Benchmarks, Algorithms, and Metrics for Hierarchical Disentanglement

Andrew Slavin Ross [1] **Finale Doshi-Velez** [1]

Abstract

In representation learning, there has been recent interest in developing algorithms to disentangle the ground-truth generative factors behind data, and metrics to quantify how fully this occurs. However, these algorithms and metrics often assume that both representations and ground-truth factors are flat, continuous, and factorized, whereas many real-world generative processes involve rich hierarchical structure, mixtures of discrete and continuous variables with dependence between them, and even varying intrinsic dimensionality. In this work, we develop benchmarks, algorithms, and metrics for learning such hierarchical representations.

1. Introduction

Autoencoders aim to learn structure in data by compressing it to a lower-dimensional representation with minimal loss of information. Although this has proven useful in many applications (LeCun et al., 2015), the individual dimensions of autoencoder representations are often inscrutable, even when the underlying data is generated by simple processes. Motivated by needs for interpretability (Alvarez-Melis & Jaakkola, 2018; Marx et al., 2019), fairness (Creager et al., 2019), and generalizability (Bengio et al., 2013), as well as a basic intuition that representations should model the data correctly, a subfield has emerged which applies representation learning algorithms to synthetic datasets and checks how well representation dimensions "disentangle" the known ground-truth factors behind the dataset.

Perhaps the most common disentanglement approach has been to learn flat, continuous vector representations whose dimensions are statistically independent (and evaluate them using metrics that assume ground-truth factors are independent), reasoning that factorization is a useful proxy (Ridgeway, 2016; Higgins et al., 2017; Chen et al., 2018; Kim & Mnih, 2018). However, this problem is not identifiable (Locatello et al., 2018), and it seems unlikely that continuous, factorized, flat representations are the optimal choice for modeling many real-world generative processes, which are often highly structured.To address aspects of this problem, some approaches generalize to partially discrete representations (Jeong & Song, 2019), or encourage independence only conditionally, based on hierarchies or causal graphs (Esmaeili et al., 2019; Träuble et al., 2020). Almost all approaches require side-information, either about specific instances or about the global structure of the dataset.

Our approach in this paper is ambitious: we introduce (1) a flexible framework for modeling deep hierarchical structure in datasets, (2) novel algorithms for learning both structure and structured autoencoders entirely from data, which we apply to (3) novel benchmark datasets, and evaluate with (4) novel hierarchical disentanglement metrics. Our framework is based on the idea that data may lie on multiple manifolds with different intrinsic dimensionalities, and that certain (hierarchical groups of) dimensions may only become active for a subset of the data.[1] Though at first glance this approach seems it should worsen, not improve, identifiability, our assumption of geometric structure also serves as an inductive bias that empirically helps us learn representations that more faithfully (and explicitly) model ground-truth generative processes.

2. Related Work

Though interest in disentanglement is longstanding (Schmidhuber, 1992; Comon, 1994; Bengio, 2013), a relatively recent resurgence has focused on **flat factorized representations**. Ridgeway (2016) provide an influential survey of such representations, arguing for their usefulness. Higgins et al. (2017) develop β-VAE, which tries to encourage factorization in variational autoencoders (VAEs, Kingma & Welling (2013)) by increasing the KL divergence penalty.

[1]Harvard University, Cambridge, Massachusetts, USA. Correspondence to: Andrew Ross <andrewslavinross@gmail.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

[1]As a concrete example, consider the problem of learning representations of medical phenotypes of patients with and without diabetes mellitus, a complex disease with multiple types and subtypes (American Diabetes Association, 2005). Some underlying factors of phenotype variation—as well as the intrinsic complexity of these variations—are likely specific to the disease, its types, or its subtypes (Ahlqvist et al., 2018). A representation that faithfully modeled the true factors of variation would need to be deeply hierarchical, with some dimensions only active for certain subtypes.

Chen et al. (2018) and Kim & Mnih (2018) factorize by directly penalizing the total correlation (TC) between dimensions. Mixed discrete-continuous extensions are developed for KL by Dupont (2018) and TC by Jeong & Song (2019).

Although these innovations are specific to flat representations, there has been work on **certain forms of hierarchy**. Esmaeili et al. (2019) encourage different degrees of factorization within and across subgroups, which can be nested. However, they only apply their method in shallow contexts, and subgroups must be provided rather than learned from data. Choi et al. (2020) learn mixed discrete-continuous representations where some continuous dimensions are "public" in scope (and globally independent), while others are "private" to a categorical (and conditionally independent of siblings). However, they do not support deep hierarchies, and the structure of categorical, public, and private variables is provided rather than learned. GINs (Sorrenson et al., 2020) can infer the dimensionality of such categorical-specific continuous dimensions groups, but still must be given the (shallow) categorical structure. FineGAN (Singh et al., 2019) learns a kind of structure, but enforces a specific shallow hierarchy of background, shape, and pose. Adams et al. (2010) model data with arbitrarily wide and deep trees using Bayesian non-parametric methods. However, there is no explicit encoder (representations are inferred via MCMC), and all features are binary. Our method attempts to provide the best of all these worlds; from data alone, we learn autoencoders whose representations have mixed discrete-continuous structure of arbitrary width and depth.

Our approach is also complementary to **recent shifts in the disentanglement literature**, especially from causality researchers. Parascandolo et al. (2018) and Träuble et al. (2020) argue for learning representations that disentangle causal mechanisms rather than statistically independent factors. Locatello et al. (2018) show global independence objectives are non-identifiable, and suggest a shift in focus to inductive biases, semi-supervision of specific factors (Mathieu et al., 2016; Siddharth et al., 2017), or weak supervision. For example, Locatello et al. (2020a) learn disentangled representations given instance-pairs that differ only by sparse sets of ground-truth factors, and Klindt et al. (2021) use similar principles to disentangle factors that vary sparsely over time. In this work, we return to the problem of learning disentangled representations from data alone. As our inductive bias to reduce (though not eliminate) non-identifiability, we assume the data contains discrete hierarchical structure that can be inferred geometrically. Though this introduces challenging new problems, it also creates opportunities to learn interpretable global summaries of the data.

Other related approaches include relational autoencoders (Wang et al., 2014), which model structure between non-iid flat data, and graph neural networks (Defferrard et al., 2016), which learn flat representations of structured data. In contrast, we model structure *within* flat inputs. Also relevant are advances in object representations, such as slot attention (Locatello et al., 2020b). While this area has generally not focused on hierarchical nesting, it learns structure and seamlessly handles sets; we view our method as complementary. Finally, our hierarchy detection method is built on work in multiple- and robust manifold learning (Mahapatra & Chandola, 2017; Mahler, 2020). We contribute new robustness innovations and also introduce hierarchy (like Tino & Nabney (2002), but without requiring fixed dimensionality or human feedback).

3. Hierarchical Disentanglement Framework

In this section, we outline our framework for modeling hierarchical structure in representations. In our framework, we associate individual data points with paths down a *dimension hierarchy* (examples in Fig. 1). Dimension hierarchies consist of dimension group nodes (shown as boxes), each of which can have any number of continuous dimensions (shown as ovals) and an optional categorical variable (diamonds) that leads to other groups based on its value. For any data point, we "activate" only the dimensions along its corresponding path. Notation-wise, $\texttt{root}(Z)$ denotes the group at the root of a hierarchy, and $\texttt{children}(Z_j)$ denotes the child groups of a categorical dimension Z_j. In the context of a dataset, for a dimension Z_j or a dimension group g, $\texttt{on}(Z_j)$ or $\texttt{on}(g)$ denotes the subset of the dataset where that Z_j or g is active.

This framework can be readily extended to support multiple categorical variables per node (e.g. recursing on both segment halves in the timeseries dataset defined below) or even DAGs, such that instances can be associated with directed flows down multiple paths. For simplicity, however, we narrow our scope to tree structures in this work.

4. Hierarchical Disentanglement Benchmarks

For new frameworks, it is especially important to have synthetic benchmarks for which the true structure is known and ground truth disentanglement scores can be computed. Below we further describe the two benchmarks from Fig. 1.

4.1. Spaceshapes

Our first benchmark dataset is Spaceshapes, a binary 64x64 image dataset meant to hierarchically extend dSprites, a shape dataset common in the disentanglement literature (Matthey et al., 2017). Like dSprites, Spaceshapes images have location variables `x` and `y`, as well as a categorical `shape` with three options (in our case, `moon`, `star`, and `ship`). However, depending on shape, we add other continuous variables with very different effects: `moons` have a

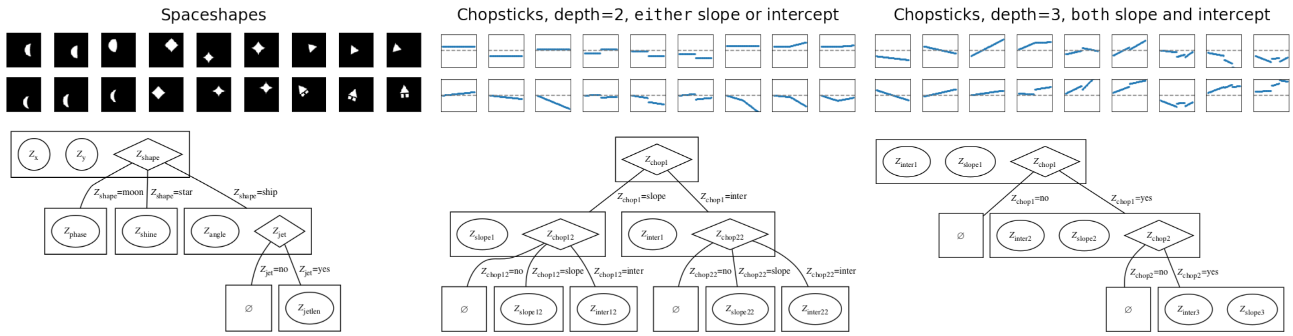


Figure 1. Examples and ground-truth variable hierarchies for Spaceshapes and two different variants of Chopsticks. Continuous variables are shown as circles and discrete variables are shown as diamonds. Discrete variables have subhierarchies of additional variables that are only active for particular discrete values.

`phase`; `stars` have a sharpness to their `shine`; and `ships` have an `angle`. Finally, `ships` can optionally have a `jet`, which has a length (`jetlen`), but this is only defined at the deepest level of the hierarchy. The presence of `jetlen` alters the intrinsic dimensionality of the representation; it can be either 3D or 4D depending on the path. As in dSprites, variables are sampled from continuous or discrete uniforms.

4.2. Chopsticks

Our second benchmark, Chopsticks, is actually a family of arbitrarily deep timeseries datasets. Chopsticks samples are 64D linear segments. Each segment can have a uniform-sampled `slope` and/or `intercept`; different Chopsticks variants can have one, the other, `both`, or `either` but not both. For all variants, segments initially span the whole interval. However, a coin is then flipped to determine whether to `chop` the segment, in which case we add a uniform offset to the slope and/or intercept of the right half. We repeat this process recursively up to a configurable maximum `depth`, biasing probabilities so that we have equal probability of stopping at each level. Each chop requires increasing local dimensionality to track additional slopes and intercepts. Although the underlying process is quite simple, the structure can be made arbitrarily deep, making it a useful benchmark for testing structure learning.

Although these datasets are designed to have clear hierarchical structure, in certain cases, there are multiple dimension hierarchies that could arguably describe the same dataset. See Fig. 14 for more and §6.1 for how we handle them.

5. Hierarchical Disentanglement Algorithms

We next present a method for learning hierarchical disentangled representations from data alone. We split the problem into two brunch-themed algorithms, MIMOSA (which infers hierarchies) and COFHAE (which trains autoencoders).

