MODEL ALIGNMENT SEARCH

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

ABSTRACT

When can we say that two neural systems are the same? The answer to this question is goal-dependent, and it is often addressed through correlative methods such as Representational Similarity Analysis (RSA) and Centered Kernel Alignment (CKA). We find ourselves chiefly interested in the relationship between representations and behavior, asking ourselves how we can isolate specific functional aspects of representational similarity to relate our measures to behavior—avoiding cause vs. correlation pitfalls in the process. In this work, we introduce Model Alignment Search (MAS), a method for causally exploring distributed representational similarity as it relates to behavior. The method learns invertible linear transformations that find an aligned subspace between two distributed networks' representations where functional information can be isolated and manipulated. We first show that the method can be used to transfer values of specific causal variables—such as the number of items in a counting task—between networks with different training seeds and different architectures. We then explore open questions in number cognition by comparing different types of numeric representations in models trained on structurally different tasks, we explore differences between MAS and preexisting functional similarity methods, and lastly, we introduce a counterfactual latent auxiliary loss that helps shape functionally relevant alignments even in cases where we do not have causal access to one of the two models for training.

1 INTRODUCTION

An important question for understanding both Artificial and Biological Neural Networks (ANNs and BNNs) is knowing what it means for one distributed system to model or represent another (Sucholutsky et al., 2023). Establishing isomorphisms between different distributed systems can be useful for simplifying their complexity and for understanding otherwise opaque inner mechanisms. We cannot yet measure from every individual neuron in most BNNs; even if we could, as is the case in ANNs, it is still difficult to find satisfying ways of understanding the neural behavior. Finding simplified models that exhibit the causal relationships of more complex distributed systems can make complex systems more interpretable and communicable, potentially leading to useful insights (Cao & Yamins, 2021; 2024; Richards et al., 2019). Furthermore, there are a number of open questions about how representations differ or converge across architectures, tasks, and modalities (Huh et al., 2024; Sucholutsky et al., 2023; Wang et al., 2024; Li et al., 2024; Hosseini et al., 2024; Zhang et al., 2024; Grant et al., 2024). Researchers often use correlational methods to measure the similarity of different neural representations. We can see examples of this in works that perform direct correlational analyses between individual ANN activations and BNN firing rates (Yamins & DiCarlo, 2016; Maheswaranathan et al., 2019; Khosla & Williams, 2023; Williams et al., 2022), and in works that use Representational Similarity Analysis (RSA)—or Centered Kernel Alignment (CKA) (Kornblith et al., 2019; Williams, 2024)—finding 2nd order isomorphisms between model and system (Kriegeskorte et al., 2008). We also see examples of this in linear decoding techniques, where linear decodability can be used as a metric for understanding the type of information encoded in distributed representations (Chen et al., 2020; Radford et al., 2021; Grill et al., 2020; Caron et al., 2021; Haxby et al., 2001; Haxby, 2013). A question remains, however, how to causally associate these similarity metrics with behavioral outcomes. Can we develop methods to understand functional representational alignment (Geiger et al., 2024; Cloos et al., 2024; Schaeffer et al., 2024)?

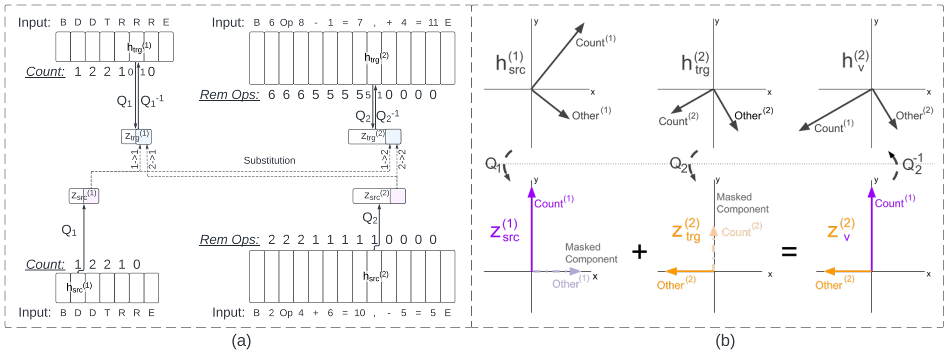

Figure 1: (a) Diagram of MAS showing all four possible intervention directions on the latent vectors (rectangles) of a Multi-Object Model$_1$ and an Arithmetic Model$_2$. Each rectangle is a latent vector produced from the input token. The values of the intervened causal variables—Count for the Multi-Object and Rem Ops for the Arithmetic models—are replaced by the values of the causal variables from the source latents, $h_{src}^{(i)}$. The value of the variable before the intervention is displayed on the left side of the arrows underneath the target latents $h_{trg}^{(}i)$. The value after the intervention is shown on the right. The dotted *Substitution* arrows each correspond to a single causal intervention. The models make predictions using the intervened vector following the intervention. (b) A theoretic causal intervention that transfers the Count information from Model$_1$'s latent representation, $h_{src}^{(1)}$, into Model$_2$'s $h_{trg}^{(2)}$. The superscripts (1) and (2) refer to the originating model. The hidden state vectors, $h_{src}^{(1)}$ and $h_{trg}^{(2)}$, are rotated into an aligned vector space using learned matrices Q_1 and Q_2. In the aligned vectors, $z_{src}^{(1)}$ and $z_{trg}^{(2)}$, the Count information lies along a disentangled component where it can be manipulated while preserving all other information. After the transfer, the rotated, intervened vector, $z_{trg}^{(2,v)}$, is returned to Model$_2$'s hidden state space, using Q_2^{-1}. Model$_2$ can then use the intervened vector to continue making predictions.

Some works have made progress toward understanding causal/functional representation similarity by transforming intermediate representations from one system into a usable form for another model. We see examples in works like Sexton & Love (2022) where they attempt to use transformed neural recordings in a trained computational model, and in *model stitching*, where a linear mapping is learned from intermediate representations in one ANN to another for the purpose of measuring similarity or improving one of the two models (Lähner et al., 2023; Moschella et al., 2023; Bansal et al., 2021; Lenc & Vedaldi, 2015). To build upon these works, we ask: 1) what do these causal mappings tell us about the underlying representations of the two systems? Are behaviorally successful mappings both necessary and sufficient for claims of functional similarity? 2) How do we compare models with disparate behavioral outputs, and how can we isolate the similarity of specific functional information in the representations? And 3) How do we achieve causal relevance in systems that we do not have causal access—as is often the case in ANN to BNN comparisons?

In this work, we introduce Model Alignment Search (MAS) to measure functional similarity between distributed networks. MAS can be thought of as a multi-model extension of Distributed Alignment Search (DAS) (Geiger et al., 2021; 2023), a technique used to align ANNs to symbolic algorithms (or directed acyclic graphs). MAS learns a rotation matrix for each model with the goal of finding an aligned representational subspace where information can be causally interchanged between the models and within each individual model. These intervened representations are then returned back to their original neural space where they can be used to produce behavior that can be compared to expected behavior.

