Mean IIA for `both equality relations`

(c) Max IIA for `left equality relation` (d) Mean IIA for `left equality relation`

(e) Max IIA for `identity of first argument` (f) Mean IIA for `identity of first argument`
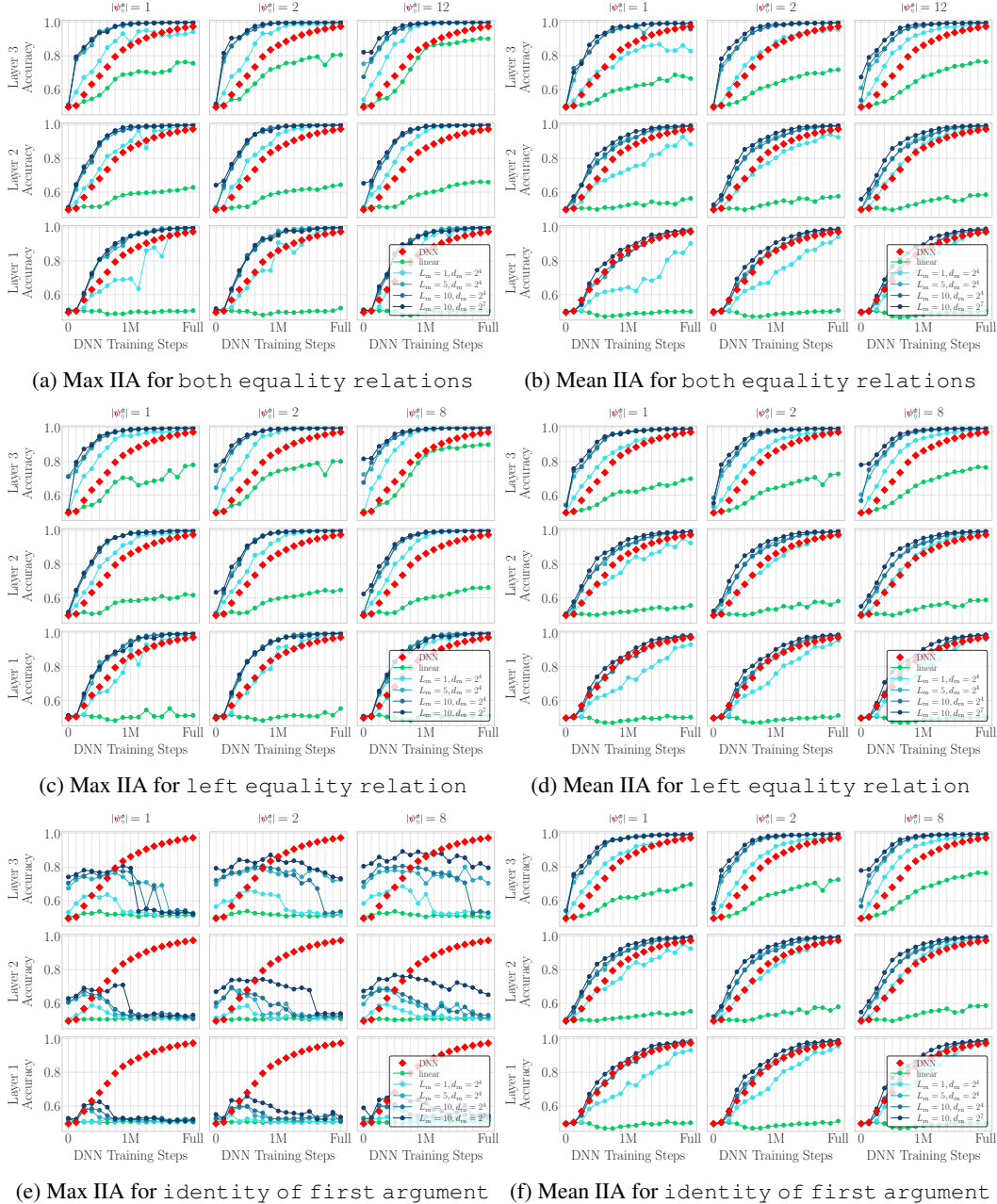
Figure 9: IIA over 5 seeds for each combination of MLP layer (rows) and intervention size (columns) during training progression for the tested algorithms.

