# Critical Percolation as a Synthetic Data Model for Interpretability

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Neural networks learn features that reflect the hierarchical, multi-scale structure of natural data. Synthetic datasets used to evaluate interpretability methods typically lack this structure, limiting their value as realistic toy models. To close this gap, we introduce a family of synthetic datasets consisting of hierarchical functions defined on critical mean-field percolation clusters embedded in a high-dimensional data space. The percolation data consists of sparse, low-dimensional fractal clusters with a power-law size distribution. Latent variables modeling a taxonomic hierarchy generate each data point's target value. The data model is analytically tractable with known critical exponents that fix its properties without requiring hyperparameter tuning. We leverage a mapping between percolation clusters, random trees, and additive coalescence to propose an almost linear-time algorithm to jointly sample a random tree and its hierarchical latent decomposition, enabling data generation at arbitrary scale. Using probing experiments, we find that the model's ground-truth latent variables can be linearly decoded from neural network activations. Together, sparsity, self-similarity, power-law statistics, and analytical tractability make critical percolation a principled testbed for interpretability research.

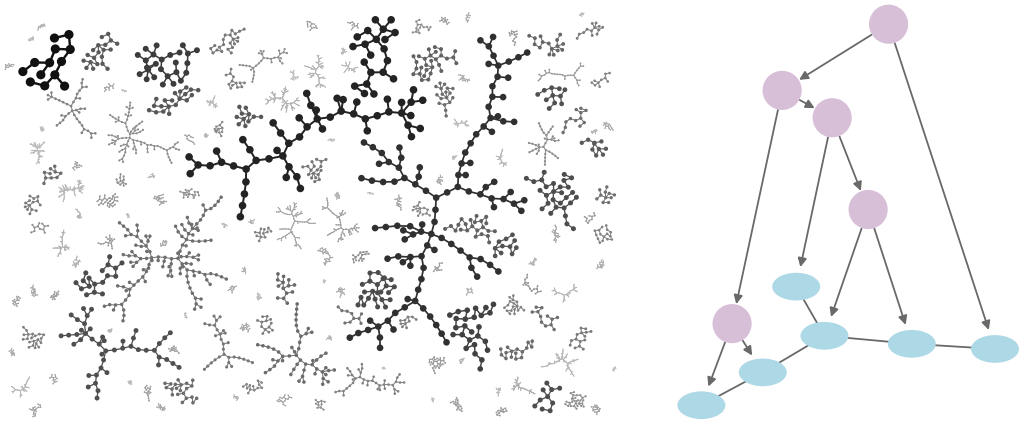
## 1 Introduction

Deep neural networks have achieved extraordinary success across hard real-world tasks [LeCun et al., 2015, Krizhevsky et al., 2012, Brown et al., 2020, Jumper et al., 2021]. To efficiently achieve strong predictive performance, a learning system must represent features that match the relevant latent factors that explain the data [Bengio et al., 2013]. By developing tractable data models that capture common structural properties of natural data, we can better understand AI systems.

Data must have structure to be efficiently learnable, due to the curse of dimensionality [Bellman, 1961]. For example, approximating a generic $d$-dimensional continuous function with uniform error $\epsilon$ requires $O\left((1/\epsilon)^d\right)$ samples, which is intractable when $d$ is large. Since both humans and machines learn to perform real-world tasks, natural data distributions must be highly structured.

Researchers have invoked a variety of structural properties of data to explain why deep learning succeeds. One important perspective comes from mechanistic interpretability, which seeks to reverse-engineer the features and mechanisms learned by neural networks [Olah et al., 2020, Elhage et al., 2022, Bereska and Gavves, 2024]. By decomposing dense representations into a large number of sparsely activating latent features, sparse autoencoders (SAEs) successfully extract interpretable features from the activations of large language models (LLMs) [Elhage et al., 2022, Cunningham et al., 2023, Bricken et al., 2023, Templeton et al., 2024, Gao et al., 2024, Lieberum et al., 2024]. These methods succeed because the distribution of natural language tasks is *sparse*: concepts relevant for prediction occur, and co-occur, rarely. The sparsity of learned features reflects the sparsity of data.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.



(a) (b)

Figure 1: The percolation data model. (a) Inputs are distributed as self-similar fractal clusters with power-law sizes. (b) Targets are generated by hierarchical latent variables decomposing each cluster.