We first validate MAS by comparing it to similarity measurements produced by RSA in multiple architectural and task variants. We then show how to use MAS even when one or both of the models use anti-Markovian states (Grant et al., 2024). We then show that MAS can reveal the representational similarity and dissimilarity in representations of number in models trained on structurally different tasks. We then show that MAS can be more restrictive than previous causal methods, like direct linear mappings (model stitching), and we provide a theoretical model to better understand desired restrictions. Lastly, we provide relevance to ANN-BNN comparisons by showing that we can introduce an auxiliary loss objective to recover causal relevance for causally inaccessible models. This loss function uses *counterfactual latent* vectors—latent vectors that match the desired representational makeup of the intervened vectors assuming a successful causal intervention—as training targets for the intervened vectors of the causally inaccessible model, thus constraining the alignment to be functionally relevant for the inaccessible system. We refer to this variant as Counterfactual latent MAS (CMAS).

The contributions of this work are as follows.

1. We introduce and validate MAS by comparing and contrasting to RSA.
2. We empirically and theoretically explore how MAS is similar to model stitching, but MAS can be more restrictive if desired and is more efficient when comparing more than 3 models.
3. We introduce a *counterfactual latent* auxiliary loss objective that can be used to find causally relevant alignments in cases where one of the two models is causally inaccessible (making the technique relevant for comparisons between ANNs and BNNs).
4. We show how MAS can be used to causally explore representations of numbers across structurally different tasks.

2 METHODS

In this work, we build upon the work of (Grant et al., 2024) to examine the causal similarity of distributed representations within models trained on next token prediction, numeric tasks. Each model is trained to $> 99.99\%$ accuracy on both training and validation data held before being analyzed and interpreted.

2.1 NUMERIC EQUIVALENCE TASKS

The goal of the numeric equivalance tasks is to reproduce a quantity of tokens that is initially observed at the start of the task. Each sequence consists of two phases: the demonstration (demo) and response phases. Each sequence has an associated *object quantity* that is uniformly sampled from 1 to 20. The ordering of the demo phase consists of a Beginning of sequence token, denoted B, a number of Demonstration (D) tokens equal in quantity to the object quantity, and a Trigger (T) token. The response phase then consists of the object quantity of Response (R) tokens and ends with the End of sequence (E) token. During the model training, we include all token types in the autoregressive, cross entropy loss, even though the number of D tokens and location of the T token is unpredictable from the sequence. A trial is considered correct when the model produces the appropriate number of R tokens followed by an E token during the response phase. We present two task variants:

Multi-Object Task: there are 3 possible demo token types $\{D_a, D_b, D_c\}$ that are uniformly sampled at each D in the sequence. There is a single response token type, R. As an example of an object quantity of 2, the sequence could be: "B D_c D_a T R R E"

Same-Object Task: there is a single token type, C, that is used as both the demo token type and the response token type. An example of a object quantity of 2 would be: "B C C T C C E".

We make a change to the Multi-Object Task when training the transformer models to prevent them from learning a solution that relies on reading out positional information (Grant et al., 2024). In this task variant, each token in the demo phase has a 0.2 probability of being sampled as a unique "void" token type, V, that is irrelevant to the completion of the numeric equivalence task. An example sequence with an object count of 2 could be: "B V D V V D T R R E". All evaluations use the original Multi-Object Task.

2.2 ARITHMETIC TASK

We include an arithmetic task consisting of addition/subtraction operations, interlaced with the intermediate, cumulative value following each operation. An example sequence is as follows: "B 3 Op 4 + 3 = 7 , + 11 = 18 , - 5 = 13 E", where the numeral between B and Op indicates the number of operations in the sequence and the numeral following Op is a sampled starting value. The number of operations is uniformly sampled from 1-10. The starting value is uniformly sampled from 0-20. All numeric values are restricted to 0-20. The arithmetic operations are uniformly sampled from $\{+, -\}$ when the cumulative value is in the range 1-19. Otherwise, the operation is selected to ensure the cumulative value stays in the range 0-20. The possible operands are uniformly sampled from the set that will restrict the subsequent cumulative value to the range 0-20. The sequence then displays the cumulative value after the "=". The "," token indicates that there are more operations in the sequence. The sequence ends with the E token after the originally indicated number of operations. We use a base 21 token system so that all values correspond to a single token.

2.3 Model Architectures

Each model in this work is autoregressively trained to perform only one of the tasks through next-token prediction (NTP). We train 2 model seeds for each task variant. We consider Gated Recurrent Units (GRUs) (Cho et al., 2014), Long-Short Term Memory recurrent networks (LSTMs) (Hochreiter & Schmidhuber, 1997), and two layer Transformers based on the Roformer architecture (Vaswani et al., 2017; Touvron et al., 2023; Su et al., 2023). The GRUs and Transformers use a dimensionality of 40, whereas the LSTM uses 20 dimensions for each the h and c vectors. We leave the details of GRU and LSTM cells to the referenced papers. The GRU and LSTM based models in this paper follow the structure:

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

$$\hat{x}_{t+1} = g(h_{t+1}) \tag{2}$$

Where h_t is the hidden state vector at step t, x_t is the input token at step t, f is the recurrent function (either a GRU or LSTM cell), and g is a two layer (two matrix) feed-forward network (FFN) used to make a prediction, $\hat{x}_{t+1}$, of the token at step $t+1$ from the updated hidden state h_{t+1}.

The transformer architecture uses Rotary Positional Encodings (RoPE) (Su et al., 2023) and GELU nonlinearities (Hendrycks & Gimpel, 2023). Transformers use a history of input tokens, $X_t = [x_1, x_2, ..., x_t]$, at each time step, t, to make a prediction: $\hat{x}_{t+1} = f(X_t)$, where f is now the transformer architecture. We show results from 2 layer, single attention head transformers. We refer readers to Figure 6 for more details. See more training details in Appendix A.1.

2.4 Model Alignment Search (MAS)

2.4.1 MAS Formulation

MAS can be thought of as a multi-model extension of DAS Geiger et al. (2021; 2023), where both methods attempt to find a representational subspace for a given model that aligns with a Symbolic Algorithm (SA) (or directed acyclic graph), and MAS further measures the degree to which two models' SA aligned subspaces align with each other. MAS and DAS are interpretability methods that operate on NNs with frozen weights. In our experiments, we first train the models to $> 99\%$ behavioral accuracy before freezing their weights. We note, however, that MAS can be used to align models with one another that are not fully trained. Refer to Figure 1 for a visual overview.

