PLOT: Progressive Localization via Optimal Transport in Neural Causal Abstraction

Jonathn Chang [1] **Arya Datla** [1] **Ziv Goldfeld** [1]

Abstract

Causal abstraction offers a principled framework for mechanistic interpretability, aligning a high-level causal model with the low-level computation realized by a neural network through counterfactual intervention analysis. Existing methods such as distributed alignment search (DAS) learn expressive subspace interventions, but the relevant neural site is unknown a priori, so finding a handle requires a computationally burdensome search over candidate sites. We introduce PLOT (Progressive Localization via Optimal Transport), a transport-based framework that localizes causal variables from the output effect geometry of abstract and neural interventions. PLOT fits an optimal transport coupling between abstract variables and candidate neural sites, yielding a global soft correspondence that can be calibrated into intervention handles. In simple settings, a single coupling over individual neurons suffices. In larger models, PLOT is applied progressively, moving from coarse sites such as tokens, timesteps, or layers to finer supports such as coordinate groups or PCA spans, and optionally guiding DAS based on the localized signal. Across experiments of increasing complexity, transport-only PLOT handles are exceedingly fast, while PLOT-guided DAS reaches DAS-level accuracy at a fraction of full DAS runtime, providing an efficient localization engine for causal abstraction research at scale. Code is available at https://github.com/jchang153/causal-abstractions-ot.

[1]Cornell University. Correspondence to: Jonathn Chang <jc3683@cornell.edu>, Arya Datla <avd42@cornell.edu>, Ziv Goldfeld <goldfeld@cornell.edu>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Causal abstractions of neural models are central to mechanistic interpretability. Given a trained neural network and a high-level model with interpretable latent variables, the goal is to determine whether the network realizes the abstract computation (Pearl, 2009; Geiger et al., 2025). Under this view, interpretation is not merely a matter of finding correlated neurons or salient features. Rather, an abstract variable is realized only if interventions on it can be matched by interventions at internal neural sites. This turns interpretability into a counterfactual localization problem of finding a faithful correspondence between the variables of an abstract algorithm and their high-dimensional, distributed neural realization.

Recent work has made this question operational through interchange interventions and distributed alignment search (DAS) (Geiger et al., 2022; 2024; Wu et al., 2024). DAS fixes a candidate neural site, such as a layer, and learns a rotation under which swapping a selected subspace between source and base activations reproduces the abstract intervention. This makes DAS expressive and accurate (Mueller et al., 2025), but local and computationally intensive. For each high-level variable, DAS must specify candidate sites, train rotations, and select among them using calibration data. Thus, DAS is powerful once a site has been proposed, but it does not provide a global mechanism for discovering where in the network to search. We address this limitation with a probabilistic approach that progressively localizes abstract variables, narrowing the search from broad regions to compact supports where handles can be extracted or DAS can be applied.

1.1. Contributions

We propose PLOT (Progressive Localization via Optimal Transport), a transport-based framework for causal abstraction localization. PLOT constructs output effect signatures for abstract and neural swap interventions, then fits an optimal transport (OT) coupling between the resulting signature collections. These signatures geometrically represent how model outputs change under a swap at a given variable or site. The resulting coupling gives a global soft correspondence between high-level variables and candidate

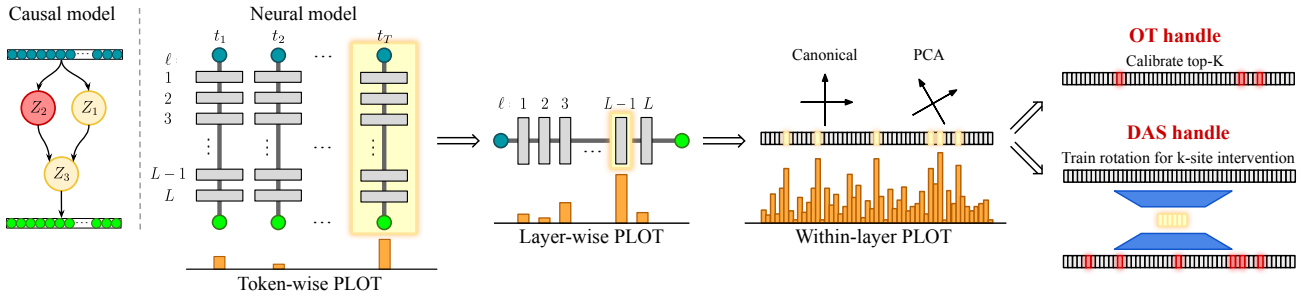


Figure 1. PLOT as a progressive localization engine. The diagram follows one high-level variable, Z_2 in red, though OT localization is performed jointly over all high-level variables and candidate neural sites. PLOT first localizes coarse sites such as tokens/layers, then refines within them to coordinates or PCA spans. The resulting signal can be calibrated into a direct handle or used to guide DAS.

neural sites. In simple tasks, high-mass sites already yield executable handles. In larger models, PLOT applies site discovery progressively, moving from coarse sites such as token positions or layers to finer supports such as coordinate groups or PCA spans. The localized support can be used directly to extract intervention handles or to guide the location and scale of DAS training.

The hierarchy is illustrated in Figure 1. At each stage, PLOT fits a single OT coupling between all target abstract variables and all candidate neural sites at that resolution; the figure follows one row of this coupling, for the variable Z_2 in red, for visualization. In a transformer, the first coupling can be over token-position sites, so different variables may localize to different parts of the sequence. Then the same joint procedure can be applied over layers within the selected tokens, and finally over coordinates or PCA spans within the selected layers. The PCA basis often identifies compact supports that are poorly aligned with canonical coordinates but yield accurate handles. The final localization can then be calibrated into a direct handle or used to guide DAS, which is no longer asked to search globally but trains only in the layer and at the scale selected by PLOT.

Empirically, we explore PLOT on a progression of tasks with increasing localization complexity. On the hierarchical equality benchmark (HEQ) of Geiger et al. (2024), single-stage PLOT over individual MLP coordinates recovers accurate, compact handles much faster than DAS. We next consider 4-bit binary addition with a GRUCell neural backbone, where the goal is to locate internal carry bits whose representations may be distributed or misaligned with the canonical basis. Here, two-stage PLOT first localizes each carry to a recurrent state and then performs within-state site discovery. We find that the PCA basis provides the strongest OT-only intervention handles on this task, while PLOT-guided DAS matches full-DAS accuracy with much lower runtime. Finally, on the multiple-choice question-answering (MCQA) benchmark (Mueller et al., 2025) with Gemma-2-2B (Gemma Team, 2024), we employ three-stage PLOT. It first localizes relevant final-token layers, then extracts native/PCA-based direct handles within those layers, and finally uses them to guide the dimensional scale of DAS. PLOT-guided DAS matches full DAS accuracy while reducing runtime by more than an order of magnitude, with OT-only handles nearly as accurate and almost two orders of magnitude faster.

1.2. Literature Review

A broad line of mechanistic interpretability work seeks to localize internal structures that mediate specific model behaviors, spanning factual associations (Meng et al., 2022), learned features (Cunningham et al., 2023), and algorithmic circuits (Mueller et al., 2025; Polyakov et al., 2025). These methods are often geared toward feature discovery, model editing, or benchmark-specific analysis rather than a general causal correspondence criterion. The causal abstraction framework sharpens this target by asking whether an intervention on a hypothesized high-level variable can be reproduced by an internal intervention in the network (Geiger et al., 2025). Methods developed for this framework include interchange intervention training (IIT) (Geiger et al., 2022), DAS (Geiger et al., 2024), and subsequent methods that automate parts of the DAS search (Wu et al., 2024; Sun et al., 2025). Boundless DAS replaces the manual sweep over subspace dimensionality with learned soft intervention boundaries (Wu et al., 2024), while HyperDAS uses a hypernetwork to automate token-position localization and concept-specific subspace construction (Sun et al., 2025). PLOT instead fits an explicit OT coupling between the abstract variables and candidate neural sites. This coupling can directly define intervention handles, or progressively restrict the search space for local subspace learners such as DAS and its variants. Other related work studies causal variable localization using PCA (Tigges et al., 2023; Marks and Tegmark, 2024), sparse autoencoders (SAEs) (Bricken et al., 2023; Huben et al., 2024), and masking-based selection methods (Chaudhary and Geiger, 2024; Mueller et al., 2025). These methods provide natural ways to select candidate sites inside the network, which can then serve as the neural site family for PLOT.

OT has also been used to align causal models across lev-

els of abstraction. Most notably, causal OT of abstractions (COTA) (Felekis et al., 2024) formulates abstraction between causal models at different granularities as a multi-marginal OT problem (Gangbo and Święch, 1998). COTA enforces causal consistency through do-calculus constraints and intervention-aware transport costs. This setting, which aligns two explicit causal models, differs from PLOT, which uses OT to align a high-level causal model with the internal computation of a neural network. More broadly, our work is also related to causal OT and OT-based distances for causal models (Acciaio et al., 2020; Lassalle, 2018; Cheridito and Eckstein, 2025), though these works do not study neural causal abstraction.

2. Background and Preliminaries

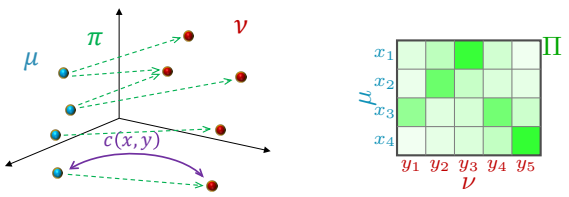
2.1. Causal Abstractions in Mechanistic Interpretability

Causal abstraction in mechanistic interpretability (Pearl, 2009; Geiger et al., 2025) compares an abstract causal model $\mathfrak{C} : \mathcal{X} \to \mathcal{Y}$, with interpretable variables $Z_1, \ldots, Z_m$, to a trained neural model $\mathfrak{N} : \mathcal{X} \to \mathcal{Y}$ that solves the same task. The central question is whether interventions on the abstract variables can be matched by interventions at suitable internal sites of the neural model.

Abstract swap. Given a base/source pair of inputs (x^b, x^s) and an abstract variable Z_i, the corresponding abstract counterfactual output is $y_i^{\text{abs_swap}}(x^b, x^s) := \mathfrak{C}\big(x^b; \text{do}\big(Z_i \leftarrow Z_i(x^s)\big)\big)$. This intervention keeps the base input fixed while replacing only the value of Z_i by its source value.

Neural swap. Let $s_1, \ldots, s_n$ be candidate neural sites (e.g., layers or groups of neurons) to be matched with $Z_1, \ldots, Z_m$, and write $h_j(x)$ for the activation at site s_j given input x. For a base/source pair (x^b, x^s), a neural swap operator at s_j injects source activations at that site and produces the counterfactual $y_j^{\text{nn_swap}}(x^b, x^s) := \mathfrak{N}\big(x^b; h_j \leftarrow \text{Swap}\big(h_j(x^b), h_j(x^s)\big)\big)$. Different methods parametrize Swap differently, ranging from activation patching to weighted or subspace-based swaps.

Correspondence criterion. A correspondence specifies which internal neural sites realize each abstract variable. We represent it by a nonnegative matrix $\Pi \in \mathbb{R}_{\geq 0}^{m \times n}$, where Π_{ij} measures the association strength between Z_i and s_j. The support of the i-th row defines the neural intervention handle for Z_i, possibly involving a single site or a distributed collection. The correspondence is faithful when the neural counterfactual induced by this handle matches the associated abstract counterfactual across relevant base/source pairs: $y_{\Pi,i}^{\text{nn_swap}}(x^b, x^s) \approx y_i^{\text{abs_swap}}(x^b, x^s)$, for all $i \in [m]$. This counterfactual agreement criterion underlies interchange-intervention methods for causal abstraction, including IIT, DAS, and its extensions (Geiger et al., 2022; 2024; 2025; Wu et al., 2024).