Besides sparsity, real-world concepts exhibit *hierarchical* organization. Such hierarchies are either compositional, relating wholes to their parts, or taxonomic, relating classes to subclasses [Woods, 1975, Brachman, 1983, Miller, 1995]. In mechanistic interpretability research, hierarchical structure has been probed via the pathologies of SAEs. If data were purely sparse, increasing an SAE's size would simply recover additional interpretable latent features. In reality, previously coarse-grained latents split into fine-grained ones, a phenomenon known as feature splitting [Bricken et al., 2023]. Related issues are feature absorption and composition, in which at larger SAE sizes previously interpretable latents fragment into special cases or into compositional features, respectively [Chanin et al., 2024, Leask et al., 2025]. These phenomena suggest that learned LLM features have hierarchical structure, reflecting the hierarchical organization of natural concepts. Indeed, SAE variants designed to recover hierarchically organized latents can achieve excellent reconstruction while recovering interpretable features at different levels of abstraction [Bussmann et al., 2025, Costa et al., 2025].

Compositional hierarchical structure in data can be modeled using a probabilistic context-free grammar (PCFG) [Allen-Zhu and Li, 2023a, Garnier-Brun et al., 2024, Lubana et al., 2024, Menon et al., 2025, Cagnetta et al., 2024, Cagnetta and Wyart, 2024, Cagnetta et al., 2025, Sclocchi et al., 2025]. Recursive application of a PCFG's production rules yields compositional hierarchical structure creating long-range correlations among the observed features or tokens. However, data models that describe taxonomic hierarchical structure have been comparatively less well studied.

Another influential idea is that natural datasets have low *intrinsic dimension*. If all natural data samples are supported on a low-dimensional data manifold, representations can be highly compressed and the curse of dimensionality curtailed [Bengio et al., 2013, Goldt et al., 2020]. Many methods exist to estimate the intrinsic dimension of data embedded in a higher-dimensional ambient space [Grassberger and Procaccia, 1983, Levina and Bickel, 2004, Facco et al., 2017, Binnie et al., 2025].

Further insight comes from neural scaling laws, which suggest the presence of *power laws* in data. [Hestness et al., 2017, Henighan et al., 2020, Kaplan et al., 2020, Hoffmann et al., 2022]. Analyses based on high-dimensional regression typically model the kernel spectrum as a power law [Spigler et al., 2020, Bordelon et al., 2020, Maloney et al., 2022, Bahri et al., 2024, Atanasov et al., 2024]. Each of the previously mentioned data properties can produce power laws. If the distribution of sparse latent features or subtasks is heavy-tailed, learning them in order of frequency or importance translates into power-law scaling [Hutter, 2021, Michaud et al., 2023, Nam et al., 2024, Pan et al., 2025, Brill, 2025a, Michaud et al., 2025]. Models of compositional hierarchical data predict data-limited scaling laws connected to the resolved context horizon [Cagnetta et al., 2025, 2026]. Finally, low intrinsic dimension naturally yields efficient power-law scaling with exponent $1/D$, where $D$ is the intrinsic dimension $D \ll d$ [Spigler et al., 2020, Sharma and Kaplan, 2022, Bahri et al., 2024].

A fractal model unites all of these properties. Fractal geometry is ubiquitous in nature [Mandelbrot, 1983]. Fractals can be disconnected, modeling sparsity. Fractals are typically *self-similar*, with the same properties at all scales, yielding hierarchical organization. A fractal fills space with a dimension distinct from both its topological dimension and its embedding dimension, reflecting intrinsically low dimensionality. Self-similar fractals are scale-free, naturally giving rise to power laws. Fractality reflects the intuition that nature's data-generating process is essentially inexhaustible. The more one learns, the more detailed distinctions one makes, and there's always more to learn.

Synthetic data models with realistic, principled structure are valuable for mechanistic interpretability research. Synthetic datasets provide ground-truth latent features, allowing researchers to validate and improve interpretability tools. Much progress has been spurred by synthetic data models with heavy-tailed sparse features [Elhage et al., 2022] and hierarchically organized features [Chanin et al., 2024, Bussmann et al., 2025, Costa et al., 2025]. Chanin and Garriga-Alonso [2026] propose a synthetic benchmark model of neural activations that incorporates feature sparsity and hierarchy, as well as superposition and correlation. These synthetic datasets model the empirical characteristics of neural activations. To ground interpretability research in meaningful properties of the data itself, principled data models are needed.

We introduce a synthetic data model based on critical mean-field percolation theory [Stauffer and Aharony, 2018]. We build on a model proposed by Brill [2024, 2025b]. The percolation model describes data that has sparsity, taxonomic hierarchy, power laws, low intrinsic dimension, and self-similarity. The data model is very simple, consisting of clusters of randomly occupied units on a high-dimensional lattice. Despite this simplicity, the model possesses rich structure, including critical phenomena, self-similarity, and fractal geometry. Figure 1 illustrates the percolation model.