5.1. Learning Hierarchies with MIMOSA

The goal of our first algorithm, MIMOSA (**M**ulti-manifold **I**so**M**ap **O**n **S**mooth **A**utoencoder), is to learn a hierarchy $\hat{H}$ from data, as well as an assignment vector $\hat{A}_n$ of data points to hierarchy leaves. MIMOSA consists of the following steps (see Appendix for Algorithms 3-7 and complexity):

Dimensionality Reduction (Algorithm 1, line 1): We start by performing an initial reduction of X to Z using a flat autoencoder. While we could start with $Z = X$, performing this reduction saves computation as later steps (e.g. finding neighbors) scale linearly with $|Z|$. Although this requires choosing $|Z|$, we find the exact value is not critical as long as it exceeds the (max) intrinsic dimensionality of the data. We also find it important to use differentiable activation functions (e.g. Softplus rather than ReLU) to keep latent manifolds smooth; see Fig. 6 for more.

Manifold Decomposition (Algorithms 3-6): We decompose Z into a set of manifold "components" by computing SVDs locally around each point and merging neighboring points with sufficiently similar subspaces. We then perform a second merging step over longer lengthscales, combining equal-dimensional components with similar local SVDs along their nearest boundary points and discarding small outliers, which we found was necessary to handle interstitial gaps when two manifolds intersect. The core of this step is based on a multi-manifold learning method (Mahapatra & Chandola, 2017), but we make efficiency as well as robustness improvements by combining ideas from RANSAC (Fischler & Bolles, 1981) and contagion dynamics (Mahler, 2020). The merging step is a new contribution.

It bears emphasis that manifold decomposition, which groups points based on the similarity of local principal components, is distinct from clustering, which groups points based on proximity. On our benchmarks, even hierarchical iterative clustering methods like OPTICS (Ankerst et al.,

Algorithm 1 MIMOSA(X)

Encode the data X using a smooth autoencoder to reduce the initial dimensionality. Store as Z.
Construct a neighborhood graph on Z using a Ball Tree (Omohundro, 1989).
Run LocalSVD (Algorithm 3) on each point in Z and its neighbors to identify local manifold directions.
Run BuildComponent (Algorithm 5) to successively merge neighboring points with similar local manifold directions.
Run MergeComponents (Algorithm 6) to combine similar components over longer distances and discard outliers.
Run ConstructHierarchy (Algorithm 7) to create a dimension hierarchy based on which components enclose others.
return the hierarchy and component assignments.

1999) will not suffice, as nearby points may lie on different manifolds.

Hierarchy Identification (Algorithm 7): We construct a tree by drawing edges from low-dimensional components to the higher-dimensional components that best "enclose" them, which we define using a ratio of inter-component to intra-component nearest neighbor distances; we believe this is novel. We use this tree and the component dimensionalities to construct a dimension hierarchy and a set of assignments from points to paths, which we output.

Hyperparameters: Each of these steps has several hyperparameters, and we provide a full listing and sensitivity study in §A.3. The one we found most critical was the minimum SVD similarity to merge neighboring points.

5.2. Training Autoencoders with COFHAE

Algorithm 2 COFHAE(X)

hierarchy, assignments = MIMOSA(X) # Algorithm 1
HAE_θ = init_hierarchical_autoencoder(hierarchy)
D_ψ = init_discriminator()
for $x, a \sim$ minibatch(X, assignments) **do**
  $a', z = \text{HAE}_\theta.\text{encode}(x; \tau)$ # Algorithm 8
  $x' = \text{HAE}_\theta.\text{decode}(\text{concat}(a', z))$ # normal NN
  $z' = \text{copy}(z)$
  for $i = 1..|z_0|$ **do**
    shuffle $z'_{:,i}$ over minibatch indices where $\texttt{on}(z_{:,i})$
  end for
  $\mathcal{L}_\theta = \sum_n \mathcal{L}_x(x'_n, x_n) + \lambda_1 \mathcal{L}_a(a'_n, a_n) - \lambda_2 \log \frac{D_\psi(z_n)}{1-D_\psi(z_n)}$
  $\mathcal{L}_\psi = \sum_n -\log D_\psi(z'_n) - \log(1 - D_\psi(z_n))$
  $\theta = \text{descent\_step}(\theta, \mathcal{L}_\theta)$
  $\psi = \text{descent\_step}(\psi, \mathcal{L}_\psi)$
end for
return HAE_θ

Our first stage, MIMOSA, gives us the hierarchy and assignments of data down it. In the second stage, COFHAE (**CO**nditionally **F**actorized **H**ierarchical **A**uto**E**ncoder, Algorithms 2 and 8), we learn an autoencoder that respects this hierarchy via (differentiable) masking operations that impose structure on flat representations.

Hierarchical Encoding: Instances x pass through a neural network encoder to an initial vector z_{pre}, whose dimensions correspond to all continuous variables in the hierarchy as well as the one-hot encoded categorical variables. Categorical dimensions (denoted a') pass through a softmax with temperature τ to softly mask z_{pre} based on the hierarchy.

Supervising Assignments: Hierarchical encoding outputs estimated assignments a'. We add a penalty $\mathcal{L}_a(a', a)$, weighted by λ_1, to make these close to MIMOSA values a.

Conditional Factorization: Kim & Mnih (2018) penalize the total correlation (TC) between dimensions of flat continuous representations z with two tricks. First, noting that TC is the KL divergence between $q(z)$ (the joint distribution of the encoded z) and $\bar{q}(z) \equiv \prod_{j=1}^{|z|} q(z_j)$ (the product of its marginals), they approximate samples from $\bar{q}(z)$ by randomly permuting the values of each z_i across batches (Arcones & Gine, 1992). Second, they approximate the KL divergence between the two distributions using the density ratio trick (Sugiyama et al., 2012) on an auxiliary discriminator $D_\psi(z)$, where $KL(q(z)||\bar{q}(z)) \approx \log \frac{D_\psi(z)}{1-D_\psi(z)}$ if $D_\psi(z)$ outputs accurate probabilities of z having been sampled from $\bar{q}$. We adopt a similar approach, except instead of permuting each z_i across the full batch $\mathcal{B}$, we only permute it where it is *active*, i.e. $\mathcal{B} \cap \texttt{on}(z_i)$ (defined using the hardened version of the mask). This approximates a hierarchical version of $\bar{q}(z)$ where each dimension distribution is a mixture of 0 and the product of its *active* marginals. $D_\psi(z)$ then lets us estimate the KL between this distribution and $q(z)$, which we penalize and weight with λ_2.

This approach presumes ground-truth continuous variables should be conditionally independent given categorical values, which is a major assumption. However, it is less strict than the assumption taken by many disentanglement methods, i.e. that continuous variables are independent marginally, and it may remain useful as an inductive bias.

6. Hierarchical Disentanglement Metrics

In this section, we develop metrics for quantifying how well learned representations and hierarchies match ground-truth.

6.1. Desiderata and Invariances

Our goal in designing metrics is to measure whether we have learned the "right representation," both in terms of global structure and specific variable correspondences. In an ideal world, we would measure whether a learned representation Z is identically equal to a ground-truth V. However, most existing disentanglement metrics are invariant to permutations, so that dimensions V_i can be reordered to match different Z_j, as well as univariate transformations, so that the values of Z_j do not need to be identical to V_i. In the case of methods like the SAP score (Kumar et al., 2017), these univariate transformations must be linear, but as the uniformity of scaling can be arbitrary, we permit general nonlinear transformations, as long as they are 1:1, or invertible. Also, in the hierarchical case, there are certain ambiguities about the right vertical placement of continuous variables. For example, on Spaceshapes, the `phase`, `shine`, and `angle` variables could all be "merged up" to a single top-level variable whose effect changes based on `shape`. Alternatively, `x` and `y` position could be "pushed down" and duplicated for each shape despite their analogous effects (see Fig. 14 for an illustration). Such "merge up" and "push down" transformations change the vector representation, but leave local dimensionality and the group structure of the hierarchy unchanged. We defer the problem of deciding the most natural vertical placement of continuous variables to future work, and make our main metrics invariant to them.

6.2. MIMOSA Metrics: H-error, Purity, Coverage

The first metric we use to evaluate MIMOSA is the **H-error**, which measures whether learned hierarchy $\hat{H}$ has the same essential structure as the ground-truth hierarchy H. To compute the H-error, we iterate over all possible paths p and p' down both H and $\hat{H}$, and attempt to pair them based on whether the minimum downstream dimensionality of p and p' matches at each respective node. The number of unpaired paths in either hierarchy is taken to be the H-error. This metric can only be 0 if both hierarchies have the same dimensionality structure, but is invariant to the "merge up" and "push down" operations described in §6.1.

The second MIMOSA metric is **purity**, which measures whether the assignments output by MIMOSA match ground-truth. To compute purity, we iterate through points assigned to each path $\hat{p}$ in $\hat{H}$, find the path p in H to which most of them belong, and then compute the fraction of points in $\hat{p}$ that belong to the majority p. Then we average these purity scores across $\hat{H}$, weighting by the number of points in $\hat{p}$. This metric only falls below 1 when we group together points with different ground-truth assignments.

The final metric we use to evaluate MIMOSA is **coverage**. Since MIMOSA discards small outlier components, it is possible that the final set of assignments will not cover the full training set. If almost all points are discarded this way, the other metrics may not be meaningful. As such, we measure coverage as the fraction of the training set which is not discarded. We note that hyperparameters can be tuned to ensure high coverage without knowing ground-truth assignments.

6.3. COFHAE Metrics: R^4 and R^4_c Scores

Per our desiderata, we seek to check whether every ground-truth variable V_i can be mapped invertibly to some learned dimension Z_j. As a preliminary definition, we say that a learned Z_j *corresponds* to a ground-truth V_i over some set $\mathcal{S} \subseteq \mathbb{R}$ if a bijection between them exists; that is,

$$\exists\, f(\cdot) : \mathcal{S} \to \mathbb{R} \text{ s.t. } f(V_i) = Z_j \text{ and } f^{-1}(Z_j) = V_i \quad (1)$$

We say that Z *disentangles* V if all V_i have a corresponding Z_j. To measure the extent to which bijections exist, we can simply try to learn them (over random splits of many paired samples of V_i and Z_j). Concretely, for each pair of learned and true dimensions, we train univariate models to map in both directions, compute their coefficients of determination (R^2), and take their geometric mean:

$$\begin{aligned} f &\equiv \min_{f \in \mathcal{F}} \mathbb{E}_{\text{train}} \left[(f(X) - Y)^2 \right] \\ R^2(X \to Y) &\equiv \mathbb{E}_{\text{test}} \left[1 - \frac{\sum (f(X) - Y)^2}{\sum (\mathbb{E}[Y] - Y)^2} \right] \\ R^2(X \leftrightarrow Y) &\equiv \sqrt{\lfloor R^2(X \to Y) \rfloor_+ \lfloor R^2(Y \to X) \rfloor_+}, \end{aligned} \quad (2)$$

where we average over train/test splits (we use 5), assume $\mathcal{F}$ is sufficiently flexible to contain the optimal bijection (we use gradient-boosted decision trees), and assume our dataset is large enough to reliably identify $f \in \mathcal{F}$. In the limit, $R^2(X \leftrightarrow Y)$ can only be 1 if a bijection exists, as any region of non-zero mass in the joint distribution of X and Y where this is false would imply $\mathbb{E}[(f(X) - Y)^2] > 0$ or $\mathbb{E}[(f(Y) - X)^2] > 0$. In the special case that Y is discrete rather than continuous, we use classifiers for f and accuracy instead of R^2, but the same argument holds.

To measure whether a *set* of variables Z disentangles another *set* of variables V, we check if, for each V_i, there is at least one Z_j for which $R^2(V_i \leftrightarrow Z_j) = 1$:

$$R^4(V, Z) \equiv \frac{1}{|V|} \sum_i \max_j R^2(V_i \leftrightarrow Z_j), \quad (3)$$

We call this the "right-representation" R^2, or R^4 score. Note that this metric is related to the existing SAP score (Kumar et al., 2017), except we allow for nonlinearity, require high R^2 in both directions, and take the maximum over each score column rather than the difference between the top two entries (which avoids assuming ground-truth is factorized).