Alg 4. *The* `ABAB-ABBA` *algorithm for the IOI task has one inner node* $\boldsymbol{\eta}_{\texttt{inner}} = \{\eta_1\}$ *and is defined as follows:*

$$f_{\texttt{A}}^{\eta_1}(\texttt{par}_{\texttt{A}}(\eta_1)) = \texttt{check\_is\_abab\_pattern}(v_{\boldsymbol{\eta_x}}) \tag{80a}$$

$$f_{\texttt{A}}^{\eta_y}(\texttt{par}_{\texttt{A}}(\eta_y)) = \begin{cases} \texttt{first\_token}(\texttt{get\_name\_b}(v_{\boldsymbol{\eta_x}})) & \text{if } v_{\eta_1} = \texttt{true} \\ \texttt{first\_token}(\texttt{get\_name\_a}(v_{\boldsymbol{\eta_x}})) & \text{if } v_{\eta_1} = \texttt{false} \end{cases} \tag{80b}$$

Here, `get_name_a(x)` *extracts the first name (denoted A, e.g.* *Juana* *in our example) and* `get_name_b(x)` *extracts the second name (denoted B, e.g.* *Kristi*) *from the input sentence* **x**. *The function* `check_is_abab_pattern(x)` *returns* `true` *if the sentence follows an "ABAB" structure (e.g., "A and B ... A gave to B", meaning B is the* IO*) and* `false` *if it follows*

37

an "ABBA" structure (e.g., "A and B ... B gave to A", meaning A is the IO). first_token(*Name*)
returns the first token of the specified name. The output **y** *is the first token of the indirect object.*

I.2.2 Training Details

We use models from the Pythia suite (Biderman et al., 2023) to evaluate the IIA performance of the different ϕ on the IOI task. Specifically, we employ the Pythia 31M, 70M, 160M, and 410M parameter models. We also examine different training checkpoints provided by these models to analyse how IIA evolves during training. To assess robustness, we replicate a subset of experiments using alternative Pythia model seeds from (van der Wal et al., 2025).

Figure 10: Cross Entropy loss (y-axis) during training (x-axis) of the alignment map for the randomly initialised *Pythia-31m*. The loss plateaus between 4k and 6k steps, and suddenly drops after 6k steps.

We train all alignment maps on 2 epochs of 10^6 interventions based on data from Muhia (2022), with a batch size of 256 and a learning rate of 10^{-4}. For all experiments, we set the intervention size $|\psi_\eta^\phi|$ to half of the DNN's hidden dimension. For smaller models (31M, 70M), we train using float64 precision and a learning rate of 10^{-3}, as these adjustments proved crucial for convergence. We also note that we observed quite severe grokking behaviour, where models had low IIA for a long time, which quickly jumped to high IIA values at a certain point of training (see Fig. 10; wandb run).

I.2.3 Additional Results

Robustness across random seeds. In Fig. 11, we examine how our main results from §6 generalise across multiple training seeds of the Pythia model. The key trends hold consistently across all 5 seeds - we can find perfect alignments using $\phi^{\texttt{nonlin}}$ in most cases. However, we observe two notable exceptions. For one seed, the DNN fails to learn the IOI task even after full training. For another seed, we cannot find an alignment using even complex alignment maps under our current setup.[17] All other seeds achieve perfect alignment under $\phi^{\texttt{nonlin}}$. We hypothesise that the alignment failure case is primarily due to suboptimal training of the alignment map. Due to computational constraints, we did not perform extensive hyperparameter tuning that might have achieved convergence.
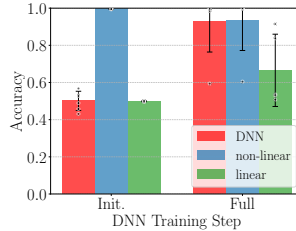

Figure 11: IIA of alignment between the ABAB-ABBA algorithm and the *Pythia-410m* model across multiple seeds (seeds 1 to 5 from van der Wal et al., 2025), with interventions at layer 12. We evaluate the IIA of both $\phi^{\texttt{lin}}$ and $\phi^{\texttt{nonlin}}$ (with $d_{\texttt{h}} = 64, K = 1$) on randomly initialised (Init.) and fully trained (Full) DNNs.

