Control and Predictivity in Neural Interpretability

Satchel Grant
Departments of Psychology and Computer Science
Stanford University
Stanford, CA 94305
grantsrb@stanford.edu

Alexa Tartaglini
Department of Computer Science
Stanford University
Stanford, CA 94305
alexart@stanford.edu

Abstract

For the goals of mechanistic interpretability, correlational methods are typically easy to scale and use, and can provide strong predictivity of Neural Network (NN) representations. However, they can lack causal fidelity which can limit their relevance to NN computation and behavior. Alternatively, causal approaches can offer strong behavioral control via targeted interventions, making them superior for understanding computational cause and effect. However, what if causal methods use out-of-distribution representations to produce their effects? Does this raise concerns about the faithfulness of the claims that can be made about the NN's native computations? In this work, we explore this possibility of this representational divergence. We ask to what degree do causally intervened representations diverge from the native distribution, and in what situations is this divergence acceptable? Using Distributed Alignment Search (DAS) as a case study, we first demonstrate the existence of causally intervened representational divergence in interventions that provide strong behavioral control, and we show that stronger behavioral control can correlate with more divergent intervened representations. We then provide a theoretical discussion showing sufficient ways for this divergence to occur in both innocuous and potentially pernicious ways. We then provide a theoretical demonstration that causal interventions typically assume principles of additivity, calling into question the use of nonlinear methods for causal manipulations. Lastly, for cases in which representational divergence is undesirable, we demonstrate how to incorporate a counterfactual latent loss to constrain intervened representations to remain closer to the native distribution. Together, we use our results to suggest that although causal methods are superior for most interpretability goals, a complete account of NN representations balances computational control with neural predictivity, with the optimal weighting depending on the goals of the research.

1 Introduction

In the many recent mechanistic interpretability developments, researchers have used a variety of methods to defend claims about how Neural Networks (NNs) perform their computations. These methods can be broadly categorized into two groups based upon their objectives. The first group analyzes NN activations to find associations between native, naturally occurring neural activity and attributes of NN inputs and outputs. The second group focuses on causally manipulating neural activity in an effort to produce an effect on the NN's computations. We will broadly refer to the former as predictive approaches and the latter as causal approaches.

Popular examples of the predictive variety include methods such as Principal Component Analysis (PCA), linear prediction, and some variants of Sparse Auto-Encoders (SAEs) [31, 6, 1, 5, 9, 20]. These methods often attempt to decompose NN latent activity into a linear sum of features (i.e. vector

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

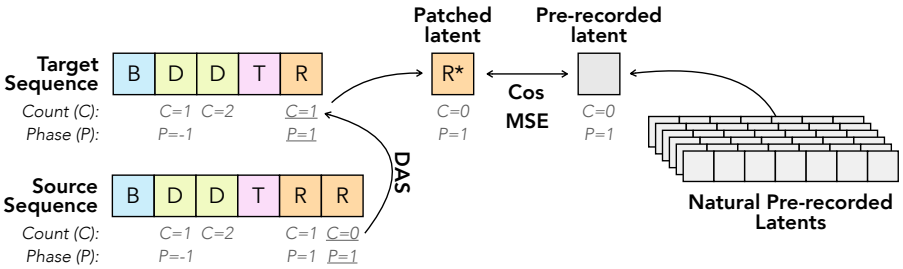

Figure 1: Figure showing counterfactual latents in counting task. The squares represent latent vectors produced from the recurrent state and the displayed input tokens. The C indicates the value of the Count of the causal abstraction, the P indicates the phase. The counterfactual latent (CL) vectors are the naturally occurring latent vectors that possess the same causal values as the post-intervention latent vector. These CL vectors can be recorded from the forward pass on curated input data.

directions) whose variability corresponds to some (hopefully) interpretable attribute that can be used to reconstruct the distribution of native latent activity. These approaches can lead to decompositions that closely match and predict the native latent distribution and have been shown to provide insight into the underlying mechanisms and neural structures of various types of NNs [5, 9]. However, they often lack causal fidelity [8], and they can exhibit a number of undesirable traits such as failing to identify atomic units [22], producing brittle concept representations [23], and under performing on causal mediation tasks and out-of-distribution probing [2, 19]. Lampinen et al. [21] have even questioned the foundations of predictive representational analyses due to disconnects between the strength of learned features and their importance for computation [20].

Alternatively, popular examples of the causal variety include activation patching and Distributed Alignment Search (DAS) [12, 14, 38, 37, 27, 28, 4]. These methods use causal relevance (i.e. NN outputs or behavior) as their main measure of success, where the goal is to controllably alter NN outputs by manipulating internal representations such that cause-and-effect claims can be made about internal mechanisms. Thus, these methods are, by definition, superior for determining cause-and-effect relationships in neural circuitry, which is perhaps of utmost importance for making mechanistic claims [30, 11, 10, 21].

What if these causal approaches, however, rely on intervened representations that strongly diverge from the distribution of native neural activity? Would that change what the methods tell us about the NN's natural mechanistic components? Some activation patching examples use features with values that are multiplied by 10-15x [24], raising questions of what these experiments say about the NN's native mechanisms. In what situations and to what degree is it okay for causal methods to deviate from the native distribution? In cases that these deviations occur, are there ways that we can mitigate the deviation? In one sense, any perspective that is critical of analyzing representations alone is suggesting that representational divergence is okay for understanding NN mechanisms, but how much deviation and in what circumstances is this deviation tolerable?

In this work, we provide theoretical and empirical insight on these issues. We first empirically explore representational divergence in a case study using DAS on Gated Recurrent Units (GRUs) [3]. We explore how choices of causal abstractions and alignment functions (AFs) affect behavioral control and faithfulness to native neural activity. We then theoretically show sufficient ways in which representational divergence can occur, and we provide a discussion on how these deviations can be okay for many mechanistic claims. Next, we show how causal interventions rely on principles of additivity, raising concerns of what nonlinear interpretability methods show us. Lastly, for cases in which researchers do care about representational divergence, we provide a demonstration on how to use a counterfactual auxiliary loss on intervened latent vectors to mitigate representational divergence between intervened and native latent vectors.

We summarize our contributions as follows:

1. We empirically explore how causally manipulated representations can control behavior, but can deviate from native neural activity. Furthermore, we show that greater behavioral accuracy can correlate with greater representational divergence.