DAS tests the assumption that the hidden state, $h^{(i)} \in R^{d_i}$, for a single model$_i$ can be written as an orthogonal rotation $Q_i h^{(i)} = z^{(i)}$, where $Q_i \in R^{d_i \times d_i}$ is a learned orthonormal matrix, $z^{(i)} \in R^{d_i}$ consists of contiguous subspaces that encode high-level variables from SAs, and d_i is the size of the hidden state.

$$z^{(i)} = \begin{bmatrix} \vec{z}_{\text{var}_1} \\ \vec{z}_{\text{var}_2} \\ \cdots \\ \vec{z}_{\text{var}_n} \end{bmatrix} = Q_i h^{(i)} \tag{3}$$

Where each $\vec{z}_{\text{var}_k} \in R^{d_{\text{var}_k}}$ is a column vector of length d_{var_k} satisfying the relation $\sum_{k=1}^{n} d_{\text{var}_k} = d_i$. The benefit of this alignment is that it allows us understand model$_i$'s neural activity in terms of interpretable variables by allowing us to isolate and causally manipulate the value of each variable, var$_k$ within the NN's latent representations. MAS builds on DAS by measuring the degree to which model$_1$ and model$_2$ share the same $\vec{z}_{\text{var}_k}$ for a given var$_k$. With this formulation, we can freely isolate and manipulate $\vec{z}_{\text{var}_k}$ encoded in $h^{(i)}$ using a causal intervention:

$$h_v^{(i)} = Q_i^{-1}((1 - D_{i,\text{var}_k})Q_i h_{trg}^{(i)} + D_{j,\text{var}_k} Q_j h_{src}^{(j)}) \tag{4}$$

Where i and j can take on either model index in the set of all models considered, Q_i is a scaled orthogonal rotation matrix for model$_i$, $D_{i,\text{var}_k} \in R^{d_i \times d_i}$ is a diagonal, binary matrix with d_{var_k} non-zero elements used to isolate the dimensions corresponding to $\vec{z}_{\text{var}_k}$, $h_{src}^{(j)}$ is the *source vector* from which the subspace is harvested, $h_{trg}^{(i)}$ is the *target vector* into which activity is substituted, and $h_v^{(i)}$ is the resulting intervened vector that then replaces $h_{trg}^{(i)}$ in the target model$_i$'s processing, allowing the model to make causally intervened predictions. In this work, we train Q_i as the product $s_i U_i$ where s_i is a learned scalar and U_i is an orthonormal matrix.

As an example, we can picture an SA where we assume that all behaviorally relevant information can be encoded in a single variable, $\vec{z}_{full}$, and all extraneous, irrelevant information is encoded in a subspace $\vec{z}_{extra}$. In this case, $z^{(i)} = \begin{bmatrix} \vec{z}_{full} \\ \vec{z}_{extra} \end{bmatrix}$, and we can freely isolate and intervene upon $\vec{z}_{full}$ using Equation 4 once we have learned each Q_i for each given $D_{i,full}$.

2.4.2 MAS Training

MAS relies on the notion of counterfactual behavior to create intervention data to train and evaluate each Q_i. For a given SA, we know what the SA's behavior will be after performing a causal intervention on one of its variables. The counterfactual behavior of the SA is the resulting behavior of the SA after changing a specific variable while keeping all other variables unchanged. This counterfactual behavior can be used as a training signal for Q_i using a standard Next-Token Prediction (NTP) autoregressive loss. It is possible for Q_i to equivalently learn any permutation of the subspaces in z_i, thus we can restrict our trainings to values of the diagonal matrices D_{i,var_k} that have contiguous, non-zero entries. It is then possible to treat d_{var_k} as a hyperparameter in independent trainings, selecting the Q_i-D_{i,var_k} pair with the best results. Unless otherwise stated, we use values of d_{var} equal to 10, and we perform our causal interventions on individual time steps in the token sequences. We run the target model$_i$ up an independently sampled timestep t on the target sequence, using its latent representation at that point as the target vector, $h_{t,trg}^{(i)}$. We do the same on the source model$_j$ to obtain the source vector, $h_{u,src}^{(j)}$, at timestep u from a separate source sequence. We then construct $h_{t,v}^{(i)}$ using Equation 4, and continue model$_i$'s predictions starting from time t, using $h_{t,\text{v}}^{(i)}$ in place of $h_{t,trg}^{(i)}$. We perform batch gradient descent on Q_1 and Q_2 using the loss as follows for a single counterfactual sequence, k, of length S_k using i as the target model index and $p_{\theta_i}(x_s)$ as the probability generated by model$_i$ for the counterfactual token label at step s:

$$\mathcal{L}_i^{(k)}(Q_i, Q_j) = -\frac{1}{S_k} \sum_{s=t}^{S_k} \log p_{\theta_i}\big(x_s^{(k)} \mid x_{<s}^{(k)}, h_{t,\text{v}}^{(k),(i)}\big) \tag{5}$$

$$\mathcal{L}_{tot} = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}_i^{(k)}(Q_i, Q_j) \tag{6}$$

Where N is the number of samples in the batch.

For the LSTM architecture, we perform MAS on a concatenation of the h and c recurrent state vectors (Hochreiter & Schmidhuber, 1997). In the GRUs, we operate on the recurrent hidden state. In the transformers, we operate on the residual stream following the first transformer layer (referred to as the Layer 1 Hidden States in Supplementary Figure 6) or the input embedding layer. We use 10000 intervention samples for training and 1000 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 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 0.003 and an Adam optimizer.

MAS Evaluation: Once the Q_1 and Q_2 training loss has converged, we can evaluate the quality of the alignment using the accuracy of each model's predictions on counterfactual outputs from held out intervention data. We consider a trial correct when all deterministic tokens are predicted correctly using the argmax over logits. We report the proportion of trials correct for the worst performing causal intervention pairing, (i, j), as the Interchange Intervention Accuracy (IIA).

2.4.3 MAS VARIANTS

Stepwise MAS: In an effort to apply MAS to transformer architectures, we include a variant that applies Equation 4 to multiple contiguous time-steps in the target and source sequences from time 0 to u—contrasted to the previously considered single time step pair (t,u). This allows us to transfer part, or all, of the representations from the source sequence into the target. There is a question of what tokens to use to create each $h^{(i)}_{t,trg}$. We show results in which each target token comes from the original target sequence padded by the resp token when u exceeds the target sequence length.

Unidirectional MAS (UniMAS): In some settings we wish to examine models for which we have neural recordings but no causal access to the model (as is often the case in BNNs). We introduce a MAS variant called UniMAS that uses the activations from both models as source activations, but only uses model$_1$ as the target model during MAS trainings. Concretely, UniMAS changes Equation 6 to the following during trainings:

$$\mathcal{L}_{UniMAS} = \sum_{j=1}^{2} \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}_1^{(k)}(Q_1, Q_j) \tag{7}$$

As a baseline, we include **Linear Maps** which are trainings that exclusively use model 2 as the source model and model 1 as the target model, $\mathcal{L}_{LinMaps} = \frac{1}{N}\sum_{k=1}^{N} \mathcal{L}_1^{(k)}(Q_1, Q_2)$.