(a) A coupling $\pi \in \Pi(\mu, \nu)$ transports mass from μ to ν. *(b)* Coupling matrix Π, representing the OT plan.

Figure 2. Optimal transport geometry & coupling.

2.2. Optimal Transport

OT compares two probability distributions by the minimum cost of moving mass from one to the other. Given $\mu \in \mathcal{P}(\mathcal{X})$, $\nu \in \mathcal{P}(\mathcal{Y})$, and a cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the Kantorovich OT problem is (Villani, 2003; Peyré and Cuturi, 2019)

$$\mathsf{OT}_c(\mu, \nu) := \inf_{\pi \in \Pi(\mu,\nu)} \int c(x, y)\, d\pi(x, y),$$

where $\Pi(\mu, \nu) := \{\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y}) : \pi(\cdot \times \mathcal{Y}) = \mu,\ \pi(\mathcal{X} \times \cdot) = \nu\}$ is the set of all couplings between μ and ν, each specifying how mass under μ is matched to mass under ν. We use the squared Euclidean cost $c(x, y) = \|x - y\|^2$; see Figure 2a. For discrete measures with m and n support points,

the optimal coupling is a nonnegative matrix $\Pi \in \mathbb{R}_{\geq 0}^{m \times n}$, as illustrated in Figure 2b, whose entry $\Pi_{i,j}$ records the mass transported from support point i of μ to support point j of ν. In PLOT, this matrix is the soft correspondence between abstract variables and candidate neural sites.

Entropic optimal transport. For computation, we use entropically regularized OT, which adds a Kullback-Leibler (KL) divergence penalty to the transport cost and makes the problem strictly convex. For a regularization parameter $\varepsilon > 0$, entropic OT (EOT) is given by

$$\mathsf{OT}_c^{\varepsilon}(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int c\, d\pi + \varepsilon \mathsf{KL}(\pi \| \mu \otimes \nu), \qquad (1)$$

where $\mathsf{KL}(\alpha \| \beta) := \int \log(d\alpha/d\beta)\, d\alpha$ if $\alpha \ll \beta$ and $+\infty$ otherwise. In the discrete setting, EOT is computed by Sinkhorn's algorithm with $O(mn)$ cost per iteration (Cuturi, 2013). The unique solution $\pi_\varepsilon^\star$ can be viewed as a smoothed version of an unregularized optimal coupling.

Unbalanced OT. Standard OT enforces both marginals exactly, which can be too rigid when some candidate neural

sites are irrelevant to the target variables. UOT relaxes the strict marginal requirements by penalizing, rather than forbidding, marginal mismatch (Chizat et al., 2018; Peyré and Cuturi, 2019):

$$\mathsf{OT}_c^{\varepsilon,\beta_1,\beta_2}(\mu,\nu) := \inf_{\pi\in\mathcal{P}(\mathcal{X}\times\mathcal{Y})} \int c\,d\pi + \varepsilon\mathsf{KL}(\pi\|\mu\otimes\nu) + \beta_1\mathsf{KL}(\pi_1\|\mu) + \beta_2\mathsf{KL}(\pi_2\|\nu). \tag{2}$$

where π_1 and π_2 are the first and second marginals of π, which need not coincide exactly with μ and ν under the UOT framework. Large β_1, β_2 values recover balanced EOT, while finite values allow some mass to remain unmatched. In our MCQA experiment with Gemma-2-2B, this relaxation becomes important at the coarse layer-localization stage, where UOT can leave irrelevant layers effectively unmatched and allocate mass to layers carrying signal. We use the one-sided version that fixes the abstract marginal $\pi_1 = \mu$ and relaxes only the neural marginal π_2, with parameter $\beta = \beta_2$.

3. Progressive Localization via Optimal Transport

Let $\mathfrak{C}, \mathfrak{N} : \mathcal{X} \to \mathcal{Y}$ denote the causal and neural models. Let $Z_1, \ldots, Z_m$ be the target abstract variables, and $\mathcal{D}_{\text{ft}}, \mathcal{D}_{\text{cal}}, \mathcal{D}_{\text{te}}$ be disjoint fit, calibration, and test banks of base/source pairs. At each localization stage, PLOT has three steps:

1. **Effect signatures:** Use $\mathcal{D}_{\text{ft}}$ to geometrically represent how abstract and neural swap interventions change model outputs. These changes are termed *effect signatures*.
2. **Transport matching:** Fit an OT coupling (or UOT, when irrelevant sites should remain unmatched) between all abstract and neural effect signatures at the current resolution. This yields a soft correspondence between abstract variables and neural sites.
3. **Handle extraction or refinement:** Use $\mathcal{D}_{\text{cal}}$ to convert the coupling into intervention handles, or to define a smaller support for the next PLOT stage or for guiding a local method such as DAS.

The extracted handles are tested on $\mathcal{D}_{\text{te}}$; see Appendix A for the full calibration and testing protocols. PLOT may be applied once or progressively, depending on the complexity of the localization task. We first delineate a single PLOT step and then discuss progressive localization.

3.1. Single PLOT Step

Causal effect signatures. As in Section 2.1, for each pair $(x_t^b, x_t^s) \in \mathcal{D}_{\text{ft}}$ and variable Z_i, the abstract swap and base outputs are $y_{i,t}^{\text{abs_swap}} = \mathfrak{C}\big(x_t^b; \text{do}(Z_i \leftarrow Z_i(x_t^s))\big)$ and $y_t^{\text{abs}} = \mathfrak{C}(x_t^b)$, respectively. For a featurizer $\phi : \mathcal{Y} \to \mathbb{R}^p$ that maps outputs to a shared feature space, the *causal effect signature* is

$$\Delta_{i,t}^{\text{abs}} := \phi\big(y_{i,t}^{\text{abs_swap}}\big) - \phi\big(y_t^{\text{abs}}\big), \quad i = 1, \ldots, m, \qquad t = 1, \ldots, T_{\text{ft}}.$$

Neural effect signatures. Choose candidate sites $s_1, \ldots, s_n$, each comprising intervention directions inside some activation vector. For site s_j, let $a_j(x) \in \mathbb{R}^{d_j}$ be the activation vector for that site, and let $W_j \in \mathbb{R}^{d_j \times r_j}$ have orthonormal columns spanning the intervention directions. Define

$$h_j(x) := W_j^\intercal\big(a_j(x) - \bar{a}_j\big) \in \mathbb{R}^{r_j}, \qquad j = 1, \ldots, n,$$

where $\bar{a}_j$ is the empirical mean of $a_j(x)$ over $\mathcal{D}_{\text{ft}}$. In the canonical basis, W_j consists of identity columns, so s_j is an individual coordinate or a group. In the PCA basis, W_j consists of PCA directions fitted to the corresponding activation vector. Given (x_t^b, x_t^s), the neural swap at s_j replaces the base site coordinates by the source coordinates while leaving the orthogonal complement fixed:

$$a_j \leftarrow a_j(x_t^b) + W_j\big(h_j(x_t^s) - h_j(x_t^b)\big).$$

Writing the output as $y_{j,t}^{\text{nn_swap}}$ and the factual output as $y_t^{\text{nn}} := \mathfrak{N}(x_t^b)$, the *neural effect signature* is

$$\Delta_{j,t}^{\text{nn}} := \phi\big(y_{j,t}^{\text{nn_swap}}\big) - \phi\big(y_t^{\text{nn}}\big), \quad j = 1, \ldots, n, \qquad t = 1, \ldots, T_{\text{ft}}.$$

This covers single/group neuron swaps, full-vector patching, and PCA-component/span swaps.

Fitting the transport matching. We pool effect signatures over fit pairs to obtain one signature per variable/site:

$$u_i := \text{Agg}\big(\Delta_{i,1}^{\text{abs}}, \ldots, \Delta_{i,T_{\text{ft}}}^{\text{abs}}\big), \quad v_j := \text{Agg}\big(\Delta_{j,1}^{\text{nn}}, \ldots, \Delta_{j,T_{\text{ft}}}^{\text{nn}}\big).$$

In our experiments, Agg is taken as stacking, optionally followed by normalization. These signatures define empirical measures $\mu := \frac{1}{m}\sum_{i=1}^m \delta_{u_i}$ and $\nu := \frac{1}{n}\sum_{j=1}^n \delta_{v_j}$, between which we compute the EOT coupling $\Pi_\varepsilon^\star \in \mathbb{R}_{\geq 0}^{m\times n}$ via Sinkhorn's algorithm (Cuturi, 2013). When the site family contains many irrelevant candidates, we instead use the UOT coupling from (2). Slightly abusing notation, write Π for the fitted EOT or UOT coupling, whose i-th row $[\Pi]_{i:} \in \mathbb{R}^n$ gives the soft correspondence from Z_i to $s_1, \ldots, s_n$.

Calibration and handle extraction. To turn $[\Pi]_{i:}$ into an executable intervention handle for Z_i, we keep the top-K highest-mass neural sites and renormalize to obtain the

intervention weights:

$$\widetilde{\Pi}_{j|i} := \frac{[\Pi]_{ij}\mathbb{1}_{\{j\in \mathrm{TopK}_K(i)\}}}{\sum_{j'}[\Pi]_{ij'}\mathbb{1}_{\{j'\in \mathrm{TopK}_K(i)\}}}, \qquad j = 1, \ldots, n.$$

The corresponding neural intervention for Z_i swaps the selected sites from source into base with weights $\widetilde{\Pi}_{j|i}$ and intervention strength $\lambda > 0$. The calibration set $\mathcal{D}_{\mathrm{cal}}$ is used only to select the handle parameters K and λ. The OT coupling itself remains fixed after fitting on $\mathcal{D}_{\mathrm{ft}}$.

We note that there are other possible choices to form a possible coupling Π; OT was chosen simply due to its principled framework and natural interpretation. We compare OT against several other coupling alternatives in Appendix A.3, where we find that OT performs similarly or better than other methods in accuracy and runtime.

3.2. Progressive Localization

PLOT becomes hierarchical by composing the OT step across nested site families; see Figure 1. Conceptually, PLOT acts as a "zoom" mechanism for causal abstraction, with each stage either yielding executable handles or restricting the search for the next, more local stage. At stage r, let $\mathcal{S}^{(r)}$ be the current candidate site family, such as token positions, layers, recurrent states, coordinate blocks, or PCA spans. PLOT fits a coupling $\Pi^{(r)}$ between the target variables $Z_1, \ldots, Z_m$ and sites in $\mathcal{S}^{(r)}$, using UOT when broad site families contain many irrelevant candidates (e.g., the coarse layer-localization stage of MCQA; see Section 4.3). Each row $[\Pi^{(r)}]_{i:}$ can be calibrated into a handle for Z_i or used to define a smaller site family $\mathcal{S}_i^{(r+1)}$ inside the high-mass support for the next localization stage. This yields the progressive pipeline:

$$\text{coarse sites} \longrightarrow \text{refined site family} \longrightarrow$$
$$\text{localized support or span} \longrightarrow \text{intervention handle.}$$

At the final stage, the calibrated PLOT handle can be used directly, or the localization signal can guide DAS. In PLOT-guided DAS, OT performs global localization while DAS learns the invariant subspace using the layer or dimension scale selected by PLOT.