2. We provide a theoretical discussion on two sufficient ways for representational divergence to occur, showing that it can arise from dormant-null subspace interactions introduced in [25] and from innocuous co-variation of causal subspaces. We use this result to suggest that representational divergence is expected and okay for most mechanistic claims.
3. We provide a theoretical demonstration that causal interventions rely on principles of additivity, which obfuscates claims of causal interpretability made with non-linear methods.
4. Lastly, we show how to use a counterfactual latent loss [17] to mitigate the divergence between the intervened and native neural distributions, reducing the gap between behavioral control and neural predictivity in causal methods.

We use our findings to inform a discussion on what our goals are as interpretability researchers. We suggest that a complete understanding of NN representations includes both control and predictivity of NN activity, while the notion of success with these criteria depends on the purpose of the research.

2 Background and Related Work

The majority of our empirical analyses use a specific form of activation patching [13, 36, 37, 27] known as Distributed Alignment Search (DAS) [12, 14, 38]. We provide a background on DAS here.

2.1 DAS Formulation

DAS is a framework for causally testing the degree of alignment between an NN's latent vectors and variables from causal abstractions (CAs) (e.g. computer programs or directed acyclic graphs). DAS does this by testing the hypothesis that a latent state vector, $h \in R^{d_m}$, within an NN can be transformed into a vector $z \in R^{d_m}$ that consists of orthogonal subspaces encoding interpretable variables from CAs. The transformation is performed by a learnable, invertible *Alignment Function* (AF), $\mathcal{A}$, as follows: $z = \mathcal{A}(h)$ [18]. The benefit of this transformation is that it allows us to formulate the NN's neural activity in terms of interpretable variables, and it allows us to manipulate the value of each variable without affecting the values of the others. We will refer to the space of h as the *native vector space* and that of z as the *aligned vector space*.

Concretely, for a given CA with variables $\text{var}_i \in \{\text{var}_1, \text{var}_2, ..., \text{var}_n\}$, DAS tests the hypothesis that z is composed of subspaces $\vec{z}_{\text{var}_i} \in R^{d_{\text{var}_i}}$ corresponding to each of the variables from the CA. We include a causally irrelevant subspace, $\vec{z}_{\text{extra}} \in R^{d_{\text{extra}}}$, to encode extraneous, functionally irrelevant activity.

$$\mathcal{A}(h) = z = \begin{bmatrix} \vec{z}_{\text{var}_1} \\ \vec{z}_{\text{var}_2} \\ \cdots \\ \vec{z}_{\text{var}_n} \\ \vec{z}_{\text{extra}} \end{bmatrix} \tag{1}$$

Each $\vec{z}_{\text{var}_i} \in R^{d_{\text{var}_i}}$ is a column vector of potentially different lengths. We refer to d_{var_i} as the *subspace size* of var_i, and all the subspace sizes together satisfy the relation $d_{\text{extra}} + \sum_{i=1}^{n} d_{\text{var}_i} = d_m$. Under this assumption, the value of a single causal variable encoded in h can be freely exchanged by performing an interchange intervention defined as follows:

$$h^v = \mathcal{A}^{-1}((1 - D_{\text{var}_i})\mathcal{A}(h^{trg}) + D_{\text{var}_i}\mathcal{A}(h^{src})) \tag{2}$$

Where $D_{\text{var}} \in R^{d_m \times d_m}$ is a manually defined, block diagonal, binary matrix that defines the subspace size d_{var_i}. Each D_{var_i} has a set of d_{var_i} contiguous ones along its diagonal to isolate the dimensions that make up $\vec{z}_{\text{var}_i}$. h^{src} is the *source vector* from which the subspace activity is harvested, h^{trg} is the *target vector* into which the harvested activity is substituted/patched, and h^v is the resulting intervened vector that we use to replace h^{trg} in the model's processing. This allows the model to make predictions using a different value of variable var_i assuming a successful intervention.

To train the AF, DAS uses *counterfactual behavior* from the pre-defined CA to create intervention data that can be used as training labels for the model's processing conditioned on the intervened latent representation after an intervention. Counterfactual behavior for a given state of a CA and its context is the behavior that would have occurred had a causal variable taken a different value

and everything else remained the same. We can simulate counterfactual behavior by freezing the state of the environment, changing one or more values in the CA, and using the CA to generate new behavior in the same environment using the new variable values. We can then use the counterfactual behavior as training labels to train the AF after each intervention while keeping the model parameters frozen. We train the AF to convergence and then use fresh intervention data to evaluate the robustness of the AF and to make claims about the NN's internal mechanisms. The model's accuracy on the counterfactual behavior following each intervention is referred to as the Interchange Intervention Accuracy (IIA).

In our experiments, we train the same AF on interventions for all causal subspaces including the extraneous subspace. We consider a trial correct when the model correctly predicts all deterministic tokens using the argmax over logits. We report the proportion of trials correct as the IIA. See Appendix A.2 for further detail.

DAS Alignment Functions: We consider three types of AFs in this work.

1. *Orthogonal Alignment Functions (OAFs)*: $\mathcal{A}(h) = Qh$ and $\mathcal{A}^{-1}(h) = Q^{-1}h$ where $Q \in R^{d_m \times d_m}$ is an orthogonal matrix.

2. *Linear Alignment Functions (LAFs)*: $\mathcal{A}(h) = W(h+b)$ and $\mathcal{A}^{-1}(z) = W^{-1}z - b$ where W is a symmetric invertible matrix [18]. See Appendix A.3 for details on how W is constructed.

3. *Reverse Resnet Alignment Functions (RRAFs)*: $\mathcal{A}(h) = RevRes(h)$ and $\mathcal{A}^{-1}(z) = RevRes^{-1}(z)$ where $RevRes$ is a reversible residual network [15, 34]. We use 3 layers with no changes in dimensionality.

3 Methods

The majority of this work consists of DAS analyses performed on Gated Recurrent Unit recurrent neural networks (GRUs) autoregressively trained on sequence-based tasks. We use DAS to align these NNs to Causal Abstractions (CAs) by performing interchange interventions on the GRUs' representations. We consider a single numeric equivalence task which has been used in prior work on human cognition [16, 7] and alignment functions [18, 17].