Counterfactual latent MAS (CMAS): We will show that the UniMAS trainings fail to learn an Q_2 with strong IIA in the inaccessible model. To address this, we introduce an auxiliary loss function to the UniMAS trainings. The auxiliary objective relies on *Counterfactual Latent (CL) vectors*. We define CL vectors as latent vectors that encode the causal variables that we would expect to exist in the intervened vector, $h^{(i)}_{trg,t}$ from Equation 4, after a successful causal intervention. We obtain these CL vectors from a neural representation dataset possessing situations and behaviors that are consistent with the information we wish to be encoded after the causal intervention. For example, if we have an SA with variables $\mathrm{v}ar_Y$, $\mathrm{v}ar_W$, and $\mathrm{v}ar_{extra}$, and following a causal intervention we expect $h^{(2)}_{t,\mathrm{v}}$ to have a value of y for variable $\mathrm{v}ar_Y$ and w for variable $\mathrm{v}ar_W$, then the CL vector can be obtained from a pre-recorded representation $h^{(i)}_{CL}$ where $Q_i^{-1} h^{(i)}_{CL}$ has the same expected variable values: $\mathrm{v}ar_Y = y$ and $\mathrm{v}ar_W = w$. The auxiliary loss $\mathcal{X}^{(k)}$ for a single sample k is composed of an L2 loss and a cosine loss using CL vectors as the ground truth:

$$\mathcal{X}^{(k)}_{L2} = \frac{1}{2} || h^{(k),(2)}_{t,\mathrm{v}} - h^{(k),(2)}_{CL} ||_2^2 \tag{8}$$

$$\mathcal{X}^{(k)}_{cos} = -\frac{1}{2} \frac{h^{(k),(2)}_{t,\mathrm{v}} \cdot h^{(k),(2)}_{CL}}{||h^{(k),(2)}_{t,\mathrm{v}}||_2 \, ||h^{(k),(2)}_{CL}||_2} \tag{9}$$

where $h^{(2)}_{t,\mathrm{v}}$ is the intervened target vector for the causally incaccessible model, and k denotes the index of the sample within the batch. The total CMAS training loss is a weighted sum of the UniMAS autoregressive loss from Equation 7, and the auxiliary loss. $\mathcal{L}_{CMAS} = \lambda(\mathcal{X}_{L2} + \mathcal{X}_{cos}) + (1 - \lambda)\mathcal{L}_{UniMAS}$ where λ is a hyperparameter.

As a baseline we include **Latent Fit** trainings which learn a single orthogonal matrix Q that maps $h^{(k),(1)}_{u,src}$ to $h^{(k),(2)}_{CL}$ minimizing only the $\mathcal{X}_{L2}$ and/or $\mathcal{X}_{cos}$. We select $\mathcal{X}_{L2}$ and/or $\mathcal{X}_{cos}$ based on the best validation IIA.

2.5 SYMBOLIC ALGORITHM VARIABLES

The choice of the counterfactual behavior defines what SA variable(s) we can align the neural activity to using MAS. The simplest option is to align all behaviorally relevant information by simply using the exact behavior of the source model as the counterfactual behavior. This is similar to model stitching in previous works such as Bansal et al. (2021); Lenc & Vedaldi (2015); Sexton & Love (2022). In these cases of complete information transfer, MAS still differs from previous works in that it performs the interventions in multiple causal directions using Q_1 and Q_2, and MAS isolates a functional subspace of the neural activity rather than using the entire latent space. MAS also has the ability to find alignments for specific types of information by conditioning the counterfactual sequences specific causal variables (i.e. the count of the sequence in the numeric equivalence tasks).

In all numeric equivalence tasks, we prevent interventions on representations resulting from the BOS, T, and EOS tokens. In the arithmetic task, we only perform interventions on representations after the "," token. We perform MAS using each of the following causal variables, where the corresponding task is denoted in parentheses:

1. **Full** (Arithmetic/Num Equivalence): Refers to cases in which we transfer all causally relevant information between models (not all activations).
2. **Count** (Num Equivalence): The difference between the number of observed demo tokens and the number of response tokens in the sequence. Example: the following sequences have a Count of 2 at the last token: "B D D" ; "B D D D T R"
3. **Last Value** (Num Equivalence): The value of the input token with respect to changing the Count of the sequence. We assign the values as D=1, R=-1, and all other tokens are 0. Example: if we change the Last Value of a single D token from 1 to -1, the counterfactual sequence should be "B D D D T R E". Example: in a partial sequence "B D D T R", changing the R token from -1 to 1 results in: "B D D T R R R R E".
4. **Cumu Val** (Arithmetic): The cumulative value of the arithmetic sequence in the Arithmetic task. Example: if we substitute in a value of 3 at the "," token in the sequence "B 2 Op 3 + 5 = 8 ," the counterfactual sequence could be "B 2 Op 3 + 5 = 8 , + 2 = 5 E" where the "+" and "2" are provided by the task.
5. **Rem Ops** (Arithmetic): The remaining number of operations in the arithmetic sequence. Example: we substitute in a value of 1 at the "," token in the sequence "B 3 Op 3 + 5 = 8 ,", the counterfactual sequence could be "B 3 Op 3 + 5 = 8 , + 1 = 9 E" where the "+" and "1" are provided.

2.6 Additional Methods

Representational Similarity Analysis (RSA): For a given model layer, we run the model on a batch of sequences consisting of 15 sequences from each object quantity 1-20. We then sample 1000 representational vectors uniformly from all time points excluding padding and end of sequence tokens. We construct a Representational Dissimilarity Matrix (RDM) as 1 minus the cosine similarity matrix over each pair-wise comparison of the representations (resulting in an RDM of dimensions 1000×1000). We create an RDM for two models and compare the RDMs using Spearman's rank correlation on the lower triangle of each matrix (Virtanen et al., 2020). We perform the RDM sampling 10 times and report the average over all 10 correlations.

3 Results/Discussion

3.1 MAS

We first turn our attention to the MAS IIA in Figure 2. The figure shows analyses where Model_1 is consistent horizontally across each row and Model_2 is displayed vertically along each column. We exclude comparisons of each model seed with itself. We see the MAS results conditioned on three different variables: Full, Count, and Last Value (see Section 2.5). The IIA shows the proportion of trials with successful counterfactual behavior after the causal interventions. A trial is considered successful if all deterministic counterfactual tokens are predicted correctly. For the Full and Count variables, causal interventions are performed on the hidden state vector after the recurrent processing in recurrent models and on the hidden states following the first transformer layer in transformers. Causal interventions conditioned on the Last Value variable are performed on the input embedding layer in all architectures.

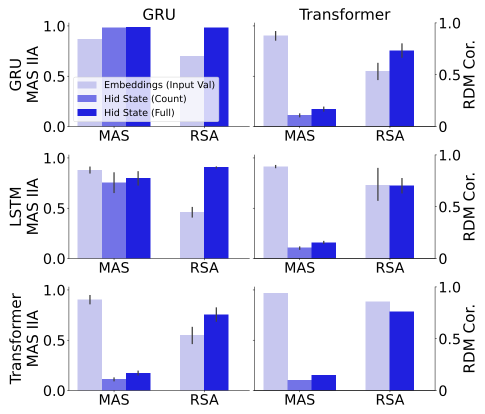

Figure 2: A performance comparison of MAS and RSA. The Model$_1$ architecture is consistent horizontally across panel rows. The Model$_2$ architecture is consistent vertically along the columns. All models are trained on the Multi-Object task. The bar colors correspond to different activation layers, and, in the case of MAS, the colors further distinguish which causal variables the analysis was conditioned upon. RDM Cor. is the value associated with an RSA whereas IIA is the intervention accuracy.