Generalisation across distinct name sets. In the main paper, we split the dataset from Muhia (2022) by ensuring that no two sentences appear in both the training and evaluation sets. However, this splitting strategy does not guarantee that the names themselves are distinct between training and evaluation sets. In Fig. 12, we examine the results when using completely different sets of names for training and evaluation. The results differ substantially: we cannot find an alignment using even complex alignment maps for the randomly initialised DNN. This suggests that IIA on the randomly initialised DNN may depend critically on overlap between the specific entities encountered during

---

[17]These failures occur in different seeds: seed 3 shows poor IIA despite learning the task, while seed 4 fails to learn the IOI task.

training and evaluation. For the fully trained DNN, we observe perfect alignment using $\phi^{\texttt{nonlin}}$ and reasonably high alignment using $\phi^{\texttt{lin}}$.
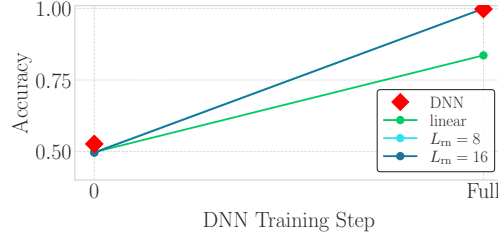

Figure 12: IIA of alignment between the `ABAB-ABBA` algorithm and the *Pythia-410m* model using a different set of names for training and evaluation, with interventions at layer 12. We evaluate the IIA of both $\phi^{\texttt{lin}}$ and $\phi^{\texttt{nonlin}}$ (with $d_{\mathrm{h}} = 64, K = 1$) on randomly initialised and fully trained (Full) DNNs.

I.3   Distributive Law Task

We now study a similar task to the hierarchical equality (in App. I.1), based on the distributive law of and ($\wedge$) and or ($\vee$).

Task 3. *The **distributive law task** is defined as follows. Let $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3 \circ \mathbf{x}_4 \circ \mathbf{x}_5 \circ \mathbf{x}_6$ be a 24-dimensional vector, where each $\mathbf{x}_i \in [-0.5, 0.5]^4$ for $i \in \{1, 2, 3, 4, 5, 6\}$, and $\circ$ denotes vector concatenation. The input space is $\mathcal{X} = [-0.5, 0.5]^{24}$, and the output space is $\mathcal{Y} = \{\texttt{false}, \texttt{true}\}$. The task function is*

$$\mathtt{T}(\mathbf{x}) = \big((\mathbf{x}_1 == \mathbf{x}_2) \wedge (\mathbf{x}_3 == \mathbf{x}_4)\big) \vee \big((\mathbf{x}_3 == \mathbf{x}_4) \wedge (\mathbf{x}_5 == \mathbf{x}_6)\big), \qquad (81)$$

where the equality $(\mathbf{x}_i == \mathbf{x}_j)$ holds if and only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are equal as vectors in $\mathbb{R}^4$.

I.3.1   Algorithms

We define the following two candidate algorithms.

Alg 5. *The `And-Or-And alg.` to solve Task 3 has*

$$\boldsymbol{\eta}_{\texttt{inner}} = \big\{ \eta_{(\mathbf{x}_1 == \mathbf{x}_2) \wedge (\mathbf{x}_3 == \mathbf{x}_4)}, \eta_{(\mathbf{x}_3 == \mathbf{x}_4) \wedge (\mathbf{x}_5 == \mathbf{x}_6)} \big\}$$

and it is defined as follows:

$$f_{\mathtt{A}}^{\eta_{(\mathbf{x}_1 == \mathbf{x}_2) \wedge (\mathbf{x}_3 == \mathbf{x}_4)}} \big(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta_{(\mathbf{x}_1 == \mathbf{x}_2) \wedge (\mathbf{x}_3 == \mathbf{x}_4)})} \big) = (v_{\eta_{\mathbf{x}_1}} == v_{\eta_{\mathbf{x}_2}}) \wedge (v_{\eta_{\mathbf{x}_3}} == v_{\eta_{\mathbf{x}_4}})$$