4. Experimental Settings and Results

Our experiments are organized by increasing localization depth: (i) HEQ, where a single-stage PLOT recovers accurate intervention handles; (ii) binary addition, where two-stage PLOT localizes each carry to a recurrent state and then refines within that state using PCA or DAS; and (iii) MCQA, where multi-stage PLOT localizes final-token layers, extracts native or PCA-based handles within them, and guides the layer and dimensional scale of DAS. See Appendix B for full experiment details.

4.1. Hierarchical Equality (HEQ)

As a first single-stage test of PLOT, we consider the HEQ benchmark from Geiger et al. (2024), where the goal is to determine whether two pairs of objects have the same equality relation. The task is small enough that direct transport over individual neurons recovers accurate handles, so no progressive refinement is needed. This setting also isolates the speed advantage of PLOT over DAS.

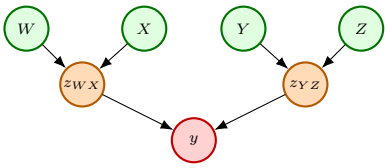


Figure 3. Causal model for HEQ.

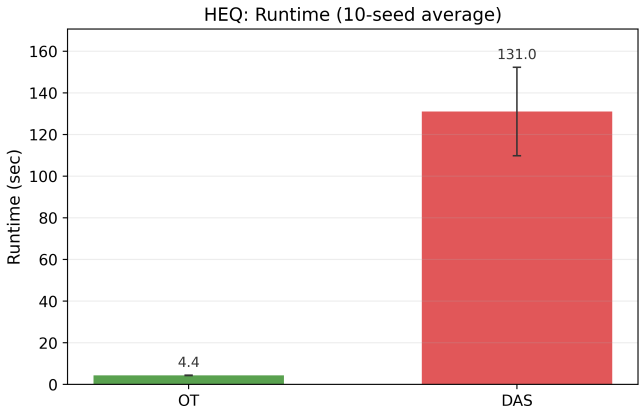
Causal model. As in Figure 3, inputs $W, X, Y, Z \in [100]$ define equality variables $z_{WX} = \mathbb{1}_{\{W=X\}}$ and $z_{YZ} = \mathbb{1}_{\{Y=Z\}}$. The output $y = \mathfrak{C}(W, X, Y, Z) = \mathbb{1}_{\{z_{WX}=z_{YZ}\}}$ is 1 precisely when both pairs are equal or both are unequal. For example, $\mathfrak{C}(1, 1, 2, 2) = \mathfrak{C}(1, 2, 3, 4) = 1$, while $\mathfrak{C}(1, 1, 1, 2) = 0$.

Neural model. We encode each tuple $(W, X, Y, Z) \in [100]^4$ using a fixed embedding map $e : [100] \to \mathbb{R}^4$ and concatenate the embeddings into $x = [e(W)\ e(X)\ e(Y)\ e(Z)] \in \mathbb{R}^{16}$. The factual model is a ReLU MLP $\mathfrak{N} : \mathbb{R}^{16} \to \mathbb{R}^2$ with three hidden layers of width 16, resulting in 850 trainable parameters. We denote the activation vector at layer ℓ by $h^{[\ell]} = \left(h_j^{[\ell]}\right)_{j=1}^{16} \in \mathbb{R}^{16}$, for $\ell = 1, 2, 3$. The MLP is trained to achieve 100% accuracy on a holdout validation set; see Appendix B.1.

Single-stage PLOT and intervention handles. We generate disjoint counterfactual pair banks $\mathcal{D}_{\mathrm{ft}}, \mathcal{D}_{\mathrm{cal}}, \mathcal{D}_{\mathrm{te}}$, each of size 1000, for fitting, calibration, and testing. The abstract variables are $Z_1 = z_{WX}$ and $Z_2 = z_{YZ}$, while the candidate sites are all $3 \times 16 = 48$ individual hidden neurons, $s_{\ell,j} = h_j^{[\ell]}$, for $\ell = 1, 2, 3$ and $j = 1, \ldots, 16$. We construct effect signatures as output-space changes over $\mathcal{D}_{\mathrm{ft}}$: neural site signatures use changes in predicted output probabilities via $\phi(y) = \mathrm{softmax}(y)$, while abstract-variable signatures directly use one-hot differences. We fit the OT coupling with Sinkhorn's algorithm at $\varepsilon = 4$. The coupling is converted into soft intervention handles by calibrating the top-K support size, $K \in \{1, \ldots, 20\}$, and intervention strength, $\lambda \in \{1, \ldots, 80\}$, on $\mathcal{D}_{\mathrm{cal}}$, which is balanced between Z_{WX}-only and Z_{YZ}-only sensitive pairs (see Appendix A.2). As a baseline, DAS trains rotations on $\mathcal{D}_{\mathrm{ft}}$ over layers $\ell = 1, 2, 3$ and subspace dimensions $\{1, \ldots, 16\}$; the best DAS handle is selected on $\mathcal{D}_{\mathrm{cal}}$.

Metric	PLOT	DAS
z_{WX} sensitivity	0.989 ± 0.014	0.989 ± 0.008
z_{WX} invariance	0.993 ± 0.007	1.000 ± 0.000
z_{YZ} sensitivity	0.991 ± 0.012	0.992 ± 0.005
z_{YZ} invariance	0.989 ± 0.010	1.000 ± 0.000
Avg. exact	0.991 ± 0.008	0.995 ± 0.004
Runtime (s)	4.4 ± 0.1	131.0 ± 22.4

(a) Accuracy and runtime summary.



(b) End-to-end runtime.

Figure 4. HEQ comparison over 10 seeds. Values are mean ± standard deviation. Runtime excludes factual-model training and includes fitting, calibration, and test evaluation.

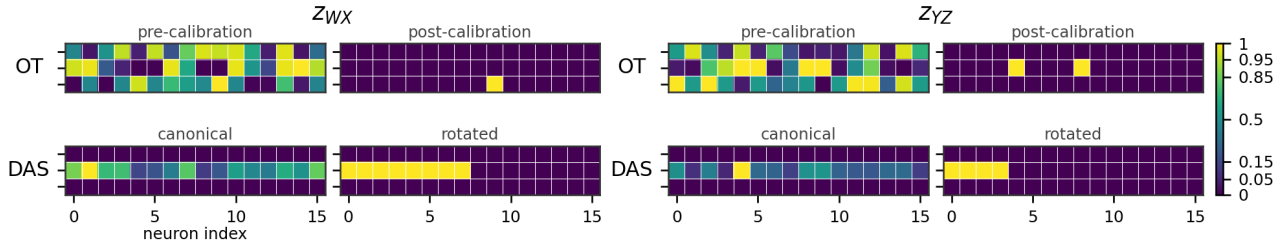


Figure 5. HEQ site-level intervention handles learned by OT (before and after calibration) and DAS (in the canonical and rotated basis). The two high-level variables are shown side by side.

Results and learned handles. The table in Figure 4 reports exact counterfactual accuracies for z_{WX} and z_{YZ}, together with their average. For each abstract variable, we separately evaluate sensitivity, on pairs that change that variable, and invariance, on pairs that leave it fixed. Both single-stage PLOT and DAS recover accurate handles on this benchmark. DAS is slightly more accurate, with average exact score 0.995 compared to 0.991 for PLOT. The larger difference is computational: PLOT takes 4.4 seconds on average, while DAS takes 131 seconds, making it roughly $30\times$ slower end-to-end. Runtime includes OT coupling fitting or DAS rotation training, hyperparameter sweeps, calibration, and final counterfactual evaluation, but excludes factual-model training. Hardware details are in Appendix B.1.

A structural difference between the methods is shown in Figure 5. DAS is fit layer by layer, whereas PLOT fits one transport problem over all candidate neurons simultaneously, so its pre-calibration mass can spread across multiple layers. Calibration then turns this diffuse profile into a sparse top-K handle. In contrast, DAS reaches similar performance through a learned rotated subspace. Thus, on HEQ, PLOT obtains compact handles directly over the original hidden neurons while achieving nearly the same counterfactual accuracy as DAS. Further layerwise summaries, OT parameter sensitivity, and learned intervention sizes are reported in Appendix B.1.

4.2. Binary Addition

HEQ is simple enough that single-stage PLOT over individual neurons works well. Binary addition with a GRUCell-based recurrent network is harder, because the carry bits are latent variables whose neural implementation is distributed across recurrent states. We therefore use a two-stage PLOT pipeline, where OT first localizes each carry to one hidden state, from which a handle is extracted by applying OT in the canonical (PLOT-NATIVE) or the PCA basis (PLOT-PCA), or by training DAS inside that state (PLOT-DAS).

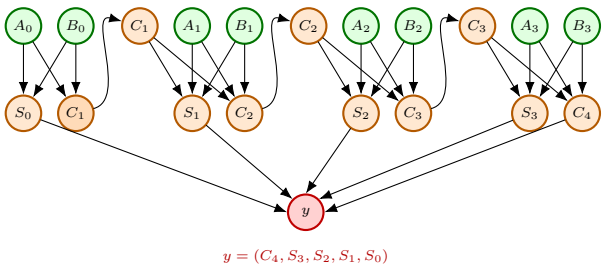


Figure 6. Causal model for 4-bit ripple-carry addition.

Causal model. We consider 4-bit binary addition with inputs $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$, as shown in Figure 6.

The abstract model is the standard ripple-carry adder, with $S_0 = (A_0 + B_0) \bmod 2$, $C_1 = \lfloor (A_0 + B_0)/2 \rfloor$, and for $i = 1, 2, 3$, $S_i = (A_i + B_i + C_i) \bmod 2$ and $C_{i+1} = \lfloor (A_i + B_i + C_i)/2 \rfloor$. The output is $y = (C_4, S_3, S_2, S_1, S_0)$, and we report intervention accuracy for the internal carry variables C_1, C_2, C_3.

Neural model. The neural model reads one bit-pair (A_i, B_i) at a time, from least to most significant, and updates a GRUCell state $h_\ell \in \mathbb{R}^d$ after timestep ℓ, for $\ell = 0, 1, 2, 3$; see Appendix B.2 for details. A linear readout then predicts the five output bits $(C_4, S_3, S_2, S_1, S_0)$. We use two GRUCell hidden dimensions, $d \in \{8, 16\}$, both trained to fit the full 4-bit truth table exactly. With this indexing, h_ℓ is the natural coarse neural site for carry $C_{\ell+1}$.

Two-stage PLOT and intervention handles. We use the same output effect pipeline as in HEQ, but with binary-addition-specific pair banks and recurrent candidate sites. Following the MIB Arithmetic benchmark (Mueller et al., 2025), we construct structured arithmetic base/source banks with operand flips and carry-targeting sources, then split them into fit, calibration, and test banks; see Appendix B.2. In Stage A, the abstract variables are the carries C_1, C_2, C_3, and the neural sites are the recurrent states $h_0, \ldots, h_3$. Fitting OT between their effect signatures localizes each carry to a timestep.

Stage B extracts a handle inside the localized state. PLOT-NATIVE runs OT over coordinate/neuron groups in that state, calibrating over resolutions $r \in \{1, 2\}$ and top-K values. PLOT-PCA fits a centered PCA basis to fit-bank hidden activations at each timestep, then runs OT over top-prefix principal-component groups. PLOT-DAS trains DAS on the full hidden state selected in Stage A. We compare to full DAS, which trains over all recurrent timesteps and calibrates to select the best handle.