We see from Figure 2 and Appendix 7 that MAS is successful at performing the Full variable interventions between different recurrent model seeds. This is consistent with the findings of previous work on direct linear mappings between networks (Lenc & Vedaldi, 2015; Bansal et al., 2021; Lähner et al., 2023; Sexton & Love, 2022; Sucholutsky et al., 2023). Furthermore, MAS is successful at transferring the Count between the Multi-Object GRUs that have previously been shown to have an interchangeable Count variable using DAS (Grant et al., 2024). This success is qualified by MAS' inability to transfer the Count when one of the two models is a Same-Object GRU (Figure 3), which is also expected from DAS. MAS on the hidden states of the transformers is largely unsuccessful with a value of 0.087, whereas MAS on the embeddings is successful with a value of 0.961. This is consistent with previous work that has shown the transformers solve the numeric equivalence tasks without encoding a Markovian cumulative state. Instead, they rely on re-solving the task at each step in the sequence (Grant et al., 2024; Behrens et al., 2024). See Figure 7 for DAS results.

3.2 MAS vs RSA

RSA is a second order correlational method that examines the similarity between sample correlation matrices constructed from two models' representations (see Appendix A.3 for details). We provide comparisons between models differing only by seed to establish an upper limit on MAS and RSA values. We also provide GRU-LSTM comparisons to establish value changes resulting from architectural differences. Turning to Figure 2 and Appendix 7, we highlight that RSA provides a lower value than MAS on the embedding layers and a larger value than MAS on the the hidden states of the transformers. In the case of the hidden states in the Transformer-Transformer comparison (the lower rightmost panel), we see an RSA value of 0.78 and a CKA value of 0.96 in the appendix. The values in this comparison are difficult to interpret, as we might expect higher values due to similarities

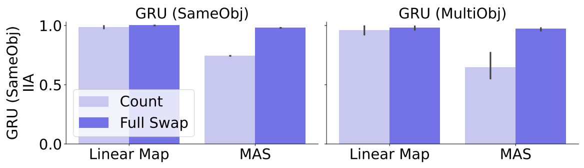

Figure 3: Performance comparison of MAS and direct linear mappings (Linear Map) between the Same-Object GRUs and between the Same-Object and Multi-Object GRUs. The Count information can be successfully transferred to and from Same-Object GRUs when using Linear Map but not in MAS.

in architecture and training, but we also know from causal experiments that there is little causal transferrability between the transformers' representations. We note that questions on how to interpret RSA values have been addressed in previous works (Kriegeskorte et al., 2008; Sucholutsky et al., 2023; Dujmović et al., 2022).

3.3 Direct Linear Mappings

Turning to Figure 3, we compare the IIA of MAS to direct linear mappings (denoted Linear Map). We restrict the Linear Map matrices to be orthogonal, making them equivalent to MAS trained in a single causal direction, where the IIA is reported from only the trained causal direction. We see that the Count variable can be successfully transferred between the Same-Object GRUs when using direct linear mappings. This is in contrast to MAS and the results of DAS from previous work (Grant et al., 2024). To better understand this result, we first formulate the direct linear mappings in terms of the latent states, h, and the interpretable z vectors. Interchange interventions can be thought of as an equality constraint, where elements that are exchanged have some functional equivalence for successful interventions. Thus, the direct linear mapping is similar to learning a matrix $W \in R^{d_1 \times d_2}$ such that $h^{(1)} = Wh^{(2)}$. We can rewrite this as follows:

$$h^{(1)} = Q_1^{-1} z^{(1)} = Wh^{(2)} = WQ_2^{-1} z^{(2)} \tag{10}$$

$$z^{(1)} = Q_1 W Q_2^{-1} z^{(2)} = X z^{(2)} \tag{11}$$

If we focus on only a single variable component, $\mathrm{v}ar_1$, in $z^{(1)}$, and assume it is only 1 dimensional for notational simplicity, we can see that $z_{\mathrm{v}ar_1} = \sum_{k=1}^{d_2} x_{1,k} * z_k^{(2)}$ where the $x_{1,\text{col}}$ are elements of the first row of X and z_k are row elements of z. This shows that $z_{\mathrm{v}ar_1}$ can include non-causal elements from $z^{(2)}$ in its calculation in the direct linear mapping case. This means that it can use non-causal information that correlates with the variable of interest. Using the same functional equivalence formulation for MAS, however, finds mappings such that for the causal variable of focus, $\mathrm{v}ar_1$, $z_{\mathrm{v}ar_1}^{(1)} = z_{\mathrm{v}ar_1}^{(2)}$ demonstrated as follows:

$$D_{\mathrm{v}ar_1} Q_1 h^{(1)} = D_{\mathrm{v}ar_1} z^{(1)} = \begin{bmatrix} \vec{z}_{\mathrm{v}ar_1}^{(1)} \\ \vec{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \vec{z}_{\mathrm{v}ar_1}^{(2)} \\ \vec{\mathbf{0}} \end{bmatrix} = D_{\mathrm{v}ar_1} z^{(2)} = D_{\mathrm{v}ar_1} Q_2 h^{(2)} \tag{12}$$

This constraint is a possible mechanism for why we see inflated IIA in the direct linear mappings seen in Figure 3.

We briefly note that while model stitching learns a transformation matrix between pairs of models, MAS learns a single transformation matrix for each model aligning them to a unified space. This reduces the number of matrices required for comparing n models from $\binom{n}{2}$ to n.

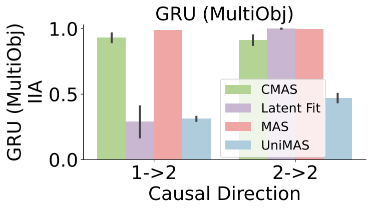

Figure 4: The IIAs in the causally untrained directions for CMAS on the Multi-Object GRU models on the Full variable. The x labels denote the intervention directions where "1" and "2" denote the model index and the arrow points from the source to the target model. Latent Fit variants fit a transformation to the latent vectors in one direction without including causal behavior training. UniMAS and MAS show a lower and upper bound on CMAS performance. The $2->2$ Latent Fit direction is trivial.

3.4 CMAS

We can see from Figure 4 the results of CMAS compared to MAS, UniMAS, and Latent Fits. It is important to note that the Latent Fit, UniMAS, and CMAS variants do not include the autoregressive

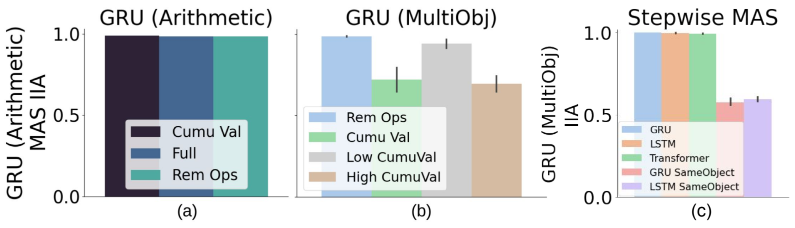

Figure 5: (a) MAS between GRUs trained on the Arithmetic task showing transferrability of the different types of information. (b) MAS used to compare the Count variable in the the Multi-Object GRUs to the Rem Ops and Cumu Val variables in the Arithmetic GRUs. Cumu Val results from MAS performed on all possible Cumu Val values. Low CumuVal results from a separate MAS analysis restricted to Cumu Val values in the range of the Rem Ops variable. High CumuVal results show a MAS analysis conditioned on values beyond the Rem Ops range. (c) Stepwise MAS doing a full behavioral transfer applied to the embedding layer between the Multi-Object GRUs and various other model types. The Stepwise procedure allows for comparisons with Transformers despite their inductive bias towards anti-Markovian solution patterns.