$$f_{\mathtt{A}}^{\eta_{(\mathbf{x}_3 == \mathbf{x}_4) \wedge (\mathbf{x}_5 == \mathbf{x}_6)}} \big(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta_{(\mathbf{x}_3 == \mathbf{x}_4) \wedge (\mathbf{x}_5 == \mathbf{x}_6)})} \big) = (v_{\eta_{\mathbf{x}_3}} == v_{\eta_{\mathbf{x}_4}}) \wedge (v_{\eta_{\mathbf{x}_5}} == v_{\eta_{\mathbf{x}_6}})$$

$$f_{\mathtt{A}}^{\eta_y} \big(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta_y)} \big) = v_{\eta_{(\mathbf{x}_1 == \mathbf{x}_2) \wedge (\mathbf{x}_3 == \mathbf{x}_4)}} \vee v_{\eta_{(\mathbf{x}_3 == \mathbf{x}_4) \wedge (\mathbf{x}_5 == \mathbf{x}_6)}}$$

Alg 6. *The `And-Or alg.` to solve Task 3 has*

$$\boldsymbol{\eta}_{\texttt{inner}} = \big\{ \eta_{\mathbf{x}_3 == \mathbf{x}_4}, \eta_{(\mathbf{x}_1 == \mathbf{x}_2) \vee (\mathbf{x}_5 == \mathbf{x}_6)} \big\}$$

and it is defined as follows:

$$f_{\mathtt{A}}^{\eta_{\mathbf{x}_3 == \mathbf{x}_4}} \big(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta_{\mathbf{x}_3 == \mathbf{x}_4})} \big) = (v_{\eta_{\mathbf{x}_3}} == v_{\eta_{\mathbf{x}_4}})$$

$$f_{\mathtt{A}}^{\eta_{(\mathbf{x}_1 == \mathbf{x}_2) \vee (\mathbf{x}_5 == \mathbf{x}_6)}} \big(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta_{(\mathbf{x}_1 == \mathbf{x}_2) \vee (\mathbf{x}_5 == \mathbf{x}_6)})} \big) = (v_{\eta_{\mathbf{x}_1}} == v_{\eta_{\mathbf{x}_2}}) \vee (v_{\eta_{\mathbf{x}_5}} == v_{\eta_{\mathbf{x}_6}})$$

$$f_{\mathtt{A}}^{\eta_y} \big(\mathbf{v}_{\mathtt{par}_{\mathtt{A}}(\eta_y)} \big) = v_{\eta_{\mathbf{x}_3 == \mathbf{x}_4}} \wedge v_{\eta_{(\mathbf{x}_1 == \mathbf{x}_2) \vee (\mathbf{x}_5 == \mathbf{x}_6)}}$$

I.3.2   Training Details

For the distributive law task, we use a 3-layer MLP (see App. E.1) with an input dimensionality of 24, hidden layers of dimensionality $|\boldsymbol{\psi}_1| = |\boldsymbol{\psi}_2| = |\boldsymbol{\psi}_3| = 24$, and an output dimensionality of 2. The model is trained using the Adam optimiser with a learning rate of 0.001 and cross-entropy loss. We use a batch size of 1024. The datasets are generated by randomly sampling input vectors $\mathbf{x} = \mathbf{x}_1 \circ \cdots \circ \mathbf{x}_6$ such that the target label $\bar{y} = \mathtt{T}(\mathbf{x})$ is true 50% of the time. We sample 1,048,576 samples for training, 10,000 for evaluation, and 10,000 for testing. Training runs for a maximum of 20 epochs with early stopping after 3 epochs of no improvement.