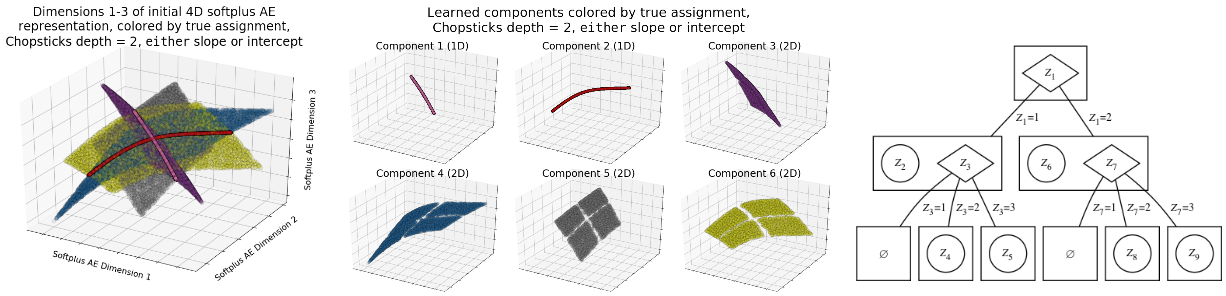


Figure 2. MIMOSA results for the depth-2 `either` version of Chopsticks, colored by ground-truth assignments. MIMOSA learns an initial 4D softplus AE representation (left), decomposes it into lower-dimensional components (middle), and infers a hierarchy (right). In this case, correspondence to ground-truth is very close (99.8% component purity, covering 93.7% of the training set, with the correct hierarchical relationships). Similar examples are shown for other datasets in Figs. 15-20 of the Appendix.

Although R^4 is useful for measuring correspondence between sets of variables that are both always active, it does not immediately apply to hierarchical representations unless inactive variables are represented somehow, e.g. as 0 (an arbitrary implementation decision that affects R^2 by changing $\mathbb{E}[Y]$). It also lacks invariance to merge-up and push-down operations. Instead, we seek *conditional correspondence* between V_i and a set of dimensions in Z, defined as

$$\begin{aligned}
&\text{for all } V_i \in \texttt{on}(V_i) \,\exists\, \mathcal{Z}_i = \{Z_j, Z_k, \ldots\} \text{ s.t.} \\
&\quad (a)\ V_i \text{ corresponds to } Z_j \text{ over } \texttt{on}(V_i) \cap \texttt{on}(Z_j), \\
&\quad (b)\ \texttt{on}(Z_j) \cap \texttt{on}(Z_k) = \emptyset \text{ for all } j \neq k, \text{ and} \\
&\quad (c)\ \textstyle\bigcup_{z \in \mathcal{Z}_i} \texttt{on}(z) = \texttt{on}(V_i),
\end{aligned} \tag{4}$$

or rather that we can find some tiling of $\texttt{on}(V_i)$ into regions where it corresponds 1:1 with different Z_j which are never active simultaneously. This allows for one Z_j to correspond to non-overlapping elements of V (e.g. merging up), as well as for one V_i to be modeled by non-overlapping elements of Z (e.g. pushing down).

We can then formulate a conditional R^4_c score which quantifies how closely conditional correspondence holds:

$$\begin{aligned}
R^2_c(V_i, g) \equiv \max\Bigg(&\max_{j \in g} \Big(R^2\big(V_i \leftrightarrow Z_j \big| \texttt{on}(V_i) \cap \texttt{on}(g)\big), \\
&\sum_{g' \in \texttt{children}(Z_j)} R^2_c(V_i, g') \frac{|\texttt{on}(V_i) \cap \texttt{on}(g')|}{|\texttt{on}(V_i)|} \Big) \Bigg),
\end{aligned}$$

for a dimension group g; the overall disentanglement is:

$$R^4_c(V \leftrightarrow Z) \equiv \frac{1}{|V|} \sum_{i=1}^{|V|} R^2_c(V_i, \texttt{root}(Z)). \tag{5}$$

In the special case that V and Z are flat, R^4_c reduces to R^4. We note that even for flat representations, the R^4 score may be a useful measure of disentanglement when ground-truth variables are not factorized.

7. Experimental Setup

Benchmarks: We ran experiments on nine benchmark datasets: Spaceshapes, and eight variants of Chopsticks (varying `slope`, `intercept`, `both`, and `either` at recursion depths of 2 and 3). See §4 for more details, and Fig. 8 for preliminary experiments on noisy data.

Algorithms: In addition to COFHAE with MIMOSA, we trained the following baselines: autoencoders (AE), variational autoencoders (Kingma & Welling, 2013) (VAE), the β-total correlation autoencoder (Chen et al., 2018) (TC-VAE), and FactorVAE (Kim & Mnih, 2018). We also ran COFHAE ablations using the ground-truth hierarchy and assignments, testing all possible combinations of loss terms and comparing conditional vs. marginal TC penalties; results are in Fig. 4. See §A.1 for training and model details.

Metrics: To evaluate hierarchies, we computed purity, coverage, and H-error (§6.2). Results are in Table 1. To measure disentanglement, we primarily use R^4_c (§6.3); results for all datasets and models are in Fig. 3. We also compute the following baseline metrics: the SAP score (Kumar et al., 2017) (SAP), the mutual information gap (Chen et al., 2018) (MIG, estimated using 2D histograms), the FactorVAE score (Kim & Mnih, 2018) (FVAE), and the DCI disentanglement score (Eastwood & Williams, 2018) (DCI). Most implementations were adapted from `disentanglement_lib` (Locatello et al., 2018). We also compute our marginal R^4 score. Results across metrics are shown for a subset of datasets and models in Fig. 5.

Hyperparameters: COFHAE is only given instances X, which complicates cross-validation. However, we can still tune parameters to ensure assignments a' match MIMOSA outputs a and reconstruction loss for x is low (which can fail to happen if the adversarial term dominates). Over a grid of τ in $\{\frac{1}{2}, \frac{2}{3}, 1\}$, λ_1 in $\{10, 100, 1000\}$, and λ_2 in $\{1, 10, 100\}$, we select the model with the lowest *training* reconstruction loss $\mathcal{L}_x$ from the $\frac{1}{3}$ with the lowest assignment loss $\mathcal{L}_a$. For MIMOSA, hyperparameters can be tuned to ensure high coverage (purity and H-error require side-information); see §A.3 for more. For β-TCVAE and FactorVAE, we show results for β=5 and γ=10, but test both across $\{5, 10, 25, 50\}$ in Fig. 11.

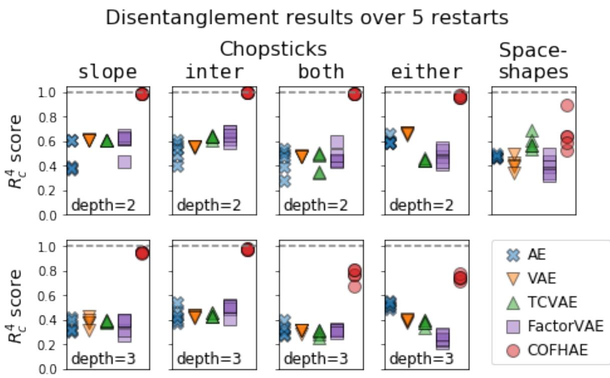


Figure 3. Hierarchical disentanglement results for representation learning methods (baselines and COFHAE + MIMOSA) over all nine datasets. COFHAE almost perfectly disentangles ground-truth on the six simplest versions of Chopsticks, with some degradations on the two most complex versions (with very deep hierarches) and on Spaceshapes (with a shallower hierarchy, but higher-dimensional inputs). Baseline methods were generally much more entangled, though β-TCVAE is competitive on Spaceshapes.

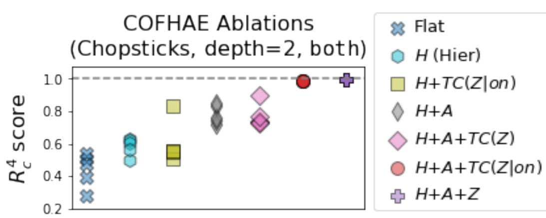


Figure 4. Ablation study for COFHAE on the depth-2 `both` version of Chopsticks (over 5 restarts). Hierarchical disentanglement is low for flat AEs (Flat); adding the ground-truth hierarchy H improves it (Hier H), as does also adding supervision for ground-truth assignments A (H+A). Adding a FactorVAE-style marginal TC penalty (H+A+$TC(Z)$) does not appear to help disentanglement, but making that TC penalty conditional (H+A+$TC(Z|on)$, i.e. COFHAE) brings it close to the near-optimal disentanglement of a hierarchical model whose latent representation is fully supervised (H+A+Z). However, the hierarchical conditional TC penalty fails to produce this same disentanglement without any supervision over assignments (H+$TC(Z)$).

8. Results and Discussion

MIMOSA consistently recovered the right hierarchies. Per Table 1, we consistently found the right hierarchy for all datasets except depth-3 `either`-Chopsticks, but even there results were close, generally recovering 12 of 14 possible hierarchy paths (see Fig. 20 for more details). Purity and coverage were also high, often near perfect as in Spaceshapes or depth-2 Chopsticks.

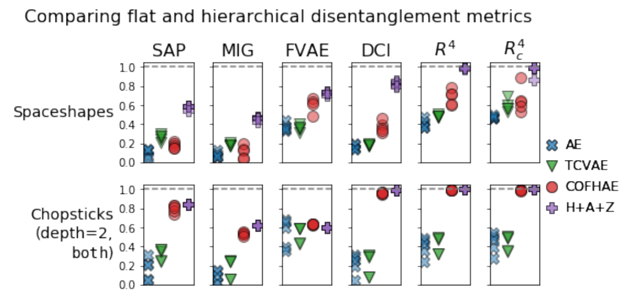


Figure 5. Comparison of disentanglement metrics across two datasets and four models. Only R^4 and R^4_c correctly and consistently award near-optimal scores to the supervised H+A+Z model.

COFHAE significantly outperformed baselines. Per Fig. 3, COFHAE R^4_c scores were near-perfect for 6 out of 9 datasets, and better than baselines on all. On Spaceshapes and the depth-3 `either` and `both` versions of Chopsticks, scores were slightly worse. Part of this suboptimality could be due to non-identifiability. For Spaceshapes and the `both` versions of Chopsticks, dimension group nodes contain multiple continuous variables, which even conditionally can be modeled by multiple factorized distributions (Locatello et al., 2018). However, optimization issues could also be at fault, as we do not see suboptimal R^4_c on Chopsticks until a depth of 3, and even supervised H+A+Z models occasionally fail to converge on Spaceshapes. Kim & Mnih (2018) note that the relatively low-dimensional discriminator used by FactorVAE is easier to optimize than the generally high-dimensional discriminators used in GANs, which are notoriously tricky to train (Mescheder et al., 2018). In our case, flattened hierarchy vectors can be high-dimensional (e.g. Fig. 21), and in any given batch, instances corresponding to different paths down the hierarchy may have different numbers of samples (potentially requiring larger batch sizes or stratified sampling to ensure sufficient coverage). Finally, alongside non-identifiability and optimization issues, MIMOSA errors (e.g. merge-up/push-down differences for Spaceshapes and suboptimal purity and coverage for Chopsticks) also may play a role, as evidenced by performance improvements in our full COFHAE ablations in Fig. 10. Despite all of these issues, COFHAE is still closer to optimal than any of our baseline algorithms.