counterfactual training signal in causal directions $1 \rightarrow 2$ and $2 \rightarrow 2$. MAS provides a theoretic upper bound on the possible IIA for CMAS. UniMAS provides a lower bound on the possible CMAS performance. We also note that the UniMAS results demonstrate that the causally relevant $\mathcal{R}_2^{-1}$ is not automatically learned. Lastly, Latent Fit—which trains a rotation matrix to map latent vectors from Model$_1$ to latent vectors of Model$_2$ without any causal behavioral training—provides a baseline of existing methods. We see that CMAS recovers much of the possible performance of MAS whereas the Latent Fit performs near the lower bound. This demonstrates the potential of CMAS to recover causally relevant intervention rotation matrices even when we do not have causal access to one of the models in the comparison.

3.5 Step-Wise MAS

In order to explore functional notions of similarity in cases where the models of interest use anti-Markovian solution structures (avoiding cumulative hidden states), we introduce a variant of MAS that is applied at each step in the sequence up to time u. We use this method to explore the similarity of embedding representations of the Multi-Object GRUs to other models as seen in Figure 5 (c). As we might expect, the GRUs have a high IIA when compared to the Multi-Object models, and the IIA drops significantly when compared to the Same-Object models which use only a single counting token type. This methodological variant opens the possibility of performing interesting analyses with LLMs, despite their possible anti-Markovian latent structures.

3.6 Arithmetic

We include a MAS analysis between Arithmetic GRUs and GRUs trained on the Numeric Equivalence tasks (see Figure 5). The leftmost panel shows that we can successfully align the Cumu Val, and the Rem Ops variables between and within the Arithmetic GRUs. The middle panel shows that MAS can successfully align the Count with the Rem Ops variables between GRUs trained on the Multi-Object and Arithmetic tasks respectively. These results are qualified by the lower IIA alignment between the Count and the Cumu Val variables. We see that when we perform MAS only on Cumu Val values that are shared with possible the Rem Ops values (Low CumuVal), the results are much higher but still do not match the results from Rem Ops. These findings are consistent with the hypothesis that these GRUs are using different types of numeric representations for arithmetic than incremental counting.

4 Conclusion

This work has introduced a technique for causally measuring the functional similarity of representations in two neural systems. We showed the potential need to complement commonly used correlational methods with MAS, and we showed how MAS improves on previous causal similarity methods. We also showed that MAS can be used to address questions of representational similarity across diverse task structures, and we introduced CMAS as a promising direction for learning causal interventions in cases where we do not have causal access to one of the two models. This work has been confined to ANNs, but we look forward to future explorations in biological neural settings.

5 Acknowledgments

Thank you to the PDP Lab, the Stanford Psychology department, and the MBCT training program for funding. Thank you to Stephen Baccus, Noah Goodman, Zen Wu, Atticus Geiger, Linas Nasvytis, Jenelle Feather, Chris Potts, Alexa Tartaglini, Daniel Wurgaft, the PDP lab, and the Stanford Mech Interp community for thoughtful discussion. Thanks to Joshua Melander, Josh Wilson, Ben Prystawski, Jerome Han, and Andrew Lampinen for thoughtful discussion and paper feedback. Thank you to Alex Williams for pointing out the matrix efficiency framing of MAS relative to model stitching. Special thanks to Jay McClelland for encouragement, paper feedback, and many thoughtful conversations.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting Model Stitching to Compare Neural Representations, June 2021. URL http://arxiv.org/abs/2106.07682. arXiv:2106.07682 [cs, stat].

Freya Behrens, Luca Biggio, and Lenka Zdeborová. Counting in small transformers: The delicate interplay between attention and feed-forward layers, 2024. URL https://arxiv.org/abs/2407.11542.

Rosa Cao and Daniel Yamins. Explanatory models in neuroscience: Part 1 – taking mechanistic abstraction seriously, April 2021. URL http://arxiv.org/abs/2104.01490. arXiv:2104.01490 [cs, q-bio].

Rosa Cao and Daniel Yamins. Explanatory models in neuroscience, Part 2: Functional intelligibility and the contravariance principle. *Cognitive Systems Research*, 85:101200, June 2024. ISSN 1389-0417. doi: 10.1016/j.cogsys.2023.101200. URL https://www.sciencedirect.com/science/article/pii/S1389041723001341.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. 2020.

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. URL http://arxiv.org/abs/1406.1078.

Nathan Cloos, Moufan Li, Markus Siegel, Scott L. Brincat, Earl K. Miller, Guangyu Robert Yang, and Christopher J. Cueva. Differentiable optimization of similarity scores between models and brains, 2024. URL https://arxiv.org/abs/2407.07059.

Marin Dujmović, Jeffrey S Bowers, Federico Adolfi, and Gaurav Malhotra. The pitfalls of measuring representational similarity using representational similarity analysis. *bioRxiv*, 2022. doi: 10.1101/2022.04.05.487135. URL https://www.biorxiv.org/content/early/2022/04/07/2022.04.05.487135.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *CoRR*, abs/2106.02997, 2021. URL https://arxiv.org/abs/2106.02997.

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

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. URL https://arxiv.org/abs/2301.04709.

Satchel Grant, Noah D. Goodman, and James L. McClelland. Emergent symbol-like number variables in artificial neural networks, 2024.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

James V Haxby. Multivariate pattern analysis of fMRI: Parcellating abstract from concrete representations. *NEuroimage*, 62(2):2013, 2013. doi: 10.1016/j.neuroimage.2012.03.016.Multivariate.

James V. Haxby, M. Ida Gobbini, Maura L. Furey, Alumit Ishai, Jennifer L. Schouten, and Pietro Pietrini. Distributed and overlapping representations of faces and objects in ventral temporal corten. *Social Neuroscience: Key Readings*, 293(September):87–96, 2001. doi: 10.4324/9780203496190.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Eghbal Hosseini, Colton Casto, Noga Zaslavsky, Colin Conwell, Mark Richardson, and Evelina Fedorenko. Universality of representation in biological and artificial neural networks. *bioRxiv*, 2024. doi: 10.1101/2024.12.26.629294. URL https://www.biorxiv.org/content/early/2024/12/26/2024.12.26.629294.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis, 2024. URL https://arxiv.org/abs/2405.07987.

Meenakshi Khosla and Alex H. Williams. Soft matching distance: A metric on neural representations that captures single-neuron tuning, 2023. URL https://arxiv.org/abs/2311.09466.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019. URL https://arxiv.org/abs/1905.00414.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2, 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008. URL https://www.frontiersin.org/articles/10.3389/neuro.06.004.2008.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence, 2015. URL https://arxiv.org/abs/1411.5908.