For training ϕ, we use a batch size of 6400 and train for up to 50 epochs with early stopping after 5 epochs of no improvement (using a threshold of 0.001 for the required change). We use the Adam

optimiser with learning rate 0.001 and cross-entropy loss. To generate the intervened datasets: For Alg. 5, we intervene with a probability of 1/3 on $\eta_{(\mathbf{x}_1==\mathbf{x}_2)\wedge(\mathbf{x}_3==\mathbf{x}_4)}$, 1/3 on $\eta_{(\mathbf{x}_3==\mathbf{x}_4)\wedge(\mathbf{x}_5==\mathbf{x}_6)}$, and 1/3 on both variables. For Alg. 6, we intervene with a probability of 1/3 on $\eta_{\mathbf{x}_3==\mathbf{x}_4}$, 1/3 on $\eta_{(\mathbf{x}_1==\mathbf{x}_2)\vee(\mathbf{x}_5==\mathbf{x}_6)}$, and 1/3 on both variables. For both algorithms, the samples for the base and source inputs are generated such that the output of the intervention changes compared to the base input 50% of the time. We sample 1,280,000 interventions for training, 10,000 for evaluation , and 10,000 for testing for each algorithm.

I.3.3 Results

In this section, we discuss the results on the distributed law task using the And-or-And and And-Or algorithms. Our findings corroborate the results presented in the main paper. As shown in Fig. 13, using linear and identity alignment maps reveals distinct dynamics. The `And-Or` algorithm achieves higher IIA using ϕ^{lin}, particularly in later layers where the IIA of ϕ^{lin} on the `And-Or-And` algorithm approaches 0.5. However, these dynamics completely vanish when using a more complex alignment map like ϕ^{nonlin}, where we achieve almost perfect IIA everywhere.
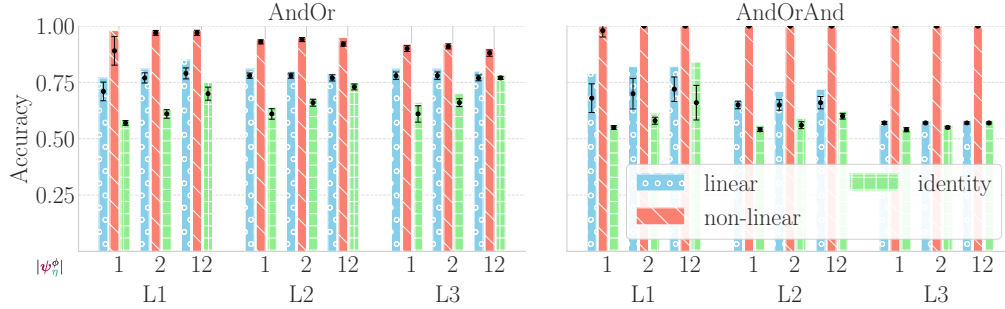
Figure 13: IIA in the distributive law task for causal abstractions trained with different alignment maps ϕ. The figure shows results for both analysed algorithms for this task. The bars represent the max IIA across 10 runs with different random seeds. The black lines represent mean IIA with 95% confidence intervals. The $|\psi_\eta^\phi|$ denotes the intervention size per node. Without interventions, all DNNs reach 100% accuracy. The used ϕ^{nonlin} uses $L_{\text{rn}} = 10$ and $d_{\text{rn}} = 24$.

Fig. 14a and 14c present the evaluated IIA throughout model training. These training progression plots show that randomly initialised models often achieve IIA above 0.8 with non-linear alignment maps, supporting our insight that when the notion of causal abstraction is equipped with ϕ^{nonlin} it may identify algorithms which are not necessarily implemented by the underlying model. In Fig. 14b and 14d, we plot the mean IIA over 5 seeds instead of the maximum IIA.

The hidden size experiments (Fig. 15a and 15b) show that even RevNets with small d_{h} of 4 achieve near-perfect IIA for And-Or-And, while And-Or never reaches perfect IIA in the second layer, regardless of the d_{h}. The training progression plots suggest a possible explanation: IIA for And-Or-And initially increases in the last two layers but then decreases, while RevNets maintain near-perfect IIA. This may indicate that And-Or-And is first implemented with simple encodings detectable by linear ϕs, before evolving into non-linear encodings that only RevNets can detect. The fact that And-Or never achieves high IIA in later layers further suggests it may not be a true abstraction of the model's behaviour, though we note this remains a hypothesis requiring further investigation.