3.1 Numeric Equivalence Task:

This task consists of a sequence of tokens produced by an environment. Each sequence starts with a beginning of sequence token, B, and ends with an end of sequence token, E. After the B token, the environment presents some number of demonstration (demo) tokens that are each sampled with replacement from the set {D_a, D_b, D_c}. The task is to produce the same number of response (R) tokens as D tokens, and end with the E token. The environment signals the end of the D tokens by producing a trigger (T) token. The number of D tokens at this point is referred to as the *object quantity* for the trial, which is uniformly sampled from 1 to 20 at the beginning. The set of possible tokens includes {B, D_a, D_b, D_c, T, R, E}. An example sequence with an object quantity of 2 is: "B D_c D_a T R R E" Each trial is considered correct when all deterministic tokens are correctly predicted. During the model training, we include all token types in a NTP cross entropy loss, even though the D and T tokens are unpredictable.

3.2 Model Architectures

In this work we consider Gated Recurrent Unit (GRU) Recurrent Neural Networks (RNNs) [3] that are autoregressively trained to perform the Numeric Equivalence task. We train 3 model seeds for each task variant up to $> 99.99\%$ accuracy on both training and validation data and freeze the weights before analysis and interpretation. The GRUs have a dimensionality of 128. We perform all DAS analyses on the output of the GRU recurrent cell, denoted h. We leave further details of the GRU recurrent cell to Appendix A.1 and the referenced paper.

3.3 Causal Abstractions (CAs)

In this work, we evaluate 2 different CAs using DAS. We briefly describe them here and offer Algorithms 1 and 2 in the appendix in addition to intervention data samples in Appendix A.3.1. Also, refer to Figure 4 for a visual depiction of the CAs.

Up-Down Abstraction: uses a single numeric variable, called the **Count**, to track the difference between the number of demo tokens and resp tokens at each step in the sequence. It also contains a **Phase** variable to determine whether it is in the demo phase—counting up—or response phase—counting down. The program ends when the Count is equal to 0 during the response phase.

Increment-Up Abstraction: this program uses progress along an interval from 0 to 1 to track quantities. To do this, it first increments a Progress variable by the value of an Increment variable. The value of the Increment is initially set to $\frac{1}{\text{max count}}$ (in our case $\frac{1}{20}$). The value of the Progress variable is then incremented with each new demo token to track the object count (number of demo tokens) during the demo phase. Upon encountering the trigger token, a new value of the Increment is calculated as the inverse of the Progress divided by the max count: $Increment = \frac{1}{Progress} \div max\,count$. The Progress is then reset to 0. The new value of Increment is now equal to $\frac{1}{\text{obj count}}$ and is used as a step size to increment the Progress variable with each new response token. The program finishes when the Progress variable is greater than 1.

3.4 Counterfactual Latent Auxiliary Loss

To encourage intervened representations to be more similar to the native distribution of NN representations, we re-purpose the counterfactual latent auxiliary loss from [17]. This auxiliary objective relies on *Counterfactual Latent (CL) vectors* as vector objectives. CL vectors are defined as vectors that encode the causal variable values that we would expect to exist in the intervened vector, h^v. We can obtain CL vectors by searching through a pre-recorded set of h vectors for situations and behaviors that are consistent with the values of the CA to which we are aligning. See Figure 1 for a visualization.

As an example, assume we have a CA with variables $\text{v}ar_y$, $\text{v}ar_w$, and $\text{v}ar_{extra}$, and following a causal intervention we expect h^v to have a value of y for variable $\text{v}ar_y$ and w for variable $\text{v}ar_w$. For this example, the CL vector can be obtained from a pre-recorded representation, h_{CL}, that has the same expected variable values: $\text{v}ar_y = y$ and $\text{v}ar_w = w$, as the intervened vector. The auxiliary loss $\mathcal{X}^{(k)}$ for a single intervention sample is composed of an L2 and a cosine distance using CL vectors as the labels:

$$\mathcal{X}_{L2} = \frac{1}{2}||h^v - h_{CL}||_2^2 \tag{3}$$

$$\mathcal{X}_{cos} = -\frac{1}{2}\frac{h^v \cdot h_{CL}}{||h^v||_2\,||h_{CL}||_2} \tag{4}$$

where h^v is the intervened vector. We combine the CL auxiliary loss with the DAS autoregressive loss into a single loss term using a weighted sum where ϵ is a hyperparameter: $\mathcal{L}_{total} = \epsilon(\mathcal{X}_{L2} + \mathcal{X}_{cos}) + \mathcal{L}_{DAS}$

4 Results

4.1 Intervened representations have varying degrees of divergence from the native distribution

We can see in Figure 2 scatter plots of NN latent vectors projected into the top two principal components. The red points come from naturally occurring latent states. The blue come from intervened latent states that have a new value of the Count subspace. The top row shows projections from latents in the model's native neural space; the intervened latents were transformed back from the aligned space, and the native were left unchanged. The bottom row shows PCA projections of the same latent vectors in the aligned space using only the Count and Phase subspaces—the extraneous subspace was set to zero. All of these PCA projections together are a qualitative demonstration of the possible divergence between the intervened and native representational distributions. It is important

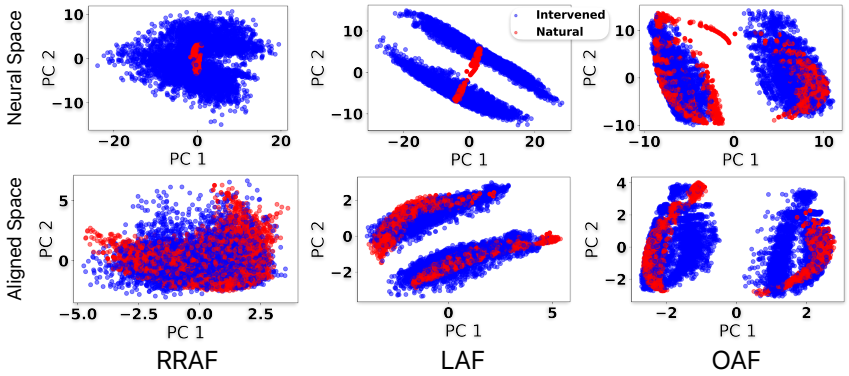

Figure 2: Principal component projections of the native latent states (in red) and intervened latent states (blue) for the Up-Down abstraction using the Count variable on a single model seed. The top row of panels show PCA performed on latent vectors in the native latent space. The bottom panels show PCA performed on vectors in the aligned space in which the extraneous subspace, $\vec{z}_{\text{extra}}$, is set to 0. These alignments do not use the counterfactual latent loss, which is equivalent to a CL Epsilon of 0 in Figure 3.

to note that these AFs each exhibit a relatively high IIA as shown in Figure 3. Furthermore, we can see that the divergence tends to increase for less restrictive AFs, and, from Supplemental Figure 4, we can see that CAs can differ in both IIA and aligned representational divergence.