Jiaang Li, Yova Kementchedjhieva, Constanza Fierro, and Anders Søgaard. Do Vision and Language Models Share Concepts? A Vector Space Alignment Study. *Transactions of the Association for Computational Linguistics*, 12:1232–1249, September 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00698. URL https://doi.org/10.1162/tacl_a_00698.

Zorah Lähner, Michael Moeller, Marco Fumero, Emanuele Rodolà, Clementine Domine, Francesco Locatello, Karolina Dziugaite, and Mathilde Caron. On the Direct Alignment of Latent Spaces. 2023.

Niru Maheswaranathan, Lane T. McIntosh, Hidenori Tanaka, Satchel Grant, David B. Kastner, Josh B. Melander, Aran Nayebi, Luke Brezovec, Julia Wang, Surya Ganguli, and Stephen A. Baccus. The dynamic neural code of the retina for natural scenes. *bioRxiv*, 2019. doi: 10.1101/340943. URL https://www.biorxiv.org/content/early/2019/12/17/340943.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication, March 2023. URL http://arxiv.org/abs/2209.15430. arXiv:2209.15430 [cs].

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. URL http://arxiv.org/abs/1912.01703.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

Blake A. Richards, Timothy P. Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J. Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W. Lindsay, Kenneth D. Miller, Richard Naud, Christopher C. Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C. Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P. Kording. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, November 2019. ISSN 1546-1726. doi: 10.1038/s41593-019-0520-2. URL https://www.nature.com/articles/s41593-019-0520-2. Publisher: Nature Publishing Group.

Rylan Schaeffer, Mikail Khona, Sarthak Chandra, Mitchell Ostrow, Brando Miranda, and Sanmi Koyejo. Does maximizing neural regression scores teach us about the brain? In *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024. URL https://openreview.net/forum?id=vbtj05J68r.

Nicholas J. Sexton and Bradley C. Love. Reassessing hierarchical correspondences between brain and deep networks through direct interface. *Science Advances*, 8(28):eabm2219, July 2022. doi: 10.1126/sciadv.abm2219. URL https://www.science.org/doi/10.1126/sciadv.abm2219. Publisher: American Association for the Advancement of Science.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

Ilia Sucholutsky, Lukas Muttenthaler, Adrian Weller, Andi Peng, Andreea Bobu, Been Kim, Bradley C. Love, Erin Grant, Iris Groen, Jascha Achterberg, Joshua B. Tenenbaum, Katherine M. Collins, Katherine L. Hermann, Kerem Oktar, Klaus Greff, Martin N. Hebart, Nori Jacoby, Qiuyi Zhang, Raja Marjieh, Robert Geirhos, Sherol Chen, Simon Kornblith, Sunayana Rane, Talia Konkle, Thomas P. O'Connell, Thomas Unterthiner, Andrew K. Lampinen, Klaus-Robert Müller, Mariya Toneva, and Thomas L. Griffiths. Getting aligned on representational alignment, 2023. URL https://arxiv.org/abs/2310.13018.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

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. URL http://arxiv.org/abs/1706.03762.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Junxuan Wang, Xuyang Ge, Wentao Shu, Qiong Tang, Yunhua Zhou, Zhengfu He, and Xipeng Qiu. Towards Universality: Studying Mechanistic Similarity Across Language Model Architectures, October 2024. URL `http://arxiv.org/abs/2410.06672`. arXiv:2410.06672 [cs].

Alex H. Williams. Equivalence between representational similarity analysis, centered kernel alignment, and canonical correlations analysis. *bioRxiv*, 2024. doi: 10.1101/2024.10.23.619871. URL `https://www.biorxiv.org/content/early/2024/10/24/2024.10.23.619871`.

Alex H. Williams, Erin Kunz, Simon Kornblith, and Scott W. Linderman. Generalized shape metrics on neural representations, 2022. URL `https://arxiv.org/abs/2110.14739`.

Daniel L.K. Yamins and James J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3):356–365, 2016. ISSN 15461726. doi: 10.1038/nn.4244.

Jerrold H Zar. Spearman rank correlation. *Encyclopedia of Biostatistics*, 7, 2005.

Le Zhang, Qian Yang, and Aishwarya Agrawal. Assessing and learning alignment of unimodal vision and language models, 2024. URL `https://arxiv.org/abs/2412.04616`.

A Appendix

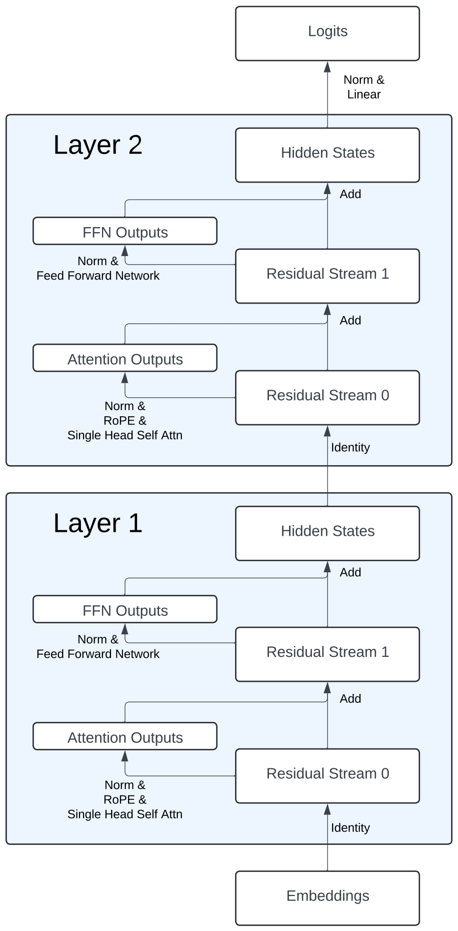

Figure 6: Figure and caption taken from Grant et al. (2024). Diagram of the transformer architecture used in this work. White rectangles represent activation vectors, arrows represent functional operations. All causal interventions were performed on either the Hidden State activations from Layer 1 or the Embeddings layer. All normalizations are Layer Norms (Ba et al., 2016).

A.1 Model Details

All artificial neural network models were implemented and trained using PyTorch (Paszke et al., 2019) on Nvidia Titan X GPUs. Unless otherwise stated, all models used an embedding and hidden state size of 48 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 GRU and LSTM model variants each consisted of a single recurrent cell followed by the output MLP. Unless otherwise stated, the transformer architecture consisted of two layers using Rotary positional encodings (Su et al., 2023). Each model variant used the same learning rate scheduler, which consisted of the original transformer (Vaswani et al., 2017) scheduling of warmup followed by decay. We used 100 warmup steps, a maximum learning rate of 0.001 , 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 MAS (and Associated Variants) Training Details

For each rotation matrix training, we use 10000 intervention samples and 1000 samples for validation and testing. We uniformly sampled corresponding indices upon which to perform interventions, excluding the B, T, and E tokens in the numeric equivalence tasks from possible intervention sample