Results and learned handles. The table in Figure 7a compares PLOT-NATIVE, PLOT-PCA, PLOT-DAS, and full DAS on the two GRU widths. Moving from canonical coordinates to the PCA basis gives the strongest OT-only handles, with PLOT-PCA raising accuracy from 0.823 to 0.937 at $d = 8$, and from 0.832 to 0.941 at $d = 16$. PLOT-DAS reaches DAS-level accuracy, with 0.974 at both widths, compared to 0.976 and 0.984 for full DAS. The main gain is again in runtime, with PLOT-DAS reducing end-to-end cost from 95.6s to 44.2s at $d = 8$, and from 210.5s to 64.2s at $d = 16$. Notably, the runtime gap widens as the model size and candidate site family grow, a trend that becomes more pronounced in MCQA; see next subsection.

Figure 8 visualizes the learned handles for a representative $d = 16$ run. Stage A localizes the carries to the expected recurrent states, with C_1, C_2, C_3 mapping to h_0, h_1, h_2. Within those states, PLOT-NATIVE and PLOT-PCA produce OT-based handles, with PLOT-PCA concentrating mass on a small prefix of principal components. Full DAS searches over all timesteps, but its final handles match those found by PLOT-DAS when DAS is restricted to the Stage A timestep. Thus, the guided method reaches essentially the same DAS solution while avoiding the full recurrent-state search.

4.3. Multiple-Choice Question Answering (MCQA)

We next evaluate PLOT in the substantially larger setting of the MCQA benchmark from MIB (Mueller et al., 2025), where Gemma-2-2B answers natural-language multiple-choice prompts. The model reads a color fact and query, identifies the answer containing the queried color, and outputs its symbol. For example, given *"The sky is blue. What color is the sky? (A) red (B) blue (C) green (D) yellow"*, the correct output is B. We use a progressive hierarchy to localize the abstract variables inside the transformer, where PLOT identifies relevant final-token layers, within-layer OT extracts native or PCA-based handles, and optional DAS uses this signal to restrict its dimension search.

Causal model. Each input is a natural-language prompt x containing a color fact, a query, and four answer choices. Let Col_i denote the color at choice position i, and Sym_i be the corresponding answer symbol, e.g., A, B, C, D. The abstract model computes the queried color $q(x)$, the answer pointer $\mathrm{AP} = \arg\mathrm{unique}_{i \in \{1,2,3,4\}}\{\mathrm{Col}_i = q(x)\}$, and the answer token $\mathrm{AT} = \mathrm{Sym}_{\mathrm{AP}}$; see Figure 9. The abstract variables of interest are AP and AT, corresponding to *AnswerPointer* and *Answer* in MIB (Mueller et al., 2025). We evaluate counterfactuals that change AP, AT, or both; see Appendix B.3.

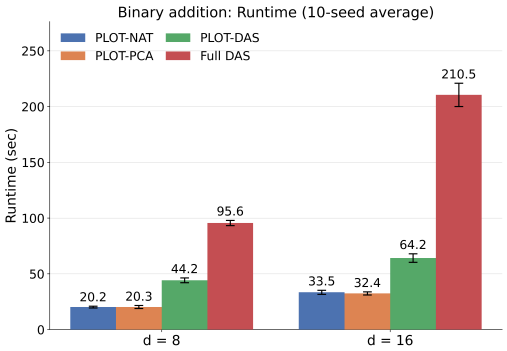
Neural model. We use Gemma-2-2B (Gemma Team, 2024), a decoder-only transformer with 26 residual-stream layers and hidden width 2304. The model receives the full MCQA prompt x and is evaluated by its next-token distribution over answer symbols.

Three-stage PLOT. We use the same output effect pipeline with signatures modified to capture alphabetic tokens relevant to the task, and with fit, calibration, and test banks constructed from the MIB MCQA counterfactual families (see Appendix B.3 for more details). Stage A performs coarse layer localization over final-token residual-stream vectors, treating each layer as a neural site. At this broad stage, some Gemma-2-2B layers need not realize either AP or AT directly, so we use UOT to avoid forcing irrelevant layers to absorb transport mass. The UOT coupling proposes candidate layers, and calibration selects one layer per abstract variable.

Stage B refines inside the selected layer using transport over

Method	$d = 8$		$d = 16$	
	Accuracy	Runtime (s)	Accuracy	Runtime (s)
PLOT-NATIVE	0.823 ± 0.040	$\mathbf{20.2 \pm 0.8}$	0.832 ± 0.027	33.5 ± 1.8
PLOT-PCA	0.937 ± 0.024	20.3 ± 1.3	0.941 ± 0.028	$\mathbf{32.4 \pm 1.4}$
PLOT-DAS	0.974 ± 0.019	44.2 ± 2.1	0.974 ± 0.013	64.2 ± 3.8
Full DAS	$\mathbf{0.976 \pm 0.018}$	95.6 ± 2.4	$\mathbf{0.984 \pm 0.009}$	210.5 ± 10.4

(a) Internal-carry exact accuracy and runtime summary.



(b) End-to-end runtime.

Figure 7. Binary-addition comparison over 10 seeds. Values are mean ± standard deviation. Accuracy is averaged over the internal carry variables C_1, C_2, C_3, and runtime excludes backbone training.

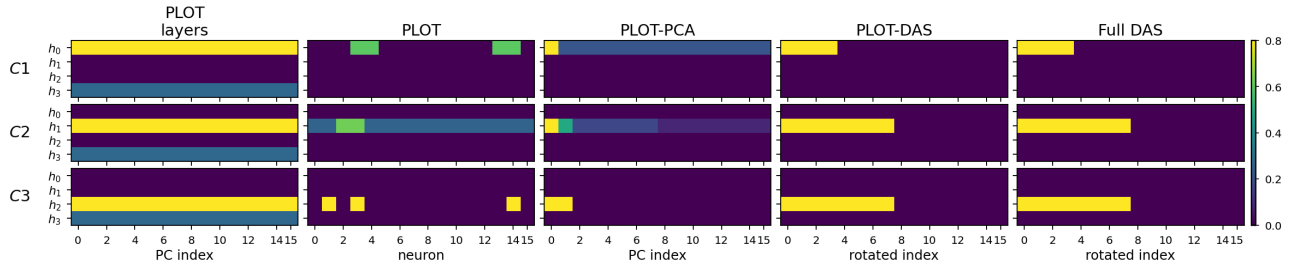


Figure 8. Binary-addition handles for $d = 16$. Rows show carries C_1, C_2, C_3, while columns show post-calibration PLOT handles and PCA/DAS handles in their rotated bases.

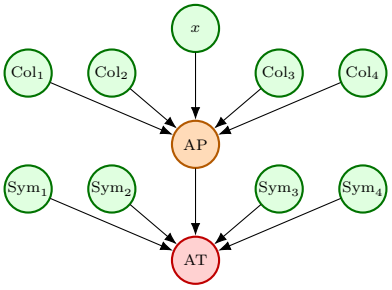


Figure 9. MCQA causal model, with abstract variables AP and AT.

native coordinate blocks or PCA spans. Calibrating top-K and λ on $\mathcal{D}_{\text{cal}}$ yields direct two-stage handles, reported as PLOT-NATIVE and PLOT-PCA. The calibrated handle sizes define the dimension hints used in Stage C. PLOT-NATIVE-DAS and PLOT-PCA-DAS train DAS inside the Stage A layer with a subspace dimension grid restricted around the native- or PCA-hinted dimension. In contrast, PLOT-DAS uses Stage A only and trains DAS on the selected layer with the standard dimension grid. PLOT in MCQA can thus provide direct handles, select layers for DAS, or guide its dimensional scale inside a chosen layer.

Results. Table 1 shows PLOT and PLOT-guided DAS results on MCQA across four seeds. Full DAS reaches 0.9310 average exact accuracy, but requires a sweep over all final-token layers and DAS subspace dimensions, taking 13107.6s. In contrast, PLOT-DAS uses only Stage A layer selection and reaches 0.9313 average exact in 1148.6s, matching full DAS accuracy while avoiding the full layer sweep. The substantial runtime gap comes from PLOT providing the layer localization that DAS otherwise has to search for. Stage B gives another way to trade accuracy for runtime. PLOT-NATIVE-DAS uses the native PLOT handle size as a dimension hint for DAS, reaching 0.9187 average exact in 346.1s. This is faster than PLOT-DAS because DAS now searches fewer dimensions, although the hinted dimension is currently not as fine as that found by the broader PLOT-DAS sweep.

Direct PLOT handles are also strong and much faster. PLOT-NATIVE reaches 0.9113 average exact in 204.6s, within two percentage points of full DAS while using no learned rotations and running roughly 65× faster. Moreover, PLOT-NATIVE does not simply intervene on the full layer, but selects handles well below full-layer width, indicating meaningful within-layer localization. PLOT-PCA and PLOT-PCA-DAS select smaller PCA-dimensional handles, especially for AP, but currently trade off AT accuracy. Overall, MCQA shows that PLOT can provide useful direct handles, localize layers for DAS, and guide the DAS dimensional scale in a large transformer. Further method details are in Appendix B.3.

Table 1. MCQA results over four seeds. Accuracy is exact counterfactual accuracy for AP and AT. Layer/dim columns show one representative seed.

Method	AP	AT	Avg.	Runtime	AP layer/dim	AT layer/dim
PLOT-NATIVE	0.8375 ± 0.0192	$\mathbf{0.9850 \pm 0.0050}$	0.9113 ± 0.0074	204.6 ± 89.4s	L18/768	L24/576
PLOT-PCA	0.8550 ± 0.0229	0.8850 ± 0.0461	0.8700 ± 0.0269	$\mathbf{188.0 \pm 4.0}$s	L18/8	L24/96
PLOT-DAS	0.8825 ± 0.0148	0.9800 ± 0.0071	$\mathbf{0.9313 \pm 0.0065}$	1148.6 ± 31.7s	L18/128	L24/2304
PLOT-NATIVE-DAS	0.8575 ± 0.0238	0.9800 ± 0.0071	0.9187 ± 0.0108	346.1 ± 88.4s	L18/1152	L24/2304
PLOT-PCA-DAS	$\mathbf{0.9075 \pm 0.0238}$	0.8225 ± 0.0148	0.8650 ± 0.0146	455.2 ± 38.0s	L18/32	L24/120
Full DAS	0.8950 ± 0.0160	0.9680 ± 0.0060	0.9310 ± 0.0070	13107.6 ± 747.7s	L17/96	L24/2304

5. Discussion and Concluding Remarks

We proposed PLOT, a progressive localization engine for causal abstractions via OT. PLOT fits an OT coupling between abstract and neural output effect signatures, then uses the high-mass support to move from coarse site families to finer ones. In simple settings, this coupling directly yields intervention handles. In larger models, the same progressive procedure can either extract multi-stage OT-only handles or restrict where and at what scale local subspace methods such as DAS are trained. Across HEQ, binary addition, and MCQA, transport-only PLOT handles are accurate and remarkably fast, while PLOT-guided DAS matches full-DAS accuracy and remains an order of magnitude faster. While this work lays the foundation for OT-based methods in mechanistic interpretability, the methodology can be improved along several axes. These include sharper PLOT localization rules that exploit finer resolutions, hierarchical effect-signature constructions, and PLOT-guided DAS methods that receive detailed support information rather than only dimensional guidance. Scaling PLOT to larger models will require engineering around batching, activation caching, approximate signatures, and distributed transport computation (Wang and Goldfeld, 2026; Tsur and Goldfeld, 2025). With appropriate parallelization, PLOT can enable systematic causal abstraction research in large neural models with progressively localizing counterfactual tools.