R^4_c provides more insight into disentanglement than baselines. One way to evaluate an evaluation metric is to test it against a precisely known quantity. In this case, we know the H+A+Z model, whose encoder is supervised

MIMOSA Metric	Chopsticks, depth=2				Chopsticks, depth=3				Space-shapes
	`inter`	`slope`	`both`	`either`	`inter`	`slope`	`both`	`either`	
Purity	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	.98±0.0	.95±0.0	.94±0.0	.93±0.0	1.0±0.0
Coverage	.99±0.0	.99±0.0	.96±0.0	.93±0.0	.98±0.0	.98±0.0	.82±0.01	.75±0.01	1.0±0.0
H-error	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	2.6±1.34	0.0±0.0

Table 1. MIMOSA results across all datasets, with means and standard deviations across 5 restarts. In general, MIMOSA components contained points only from single ground-truth sets of paths (purity), were inclusive of most points in the training set (coverage), and resulting in perfectly accurate hierarchies (H errors), with the greatest or only exception being the Chopsticks depth-3 `either` dataset (where we tended to recover only 12 of its 14 possible paths).

to match ground-truth, should receive a near-perfect score. The only metrics to do this consistently are R^4 and R^4_c. Note that the DCI disentanglement score, based on the entropy of normalized feature importances from an estimator predicting single ground-truth factors from all learned dimensions, comes close. Intuitively, this metric could behave similarly to R^4 if its estimator was trained to be sparse (placing importance on as few dimensions as possible). However, using R^2s of univariate estimators is more direct, and also incorporates information from the DCI informativeness score.

Another way to evaluate an evaluation metric is to test whether quantitative differences capture salient qualitative differences. To this point, specifically to compare R^4 and R^4_c, we consider several examples in Fig. 12 and Fig. 13. First, we see that for the Spaceshapes COFHAE model in Fig. 12c, its R^4_c score (0.89) is higher than its R^4 (0.79). This increase is due to the fact that R^4 penalizes the "push-down" differences (§6.1) between the learned and true factors representing `x` and `y` position, while R^4_c is invariant to them. However, the overall increase is less dramatic than one might expect due to modest decreases in correspondence scores for other dimensions (e.g. $0.98 \to 0.89$ for `jetlen`), which occur because R^4_c is not biased by spurious equality between dimensions which are both inactive. Another example of a difference between R^4 and R^4_c (illustrating invariance to "merging up" rather than "pushing down") is for the Spaceshapes β-TCVAE in Fig. 12b. In this case, histograms show that one β-TCVAE variable (Z_3) corresponds closely to both moon `phase` and star `shine` (and to a lesser extent, `jetlen`), only one of which is active at a time. The R^4 score (0.47) assigns low scores to these correspondences, but R^4_c (0.69) properly factors them in.

COFHAE and MIMOSA subcomponents improve performance. Though COFHAE contains many moving parts, results in Fig. 4 and Fig. 10 suggest they all count. Autoencoders only achieve optimal disentanglement if provided with the hierarchy, assignments, and a conditional (not marginal) penalty on the TC of continuous variables; no partial subset suffices. In the Appendix, Fig. 9 shows ablations and sensitivity analyses for MIMOSA that validate its subcomponents are important as well.

9. Conclusion

In this work, we introduced the problem of hierarchical disentanglement, where ground-truth representation dimensions are organized into a tree and activated or deactivated based on the values of categorical dimensions. We presented benchmarks, algorithms, and metrics for learning and evaluating such hierarchical representations.

There are a number of promising avenues for future work. One is extending the method to handle a wider variety of underlying structures, e.g. non-hierarchical dimension DAGs, or integrating our method with object representation techniques to better model generative processes involving ordinal variables or unordered sets (Locatello et al., 2020b). Another is to better solve or understand hierarchical disentanglement as we have already formulated it, e.g. by improving robustness to noise, or providing a better theoretical understanding of identifiability and when we can guarantee methods will succeed. Finally, there are ample opportunities to apply these techniques to real-world cases that we expect to have hierarchical structure, such as causal inference, patient phenotype, or population genetics datasets.

More generally, we feel it is important for representation learning to move beyond flat vectors, and work towards explicitly modeling the rich structure contained in the real world. For a long time, many symbolic AI and cognitive science researchers have argued that AI progress should be evaluated not by improvements in accuracy or reconstruction error, but by how well we can build models that build their own interpretable models of the world (Lake et al., 2017). Our work takes steps in this direction.

References

Adams, R., Wallach, H., and Ghahramani, Z. Learning the structure of deep sparse graphical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Ahlqvist, E., Storm, P., Käräjämäki, A., Martinell, M., Dorkhan, M., Carlsson, A., Vikman, P., Prasad, R. B., Aly, D. M., Almgren, P., et al. Novel subgroups of adult-onset diabetes and their association with outcomes: a data-driven cluster analysis of six variables. *The Lancet Diabetes & Endocrinology*, 6(5):361–369, 2018.

Alvarez-Melis, D. and Jaakkola, T. S. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, 2018.

American Diabetes Association. Diagnosis and classification of diabetes mellitus. *Diabetes Care*, 2005.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. Optics: Ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2):49–60, 1999.

Arcones, M. A. and Gine, E. On the bootstrap of u and v statistics. *The Annals of Statistics*, pp. 655–674, 1992.

Bengio, Y. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pp. 1–37. Springer, 2013.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 2013.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in beta-vae. *arXiv:1804.03599*, 2018.

Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

Choi, J., Hwang, G., and Kang, M. Discond-vae: disentangling continuous factors from the discrete. *arXiv:2009.08039*, 2020.

Comon, P. Independent component analysis, a new concept? *Signal processing*, 1994.

Creager, E., Madras, D., Jacobsen, J.-H., Weis, M., Swersky, K., Pitassi, T., and Zemel, R. Flexibly fair representation learning by disentanglement. In *International Conference on Machine Learning*, 2019.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 2016.

Dupont, E. Learning disentangled joint continuous and discrete representations. In *Advances in Neural Information Processing Systems*, 2018.

Eastwood, C. and Williams, C. K. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

Esmaeili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J., and van de Meent, J.-W. Structured disentangled representations. *International Conference on Artificial Intelligence and Statistics*, 2019.

Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Jeong, Y. and Song, H. O. Learning discrete and continuous factors of data via alternating disentanglement. *arXiv:1905.09432*, 2019.

Kim, H. and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

Klindt, D. A., Schott, L., Sharma, Y., Ustyuzhaninov, I., Brendel, W., Bethge, M., and Paiton, D. Towards nonlinear disentanglement in natural data with temporal sparse coding. In *International Conference on Learning Representations*, 2021.

Kumar, A., Sattigeri, P., and Balakrishnan, A. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv:1711.00848*, 2017.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

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

Little, A. V., Lee, J., Jung, Y.-M., and Maggioni, M. Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale svd. In *IEEE/SP 15th Workshop on Statistical Signal Processing*, 2009.

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv:1811.12359*, 2018.

Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., and Tschannen, M. Weakly-supervised disentanglement without compromises. *arXiv:2002.02886*, 2020a.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020b.

Mahapatra, S. and Chandola, V. S-isomap++: multi manifold learning from streaming data. In *IEEE International Conference on Big Data*, 2017.

Mahler, B. I. Contagion dynamics for manifold learning. *arXiv:2012.00091*, 2020.

Marx, C., Phillips, R., Friedler, S., Scheidegger, C., and Venkatasubramanian, S. Disentangling influence: Using disentangled representations to audit model predictions. *Advances in Neural Information Processing Systems*, 2019.

Mathieu, M. F., Zhao, J. J., Zhao, J., Ramesh, A., Sprechmann, P., and LeCun, Y. Disentangling factors of variation in deep representation using adversarial training. *Advances in Neural Information Processing Systems*, 2016.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Mescheder, L., Geiger, A., and Nowozin, S. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, 2018.

Omohundro, S. M. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

Parascandolo, G., Kilbertus, N., Rojas-Carulla, M., and Schölkopf, B. Learning independent causal mechanisms. In *International Conference on Machine Learning*, 2018.

Ridgeway, K. A survey of inductive biases for factorial representation-learning. *arXiv:1612.05299*, 2016.

Schmidhuber, J. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

Siddharth, N., Paige, B., Van de Meent, J.-W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., and Torr, P. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, 2017.

Singh, K. K., Ojha, U., and Lee, Y. J. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Sorrenson, P., Rother, C., and Köthe, U. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *arXiv:2001.04872*, 2020.

Sugiyama, M., Suzuki, T., and Kanamori, T. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

Tino, P. and Nabney, I. Hierarchical gtm: Constructing localized nonlinear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):639–656, 2002.

Träuble, F., Creager, E., Kilbertus, N., Goyal, A., Locatello, F., Schölkopf, B., and Bauer, S. Is independence all you need? on the generalization of representations learned from correlated data. *arXiv:2006.07886*, 2020.

Wang, W., Huang, Y., Wang, Y., and Wang, L. Generalized autoencoder: A neural network framework for dimensionality reduction. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

A. Appendix

A.1. Training and Architecture Details

For Chopsticks, our encoders and decoders used two hidden layers of width 256, and our loss function $\mathcal{L}_x$ was defined as a zero-centered Gaussian negative log likelihood with $\sigma = 0.1$. For Spaceshapes, encoders and decoders used the 7-layer convolutional architecture from Burgess et al. (2018), and our loss function $\mathcal{L}_x$ was Bernoulli negative log likelihood. All models were implemented in Tensorflow. Code to reproduce experiments is at `https://github.com/dtak/hierarchical-disentanglement`.

For both models, the assignment loss $\mathcal{L}_a$ was set to mean-squared error, but only for assignments that were defined: we set undefined assignment components to -1, then let $\mathcal{L}_a(a, a') = \sum_i \mathbb{1}[a'_i \geq 0](a_i - a'_i)^2$.

All activation functions were set to ReLU $(\max(0, x))$, except in the case of the initial smooth autoencoder, where they were replaced with Softplus $(\ln(1 + e^x))$. This initial autoencoder was trained with dimensionality equal to one plus the maximum intrinsic dimensionality of the dataset. We investigate varying this parameter in Fig. 9 and find it can be much larger, and perhaps would have produced better results (though nearest neighbor calculation and local SVD computations would have been slower).

All models were trained for 50 epochs with a batch size of 256 on a dataset of size 100,000, split 90%/10% into train/test. We used the Adam optimizer with a learning rate starting at 0.001 and decaying by $\frac{1}{10}$ halfway and three-quarters of the way through training.

For COFHAE, we selected softmax temperature τ, the assignment penalty strength λ_1, and the adversarial penalty strength λ_2 based on *training set* reconstruction error and MIMOSA assignment accuracy. Splitting off a separate validation set was not necessary, as the most common problem we faced was poor convergence, not overfitting; the adversarial penalty would dominate and prevent the procedure from learning a model that could reconstruct X or A.

Specifically, for each restart, we ran COFHAE with τ in $\{\frac{1}{2}, \frac{2}{3}, 1\}$, λ_1 in $\{10, 100, 1000\}$, and λ_2 in $\{1, 10, 100\}$. We then selected the model with the lowest training MSE $\sum_n ||x_n - x'_n||_2^2$, but restricting ourselves to the 33.3% of models with the lowest assignment loss $\sum_n \mathcal{L}_a(a_n, a'_n)$.

For evaluating R^4 and R^4_c, we used gradient boosted decision trees, which were faster to train than neural networks.

A.2. Complexity and Runtimes

Per Fig. 7, the total runtime of our method is dominated by COFHAE, an adversarial autoencoder method which has the same complexity as FactorVAE (Kim & Mnih, 2018) (linear in dataset size N and number of training epochs, and strongly affected by GPU speed).

MIMOSA could theoretically take more time, however, as the complexity of constructing a ball tree (Omohundro, 1989) for nearest neighbor queries is $O(|Z|N \log N)$. As such, initial dimensionality reduction is critical; in our Spaceshapes experiments, $|Z|$ is 7, whereas $|X|$ is 4096.