4.2 How can causal interventions lead to representational divergence?

In this section, we enumerate distinct cases in which systematic intervened representational divergence can occur. The first draws on the work of [25] who demonstrated that it is possible for an interaction between dormant and null subspaces to occur that can create interventions that produce the correct counterfactual behavior without using native causal subspaces, where dormant subspaces are those that do not vary between inputs, null subspaces are those that exist in the null space of the NN's next layer, and causal subspaces are those that both vary between inputs and causally affect behavior. These cases of null-dormant subspace interactions can lead to representations that deviate from the native distribution by definition, due to the fact that the value along the dormant direction is different than the native dormant value.

Another way in which representational divergence can occur that ignores null and dormant subspaces is in cases where segregated causal subspaces have covariance within the bounds of their behavioral decision boundaries. We use a behaviorally binary subspace as a concrete example, where we define a behaviorally binary subspace as one in which the behavior of the NN depends on the sign of the subspace and is invariant to magnitude and angle.

Suppose we have an NN with two causal subspaces, $\tilde{\mathbf{z}}_{\text{var}_\mathbf{a}}$ and $\tilde{\mathbf{z}}_{\text{var}_\mathbf{b}}$ with values $\vec{z}^{(x_i)}_{\text{var}_a}$ and $\vec{z}^{(x_i)}_{\text{var}_b}$ for a model input x_i, where we use the bold notation to distinguish variables from their (non-bold) values. Furthermore, assume that $\tilde{\mathbf{z}}_{\text{var}_\mathbf{b}}$ is a behaviorally binary subspace that co-varies with $\tilde{\mathbf{z}}_{\text{var}_\mathbf{a}}$. Using $h^{(x_i)}$ and $z^{(x_i)}$ from Equation 1 under a given input x_i, we use the following definition:

$$\mathcal{A}(h^{(x_i)}) = z^{(x_i)} = \begin{bmatrix} \tilde{\mathbf{z}}^{(\mathbf{x_i})}_{\text{var}_\mathbf{a}} = \vec{z}^{(x_i)}_{\text{var}_a} \\ \tilde{\mathbf{z}}^{(\mathbf{x_i})}_{\text{var}_\mathbf{b}} = \vec{z}^{(x_i)}_{\text{var}_b} \end{bmatrix} \tag{5}$$

Due to the assumption of covariance in $\tilde{\mathbf{z}}_{\text{var}_\mathbf{a}}$ and $\tilde{\mathbf{z}}_{\text{var}_\mathbf{b}}$, it is reasonable to assume that the values $\vec{z}^{(x_{low})}_{\text{var}_b}$ and $\vec{z}^{(x_{high})}_{\text{var}_b}$ are systematically distinct for distinct values of $\tilde{\mathbf{z}}_{\text{var}_\mathbf{a}}$ under some pedagogically contrived classes of inputs x_{low} and x_{high}, while $sign(\vec{z}^{(x_{low})}_{\text{var}_b}) = sign(\vec{z}^{(x_{high})}_{\text{var}_b})$. Under these assumptions, if we perform an interchange intervention on $\tilde{\mathbf{z}}_{\text{var}_\mathbf{b}}$ using source representations from input x_{low} and target representations from input x_{high}, the intervened representation will have values:

$$z^v = \begin{bmatrix} \tilde{\mathbf{z}}_{\text{var}_\mathbf{a}} = \vec{z}^{(x_{high})}_{\text{var}_a} \\ \tilde{\mathbf{z}}_{\text{var}_\mathbf{b}} = \vec{z}^{(x_{low})}_{\text{var}_b} \end{bmatrix} \tag{6}$$

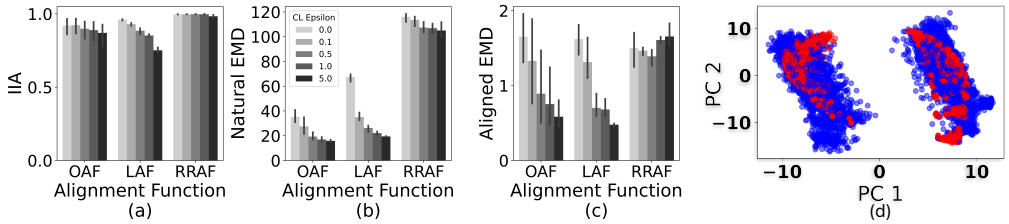

Figure 3: All panels show results for DAS trained on the Up-Down abstraction in which the reported IIA is for the Count variable for 3 model seeds. **(a)** The final validation Interchange Intervention Accuracy (IIA) for different weights (values of epsilon) of the Counterfactual Latent (CL) loss for the Orthogonal (OAF), Linear (LAF), and Reverse Resnet (RRAF) Alignment Functions. **(b)** The Earth Mover's Distance between native latent vectors and intervened latent vectors in the NN's neural space in which only the Count subspace has been manipulated. **(c)** The Earth Mover's Distance between native latent vectors and intervened latent vectors in the aligned space with the extraneous subspaces set to zero. **(d)** PCA projections of the intervened and native latent states in the native space for an LAF using a CL epsilon of 5. This panel is the same as the top panel for the LAF from Figure 2 except that the CL epsilon is non-zero. Blue shows projections from intervened latents whereas red shows native latents.

Because we assumed that the value of $\vec{z}^{(x_{low})}_{\text{var}_b}$ is systematically unique due to covariance in $\tilde{\mathbf{z}}_{\text{var}_\mathbf{a}}$ and $\tilde{\mathbf{z}}_{\text{var}_\mathbf{b}}$, then the values $\vec{z}^{(x_{high})}_{\text{var}_a}$ and $\vec{z}^{(x_{low})}_{\text{var}_b}$ in Equation 6 will have never existed together in the native distribution, but the behavior of the NN will remain unchanged because $\tilde{\mathbf{z}}_{\text{var}_\mathbf{b}}$ is behaviorally binary and its sign has not changed.

We note that this divergent effect stemming from covariance could occur in the null-space as well as causal subspaces. We argue that due to the causal irrelevance of such divergence, these deviations are innocuous to causal mechanistic claims. We provide the lower half of Figure 2 to visualize how the intervened values of z can diverge in practice even in the absence of the extraneous/null subspace. These panels show the top two PCs of aligned z vectors in which the extraneous subspace has been projected out of both the intervened and native latent vectors.