References

Beatrice Acciaio, Julio Backhoff-Veraguas, and Anastasiia Zalashko. Causal optimal transport and its links to enlargement of filtrations and continuous-time stochastic optimization. *Stochastic Processes and their Applications*, 130(5):2918–2953, 2020. doi: 10.1016/j.spa.2019.08.009. URL https://doi.org/10.1016/j.spa.2019.08.009.

T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Maheep Chaudhary and Atticus Geiger. Evaluating open-source sparse autoencoders on disentangling factual knowledge in GPT-2 small. *CoRR*, abs/2409.04478, 2024. doi: 10.48550/arXiv.2409.04478. URL https://arxiv.org/abs/2409.04478.

Patrick Cheridito and Stephan Eckstein. Optimal transport and Wasserstein distances for causal models. *Bernoulli*, 31(2):1351–1376, 2025. doi: 10.3150/24-BEJ1773. URL https://doi.org/10.3150/24-BEJ1773.

Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

Hoagy Cunningham, Aidan Ewart, Logan Riggs Smith, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *CoRR*, abs/2309.08600, 2023. doi: 10.48550/arXiv.2309.08600. URL https://arxiv.org/abs/2309.08600.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Yorgos Felekis, Fabio Massimo Zennaro, Nicola Branchini, and Theodoros Damoulas. Causal optimal transport of abstractions. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pages 462–498, 2024. URL https://proceedings.mlr.press/v236/felekis24a.html.

Wilfrid Gangbo and Andrzej Święch. Optimal maps for the multidimensional Monge–Kantorovich problem. *Communications on Pure and Applied Mathematics*, 51(1): 23–45, 1998.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah D. Goodman, and

Christopher Potts. Inducing causal structure for interpretable neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7324–7338, 2022. URL https://proceedings.mlr.press/v162/geiger22a.html.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pages 160–187, 2024. URL https://proceedings.mlr.press/v236/geiger24a.html.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah D. Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025.

Gemma Team. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024. doi: 10.48550/arXiv.2408.00118. URL https://arxiv.org/abs/2408.00118.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2017. URL https://arxiv.org/abs/1412.6980.

Rémi Lassalle. Causal transport plans and their Monge–Kantorovich problems. *Stochastic Analysis and Applications*, 36(3):452–484, 2018. doi: 10.1080/07362994.2017.1422747. URL https://doi.org/10.1080/07362994.2017.1422747.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=aajyHYjjsk.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, et al. MIB: A mechanistic interpretability benchmark. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 45069–45108, 2025. URL https://proceedings.mlr.press/v267/mueller25a.html.

Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.

Gabriel Peyré and Marco Cuturi. *Computational optimal transport*. Now Publishers, 2019.

Gregory Polyakov, Christian Hepting, Carsten Eickhoff, and Seyed Ali Bahrainian. Interpretability analysis of arithmetic in-context learning in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1758–1777, 2025. doi: 10.18653/v1/2025.emnlp-main.92. URL https://aclanthology.org/2025.emnlp-main.92/.

Jiuding Sun, Jing Huang, Sidharth Baskaran, Karel D'Oosterlinck, Christopher Potts, Michael Sklar, and Atticus Geiger. HyperDAS: Towards automating mechanistic interpretability with hypernetworks. *CoRR*, abs/2503.10894, 2025. URL https://arxiv.org/abs/2503.10894.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *CoRR*, abs/2310.15154, 2023. doi: 10.48550/arXiv.2310.15154. URL https://arxiv.org/abs/2310.15154.

Dor Tsur and Ziv Goldfeld. Neural entropic multi-marginal optimal transport and Gromov–Wasserstein alignment. *CoRR*, abs/2506.00573, 2025. doi: 10.48550/arXiv.2506.00573. URL https://arxiv.org/abs/2506.00573.

Cédric Villani. *Topics in optimal transportation*, volume 58 of *Graduate Studies in Mathematics*. American Mathematical Society, 2003.

Tao Wang and Ziv Goldfeld. Neural entropic optimal transport and Gromov–Wasserstein alignment. *IEEE Transactions on Information Theory*, 2026. doi: 10.1109/TIT.2026.3661439. URL https://doi.org/10.1109/TIT.2026.3661439. Early access.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah D. Goodman. Interpretability at scale: Identifying causal mechanisms in Alpaca. *CoRR*, abs/2305.08809, 2024. URL https://arxiv.org/abs/2305.08809.

Appendix

A. Additional Methodology Details

We evaluate learned handles by interchange-intervention accuracy (Geiger et al., 2024). For a handle associated with Z_i, calibration accuracy is

$$\mathcal{L}_{\text{cal}}(Z_i) \coloneqq \frac{1}{T_{\text{cal}}} \sum_{t=1}^{T_{\text{cal}}} \mathbb{1}\left[y_{j,t}^{\text{nn_swap}} = y_{i,t}^{\text{abs_swap}}\right]$$

with test accuracy defined analogously on $\mathcal{D}_{\text{te}}$. Note that the equals sign inside of the indicator is a heuristic and is task-dependent. For example, with an LLM output, it is satisfied if the token corresponding to the argmax of the output logits $y^{\text{nn_swap}}$ equals the one-hot token $y^{\text{abs_swap}}$.

A.1. PLOT Calibration Details

Recall that keeping the top-K highest-mass neural sites for the abstract variable Z_i gives the renormalized coupling

$$\widetilde{\Pi}_{j|i} \coloneqq \frac{[\Pi]_{ij}\mathbb{1}_{\{j \in \text{TopK}_K(i)\}}}{\sum_{j'}[\Pi]_{ij'}\mathbb{1}_{\{j' \in \text{TopK}_K(i)\}}}, \qquad j = 1, \ldots, n.$$

The soft intervention handle for Z_i intervenes on all top-K sites simultaneously, with weights $\widetilde{\Pi}_{j|i}$ and intervention strength λ. First, suppose the selected sites span one activation vector $a(x) \in \mathbb{R}^d$. Applied to a calibration or test pair $(x_t^b, x_t^s) \in \mathcal{D}_{\text{cal}}$, the handle is

$$a \leftarrow a(x_t^b) + \lambda \sum_{j \in \text{TopK}_K(i)} \widetilde{\Pi}_{j|i} \cdot W_j\big(h_j(x_t^s) - h_j(x_t^b)\big)\,.$$

Thus, each selected site performs its usual neural swap through h_j and W_j, weighted by the calibrated transport mass and the global strength parameter λ.

If the top-K sites span multiple activation vectors or layers, we apply the soft intervention in forward-pass order. At the earliest intervened layer, we update all selected sites in that layer, continue the forward pass, and repeat at each later layer containing selected sites. For later layers, the "base" activation is the current activation produced by the already-intervened forward pass, so earlier interventions are preserved.

For each calibration pair, we apply the soft intervention and evaluate counterfactual accuracy. We calibrate by sweeping K and λ, then selecting the pair (K, λ) with the best accuracy on $\mathcal{D}_{\text{cal}}$.

A.2. Test Set Details

After calibrating K and λ for each abstract variable Z_i, we evaluate on separate holdout test sets $\mathcal{D}_{\text{te}}$:

- **Z_i-sensitivity set:** containing pairs (x^b, x^s) for which x^s changes the value of Z_i relative to x^b.
- **Z_i-invariance set:** containing pairs (x^b, x^s) for which x^s leaves the value of Z_i fixed.

We evaluate each calibrated handle separately on both sets, checking both sensitivity to interventions that change Z_i and invariance to interventions that leave Z_i unchanged. This differs from prior works (Geiger et al., 2024; Wu et al., 2024; Mueller et al., 2025) which mixed sensitive and invariant pairs, leading to test accuracies that are less interpretable. Indeed, a method that learns no handle can score perfectly on invariance-heavy test sets, yielding high average accuracy while being weak on sensitivity. Unless stated otherwise, main-text accuracies average over all $2m$ sensitive and invariant test sets, two for each variable $Z_1, \ldots, Z_m$.

A.3. Non-OT Coupling Alternatives

Although OT was chosen as the primary coupling method throughout the paper due to its natural compatibility in solving the global matching problem between abstract variables and neural sites, we propose and compare two non-OT alternatives for defining the coupling Π:

- **Cosine coupling:** collect effect signatures as normal, then define Π using the cosine similarity of the effect signatures,

  $$\Pi_{ij} \coloneqq \frac{u_i^\intercal v_j}{\|u_i\|_2 \|v_j\|_2}$$

  and take the softmax of each row.
- **Brute-force coupling:** for each abstract variable i and each neural site j, directly compute their alignment quality using the interchange intervention accuracy on $\mathcal{D}_{\text{ft}}$:

  $$\Pi_{ij} \coloneqq \frac{1}{T_{\text{ft}}} \sum_{t=1}^{T_{\text{ft}}} \mathbb{1}\left[y_{j,t}^{\text{nn_swap}} = y_{i,t}^{\text{abs_swap}}\right],$$

  and take the softmax of each row.

Note that the OT and cosine couplings each take $\mathcal{O}(nT_{\text{ft}})$ neural interventions to construct Π, while the brute-force coupling takes $\mathcal{O}(mnT_{\text{ft}})$. This implies that as the number of causal variables in the task scales, brute-force will slow down significantly. However, we don't yet observe a significant difference here due to the small size of our causal models and the relatively longer time it takes to sweep calibration parameters compared to forming the coupling.

Table 2 displays a comparison between OT, cosine, and brute-force coupling methods on binary addition ($d = 16$) and MCQA tasks. We omit HEQ, for which we didn't observe any meaningful differences. For MCQA, we compare

the coupling methods on their within-layer localization abilities; we standardize the comparison by first performing Stage A coarse layer localization via UOT for all the methods, then proceed with comparing Stage B native handle extraction for each method. Note that Table 2 only reports each method's Stage B runtime.

For binary addition, OT slightly outperforms cosine and brute-force in accuracy and runtime. OT runtime is comparable with cosine, and brute-force lags behind due to the slightly more expensive coupling cost. For MCQA, OT performs the best for recovering AT, but cosine and brute-force outperform OT for recovering AP. Averaging the variables, the methods have comparable accuracies, with the runtime cost of brute-force being more apparent here.

B. Additional Task Descriptions and Results

B.1. Hierarchical Equality (HEQ)

Neural backbone training. We train the MLP with cross-entropy loss using Adam (Kingma and Ba, 2017), learning rate 10^{-3}, and batch size 1024 for three epochs on 1,048,576 factual examples, matching the setup in Geiger et al. (2024). The model reaches exact accuracy 1.0 on a disjoint validation set of size 10,000.

Pair bank construction. We construct the HEQ fit, calibration, and test banks as follows:

- **Fit bank:** 1000 pairs from a mixed policy with roughly 50% pairs sensitive in either z_{WX} or z_{YZ}, and 50% invariant in both.
- **Calibration bank:** 1000 pairs, half only z_{WX}-sensitive and half only z_{YZ}-sensitive.
- **Test banks:** four holdout banks of size 1000, one sensitive and one invariant for each of z_{WX} and z_{YZ}.