And-Or-And Training. In this section, we analyse a DNN when this model is trained specifically to rely on the `And-Or-And` algorithm (and, consequently, to encode the values of its hidden nodes). We do so with the method from Geiger et al. (2022), training the DNN to encode `And-Or-And`'s hidden nodes' values in its second layer, with an intervention size of 12. This method is similar to how we train ϕ (see App. I.3.2), but ϕ is fixed to the identity function, and the DNN itself is trained; further, the training dataset is composed of 1/4 non-intervened samples, 1/4 samples with interventions on $\eta_{(\mathbf{x}_1==\mathbf{x}_2)\wedge(\mathbf{x}_3==\mathbf{x}_4)}$, 1/4 on $\eta_{(\mathbf{x}_3==\mathbf{x}_4)\wedge(\mathbf{x}_5==\mathbf{x}_6)}$, and 1/4 on both variables. We then evaluate if this DNN abstracts both the `And-Or-And` and `And-Or` using different ϕ (as before, after freezing the DNN). The IIA performance of these ϕ is presented in Fig. 16. We can see here that, when using identity and linear alignment maps ϕ, IIA scores suggest that the `And-Or-And` algorithm seems to be implemented perfectly given the second layer, where we have only around 0.75 IIA for

(a) Max IIA for And-Or

(b) Max IIA for And-Or

(c) Max IIA for And-Or-And
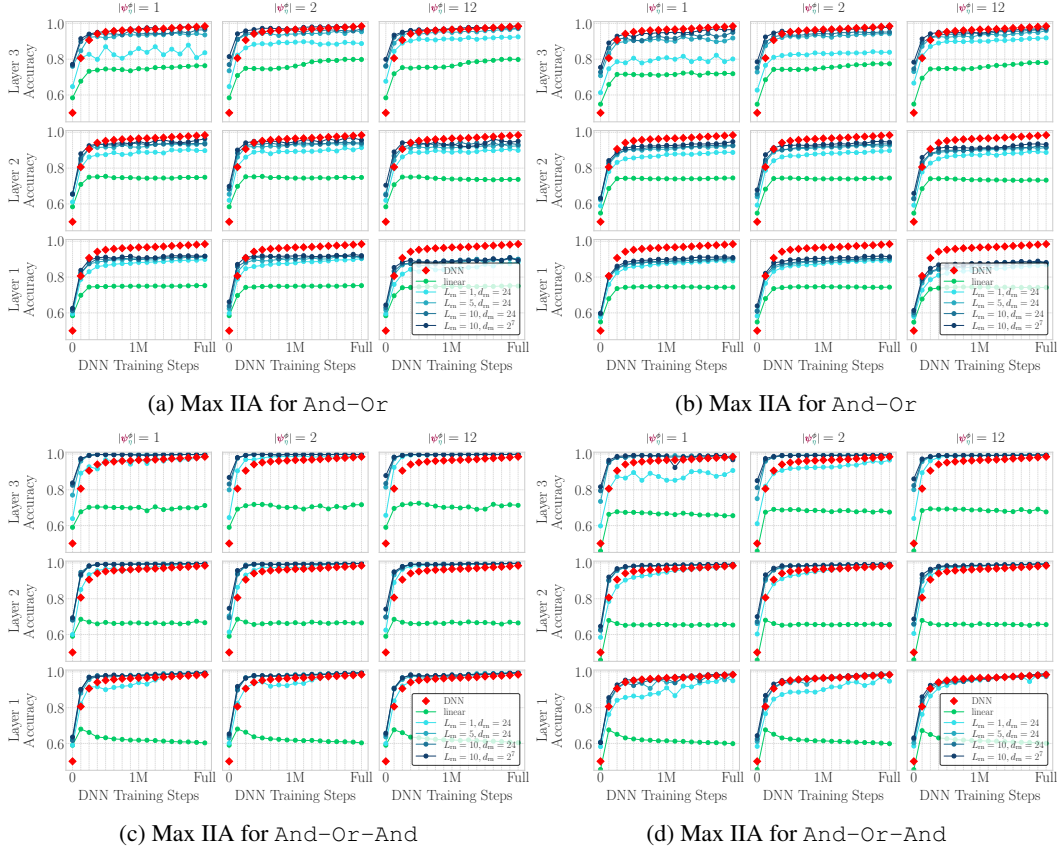
(d) Max IIA for And-Or-And

Figure 14: IIA over 5 seeds for each combination of MLP layer (rows) and intervention size (columns) during training progression for the evaluated algorithms.

the And-Or algorithm. However, these differences vanish almost completely using $\phi^{\mathtt{nonlin}}$ as our alignment map.