If we focus only on the divergence in causal subspaces and ignore cases of null and dormant interactions, what claims can we make about the model's native neural mechanisms using divergent, intervened representations? An interpretation consistent with the principles of superposition [33, 32, 26, 6] is that differences in the exact values of each causal subspace do not matter, *only the decision boundaries* along these subspaces matter. Under this interpretation, any divergence in the intervened distribution arising from covariance in causal subspaces is okay, because the separation of the causal subspaces and their decision boundaries—the functionally important aspects of the NN—are respected in the alignment. This interpretation allows us to ignore/abstract away functionally irrelevant covariance between the causal subspaces in our attempt to understand the NN mechanisms, allowing us to focus entirely on the functional/behavioral computations. This perspective emphasizes behavioral control over neural predictivity and is the perspective that we subscribe to.

4.3 Non-linear alignment functions violate assumptions of additivity

In this section we explore how the DAS method for computing interchange interventions (Equation 2) assumes that the AF exhibits principles of additivity, corroborating Sutter et al.'s [34] concerns about the validity of non-linear AFs, as additivity is only guaranteed in linear AFs.

Let $\mathcal{A} : \mathbb{R}^{d_m} \rightarrow \mathbb{R}^{d_a}$ be an AF that maps a model's native latent representation $h \in \mathbb{R}^{d_m}$ to a shared aligned space $z = \mathcal{A}(h)$. We assume that $\mathcal{A}$ is invertible and denote its inverse as $\mathcal{A}^{-1}$. In practice, $\mathcal{A}$ is a learned transformation.

Interchange interventions are motivated by the desire to substitute values of causal subspaces between latent vectors, where the resulting intervened vector h^v can be defined equivalently to Equation 2 as follows:

$$h^v = \mathcal{A}^{-1}\left(\sum_{j \neq i} \hat{z}^{\text{trg}}_{\text{var}_j} + \hat{z}^{\text{src}}_{\text{var}_i}\right) \tag{7}$$

Each $\hat{z}_{\text{var}_j} \in \mathbb{R}^{d_m}$ corresponds to a masked subspace of the aligned representation defined as:

$$\hat{z}_{\text{var}_j} = D_{\text{var}_j} z \tag{8}$$

and $D_{\text{var}_j} \in \mathbb{R}^{d_m \times d_m}$ is a block diagonal binary matrix isolating the subspace dimensions associated with variable var_j. The D matrices form a partition of the aligned space: $\sum_{j=1}^{n} D_{\text{var}_j} = I$ and $D_{\text{var}_i} D_{\text{var}_j} = \mathbf{0}$ when $i \neq j$.

The standard implementation of interchange interventions assumes that one can perform the substitution at the level of aligned representations, and then apply the inverse transform to return the representation to the native space (defined in Equation 2, or equivalently in Equation 7). However, this procedure implicitly assumes that the inverse transform $\mathcal{A}^{-1}$ is *additive*, i.e., for any $x, y \in \mathbb{R}^{d_a}$:

$$\mathcal{A}^{-1}(x + y) = \mathcal{A}^{-1}(x) + \mathcal{A}^{-1}(y) \tag{9}$$

Without this property, the inverse of a sum is not equal to the sum of the inverses, which breaks the modular interpretation of subspace substitutions.

To illustrate this, consider the decomposition of a target latent vector:

$$z^{\text{trg}} = \mathcal{A}(h^{\text{trg}}) = \sum_{j=1}^{n} \hat{z}^{\text{trg}}_{\text{var}_j} \tag{10}$$

$$h^{\text{trg}} = \mathcal{A}^{-1}\left(\sum_{j=1}^{n} \hat{z}^{\text{trg}}_{\text{var}_j}\right) \tag{11}$$

If $\mathcal{A}^{-1}$ is additive, this becomes:

$$h^{\text{trg}} = \sum_{j=1}^{n} \mathcal{A}^{-1}(\hat{z}^{\text{trg}}_{\text{var}_j}) \tag{12}$$

Thus, we may cleanly isolate and manipulate individual subspace contributions in the aligned space before inverting.

The same logic applies to an intervention where the i-th subspace is replaced from a source latent vector:

$$h^v = \mathcal{A}^{-1}\left(\sum_{j \neq i} \hat{z}^{\text{trg}}_{\text{var}_j} + \hat{z}^{\text{src}}_{\text{var}_i}\right) \tag{13}$$

$$= \sum_{j \neq i} \mathcal{A}^{-1}(\hat{z}^{\text{trg}}_{\text{var}_j}) + \mathcal{A}^{-1}(\hat{z}^{\text{src}}_{\text{var}_i}) \quad \text{(if } \mathcal{A}^{-1} \text{ is additive)} \tag{14}$$

The standard interchange intervention approach to computing and isolating $\sum_{j \neq i} \hat{z}^{\text{trg}}_{\text{var}_j}$ and $\hat{z}^{\text{src}}_{\text{var}_i}$ first independently computes each $\hat{z}$ from the respective h^{trg} and h^{src} before adding them together. Thus, the standard method for computing h^v relies on the principle of additivity.

Proposition. Suppose $\mathcal{A}^{-1}$ is not additive. Then there exist vectors $x, y \in \mathbb{R}^{d_a}$ such that:

$$\mathcal{A}^{-1}(x + y) \neq \mathcal{A}^{-1}(x) + \mathcal{A}^{-1}(y). \tag{15}$$

Consequently, there exist source and target latent representations for which:

$$h^v = \mathcal{A}^{-1}\left(\sum_{j \neq i} \hat{z}^{\text{trg}}_{\text{var}_j} + \hat{z}^{\text{src}}_{\text{var}_i}\right)$$

does not equal:

$$\sum_{j \neq i} \mathcal{A}^{-1}(\hat{z}^{\text{trg}}_{\text{var}_j}) + \mathcal{A}^{-1}(\hat{z}^{\text{src}}_{\text{var}_i}).$$

This means that the inverse transform introduces non-linear interactions between subspaces, undermining any clean attribution of h^v to its constituent parts.

Implication. The correctness and interpretability of standard interchange interventions depend on the additivity of $\mathcal{A}^{-1}$. In the absence of this property, it becomes unclear whether the reconstructed vector h^v reflects an interpretable combination of the intended latent subspaces. Instead, $\mathcal{A}^{-1}$ may behave like an arbitrary function $\mathcal{F}$ with no guaranteed semantic alignment to $\mathcal{A}$.