Hardware and runtime details. Due to the small size of HEQ, we run single-stage OT and DAS locally on an Apple M1 Pro with 8 CPU cores and 16GB of memory. Runtime excludes factual-model training and is measured as follows:

- **PLOT:** for a fixed entropic parameter ε, runtime includes effect-signature construction over $\mathcal{D}_{\text{ft}}$ for the two abstract variables and $3 \times 16 = 48$ neural sites, Sinkhorn fitting of the EOT coupling, calibration of each coupling row over the (K, λ) grid on $\mathcal{D}_{\text{cal}}$, and final evaluation on $\mathcal{D}_{\text{te}}$ of the calibrated intervention.
- **Full DAS:** for a fixed learning rate and maximum epoch budget with early stopping, runtime includes training rotations on $\mathcal{D}_{\text{ft}}$ for each abstract variable, hidden layer ℓ, and subspace dimension k, calibration over all (ℓ, k) pairs on $\mathcal{D}_{\text{cal}}$, and final evaluation of the selected rotation on $\mathcal{D}_{\text{te}}$.

Heatmap figure details. To generate Figure 5, we select one HEQ seed to display learned handles. The accuracies for this seed are displayed in Table 3.

Each panel in Figure 5 displays a learned distribution over the 3×16 hidden neurons of the HEQ MLP. For PLOT before calibration, we show the coupling rows for z_{WX} and z_{YZ} over the 48 candidate sites, renormalized to $[0, 1]$ for visualization. For PLOT after calibration, we display only the calibrated top-K support, again renormalized to $[0, 1]$.

For DAS, we display the best calibration-selected rotation after sweeping layers ℓ and subspace dimensions k. For z_{WX}, $\ell = 2$ and $k = 8$, while for z_{YZ}, $\ell = 2$ and $k = 4$. In the canonical-basis panel, we plot the row-wise Euclidean norm of the learned rotation matrix, normalized to $[0, 1]$, as a summary of each neuron's contribution. In the rotated-basis panel, we mark the first k rotated coordinates in layer ℓ, which are the actual coordinates intervened on by DAS.

Effect of entropic regularization. We test sensitivity to the entropic regularization parameter ε in (1) by rerunning the HEQ PLOT pipeline on the same backbone and calibration protocol while varying only ε. As shown in Figure 10a, performance is stable once ε leaves the nearly unregularized regime. Very small values degrade performance because the transport plan becomes too sharp, making site rankings sensitive to small errors in the estimated effect signatures. We therefore view ε primarily as a numerical regularizer rather than a main performance knob. In practice, this stability of PLOT means that large sweeps over ε values are not needed.

Learned intervention sizes. Figure 10b shows the average learned intervention size for each method and variable on HEQ. Across ten seeds, PLOT learns both z_{WX} and z_{YZ} with interventions on about 4 neurons, while DAS uses rotated subspaces of about 7 dimensions. Thus, PLOT tends to recover more localized handles in the original neuron basis.

B.2. Binary Addition

Neural backbone training. We use a single-layer GRUCell recurrent network, which utilizes one gated recurrent unit (GRU) cell[1]. The input consists of two 4-bit numbers

$$a = (a_0, a_1, a_2, a_3), \qquad b = (b_0, b_1, b_2, b_3),$$

processed from least to most significant bit. At timestep ℓ,

$$x_\ell = [a_\ell, b_\ell] \in \mathbb{R}^2,$$
$$h_\ell = \text{GRUCell}(x_\ell, h_{\ell-1}) \in \mathbb{R}^d, \quad h_{-1} = 0.$$

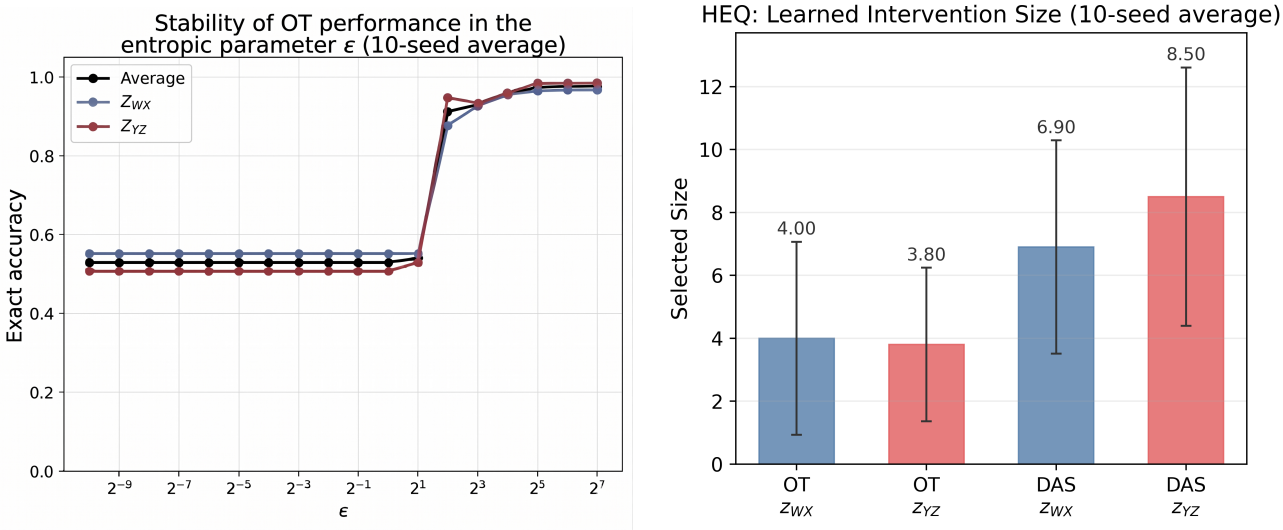
[1] https://docs.pytorch.org/docs/2.11/generated/torch.nn.GRUCell.html

Table 2. Coupling methods comparison across tasks. Values are mean $\pm$ standard deviation over three seeds. Note that runtime number may differ from previous ones, as different hardware may have been used.

Task	Metric	OT	Cosine	Brute-force
Binary addition ($d = 16$)	Average exact acc.	$\mathbf{0.7928 \pm 0.0466}$	0.7598 ± 0.0305	0.7611 ± 0.0139
	Runtime (s)	$\mathbf{5.12 \pm 0.09}$	5.14 ± 0.04	5.66 ± 0.03
MCQA	AP exact acc.	0.8200 ± 0.0216	$\mathbf{0.8500 \pm 0.0432}$	0.8433 ± 0.0411
	AT exact acc.	$\mathbf{0.9833 \pm 0.0047}$	0.9800 ± 0.0082	0.9800 ± 0.0082
	Average exact acc.	0.9017 ± 0.0125	$\mathbf{0.9150 \pm 0.0248}$	0.9117 ± 0.0232
	Runtime (s)	11.54 ± 3.19	$\mathbf{11.32 \pm 3.17}$	14.71 ± 3.97

Table 3. Selected HEQ seed used to generate Figure 5. Runtime includes alignment, calibration, and counterfactual evaluation, excluding factual-model training.

Method	z_{WX}	z_{YZ}	Average	Runtime (s)	Best setting
OT	0.9680	0.9970	0.9825	4.45	$\varepsilon = 4, K = (1, 2), \lambda = (7.4, 6.1)$, layers $(3, 2)$
DAS	0.9910	0.9970	0.9940	122.08	layer-2 sites with learned subspace sizes 8 (z_{WX}) and 4 (z_{YZ})



(a) PLOT stability in the entropic parameter ε.

(b) Average learned intervention size across methods.

Figure 10. Additional HEQ diagnostics: sensitivity to entropic regularization and average learned intervention size.

The hidden state therefore carries information from lower-order bits to higher-order bits, enabling carry propagation. Sum-bit logits are read from each timestep,

$$y_\ell^{(S)} = w_S^\intercal h_\ell + b_S, \qquad \ell = 0, 1, 2, 3,$$

and the final carry logit is read from the last state,

$$y^{(C)} = w_C^\intercal h_3 + b_C,$$
$$y = [y^{(C)}, y_3^{(S)}, y_2^{(S)}, y_1^{(S)}, y_0^{(S)}] \in \mathbb{R}^5.$$

We train models with $d \in \{8, 16\}$ using binary cross entropy and Adam at learning rate 10^{-2}, batch size 64, and 250 epochs. Both models achieve exact accuracy on all $2^4 \times 2^4 = 256$ addition pairs.

Pair bank construction. There are $2^4 \times 2^4 = 256$ possible inputs, which we collect as a bank of base examples. For each base example, we use a policy to sample 26 source examples:

- 8 source examples where each one flips the value of one of the 8 input bits, $a_0, \ldots, a_3, b_0, \ldots, b_3$.
- 18 carry-targeted source examples: 3 that change the value of C_1, 5 for C_2, 7 for C_3, and 3 for C_4.

We use a $128 - 64 - 64$ split for the 256 base examples, which means $128 \times 26 = 3328$ pairs for the fit set, $64 \times 26 = 1664$ for the calibration set, and $64 \times 26 = 1664$ for the test set.

For the test set, we record for each pair whether or not it changed the value of C_1, C_2, and C_3, the three abstract variables of interest. Then, we partition the 1664 test pairs into sensitive and invariant sets for each of these variables:

- C_1: 417 sensitive examples, 1247 invariant examples.
- C_2: 572 sensitive examples, 1092 invariant examples.
- C_3: 700 sensitive examples, 964 invariant examples.

The final calibrated handles for each method are tested on all six of these test sets, and their accuracies are averaged.

Hardware, runtime, and algorithm details. We run all methods locally on an 11th Gen Intel Core i7-1185G7 CPU with 16GB of memory. The algorithm and runtime for each method are detailed below.

- PLOT: performs Stage A OT to calibrate the best timestep per carry variable. See Algorithm 1 for the exact pseudocode.
- PLOT-NATIVE: performs Stage A OT, then Stage B OT in the canonical basis to further refine to sites within the selected timesteps, generating soft intervention handles; Algorithm 2.
- PLOT-PCA: performs Stage A OT over timesteps, fits PCA rotations, then Stage B PCA-OT through the PCA rotations to generate soft intervention handles inside the projected and rotated space; Algorithm 3.
- PLOT-DAS: performs Stage A OT over timesteps, then learns a DAS rotation handle per carry variable on the chosen timestep, sweeping over subspace intervention sizes; Algorithm 4.
- Full DAS: sweeps over all timesteps and subspace intervention sizes to learn a DAS rotation handle per carry variable; Algorithm 5.