Other MIMOSA steps can also take time. With a `num_nearest_neighbors` of k, the complexity of running local SVD on every point in the dataset is $O(N(|Z|^2k + |Z|k^2 + k^3))$, providing another reason to reduce initial dimensionality and keep neighborhood size manageable (though ideally k should increase with $|Z|$ to robustly learn local manifold directions). Iterating over the dataset in BuildComponent and computing cosine similarity will also have complexity at least $O(Nkd^3(d + |Z|))$ for components of local dimensionality d, and detecting component boundaries can actually have complexity $O(Nke^d)$ (if this is implemented, as in our experiments, by checking if projected points are contained in their neighbors' convex hulls—though we also explored a much cheaper $O(Nk^2d)$ strategy of checking for the presence of neighbors in all principal component directions that worked almost as well).

Although these scaling issues are worth noting, MIMOSA was still relatively fast in our experiments, where runtimes were dominated by neural network training (Fig. 7).

A.3. MIMOSA Hyperparameters

In this section, we list and describe all hyperparameters for MIMOSA, along with values that we used for our main results. We also present sensitivity analyses in Fig. 9.

MIMOSA initial autoencoder (Algorithm 1, line 1)

- `initial_dim` - the dimensionality of the initial smooth autoencoder. As Fig. 9 shows, this can be larger than the intrinsic dimensionality of the data, which MIMOSA will estimate. We defaulted to using the max. intrinsic dimensionality plus 1; in a real-world context where this information is not available, it can be estimated by reducing from `initial_dim` $= |X|$ until reconstruction error starts increasing.
- Training and architectural details appropriate for the data modality (e.g. convolutional layers for images). See §A.1 for our choices.

LocalSVD (Algorithm 3)

- `num_nearest_neighbors` - neighborhood size for LocalSVD and traversal; we used 40. Must exceed `initial_dim`; could replace with a search radius.
- `ransac_frac` - the fraction of neighbors to refit SVD. We used $2/3$. Note that we do not run traditional multi-step RANSAC (Fischler & Bolles, 1981), but a two-step approximation, where we define loss by aggregating reconstruction errors across dimensions. Another (slower but potentially more robust) option would be to iteratively refit SVD on the points with lowest reconstruction error at each dimension, and check if the resulting eigenvalues meet our cutoff criteria.
- `eig_cumsum_thresh` - the minimum fraction of variance SVD dimensions must explain to determine local dimensionality. We used 0.95. For noisy or sparse data, it might be useful to reduce this parameter.
- `eig_decay_thresh` - the minimum multiplicative factor by which SVD eigenvalues must decay to determine local dimensionality. We used 4. It might also be useful to reduce this parameter for sparse data.

Note that our LocalSVD algorithm can be seen as a faster version of Multiscale SVD (Little et al., 2009), which is used in an analogous way by Mahapatra & Chandola (2017), but would require repeatedly computing singular value decompositions over different search radii for each point.

BuildComponent (Algorithm 5)

- `cos_simil_thresh` - neighbors' local SVDs must be this similar to add to the component. This corresponds to the ϵ parameter from Mahapatra & Chandola (2017). We used 0.99 for Chopsticks and 0.95 for Spaceshapes; in general, we feel this is one of the most important parameters to tune, and should generally be reduced in the presence of noise or data scarcity.
- `contagion_num` - only add similar points to a manifold component when a threshold fraction of their

Algorithm 3 LocalSVD(Z)

Run SVD on Z (a design matrix of dimension `num_nearest_neighbors` by `initial_dim`)
if `ransac_frac` < 1 **then**
  for each dimension d from 1 to `initial_dim` $- 1$ **do**
    for each point z_n **do**
      Compute the reconstruction error for z_n using the only first d SVD dimensions
    end for
  end for
  Take the norm of reconstruction errors across dimensions, giving a vector of length `num_nearest_neighbors`
  Re-fit SVD on points whose error-norms are less than the $100 \times$ `ransac_frac` percentile value.
end if
for each dimension d from 1 to `initial_dim` $- 1$ **do**
  Check if the cumulative sum of the first d eigenvalues is at least `eig_cumsum_thresh`
  Check if the ratio of the dth to the $d + 1$st eigenvalue is at least `eig_decay_thresh`
  if both of these conditions are true **then**
    return only the first d SVD components
  end if
end for
return the full set of SVD components otherwise

Algorithm 4 TangentPlaneCos(U, V)

if U and V are equal-dimensional **then**
  return $|\det(U \cdot V^T)|$
else
  return 0
end if

Algorithm 5 BuildComponent(z_i, neighbors, svds)

Initialize component to z_i and neighbors z_j not already in other components where $\mathrm{TangentPlaneCos}(\mathrm{svds}_i, \mathrm{svds}_j) \geq$ `cos_simil_thresh` (Algorithm 4).
while the component is still growing **do**
  Add all points z_k for which at least `contagion_num` of their neighbors z_ℓ are already in the component with $\mathrm{TangentPlaneCos}(\mathrm{svds}_k, \mathrm{svds}_\ell) \geq$ `cos_simil_thresh`.
  Skip adding any z_k already in another component.
end while
return the set of points in the component

Algorithm 6 MergeComponents(components, svds)

Discard components smaller than `min_size_init`.
for each component c_i **do**
  Construct a local ball tree for the points in c_i.
  Set c_i.edges to points not contained in the convex hull of their neighbors in local SVD space.
end for
Initialize edge overlap matrix M of size |components| by |components| to 0.
for each ordered pair of equal-dimensional components (c_i, c_j) **do**
  Set M_{ij} to the fraction of points in c_i.edges for which the closest point in c_j.edges has local SVD tangent plane similarity above `cos_simil_thresh`.
end for
Average M with its transpose to symmetrize.
Merge all components $c_i \neq c_j$ of equal dimensionality d where $M_{ij} \geq$ `min_common_edge_frac`(d).
Discard components smaller than `min_size_merged`.
return the merged set of components

Algorithm 7 ConstructHierarchy(components)

for each component c_i **do**
  Set c_i.neighbor_lengthscale to the average distance of points to their nearest neighbors inside the component (computed using the local ball tree from Algorithm 6)
end for
for each pair of different-dimensional components (c_i, c_j), c_i higher-dimensional **do**
  Compute the average distance from points in c_i to their nearest neighbors in c_j (via ball tree).
  Divide this average distance by $c_i.\text{neighbor\_lengthscale}$.
  if $\text{the resulting ratio} \leq$ `neighbor_lengthscale_mult` **then**
    Set $c_j \in c_i$ (c_j is enclosed by c_i)
  end if
end for
Create a root node with edges to all components which do not enclose others.
Transform the component enclosure DAG into a tree (where enclosing components are children of enclosed components) by deleting edges which:

1. are redundant because an intermediate edge exists, e.g. if $c_1 \in c_2 \in c_3$, we delete the edge between c_1 and c_3.
2. are ambiguous because a higher-dimensional component encloses multiple lower-dimensional components (i.e. has multiple parents). In that case, preserve only the edge with the lowest distance ratio.

Convert the resulting component enclosure tree into a dimension hierarchy:

1. If the root node has only one child, set it to be the root. Otherwise, begin with a dimension group with a single categorical dimension whose options point to groups containing each child.
2. For the rest of the component tree, add continuous dimensions until the total number of continuous dimensions up to the root equals the component's dimensionality.
3. If a component has children, add a categorical dimension that includes those child groups as options (recursing down the tree), along with an empty group ($\boxed{\emptyset}$) option.

return the dimension hierarchy

Algorithm 8 HAE_θ.encode$(x; \tau)$

Encode x using any neural network architecture as a flat vector z_{pre}, with size equal to the number of continuous variables plus the number of categorical options in $\text{HAE}_\theta.\text{hierarchy}$.
Associate each group of dimensions in the flat vector with variables in the hierarchy.
For all of the categorical variables, pass their options through a softmax with temperature τ.
Use the resulting vector to mask all components of z_{pre} corresponding to variables *below* each option in $\text{HAE}_\theta.\text{hierarchy}$.
return the masked representation, separated into discrete a', continuous z, as well as the mask m (for determining active dimensions later).

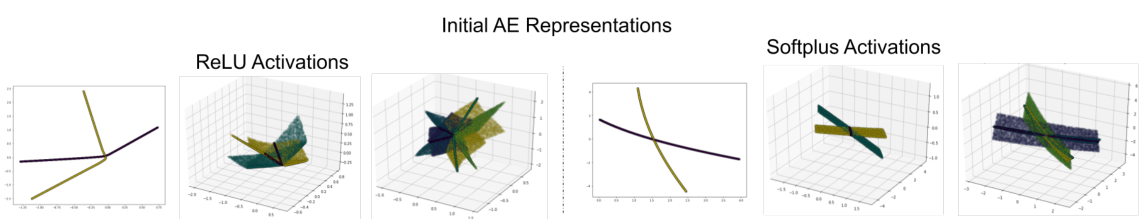


Figure 6. Comparison of the latent spaces learned by MIMOSA initial autoencoders with ReLU (top) vs. Softplus (bottom) activations. Each plot shows encoded Chopsticks data samples colored by their ground-truth location in the dimension hierarchy. Because ReLU activations are non-differentiable at 0, the resulting latent manifolds contain sharp corners, which makes it difficult for MIMOSA's local SVD procedure to merge points together into the correct components.

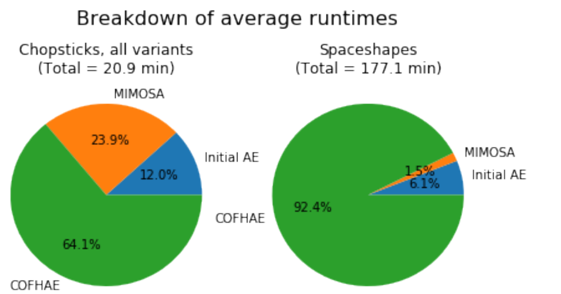
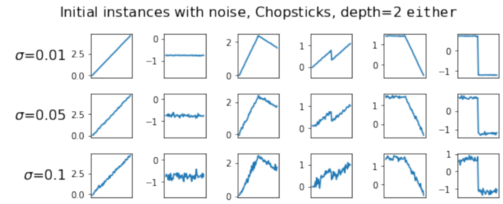
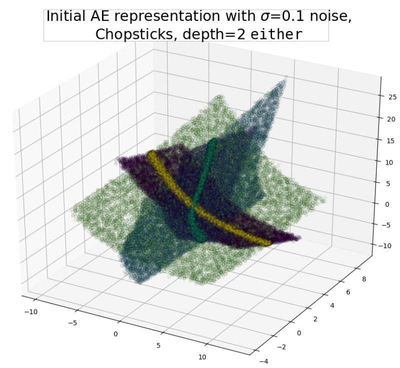


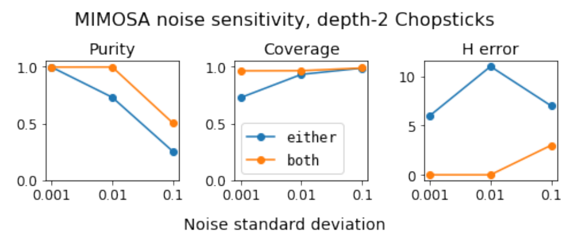
Figure 7. Mean runtimes and percentage breakdowns for COFHAE and MIMOSA on Chopsticks and Spaceshapes, based on Tensorflow implementations running on single GPUs (exact model varies between Tesla K80, Tesla V100, GeForce RTX 2080, etc). Runtimes tend to be dominated by COFHAE, which is similar in complexity to existing adversarial methods (e.g. FactorVAE).



(a) Chopsticks Xes corrupted by Gaussian noise.



(b) Effect of noise on initial autoencoder Zs.