4.4 Counterfactual Latent Vectors alleviate post-intervention divergence

Although we have shown in Section 4.2 why some types of representational divergence are acceptable for many mechanistic claims, there are still some cases in which it may be desirable for intervened representations to be predictive of the native distribution. For these cases, we explore the use of a CL auxiliary loss, which is one that minimizes the L2 and cosine distances between the intervened and native representations. We can see in Figure 3 that we can successfully reduce the Earth Mover's distance between the intervened and native distributions by applying the CL loss during the DAS training. This is a step towards making DAS more relevant for goals of neural predictivity.

5 Limitations/Future Directions

The results presented in this work has been confined to synthetic GRUs and simplistic tasks. A more complete demonstration of the intervened distribution shift would include evaluations performed on larger, more practically oriented transformers and Large Language Models (LLMs). This is particularly noteworthy for this work as the GRU architecture imposes a Tanh nonlinearity on the representations analyzed in this work, whereas Transformers potentially have more linear representations due to the nature of their residual stream [6]. We look forward to exploring LLMs in future work.

6 Discussion/Conclusion

In this work we examined autoregressive GRUs trained on numeric tasks to demonstrate the following: we showed the degree to which intervened representations can diverge from naturally occurring; we showed how intervened representations can occur in ways that are mechanistically innocuous to many claims; we showed how AFs that don't exhibit principles of additivity violate implicit assumptions in interchange interventions; and we showed how to use a CL auxiliary loss to reduce the divergence between native and intervened representations. Where does this leave us with respect to neural interpretability?

We return to our goals underlying the notion of "understanding" neural activity. In general, it is reasonable to equate the notion of "understanding neural activity" to one's ability to predict and/or control the activity in ways that are deemed interpretable. Causal methods such as DAS manage to control NN behavior quite well as exemplified by the strong IIA in this work, and these methods do so through *interpretable* causal abstractions. However, causal methods may be limited in their ability to predict native neural activity. Different research objectives will place different weights on the importance of neural predictivity and behavioral control. In some cases, for example, we may wish to classify native neural activity, in which case, predictivity is potentially useful. In other cases, we may wish to exert influence over the computations of the NN for the purpose of AI safety or to characterize the space of *potential* computations as a means of predicting NN generalization.

In light of classic causal mediation philosophy corroborated by more recent findings of complications in non-causal representational analyses, we find ourselves favoring causal methods for their ability to ignore computationally irrelevant NN details as shown in this work, and their ability to unify diverse neural systems [17, 20, 21] while providing useful ways of understanding neural mechanisms [11, 10]. We hesitate to diminish goals of neural predictivity, however, and we remind ourselves that there are no guarantees that we find satisfying, interpretable ways of understanding the complete complexities of NNs.

Acknowledgments and Disclosure of Funding

Thank you to the PDP lab for computational resources, Jay McClelland for thought provoking discussions in previous work, Jerome Han for providing paper feedback, and Chris Potts for providing paper feedback.

References

[1] Trenton Bricken, Neel Nanda, Nicholas Joseph, Arthur Conmy, Andy Jones, Anna Chen, Neal DasSarma, Nelson Elhage, Ben Mann, Catherine Olsson, Kamal Ndousse, Sam Ringer, Alex Tran-Johnson, Yuntao Bai, Liane Lovitt, Zac Hatfield-Dodds, Amanda Askell, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, and Sam McCandlish. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

[2] Maheep Chaudhary and Atticus Geiger. Evaluating open-source sparse autoencoders on disentangling factual knowledge in gpt-2 small. *arXiv*, 2024. arXiv preprint.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[4] Róbert Csordás, Christopher Potts, Christopher D. Manning, and Atticus Geiger. Recurrent neural networks learn to store and generate sequences using non-linear representations, 2024.

[5] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[6] 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. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

[7] Michael C. Frank, Daniel L. Everett, Evelina Fedorenko, and Edward Gibson. Number as a cognitive technology: Evidence from pirahã language and cognition. *Cognition*, 108(3):819–824, 2008.

[8] Dan Friedman, Andrew Lampinen, Lucas Dixon, Danqi Chen, and Asma Ghandeharioun. Interpretability illusions in the generalization of simplified models. *arXiv preprint arXiv:2312.03656*, 2023.

[9] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[10] Atticus Geiger, Jacqueline Harding, and Thomas Icard. How causal abstraction underpins computational explanation. *arXiv preprint arXiv:2508.11214*, 2025.

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

[12] Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *CoRR*, abs/2106.02997, 2021.

[13] Atticus Geiger, Kyle Richardson, and Christopher Potts. Neural natural language inference models partially embed theories of lexical entailment and negation. *arXiv preprint arXiv:2004.14623*, 2020.

[14] Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations, 2023.

[15] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

[16] Peter Gordon. Numerical cognition without words: Evidence from Amazonia. *Science*, 306(5695):496–499, 2004.

[17] Satchel Grant. Model alignment search. *arXiv preprint arXiv:2501.06164*, 2025.

[18] Satchel Grant, Noah D. Goodman, and James L. McClelland. Emergent symbol-like number variables in artificial neural networks. *Transactions on Machine Learning Research*, 2024.

[19] Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing. *arXiv preprint arXiv:2502.16681*, 2025.

[20] Andrew Kyle Lampinen, Stephanie CY Chan, and Katherine Hermann. Learned feature representations are biased by complexity, learning order, position, and more. *arXiv preprint arXiv:2405.05847*, 2024.

[21] Andrew Kyle Lampinen, Stephanie CY Chan, Yuxuan Li, and Katherine Hermann. Representation biases: will we achieve complete understanding by analyzing representations? *arXiv preprint arXiv:2507.22216*, 2025.

[22] Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. *arXiv*, 2025. arXiv preprint.

[23] Aaron J. Li, Suraj Srinivas, Usha Bhalla, and Himabindu Lakkaraju. Interpretability illusions with sparse autoencoders: Evaluating robustness of concept representations. *arXiv*, 2025. arXiv preprint.

[24] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

[25] Aleksandar Makelov, Georg Lange, and Neel Nanda. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. *arXiv preprint arXiv:2311.17030*, 2023.

[26] J. L. McClelland, D. E. Rumelhart, and PDP Research Group, editors. *Parallel Distributed Processing. Volume 2: Psychological and Biological Models*. MIT Press, Cambridge, MA, 1986.