Algorithm 1 Binary Addition PLOT

Require: OT parameter ε
Require: Splits $\mathcal{D}_{\text{ft}}$, $\mathcal{D}_{\text{cal}}$, and $\mathcal{D}_{\text{te}}$, factual pre-trained GRUCell with hidden size d
Start timer
Construct effect signatures over $\mathcal{D}_{\text{ft}}$ for neural timesteps $h_0, \ldots, h_3$
Construct effect signatures over $\mathcal{D}_{\text{ft}}$ for abstract carry variables C_1, C_2, C_3
Compute the Sinkhorn EOT coupling in $\mathbb{R}^{3\times 4}$ using ε
for carry variable $C_i \in \{C_1, C_2, C_3\}$ **do**
    Extract the top timestep ℓ_i for row C_i
    Evaluate timestep h_{ℓ_i} for C_i on $\mathcal{D}_{\text{te}}$
end for
End timer and report PLOT runtime
Return selected timesteps ℓ_1, ℓ_2, ℓ_3

Algorithm 2 Binary Addition PLOT-NATIVE

Require: OT parameter ε
Require: Splits $\mathcal{D}_{\text{ft}}$, $\mathcal{D}_{\text{cal}}$, and $\mathcal{D}_{\text{te}}$, factual pre-trained GRUCell with hidden size d
Start timer
Run PLOT (Algorithm 1) to obtain ℓ_1, ℓ_2, ℓ_3
for $\ell \in \{\ell_1, \ell_2, \ell_3\}$ **do**
    for resolution $r \in \{1, 2\}$ **do**
        Partition h_ℓ into neural sites with resolution r
        Construct effect signatures over $\mathcal{D}_{\text{ft}}$ for the partitioned neural sites
        Compute the Sinkhorn EOT coupling in $\mathbb{R}^{3\times\lfloor d/r\rfloor}$ using ε
        for C_i with top timestep ℓ **do**
            for $(\text{top-}K, \lambda) \in \{1, 2, 4\} \times \{0.25, 0.5, 1, 2, 4, 8\}$ **do**
                Evaluate the soft intervention handle for C_i on $\mathcal{D}_{\text{cal}}$
            end for
            Evaluate the best calibrated intervention for C_i on $\mathcal{D}_{\text{te}}$
        end for
    end for
end for
End timer and report PLOT-NATIVE runtime

Algorithm 3 Binary Addition PLOT-PCA

Require: OT parameter ε
Require: Splits $\mathcal{D}_{\text{ft}}$, $\mathcal{D}_{\text{cal}}$, and $\mathcal{D}_{\text{te}}$, factual pre-trained GRUCell with hidden size d
Start timer
Run PLOT (Algorithm 1) to obtain ℓ_1, ℓ_2, ℓ_3
for $\ell \in \{\ell_1, \ell_2, \ell_3\}$ **do**
  Collect hidden states from $\mathcal{D}_{\text{ft}}$ at timestep h_ℓ
  Compute the centered PCA rotation with full rank; store it as R_ℓ
  Define neural sites to be the first $1, 2, 4, \ldots, d$ principal component dimensions
  Set $W_j = R_\ell$ for each PCA site s_j contained in timestep h_ℓ
  Construct effect signatures over $\mathcal{D}_{\text{ft}}$ for the PCA-prefix neural sites
  Compute the Sinkhorn EOT coupling in $\mathbb{R}^{3\times(\log_2(d)+1)}$ using ε
  for C_i with top timestep ℓ **do**
    for (top-K, λ) $\in \{1, 2, 4\} \times \{0.25, 0.5, 1, 2, 4, 8\}$ **do**
      Evaluate the soft intervention handle for C_i on $\mathcal{D}_{\text{cal}}$
    end for
    Evaluate the best calibrated intervention for C_i on $\mathcal{D}_{\text{te}}$
  end for
end for
End timer and report PLOT-PCA runtime

Algorithm 4 Binary Addition PLOT-DAS

Require: DAS learning rate lr, DAS maximum epochs N
Require: Splits $\mathcal{D}_{\text{ft}}$, $\mathcal{D}_{\text{cal}}$, and $\mathcal{D}_{\text{te}}$, factual pre-trained GRUCell with hidden size d
Start timer
Run PLOT (Algorithm 1) to obtain ℓ_1, ℓ_2, ℓ_3
for $C_i \in \{C_1, C_2, C_3\}$ **do**
  for $k \in \{1, 2, 4, \ldots, d\}$ **do**
    Train a DAS rotation for C_i on $\mathcal{D}_{\text{ft}}$ at timestep h_{ℓ_i} with subspace intervention size k
    Evaluate the rotation handle for C_i on $\mathcal{D}_{\text{cal}}$
  end for
  Evaluate the best calibrated rotation for C_i on $\mathcal{D}_{\text{te}}$
end for
End timer and report PLOT-DAS runtime

Algorithm 5 Binary Addition Full DAS

Require: DAS learning rate lr, DAS maximum epochs N
Require: Splits $\mathcal{D}_{\text{ft}}$, $\mathcal{D}_{\text{cal}}$, and $\mathcal{D}_{\text{te}}$, factual pre-trained GRUCell with hidden size d
Start timer
for $C_i \in \{C_1, C_2, C_3\}$ **do**
  for timestep $\ell \in \{0, 1, 2, 3\}$ **do**
    for $k \in \{1, 2, 4, \ldots, d\}$ **do**
      Train a DAS rotation for C_i on $\mathcal{D}_{\text{ft}}$ at timestep h_ℓ with subspace intervention size k
      Evaluate the rotation handle for C_i on $\mathcal{D}_{\text{cal}}$
    end for
  end for
  Evaluate the best calibrated rotation for C_i on $\mathcal{D}_{\text{te}}$
end for
End timer and report Full DAS runtime

B.3. Multiple-Choice Question Answering (MCQA)

Pair bank construction. For the MCQA task, MIB-Bench (Mueller et al., 2025) open-sources a dataset[2] used to train and test their methods. However, this public dataset only contains 210 rows, so we synthetically generate a dataset[3] with 10,000 rows in the same format as their dataset. See Table 5 for a base example and associated counterfactuals. Recall that the causal model in Figure 9 contains two abstract variables: AP and AT. As in MIB-Bench, we consider the following three counterfactuals:

- `answerPosition`: permute the positions of the answer choice colors $\mathrm{Col_i}$, without changing the answer choice symbols $\mathrm{Sym_i}$. This typically affects both AP and AT.
- `randomLetter`: randomly sample four different letters of the alphabet to replace the answer choice symbols $\mathrm{Sym_i}$. This typically affects AT.
- `answerPosition + randomLetter`: apply both counterfactuals above at the same time. This typically affects both AP and AT.

Before constructing pair banks, we filter examples to ensure that Gemma-2-2B correctly completes the prompts before any interventions. We first sample 2,000 rows from our dataset, forming $2{,}000 \times 3 = 6{,}000$ (base, source) pairs from the three counterfactuals. Then we pass each pair into Gemma-2-2B: a (base, source) pair passes the filter if Gemma-2-2B correctly completes both the base and source examples, up to capitalization, whitespace, and punctuation. Finally, we sample the following fit, calibration, and test banks:

- Fit bank: 200 pairs sampled randomly from the filtered bank.
- Calibration banks: 100 sensitive pairs for each AP and AT.
- Test banks: 100 sensitive pairs for each AP and AT.

Effect signature construction. Gemma-2-2B has a vocabulary size of 256,000, so it would be computationally impractical to concatenate $\mathcal{D}_{\mathrm{ft}}$ many 256,000-sized vectors to form each neural or abstract effect signature. Instead, because we care mostly about the effect of intervention on the alphabet tokens $A, \ldots, Z$, we first truncate the LLM's output to these 26 tokens. Let

$$\Delta^{\mathrm{nn}}_{j,t} = y^{\mathrm{nn_swap}}_{j,t}(A:Z) - y^{\mathrm{nn}}_{t}(A:Z) \in \mathbb{R}^{26}$$

denote the truncated neural effect of intervening on neural site j for example t. Next, we average these effects across counterfactual families,

$$\bar{\Delta}^{\mathrm{nn}}_{j,g} = \frac{1}{|\mathcal{D}_{\mathrm{ft,g}}|} \sum_{t \in \mathcal{D}_{\mathrm{ft,g}}} \Delta^{\mathrm{nn}}_{j,t},$$

where g is one of `answerPosition`, `randomLetter`, and `answerPosition + randomLetter` before aggregating the normalized effects into one family-wise effect:

$$v_j = \left[\hat{\Delta}^{\mathrm{nn}}_{j,\mathtt{aP}}; \hat{\Delta}^{\mathrm{nn}}_{j,\mathtt{rL}}; \hat{\Delta}^{\mathrm{nn}}_{j,\mathtt{aP+rL}}\right] \in \mathbb{R}^{78}$$

where $\hat{\Delta}$ is $\bar{\Delta}$ after being centered and ℓ_2-normalized. We can similarly compute the family-wise abstract effect signature using one-hot encodings of the alphabet tokens.

Effect signature ablation. As an ablation study, we additionally consider total-variation (TV) and KL effect signatures over the full output vocabulary. For both variants, the abstract effect signature records whether the abstract intervention changes the output:

$$u_i = \left[\mathbb{1}\left[y^{\mathrm{abs_swap}}_{i,t} \neq y^{\mathrm{abs}}_{t}\right]\right]_{t=1}^{T_{\mathrm{ft}}}.$$

For the TV signature, the effect of neural site s_j on example t is

$$\Delta^{\mathrm{nn}}_{j,t} = \frac{1}{2}\left\|\mathrm{softmax}\big(y^{\mathrm{nn_swap}}_{j,t}\big) - \mathrm{softmax}(y^{\mathrm{nn}}_{t})\right\|_1,$$

giving the neural signature

$$v^{\mathrm{TV}}_j = \left[\Delta^{\mathrm{TV}}_{j,t}\right]_{t=1}^{T_{\mathrm{ft}}}.$$

For the KL signature, we have

$$\begin{aligned}\Delta^{\mathrm{nn}}_{j,t} =& 1 - \exp\big(\\ & -D_{\mathrm{KL}}\big(\mathrm{softmax}\big(y^{\mathrm{nn_swap}}_{j,t}\big) \,\big\|\, \mathrm{softmax}(y^{\mathrm{nn}}_{t})\big)\big),\end{aligned}$$

where we use the transformation $1 - \exp(-D_{\mathrm{KL}})$ to bound the raw KL value to $[0, 1)$, then construct

$$v^{\mathrm{KL}}_j = \left[\Delta^{\mathrm{KL}}_{j,t}\right]_{t=1}^{T_{\mathrm{ft}}}.$$

We compare the performance of the default family-wise truncated signature to the TV and KL signatures for MCQA by utilizing each signature with PLOT-NATIVE in layer localization in Stage A and within-layer native handle extraction in Stage B. Table 4 displays the results: for one seed, the TV signature matches the family-wise signature's accuracy. However, the KL signature is notably worse, and unable to recover a meaningful layer for AT in Stage A.

[2] https://huggingface.co/datasets/mib-bench/copycolors_mcqa

[3] placeholder for anonymous submission

Table 4. Effect-signature comparison on MCQA for one seed.

Metric	Family-wise	TV	KL
Stage A selected AP layer	18	17	18
Stage A selected AT layer	24	24	21
AP exact acc.	0.8100	**0.8200**	0.7400
AT exact acc.	**0.9400**	0.9300	0.8500
Average exact acc.	**0.8750**	**0.8750**	0.7950

OT vs. UOT details. In Table 6, we compare OT and UOT as Stage-A layer-selection methods for the final-token residual stream. Each method produces a 2×26 coupling, with rows corresponding to AP and AT and columns corresponding to transformer layers. For each row, we take the top-6 layers by mass, calibrate the associated full-layer interventions on that abstract variable, and report the layer with the highest calibration accuracy. Because standard OT must distribute mass across all 26 layer sites using only two abstract rows, it yields weak layer selection for AP. In contrast, UOT can leave irrelevant layers unmatched, which allows it to select high-scoring AP layers across all three seeds. Notably, layer 17 also agrees with the layer selected by full DAS across the four-seed run.

Hardware, runtime, and algorithm details. We run all methods on an NVIDIA H100 SXM GPU with 80 GB VRAM, 188 GB system RAM, and 8 vCPUs. The algorithm and runtime for each method are detailed below.