This work presents several contributions. First, we introduce an explicitly hierarchical construction of the percolation data, revealing its innate self-similarity by constructing target values in accordance with latent variables organized in binary trees. Second, we highlight a mapping between critical mean-field percolation clusters, random labeled trees [Moon, 1970], and the additive coalescent process [Aldous and Pitman, 1998, Pitman, 1999], and leverage this mapping to propose a *cyclic coalescent* algorithm to jointly sample a random tree and its hierarchical latent decomposition in almost linear $O(n\,\alpha(n))$ time. Third, we develop tools to generate synthetic hierarchical percolation datasets using the cyclic coalescent. Fourth, we empirically investigate representations of neural networks trained on synthetic percolation data via probing experiments.[1]

In Sec. 2, we introduce the percolation data model. In Sec. 3, we propose the cyclic coalescent algorithm. In Sec. 4, we describe a procedure to generate synthetic data following the percolation model. We describe our experiments in Sec. 5 and conclude with discussion in Sec. 6.

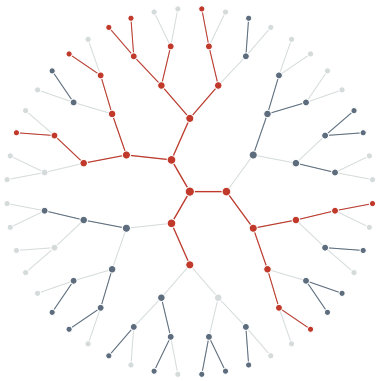


Figure 2: Bethe lattice percolation. Red: infinite cluster. Blue: finite clusters.

## 2 Percolation Model

### 2.1 Percolation theory

The branch of statistical physics concerned with the properties of clusters of randomly occupied sites or bonds on a lattice is called percolation theory [Stauffer and Aharony, 2018]. In this framework, sites or bonds on a specified lattice are occupied independently at random with probability $p$, forming contiguous clusters.[2] We consider a hypercubic lattice of dimension $d$ and scale $L$. Percolation's most notable property is a phase transition. Above a critical occupation probability $p = p_c$, the system *percolates*: an infinite cluster emerges that scales with the system size. Below this transition, only finite clusters exist.

Above an upper critical dimension believed to equal $d_u = 6$, percolation enters an exactly solvable mean-field regime. A random path in high-dimensional

[1] All code will be made publicly available.

[2] Appendix A describes field-specific terms used in this paper.

space is vanishingly unlikely to self-intersect, so percolation clusters are cycle-free, and the lattice can be approximated as a tree. Specifically, mean-field percolation is modeled using the Bethe lattice, an infinite tree in which all nodes have equal degree $z$. Fig. 2 illustrates this with $z = 3$. The percolation threshold in this setting is $p_c = 1/(z-1)$. Using the Bethe lattice to approximate a hypercubic lattice gives $z = 2d$ and $p_c = 1/(2d-1)$. For mean-field percolation, site and bond percolation are essentially equivalent.

Percolation clusters have interesting statistical and geometric properties [Stauffer and Aharony, 2018]. Let $s \geq 1$ denote a cluster's size. Close to criticality for $s \gg 1$, mean-field percolation clusters have a size distribution $n_s(p) = s^{-\tau} e^{-cs}$, where $\tau = 5/2$, $c \propto (p_c - p)^{1/\sigma}$, and $\sigma = 1/2$. Exactly at criticality with $p = p_c$, the cluster size distribution becomes a pure power law. Geometrically, percolation clusters are fractal objects with fractal dimension $D < d$. A critical mean-field percolation cluster has $D = 4$ and can be visualized as a tree embedded as a branching random walk on a lattice.

### 2.2 Data model

Our data model is based on site percolation. Let $\mathcal{X} = [0,1)^d \subset \mathbb{R}^d$ denote the $d$-dimensional data space from which potential data points $\mathbf{x} \in \mathcal{X}$ are drawn. We fix the unit cube without loss of generality. We discretize $\mathcal{X}$ as a hypercubic lattice of linear size $L$ and spacing $1/L$,

$$\Lambda = \tfrac{1}{L}\mathbb{Z}^d \cap \mathcal{X}. \tag{1}$$

Each site of $\Lambda$ is independently in-distribution with probability $p \in [0,1]$, giving a random subset $\mathcal{S} \subseteq \Lambda$ of expected size $\mathbb{E}|\mathcal{S}| = pL^d$. The continuum limit is recovered as $L \to \infty$. An empirical dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{D}|}\}$ is then drawn i.i.d. uniformly from $\mathcal{S}$.