[27] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[28] Neel Nanda. Attribution patching: Activation patching at industrial scale. `https://www.neelnanda.io/mechanistic-interpretability/attribution-patching`, 2022.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[30] Judea Pearl. An Introduction to Causal Inference. *The International Journal of Biostatistics*, 6(2):7, February 2010.

[31] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[32] D. E. Rumelhart, J. L. McClelland, and PDP Research Group, editors. *Parallel Distributed Processing. Volume 1: Foundations*. MIT Press, Cambridge, MA, 1986.

[33] Paul Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11(1):1–23, 1988.

[34] Denis Sutter, Julian Minder, Thomas Hofmann, and Tiago Pimentel. The non-linear representation dilemma: Is causal abstraction enough for mechanistic interpretability? *arXiv preprint arXiv:2405.05847*, 2025.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[36] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*, 2020.

[37] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022.

[38] Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah D. Goodman. Interpretability at scale: Identifying causal mechanisms in alpaca, 2024.

Input Tokens →		B	D	D	T	R	R	E
Count Up Down Variables →	Count:	0	1	2	2	1	0	NA
	Phase:	0	0	0	1	1	1	NA
Increment Up Up Variables →	Progress:	1/21	1/21	2/21	0	1/2	1	NA
	Increment:	1/21	1/21	1/21	1/2	1/2	1/2	NA

Figure 4: Visual depiction of the causal abstractions considered in this work.

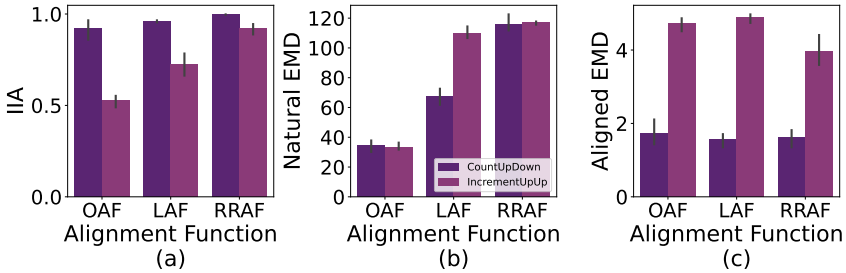

Figure 5: Compares results from different CAs. CountUpDown refers to the Count variable from the Up-Down abstraction, and IncrementUpUp refers to the Progress variable from the Increment-Up abstraction. **(a)** The final validation interchange intervention accuracy for different AFs on the x-axis and causal abstractions denoted by color. **(b)** The Earth Mover's Distance between naturally occurring latent vectors and intervened latent vectors (in which only the Count subspace or the Progress subspace have been manipulated depending on the causal abstraction) in the NN's neural space. **(c)** The Earth Mover's Distance between naturally occurring vectors and intervened vectors in the aligned space with the extraneous (i.e. non-causal) subspace set to zero.

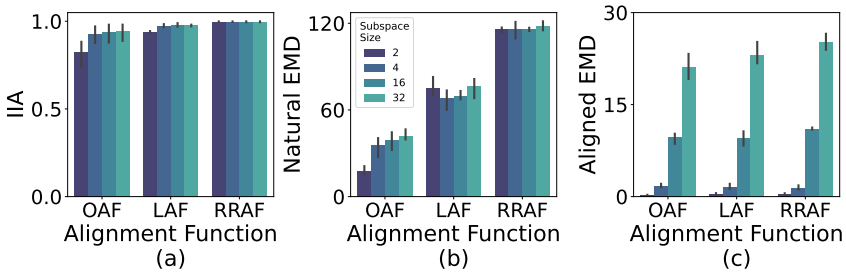

Figure 6: All panels show results for DAS trained on the Up-Down abstraction in which the reported IIA is for the Count variable for 3 model seeds. **(a)** The final validation IIA for different causal subspace sizes denoted by color. Each subspace size is used for both the Count and Phase variables in each alignment training. **(b)** The Earth Mover's Distance between naturally occurring latent vectors and intervened latent vectors (in which only the Count subspace has been manipulated) in the NN's neural space. **(c)** The Earth Mover's Distance between naturally occurring vectors and intervened vectors in the aligned space with the extraneous (i.e. non-causal) subspace set to zero.

A Appendix

A.1 Model Details

All artificial neural network models were implemented and trained using PyTorch [29] on Nvidia Titan X GPUs. Unless otherwise stated, all models used an embedding and hidden state size of 128 dimensions. To make the token predictions, each model used a two layer multi-layer perceptron (MLP) with GELU nonlinearities, with a hidden layer size of 4 times the hidden state dimensionality

with 50% dropout on the hidden layer. The model consisted of a single GRU recurrent cell followed by an output multi-layer perceptron (MLP). We show the GRU model structure:

$$h_{t+1} = f(h_t, x_t) \tag{16}$$

$$\hat{x}_{t+1} = g(h_{t+1}) \tag{17}$$

Where h_t is the hidden state vector at step t, x_t is the input token at step t, f is the GRU cell, and g is a two layer (two matrix) MLP used to make a prediction, $\hat{x}_{t+1}$, of the token at step $t+1$ from the updated hidden state h_{t+1}. Models were trained using a learning rate scheduler, which consisted of the original transformer [35] scheduling of warmup followed by decay. We used 100 warmup steps, a maximum learning rate of 0.0001 , a minimum of 1e-7, and a decay rate of 0.5. We used a batch size of 128, which caused each epoch to consist of 8 gradient update steps.

A.2 DAS Details

In our experiments, we perform causal interventions on individual time steps in the sequence. We run the model up to a sampled timestep t in the target sequence, taking its latent representation at that point as the target vector, h_t^{trg}. We do the same for the source vector, h_u^{src}, at timestep u from a separate source sequence. We then construct h_t^v using Equation 2, and continue the model's predictions starting from time t, using h_t^v in place of h_t^{trg}.

We use 10,000 intervention samples for training and 1,000 samples for validation and testing. For all data, we uniformly sample trial object quantities, and unless otherwise stated, we uniformly sample intervention time points, t and u, from sequence positions containing demo tokens or response tokens (excluding BOS, trigger, and EOS tokens).

We orthogonalize the rotation matrix for OAFs using PyTorch's orthogonal parameterization with default settings. We train Q with a batch size of 512 until convergence, selecting the checkpoint with the best validation performance for analysis. We use a learning rate of 0.001 and an Adam optimizer.