(c) Effect of noise on MIMOSA for two variants.

Figure 8. Illustration of the sensitivity of MIMOSA to data noise. In preliminary experiments, we find that noise poses the greatest problem for identifying the lowest-dimensional components, e.g. the 1D components in (b) that end up being classified as 2D or 3D. Tuning parameters would help, but we lack labels to cross-validate.

neighbors have already been added. This is useful for robustness, and corresponds to the T parameter from Mahler (2020) (but expressed as a number rather than a fraction). We used 5 for Chopsticks and 3 for Spaceshapes. Values above 20% of `num_nearest_neighbors` will likely produce poor results, and we found the greatest increases in robustness just going from 1 (or no contagion dynamics) to 2.

MergeComponents (Algorithm 6)

- `min_size_init` - discard initial components smaller than this. We used 0.02% of the dataset size, or about 20 points. This parameter helps speed up the algorithm (by reducing the number of pairwise comparisons) and avoids merges through single-point components.
- `min_size_merged` - discard merged components smaller than this. We used 2% of the dataset size, or about 2000 points. This parameter helps exclude spurious interstitial points that appear at boundaries where low-dimensional components intersect.
- `min_common_edge_frac(d)` - the minimum fraction of edges that two manifold components must share in common to merge, as a function of dimensionality d. We used $2^{-d-1}+2^{-d-2}$; this is based on the idea that two neighboring (possibly distorted) hypercubes of dimension d should match on one of their sides; since they have 2^d sides, the fraction of matching edge points would be 2^{-d}. However, for robustness (as not all components will be hypercubes, and even then some edge points may not match), we average this with the smaller fraction that a $d{+}1$ dimensional hypercube would need, or 2^{-d-1}, for our resulting $2^{-d-1}+2^{-d-2}$. We found that this choice was not critical in preliminary experiments, as matches were common for components with the same true assignments and rare for others, but it could become more important for sparse or noisy data.

ConstructHierarchy (Algorithm 7)

- `neighbor_lengthscale_mult` - the threshold for deciding whether a higher-dimensional component "encloses" a lower-dimensional component, expressed as a ratio of (1) the average distance from lower-dimensional component points to their nearest neighbors in the higher-dimensional component (inter-component distance), to (2) the average distance of points in the higher-dimensional component to their nearest neighbors in that same component (intra-component distance). We used 10, which we found was robust for our benchmarks, though it may need to be increased if ground-truth components are higher-dimensional than those in our benchmarks.

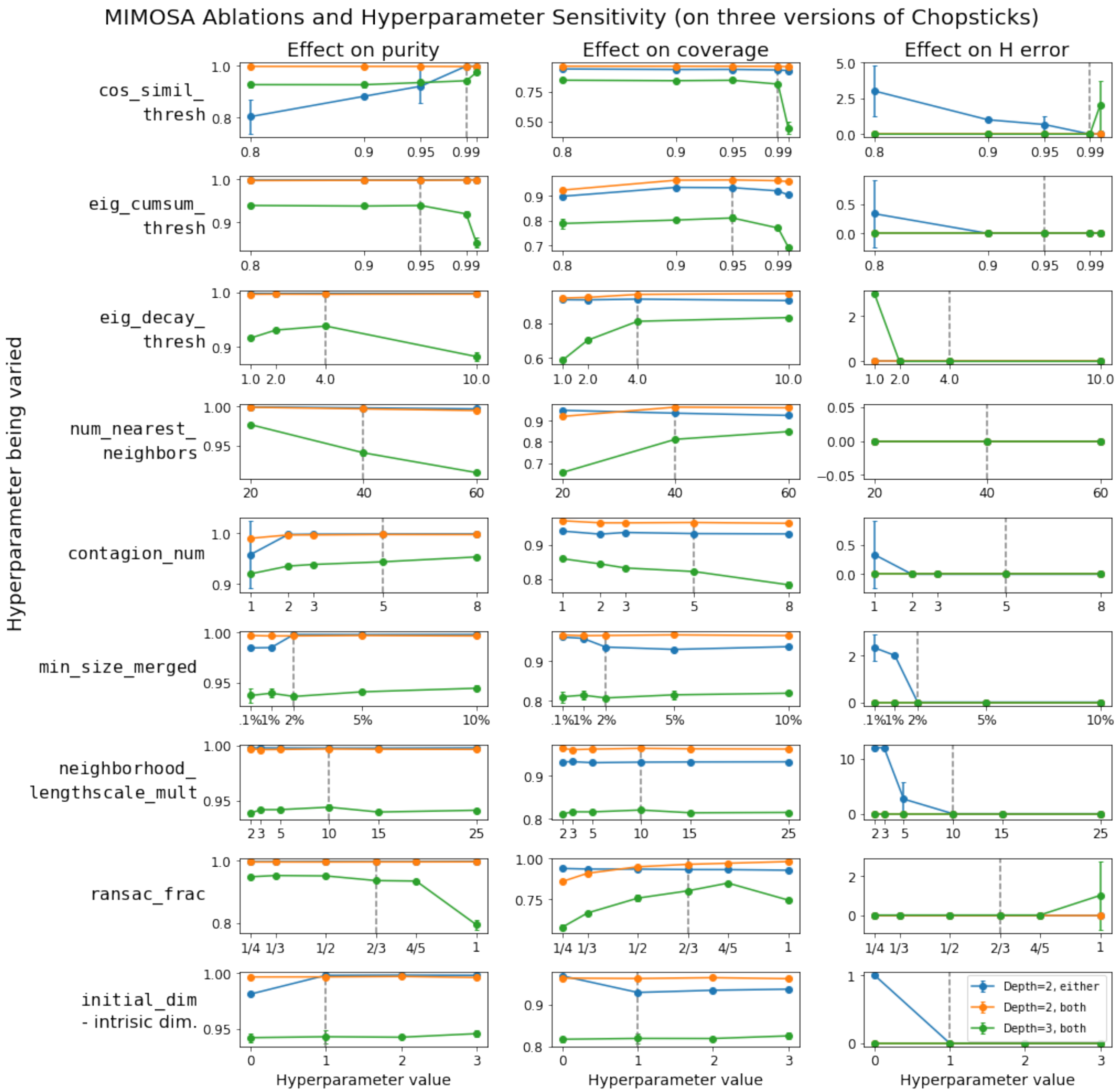


Figure 9. Effect of varying different hyperparameters (and ablating different robustness techniques) on MIMOSA. Default values are shown with vertical gray dotted lines, and for each hyperparameter (top to bottom), average coverage (left), purity (middle), and H error (right) when deviating from defaults are shown for three versions of the Chopsticks dataset. Results suggest both a degree of robustness to changes (degradations tend not to be severe for small changes), but also the usefulness of various components; for example, results markedly improve on some datasets with `contagion_num`>1 and `ransac_frac`<1 (implying contagion dynamics and RANSAC both help). Many parameters exhibit tradeoffs between component purity and dataset coverage.

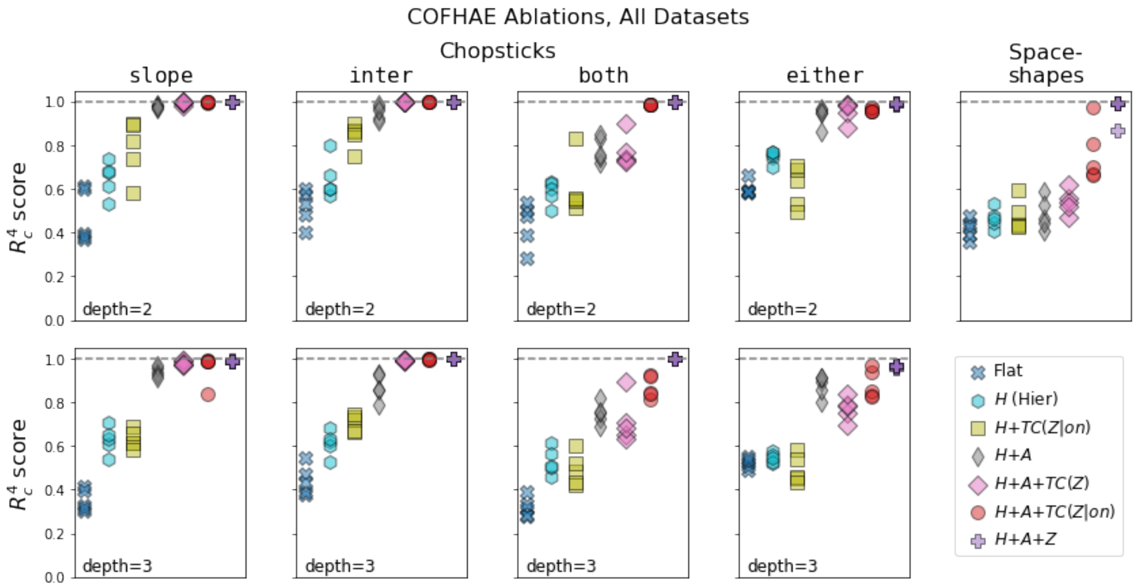


Figure 10. A fuller version of main paper Fig. 4 showing COFHAE ablations on all datasets. Hierarchical disentanglement tends to be low for flat AEs (Flat), better with ground-truth hierarchy H (Hier H), and even better after adding supervision for ground-truth assignments A (H+A). Adding a FactorVAE-style marginal TC penalty (H+A+$TC(Z)$) sometimes helps disentanglement, but making that TC penalty conditional (H+A+$TC(Z|on)$, i.e. COFHAE) tends to help more, bringing it close to the near-optimal disentanglement of a hierarchical model whose latent representation is fully supervised (H+A+Z). Partial exceptions include the hardest three datasets (Spaceshapes and depth-3 compound Chopsticks), where disentanglement is not consistently near 1; this may be due to non-identifiability or adversarial optimization difficulties.