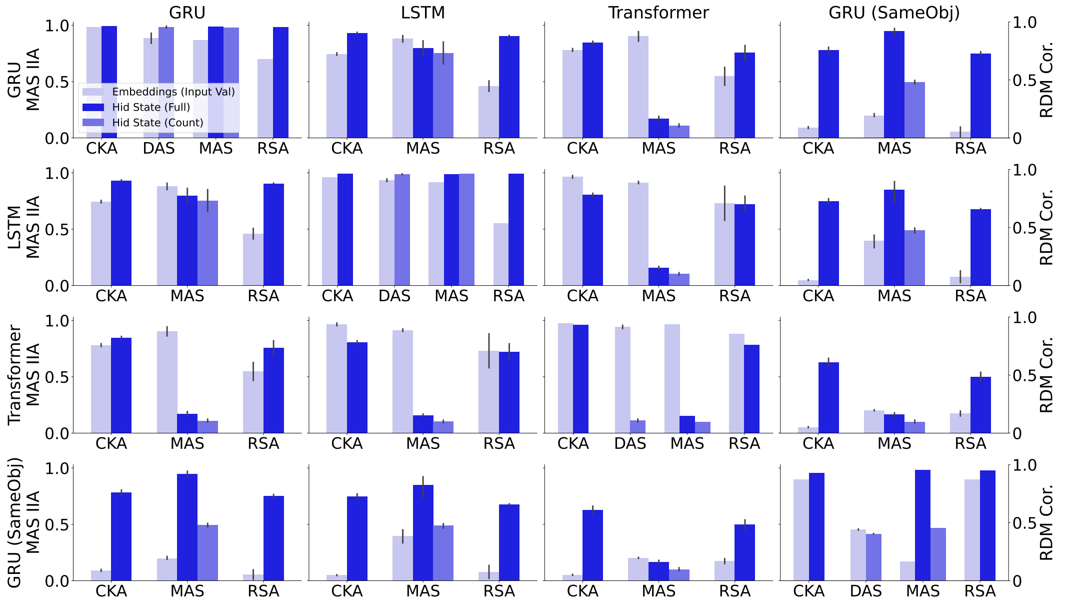

Figure 7: A performance comparison of MAS against RSA. The rows of panels show Model 1 whereas the columns represent the Model 2 in the context of the MAS diagram Figure 1. The different colors represent different activation layers within the models, and, in the case of MAS, the colors further distinguish which causal variables the analysis was conditioned upon.

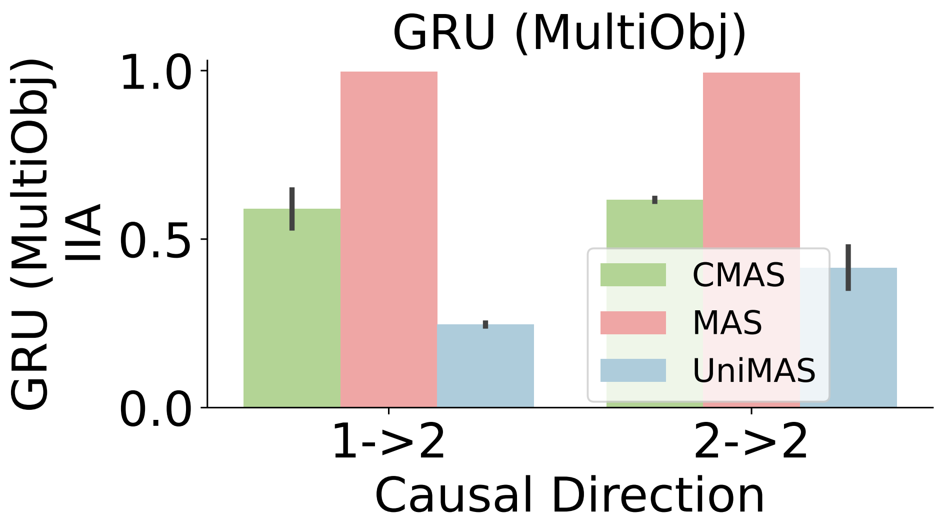

Figure 8: The IIAs in the causally untrained directions for CMAS on the Multi-Object GRU models on the Count variable. The x labels denote the intervention directions where the numbers 1 and 2 denote the model index and the arrow points from the source to the target model. UniMAS and MAS show a lower and upper bound on CMAS performance.

indices. In the Arithmetic task, we used the comma token for Rem Ops and Cumu Val interventions. When intervening upon a state in the demo phase in the numeric equivalence tasks, we uniformly sample 0-3 steps to continue the demo phase before changing the phase by inserting the trigger token. We orthongonalize the matrices, Q_i, using PyTorch's orthogonal parametrization with default settings. PyTorch creates the orthogonal matrix as the exponential of a skew symmetric matrix. We train the rotation matrices for 1000 epochs, with a batch size of 512 used for each model index pairing. We only perform experiments considering two models. Each gradient step uses the average gradient over batches of all 4 i, j pairings. We select the checkpoint with the best validation performance for analysis. We use a learning rate of 0.003 and an Adam optimizer.

A.3 RSA Details

We performed RSA on a subsample of a dataset of 15 sampled sequences for each object quantity ranging from 1-20 on the task that each model was trained on for each model. We first ran the models on their respective datasets to collect the latent representations. We sampled 1000 of these latent vectors as the sample representations in a matrix $M_k \in R^{N \times d_k}$ where k refers to the model index, N is the number of latent vectors ($N = 1000$ in our analyses), d_k is the dimensionality of a single latent vector for model k. We then calculated the sample cosine distance matrices (1-cosine similarity) for each model resulting in matrices $C_k \in R^{N \times N}$. Lastly we calculated the Spearman's Rank Correlation Coefficient between the lower triangles of the matrices C_1 and C_2 as the RSA value using python's SciPy package (Zar, 2005; Kriegeskorte et al., 2008; Virtanen et al., 2020). We resampled the 1000 vectors 10 times and recalculated the RSA score 10 times and report the average over these scores.

A.4 CKA Details

We performed CKA on a subsample of a dataset of 15 sampled sequences for each object quantity ranging from 1-20 on the task that each model was trained on for each model. We first ran the models on their respective datasets to collect the latent representations. We sampled 1000 of these latent vectors as the sample representations in a matrix $M_k \in R^{N \times d_k}$ where k refers to the model index, N is the number of latent vectors ($N = 1000$ in our analyses), d_k is the dimensionality of a single latent vector for model k. We then normalized the vectors along the sample dimension by subtracting the mean and dividing by the standard deviation (using d_k means and d_k standard deviations calculated over 1000 samples). Using these samples we calculated the kernel matrices using cosine similarity to create matrices $C_k \in R^{N \times N}$. Using these matrices, we computed the Hilbert-Schmidt Independence Criterion (HSIC) where $I \in R^{N \times N}$ is the identity and $J \in R^{N \times N}$ is a matrix of values all equal to 1:

$$H = I - \frac{1}{N} J \tag{13}$$

$$\text{HSIC}(C_1, C_2) = \frac{\text{trace}(C_1 H C_2 H)}{(N-1)^2} \tag{14}$$

and lastly we computed CKA as the following:

$$\text{CKA} = \frac{\text{HSIC}(C_1, C_2)}{\sqrt{\text{HSIC}(C_1, C_1)\text{HSIC}(C_2, C_2)}} \tag{15}$$

(Kornblith et al., 2019). We resampled the 1000 vectors 10 times and recalculated the CKA score 10 times and report the average over these scores.