To train the AFs, we sampled 10,000 sequence pairs for the intervention training datasets. See Supplement A.3.1 for more details on intervention data construction and examples. We use a learning rate of 0.001 and a batch size of 512. We removed models with performance below 99% to limit our DAS results to perfectly performing models thus simplifying our interpretations of the results. We chose 99% accuracy instead of 100% due to slight numerical underflow in accuracy calculations and due the fact that half of the Variable-Length Same-Object models would have been dropped due to low performance.

A.3 Linear Alignment Functions

To construct the W matrix used in the LAF, we use the following equation: $W = (MM^\top + \epsilon I)S$. $M \in R^{d_m \times d_m}$ is a matrix of learned parameters initially sampled from a centered gaussian distribution with a standard deviation of $\frac{1}{d_m}$, $I \in R^{d_m \times d_m}$ is the identity matrix, $\epsilon = 0.1$ to prevent singular values equal to 0, and $S \in R^{d_m \times d_m}$ is a diagonal matrix to learn a sign for each column of X using diagonal values $s_{i,i} = \text{Tanh}(a_i) + \epsilon(\text{sign}(\text{Tanh}(a_i)))$ where each a_i is a learned parameter and $\epsilon = 0.1$ to prevent 0 values.

A.3.1 DAS Intervention Data

In this section we provide intervention data examples. To construct an intervention sample, we first sample a target sequence and a source squence and a positional index from each sequence. We exclude trigger tokens and beginning and end of sequence tokens from the possible positional indices. We then compute the values of each of the variables at the sampled indices using the specified CA for both the target and source. We then transfer the value of the variable of focus from the source into the the target variable in the CA. We then continue the CA using the new variable values to produce the counterfactual sequence.

A.3.2 Up-Down Program Examples

Count Variable: Interventions attempt to transfer the representation corresponding to the difference between the number of resp tokens and demo tokens. Interventions are only performed at positional

indices corresponding to demo or resp tokens. The target sequence maintains its original object count when the Count variable is changed in the demo phase. In cases where the new value exceeds the object count, the CA immediately produces the trigger token.

Multi-Object Examples	*1*	*2*	*3*	*4*
Source Sequence	BOS D_1	BOS D_2 D_1 D_1	BOS D_2 D_1 T R	BOS D_1 D_3 T R R
Target Sequence	BOS D_3 D_2	BOS D_2 T R	BOS D_1 D_2 D_1 T R	BOS D_2
Original Labels	D_2 D_3 T R R R R EOS	EOS	R R EOS	D_2 T R R EOS
Counterfactual	D_2 D_3 T R R R EOS	R R R EOS	R EOS	D_2 T R EOS
Single-Object Examples	*1*	*2*	*3*	*4*
Source Sequence	BOS D	BOS D D D	BOS D D T R	BOS D D T R R
Target Sequence	BOS D D	BOS D T R	BOS D D D T R	BOS D
Original Labels	D D T R R R R EOS	EOS	R R EOS	D T R R EOS
Counterfactual	D D T R R R EOS	R R R EOS	R EOS	D T R EOS
Same-Object Examples	*1*	*2*	*3*	*4*
Source Sequence	BOS C	BOS C C C	BOS C C T C	BOS C C T C C
Target Sequence	BOS C C	BOS C T C	BOS C C C T C	BOS C
Original Labels	C C T C C C C EOS	EOS	C C EOS	C T C C EOS
Counterfactual	C C T C C C EOS	C C C EOS	C EOS	C T C EOS

Phase Variable: Interventions transfer the representation corresponding to the Phase of the sequence (whether it is counting up or counting down). Interventions are only performed at positional indices corresponding to demo or resp tokens.

Multi-Object Examples	*1*	*2*	*3*	*4*
Source Sequence	BOS D_1	BOS D_3 D_1 D_2	BOS D_2 D_1 T R	BOS D_2 D_3 T R R
Target Sequence	BOS D_2 D_1	BOS D_3 T R	BOS D_1 D_3 D_1 T R	BOS D_2
Original Labels	D_3 D_1 T R R R R EOS	EOS	R R EOS	D_1 T R R EOS
Counterfactual	D_3 D_1 T R R R R EOS	D_2 T R EOS	R R EOS	R EOS
Single-Object Examples	*1*	*2*	*3*	*4*
Source Sequence	BOS D	BOS D D D	BOS D D T R	BOS D D T R R
Target Sequence	BOS D D	BOS D T R	BOS D D D T R	BOS D
Original Labels	D D T R R R R EOS	EOS	R R EOS	D T R R EOS
Counterfactual	D D T R R R R EOS	D T R EOS	R R EOS	R EOS
Same-Object Examples	*1*	*2*	*3*	*4*
Source Sequence	BOS C	BOS C C C	BOS C C T C	BOS C C T C C
Target Sequence	BOS C C	BOS C T C	BOS C C C T C	BOS C
Original Labels	C C T C C C C EOS	EOS	C C EOS	C T C C EOS
Counterfactual	C C T C C C C EOS	C T C EOS	C C EOS	C EOS

Algorithm 1 One sequence step of the Up-Down Program

```
q ← Count
p ← Phase
y ← input token
if y == BOS then                          ▷ BOS is beginning of sequence token
    q ← 0, p ← 0
    return sample(D)                      ▷ sample a demo token
else if y ∈ D then                        ▷ D is set of demo tokens
    q ← q + 1
    return sample(D)
else if y == T then                       ▷ T is trigger token
    p ← 1
else if y == R then                       ▷ R is response token
    q ← q − 1
end if
if (q == 0) & (p == 1) then
    return EOS                            ▷ EOS is end of sequence token
end if
return R
```

Algorithm 2 One sequence step of the Increment-Up Program

```
m ← Interval
q ← Progress
p ← Phase
i ← Increment
y ← input token
if y == BOS then                          ▷ BOS is beginning of sequence token
    q ← 0, p ← 0, i ← 1/m
    return sample(D)                      ▷ sample a demo token
else if (y ∈ D or y == R) and q < m then
    q ← q + i * m
else if y == T then                       ▷ T is trigger token
    p ← 1
    i ← 1/q
    q ← 0
end if
if (q ≥ m) and (p == 1) then
    return EOS                            ▷ EOS is end of sequence token
else if q ≥ m and p == 0 then
    return T
else if p == 0 then
    return sample(D)
else
    return R
end if
```