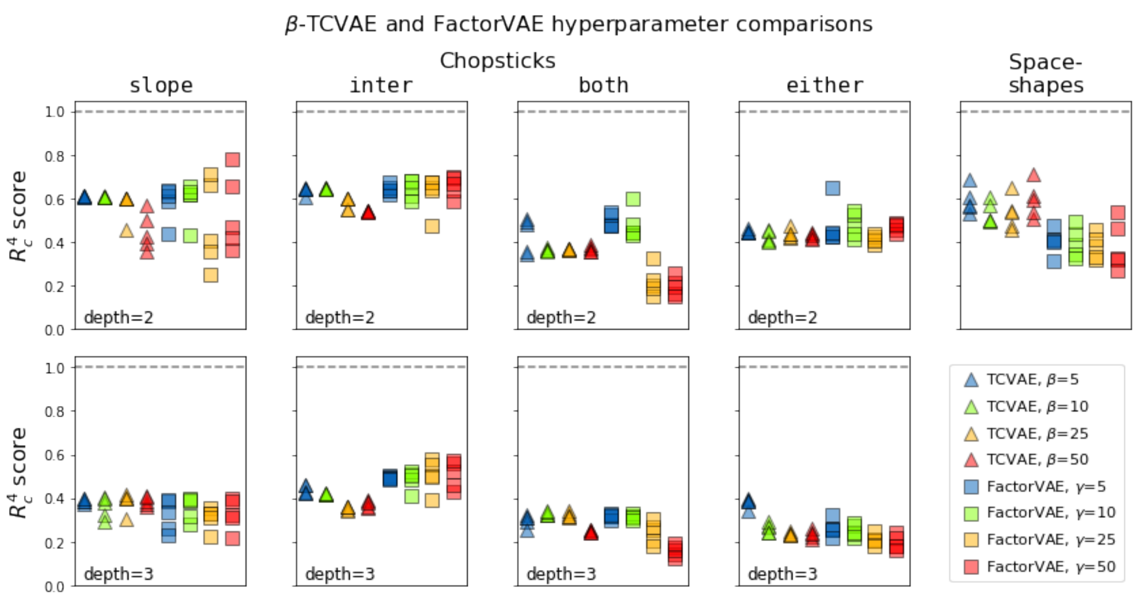
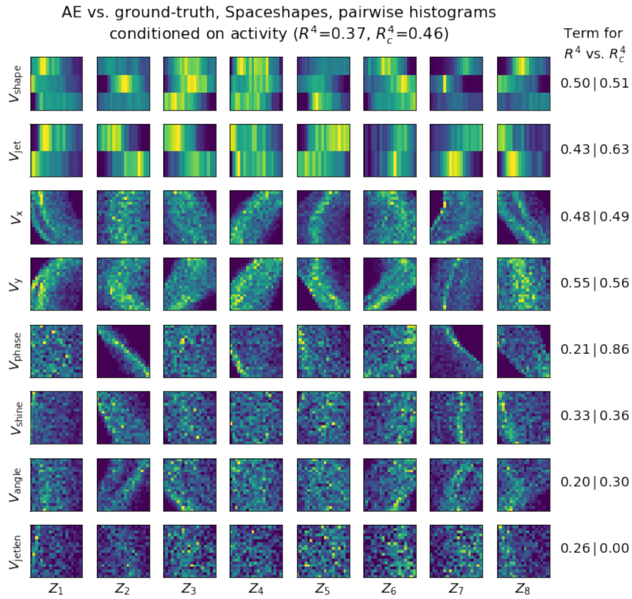
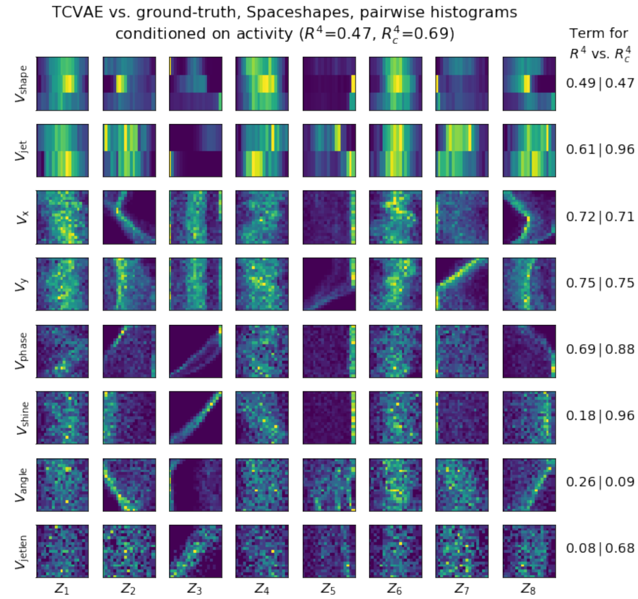


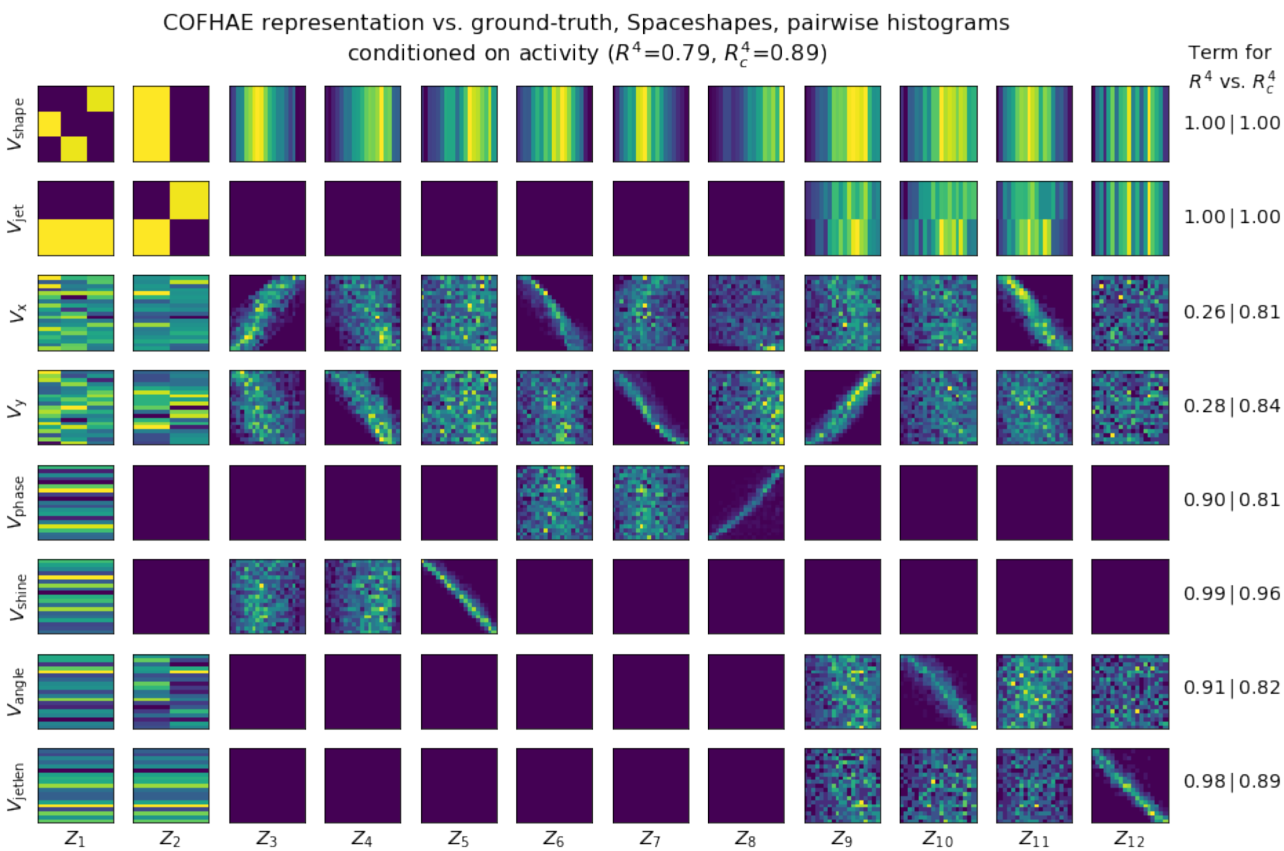
Figure 11. Varying disentanglement penalty hyperparameters for baseline algorithms (TCVAE and FactorVAE). In contrast to COFHAE, no setting produces near-optimal disentanglement, even sporadically.



(a) AE pairwise histograms and R^4/R_c^4 scores



(b) TCVAE pairwise histograms and R^4/R_c^4 scores



(c) COFHAE pairwise histograms and R^4/R_c^4 scores

Figure 12. Pairwise histograms of ground-truth vs. learned variables for a flat autoencoder (top left), β-TCVAE (top right), and the best-performing run of COFHAE (bottom) on Spaceshapes. Histograms are conditioned on both variables being active, and dimension-wise components of the R_c^4 score are shown on the right. β-TCVAE does a markedly better job disentangling certain components than the flat autoencoder, but in this case, COFHAE is able to fully disentangle the ground-truth by modeling the discrete hierarchical structure. See Fig. 13 for a latent traversal visualization.

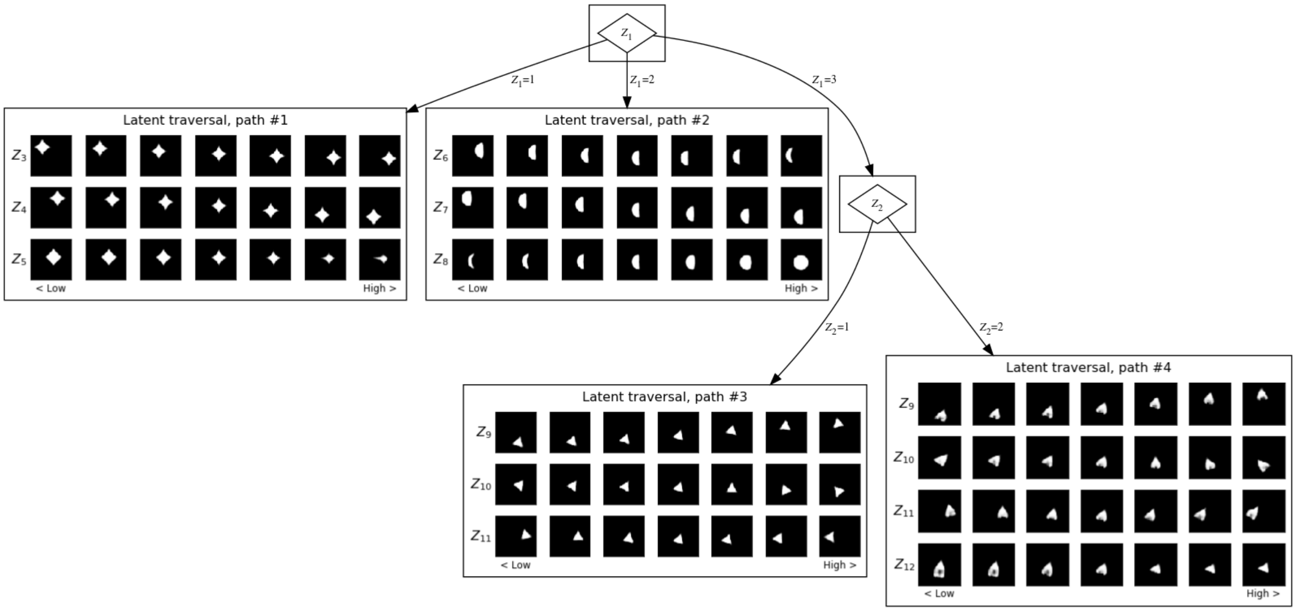


Figure 13. Hierarchical latent traversal plot for the Spaceshapes COFHAE model shown in Fig. 12c. Individual latent traversals show the effects of linearly sweeping each *active* dimension from its 1st to 99th percentile value (center column shows the same input with intermediate values for all active dimensions). Consistent with Fig. 12c, the model is not perfectly disentangled, though primary correspondences are clear: star `shine` is modeled by Z_5, moon `phase` is modeled by Z_8, ship `angle` is modeled by Z_{10}, ship `jetlen` is modeled by Z_{12}, and (`x`, `y`) are modeled by (Z_3, Z_4), (Z_6, Z_7), and (Z_{11}, Z_9) respectively for each shape.

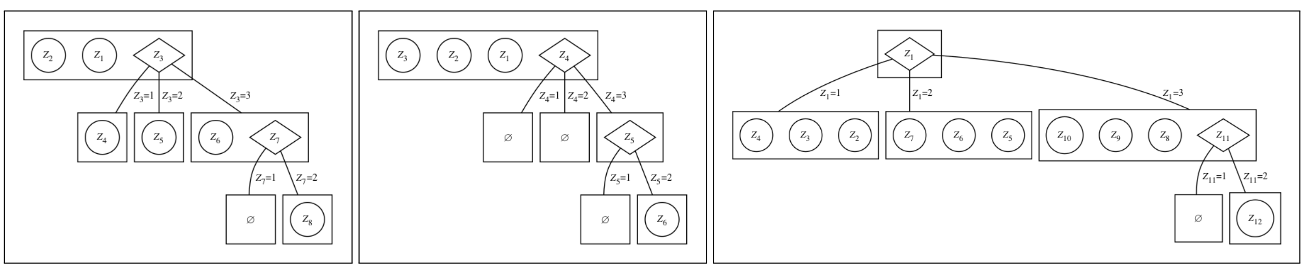


Figure 14. Three different potential hierarchies for Spaceshapes which all have the same structure of variable groups and dimensionalities, but with different distributions of continuous variables across groups. The ambiguity in this case is that the continuous variable that modifies each shape (`phase`, `shine`, `angle`) could either be a child of the corresponding shape category, or be "merged up" and combined into a single top-level continuous variable that controls the shape in different ways based on the category. Alternatively, the location variables `x` and `y` could instead be "pushed down" from the top level and duplicated across each shape category. In each of these cases, the learned representation still arguably disentangles the ground-truth factors—in the sense that for any fixed categorical assignment, there is still 1:1 correspondence between all learned and ground-truth continuous factors. We deliberately design our R^4_c and H-error metrics in §6 to be invariant to these transformations, leaving this specific disambiguation to future work.