- PLOT: performs Stage A UOT to calibrate the best layers per abstract variable. See Algorithm 6 for the exact pseudocode.
- PLOT-NATIVE: performs Stage A UOT, then Stage B OT to further refine to sites within the layer, generating soft intervention handles; Algorithm 7.
- PLOT-PCA: performs Stage A UOT, fits a PCA rotation, then Stage B PCA-OT through the PCA rotation to generate soft intervention handles inside the projected and rotated space; Algorithm 8.
- PLOT-DAS: performs Stage A UOT, then learns a DAS rotation handle per abstract variable on the chosen layers, sweeping over subspace intervention sizes; Algorithm 9.
- PLOT-NATIVE-DAS: performs Stage A UOT, Stage B OT, then learns a DAS rotation handle per abstract variable on the chosen layers, sweeping over selected subspace intervention sizes informed by the Stage B OT; Algorithm 10.
- PLOT-PCA-DAS: performs Stage A UOT, Stage B PCA-OT, then learns a DAS rotation handle on top of the PCA rotation per abstract variable on the chosen layers, sweeping over selected subspace intervention sizes informed by the Stage B PCA-OT; Algorithm 11.
- Full DAS: sweeps over all layers and subspace intervention sizes to learn a DAS rotation handle per abstract variable; Algorithm 12.

Table 5. An MCQA example and selected associated counterfactuals.

Counterfactual	Text	Correct Completion
Original Prompt	*Question: A banana is yellow. What color is a banana? A. red\nB. blue\nC. yellow\nD. green\nAnswer:*	C
`answerPosition`	*Question: A banana is yellow. What color is a banana? A. green\nB. yellow\nC. red\nD. blue\nAnswer:*	B
`randomLetter`	*Question: A banana is yellow. What color is a banana? H. red\nT. blue\nP. yellow\nX. green\nAnswer:*	P
`answerPosition + randomLetter`	*Question: A banana is yellow. What color is a banana? H. green\nT. yellow\nP. red\nX. blue\nAnswer:*	T

Table 6. Selected MCQA layers and accuracies for Stage A OT vs UOT across four seeds.

	OT		UOT	
Seed	AP	AT	AP	AT
0	L6 (0.000)	L24 (0.980)	L18 (0.700)	L24 (0.980)
1	L6 (0.000)	L23 (0.970)	L17 (0.670)	L23 (0.970)
2	L6 (0.000)	L23 (0.950)	L17 (0.750)	L23 (0.950)
3	L6 (0.000)	L23 (0.970)	L17 (0.800)	L23 (0.970)

Algorithm 6 MCQA PLOT

Require: UOT parameters ε and β_{neural}
Require: Splits $\mathcal{D}_{\text{ft}}$, $\mathcal{D}_{\text{cal}}$, and $\mathcal{D}_{\text{te}}$, factual pre-trained Gemma-2-2B
Start timer
Construct effect signatures over $\mathcal{D}_{\text{ft}}$ for neural sites $\ell_0, \ldots, \ell_{25}$
Construct effect signatures over $\mathcal{D}_{\text{ft}}$ for abstract variables AP, AT
Compute the one-sided Sinkhorn UOT coupling in $\mathbb{R}^{2\times 26}$ using ε and β_{neural}
for abstract variable $Z \in \{\text{AP}, \text{AT}\}$ **do**
    Extract the top-6 highest mass sites for Z; evaluate each for Z on $\mathcal{D}_{\text{cal}}$
    Select the layer ℓ_Z with the best calibration accuracy; evaluate it for Z on $\mathcal{D}_{\text{te}}$
end for
End timer and report PLOT runtime
Return ℓ_{AP} and ℓ_{AT}

Algorithm 7 MCQA PLOT-NATIVE

Require: OT parameter ε
Require: Splits $\mathcal{D}_{\mathrm{ft}}$, $\mathcal{D}_{\mathrm{cal}}$, and $\mathcal{D}_{\mathrm{te}}$, factual pre-trained Gemma-2-2B
Start timer
Run PLOT (Algorithm 6) to obtain ℓ_{AP} and ℓ_{AT}
for $Z \in \{\mathrm{AP}, \mathrm{AT}\}$ **do**
  for resolution $r \in \{128, 144, 192, 256, 288, 384, 576, 768\}$ **do**
    Partition ℓ_Z into sites with resolution r
    Construct effect signatures over $\mathcal{D}_{\mathrm{ft}}$ for the partitioned neural sites
    Compute the Sinkhorn EOT coupling in $\mathbb{R}^{2\times\lfloor 2304/r\rfloor}$ using ε
    for (top-K, λ) $\in \{1, 2, 4\} \times \{0.5, 1, 2, 4\}$ **do**
      Evaluate the soft intervention handle for Z on $\mathcal{D}_{\mathrm{cal}}$
    end for
  end for
  Evaluate the best calibrated intervention for Z on $\mathcal{D}_{\mathrm{te}}$
end for
End timer and report PLOT-NATIVE runtime
Return the optimal handles $H^{\mathrm{nat}}_{\mathrm{AP}}, H^{\mathrm{nat}}_{\mathrm{AT}}$ (along with their chosen $r^\star, K^\star, \lambda^\star$)

Algorithm 8 MCQA PLOT-PCA

Require: OT parameter ε
Require: Splits $\mathcal{D}_{\mathrm{ft}}$, $\mathcal{D}_{\mathrm{cal}}$, and $\mathcal{D}_{\mathrm{te}}$, factual pre-trained Gemma-2-2B
Start timer
Run PLOT (Algorithm 6) to obtain ℓ_{AP} and ℓ_{AT}
for $Z \in \{\mathrm{AP}, \mathrm{AT}\}$ **do**
  Collect hidden states from $\mathcal{D}_{\mathrm{ft}}$ at layer ℓ_Z
  Compute the centered PCA rotation with full rank; store the rotation as R_Z
  for band count $b \in \{8, 16\}$ **do**
    Partition ℓ_Z under rotation R_Z (i.e., a rotated space of dimension $\mathrm{rank}(R_Z)$) into b sites.
    Set $W_j = R_Z$ for each site s_j contained in h_{ℓ_Z}
    Construct effect signatures over $\mathcal{D}_{\mathrm{ft}}$ for the partitioned neural sites
    Compute the Sinkhorn EOT coupling in $\mathbb{R}^{2\times b}$ using ε
    for (top-K, λ) $\in \{1, 2, 4\} \times \{0.5, 1, 2, 4\}$ **do**
      Evaluate the soft intervention handle for Z on $\mathcal{D}_{\mathrm{cal}}$
    end for
  end for
  Evaluate the best calibrated intervention for Z on $\mathcal{D}_{\mathrm{te}}$
end for
End timer and report PLOT-PCA runtime
Return the rotations $R_{\mathrm{AP}}, R_{\mathrm{AT}}$ and the optimal handles $H^{\mathrm{pca}}_{\mathrm{AP}}, H^{\mathrm{pca}}_{\mathrm{AT}}$ (along with their chosen $b^\star, K^\star, \lambda^\star$)

Algorithm 9 MCQA PLOT-DAS

Require: DAS learning rate lr, DAS maximum epochs N
Require: Splits $\mathcal{D}_{\mathrm{ft}}$, $\mathcal{D}_{\mathrm{cal}}$, and $\mathcal{D}_{\mathrm{te}}$, factual pre-trained Gemma-2-2B
Start timer
Run PLOT (Algorithm 6) to obtain ℓ_{AP} and ℓ_{AT}
for $Z \in \{\mathrm{AP}, \mathrm{AT}\}$ **do**
  for $k \in \{32, 64, 96, 128, 256, 512, 768, 1024, 1536, 2048, 2304\}$ **do**
    Train a DAS rotation for Z on $\mathcal{D}_{\mathrm{ft}}$ at layer ℓ_Z with subspace intervention size k
    Evaluate the rotation handle for Z on $\mathcal{D}_{\mathrm{cal}}$
  end for
  Evaluate the best calibrated rotation for Z on $\mathcal{D}_{\mathrm{te}}$
end for
End timer and report PLOT-DAS runtime

Algorithm 10 MCQA PLOT-NATIVE-DAS

Require: DAS learning rate lr, DAS maximum epochs N
Require: Splits $\mathcal{D}_{\mathrm{ft}}$, $\mathcal{D}_{\mathrm{cal}}$, and $\mathcal{D}_{\mathrm{te}}$, factual pre-trained Gemma-2-2B
Start timer
Run PLOT-NATIVE (Algorithm 7) to obtain ℓ_{AP}, ℓ_{AT}, $H^{\mathrm{nat}}_{\mathrm{AP}}$, $H^{\mathrm{nat}}_{\mathrm{AT}}$
for $Z \in \{\mathrm{AP}, \mathrm{AT}\}$ **do**
  Extract $r^\star$ and $K^\star$ from H^{nat}_Z
  Set effective dimension $e := r^\star K^\star$
  for $k \in \{0.5e, 0.75e, e, 1.5e, 2.0e\}$ **do**
    Train a DAS rotation for Z on $\mathcal{D}_{\mathrm{ft}}$ at layer ℓ_Z with subspace intervention size k
    Evaluate the rotation handle for Z on $\mathcal{D}_{\mathrm{cal}}$
  end for
  Evaluate the best calibrated rotation for Z on $\mathcal{D}_{\mathrm{te}}$
end for
End timer and report PLOT-NATIVE-DAS runtime

Algorithm 11 MCQA PLOT-PCA-DAS

Require: DAS learning rate lr, DAS maximum epochs N
Require: Splits $\mathcal{D}_{\mathrm{ft}}$, $\mathcal{D}_{\mathrm{cal}}$, and $\mathcal{D}_{\mathrm{te}}$, factual pre-trained Gemma-2-2B
Start timer
Run PLOT-PCA (Algorithm 8) to obtain ℓ_{AP}, ℓ_{AT}, R_{AP}, R_{AT}, $H^{\mathrm{pca}}_{\mathrm{AP}}$, $H^{\mathrm{pca}}_{\mathrm{AT}}$
for $Z \in \{\mathrm{AP}, \mathrm{AT}\}$ **do**
  Extract $b^\star$ and $K^\star$ from H^{pca}_Z
  Set effective dimension $e := \lfloor \mathrm{rank}(R_Z)/b^\star \rfloor K^\star$
  for $k \in \{0.5e, 0.75e, e, 1.5e, 2.0e\}$ **do**
    Train a DAS rotation on top of R_Z for Z on $\mathcal{D}_{\mathrm{ft}}$ (DAS rotation has size $\mathbb{R}^{k \times \mathrm{rank}(R_Z)}$)
    Evaluate the rotation for Z on $\mathcal{D}_{\mathrm{cal}}$
  end for
  Evaluate the best calibrated rotation for Z on $\mathcal{D}_{\mathrm{te}}$
end for
End timer and report PLOT-PCA-DAS runtime

Algorithm 12 MCQA Full DAS

Require: DAS learning rate lr, DAS maximum epochs N
Require: Splits $\mathcal{D}_{\mathrm{ft}}$, $\mathcal{D}_{\mathrm{cal}}$, and $\mathcal{D}_{\mathrm{te}}$, factual pre-trained Gemma-2-2B

```
Start timer
for Z ∈ {AP, AT} do
    for ℓ ∈ {ℓ_0, ..., ℓ_25} do
        for k ∈ {32, 64, 96, 128, 256, 512, 768, 1024, 1536, 2048, 2304} do
            Train a DAS rotation on D_ft at layer ℓ with subspace intervention size k
            Evaluate the rotation handle for Z on D_cal
        end for
    end for
    Evaluate the best calibrated rotation for Z on D_te
end for
End timer and report Full DAS runtime
```