And-Or Training. In this section, we report an experiment similar to the above, but we train our DNN to rely on the And-Or algorithm instead. These results are shown in Fig. 17. In this figure, we again see that, when using identity and linear as alignment map ϕ, IIA performance suggests that the And-Or algorithm seems to be implemented perfectly given the second layer, where we have only around 0.65 IIA for the And-Or-And algorithm. These differences however vanish when using $\phi^{\mathtt{nonlin}}$ as alignment map, which leads to perfect IIA scores with either algorithm.

J Computational Resources

The experiments on MLP were executed on CPU (10 computers with i7-4770 or newer) over 3 weeks, as we noticed that DAS on small MLPs are faster on CPU than on GPU. The experiments on the Pythia models were executed on a single A100 GPU with 80GB of memory using approximately 30 GPU hours, including the hyperparameter tuning.

(a) `And-Or`

(b) `And-Or-And`
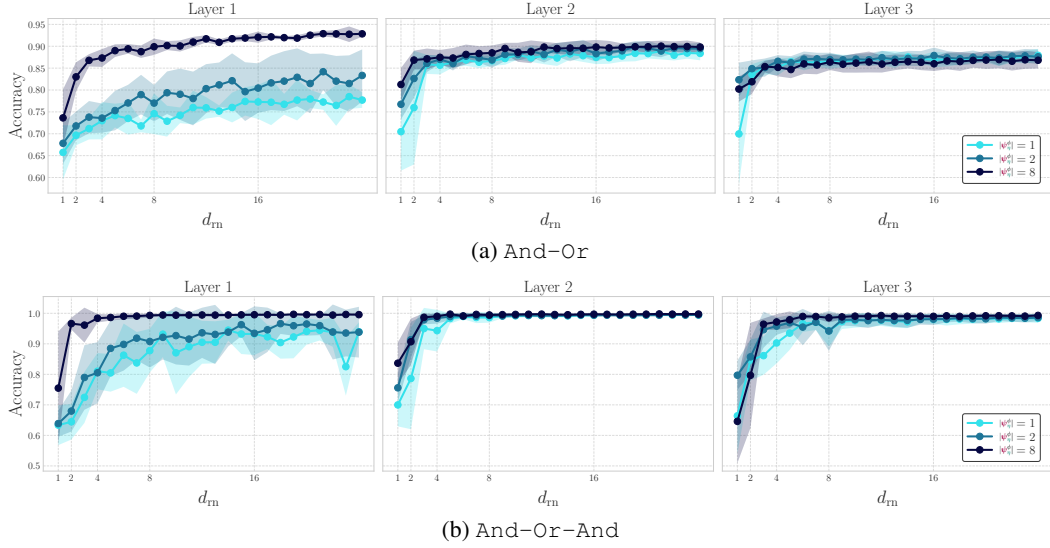
Figure 15: Mean IIA over 5 seeds using $\phi^{\mathtt{nonlin}}$ ($L_{\mathrm{rn}} = 1$) on the trained DNN. Performance improves with larger hidden dimension d_{rn} and intervention size $|\boldsymbol{\psi}_\eta^\phi|$. Each subplot corresponds to one of the two candidate algorithms for the distributed law task, showing how ϕ's representational capacity influences performance.
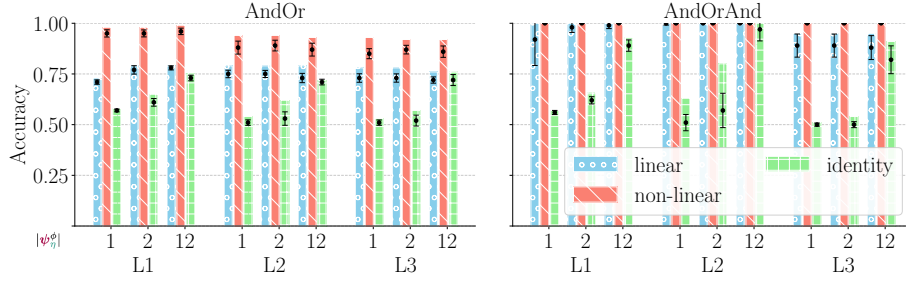