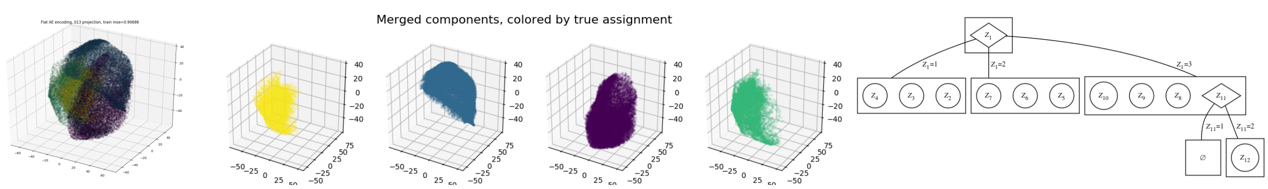


Figure 15. MIMOSA-learned initial encoding (left), components (middle), and hierarchy (right) for Spaceshapes. Initial points are in 7 dimensions and projected to 3D for plotting. Three identified components are 3D and one is 4D. Analogue of Fig. 2 in the main text.

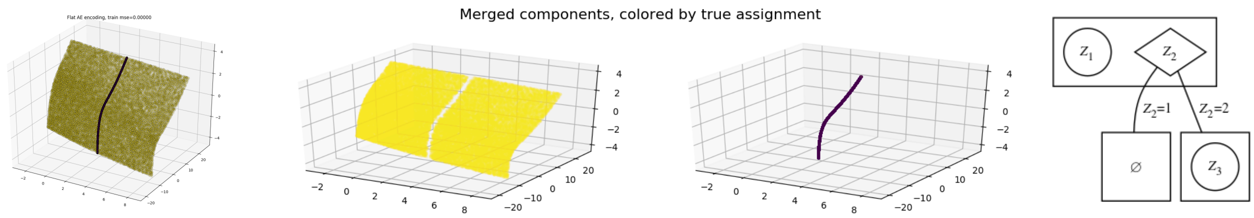


Figure 16. MIMOSA-learned initial encoding (left), 2D and 1D components (middle), and hierarchy (right) for depth-2 Chopsticks varying the slope. Analogue of Fig. 2 in the main text.

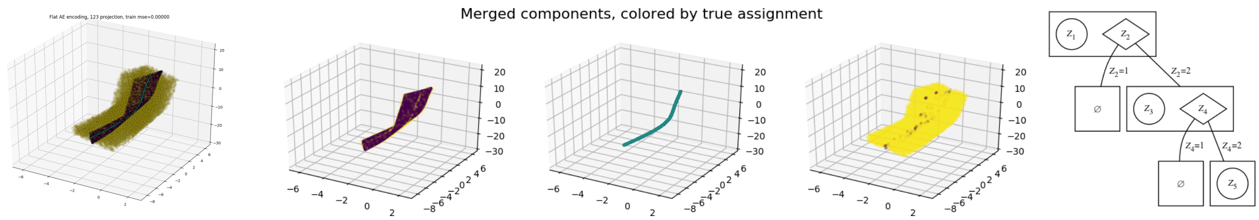


Figure 17. MIMOSA-learned initial encoding (left), 2D, 1D, and 3D components (middle), and hierarchy (right) for depth-3 Chopsticks varying the slope. Initial points are in 4 dimensions and projected to 3D for plotting. Analogue of Fig. 2 in the main text.

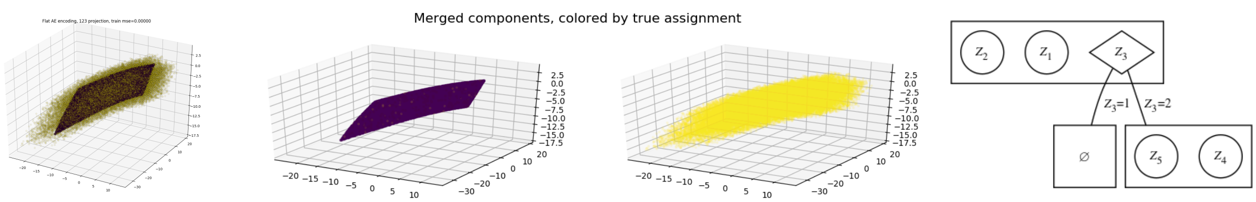


Figure 18. MIMOSA-learned initial encoding (left), 2D and 4D components (middle), and hierarchy (right) for depth-2 Chopsticks varying `both` slope and intercept. Initial points are in 5 dimensions and projected to 3D for plotting. Analogue of Fig. 2 in the main text.

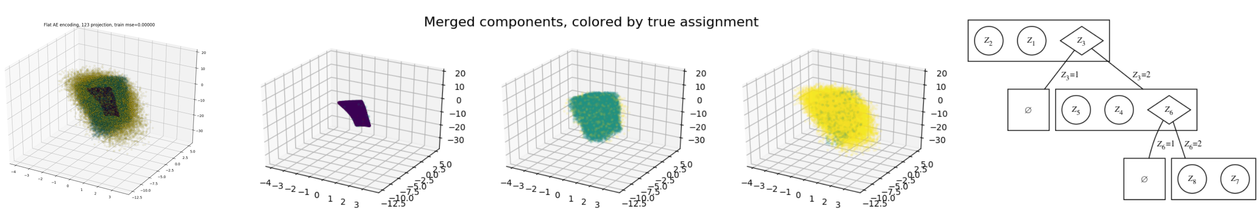


Figure 19. MIMOSA-learned initial encoding (left), 2D, 4D, and 6D components (middle), and hierarchy (right) for depth-2 Chopsticks varying `both` slope and intercept. Initial points are in 7 dimensions and projected to 3D for plotting. Analogue of Fig. 2 in the main text.

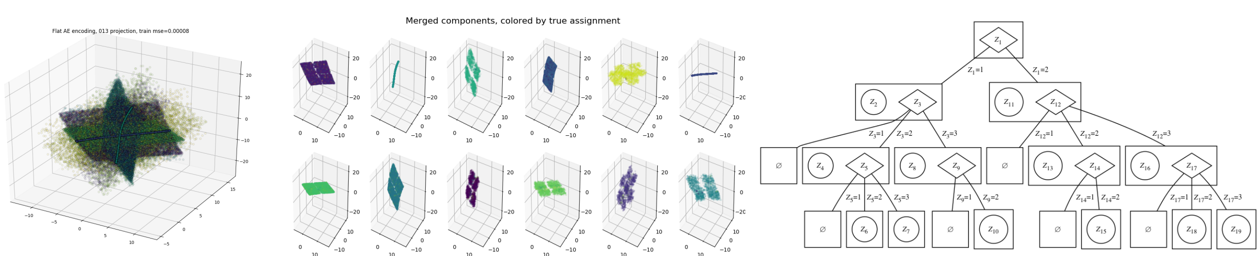


Figure 20. MIMOSA-learned initial encoding (left), 1D-3D components (middle), and hierarchy (right) for depth-3 Chopsticks varying either slope or intercept. Note that the learned hierarchy is not quite correct (two nodes at the deepest level are missing). Initial points are in 5 dimensions and projected to 3D. Analogue of Fig. 2.

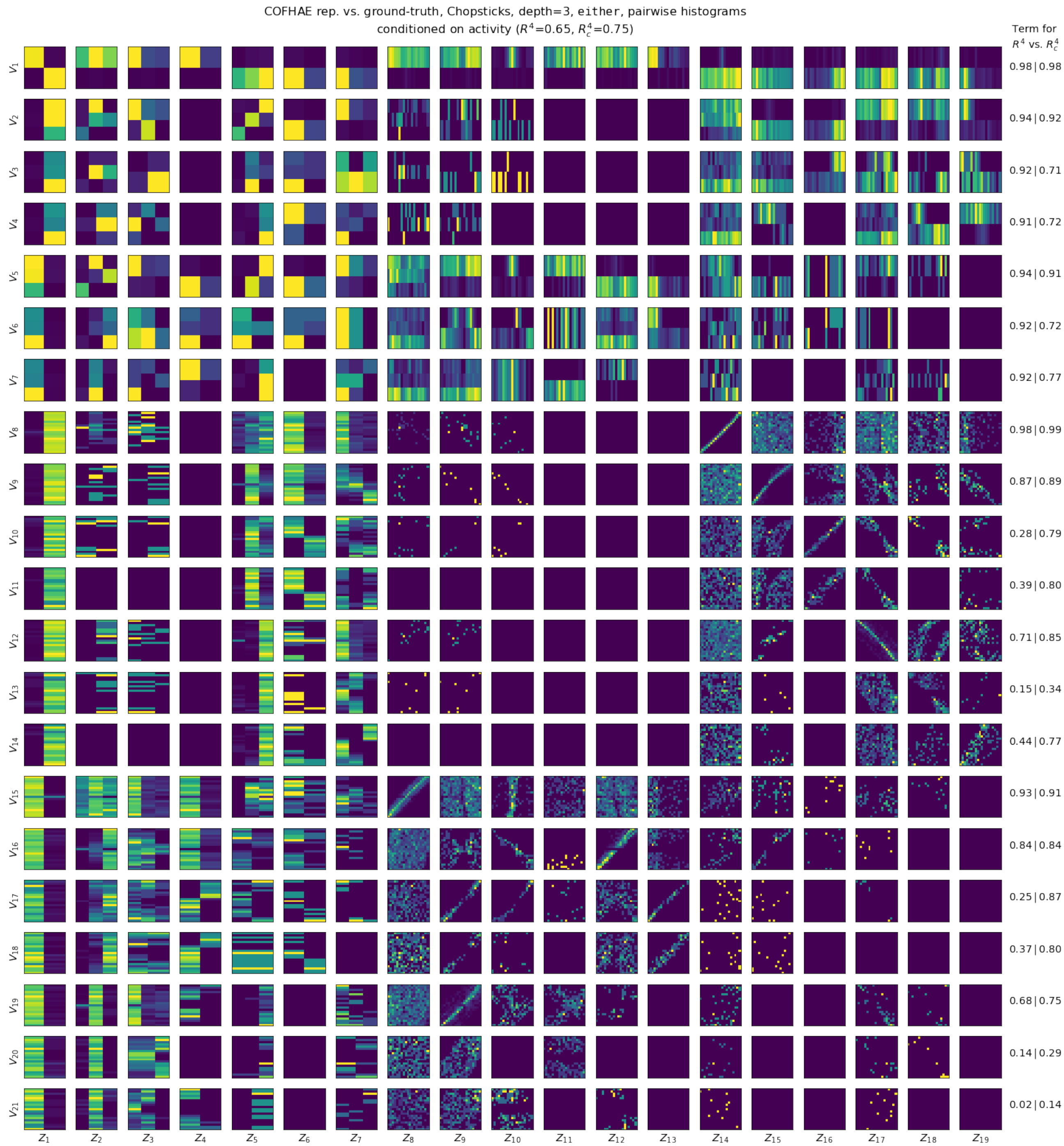


Figure 21. Pairwise histograms of ground-truth vs. learned variables for COFHAE on the most complicated hierarchical benchmark (Chopsticks at a recursion depth of 3 varying `either` slope or intercept). Histograms are conditioned on both variables being active, and dimension-wise components of the R^4_c score are shown on the right. Despite the depth of the hierarchy, COFHAE representations model it fairly well.