Figure 16: IIA in the distributive law task for causal abstractions trained with different alignment maps ϕ and a DNN trained to use the `And-Or-And` algorithm. The figure shows results when evaluating if the DNN encodes either of the analysed algorithms for this task. The bars represent the max IIA across 10 runs with different random seeds. The black lines represent mean IIA with 95% confidence intervals. The $|\boldsymbol{\psi}_\eta^\phi|$ denotes the intervention size per node. All DNNs reach >99.9% accuracy after training. The used $\phi^{\mathtt{nonlin}}$ uses $L_{\mathrm{rn}} = 10$ and $d_{\mathrm{rn}} = 16$.
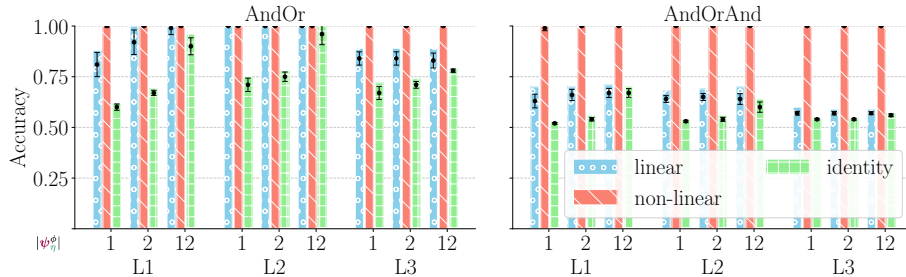
Figure 17: IIA in the distributive law task for causal abstractions trained with different alignment maps ϕ and a DNN trained to use the `And-Or` algorithm. The figure shows IIA results when evaluating if the DNN encodes either of the analysed algorithms for this task. The bars represent the max IIA across 10 runs with different random seeds. The black lines represent mean IIA with 95% confidence intervals. The $|\boldsymbol{\psi}_\eta^\phi|$ denotes the intervention size per node. Without interventions, all DNNs reach >99.9% accuracy. The used $\phi^{\mathtt{nonlin}}$ uses $L_{\mathrm{rn}} = 10$ and $d_{\mathrm{rn}} = 16$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The theoretical and empirical results mentioned in the abstract are given in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a limitations section and clearly label our theorem's assumptions, discussing when they might not hold in practice.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Our proofs are based on assumptions given in the main paper and detailed proofs and assumptions for our theorems are given in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We discuss the experimental settings and data used in the main paper as well as providing detailed information in the appendix. We additionally link to the code in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We link to the code used to generate our results in the paper with a description of how to reproduce it.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setting is discussed in the main paper and details are provided in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For some of our results we provide, besides the mean, 95% confidence intervals. For our experiments on Pythia, due to computational requirements we only ran one seed per configuration; the consistent trends in these results across model sizes and training steps, however, still enable us to have confidence in their robustness.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

    Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

    Answer: [Yes]

    Justification: We mention the used computation in the appendix.

    Guidelines:

    - The answer NA means that the paper does not include experiments.
    - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
    - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
    - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

    Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

    Answer: [Yes]

    Justification: Our work does not provide a new method, but rather analyses an already published research methodology. Therefore, we believe it adds no additional ethical concerns to the current research field.

    Guidelines:

    - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
    - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
    - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: Our method analyses an already published research methodology and we believe it adds no additional societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not publish any new method, model or dataset.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The code is implemented by us (except standard libraries) and the only used external dataset is referenced.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We link to the created code in the paper.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No human subjects were involved in our research.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human subjects were involved in our research.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs were only used for writing, editing, or formatting purposes and did not impact the core methodology.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.