The above model was previously described by Brill [2024]. In that work, the target function was modeled as a set of random continuous functions, each supported on a separate cluster in $\mathcal{S}$. We now simplify and augment that picture by making three crucial assumptions.

- **Mean-field**: We assume $d \gg d_u$ so that the Bethe lattice model applies. High-dimensional inputs are a realistic assumption for natural data.
- **Criticality**: We set $p = p_c$. This regime may enable both useful and efficient learning [Brill, 2024]. Combined with mean-field, this makes the model analytically tractable.
- **Hierarchy**: We constrain targets to follow a simple hierarchical construction, stated next.

We generate target values hierarchically. Let $\mathcal{F}$ be a binary forest of latent variables. Initialize it as the trivial forest with one leaf $z_{\mathbf{x}}$ associated with each site $\mathbf{x} \in \mathcal{S}$. As a construction device, we activate the bonds between each pair of adjacent sites in $\mathcal{S}$ sequentially in a uniformly random order. However, only a partial order on bonds recorded by $\mathcal{F}$ plays any further role.

The mean-field assumption ensures that every bond connects two distinct clusters $\mathcal{C}_1$ and $\mathcal{C}_2$. For each bond, we introduce a fresh latent, associate it with each site in $\mathcal{C}_1 \cup \mathcal{C}_2$, and make it the parent of the existing topmost latents of $\mathcal{C}_1$ and $\mathcal{C}_2$. Writing $\mathcal{Z}_{\mathbf{x}}$ for the top-down path of latents in $\mathcal{F}$ associated with site $\mathbf{x}$, the target at $\mathbf{x}$ is, for some function $f$,

$$y_{\mathbf{x}} = f(\mathcal{Z}_{\mathbf{x}}). \tag{2}$$

### 2.3 Model properties

The percolation model is quite simple, but it has rich structure matching properties associated with natural data distributions. Mean-field percolation is analytically tractable and well studied, and its statistical and geometric properties are textbook results [Stauffer and Aharony, 2018]. At criticality, the data consist of sparse clusters power-law-distributed with exponent $5/2$. Furthermore, clusters are low-dimensional fractals with intrinsic dimension $D = 4$. Brill [2024] analyzes the scaling laws that result from the tradeoff between these two competing sources of power-law structure. The graph of each percolation cluster has degree distribution $k \sim 1 + \text{Binomial}(z-1,\ p)$. For large $z \sim 2d$ and $p = p_c$, we make the Poisson approximation $k \sim 1 + \text{Poisson}(1)$.

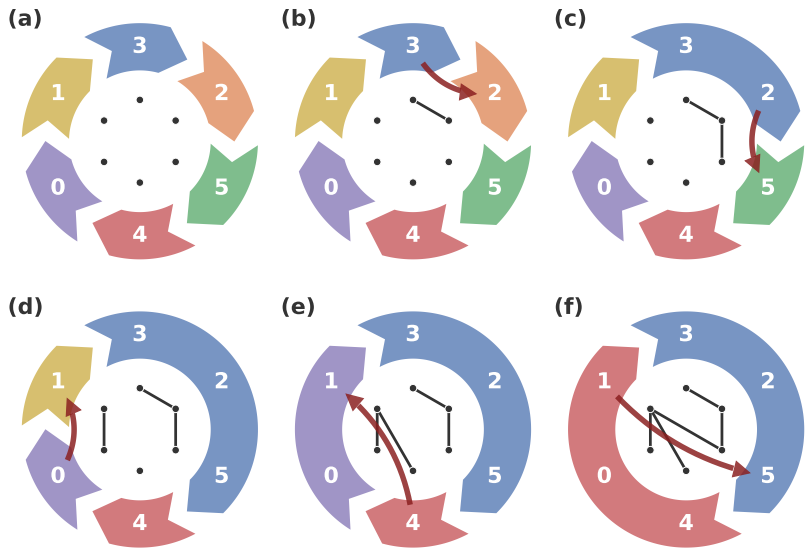



Figure 3: Example run of the cyclic coalescent. (a) Initialize nodes in a random cycle. (b–f) At each step, connect a random node to a random node in its block's successor, merging those blocks.

Each cluster forms a functional equivalence class, in the sense that any subset of its inputs shares at least one latent. The construction is statistically self-similar. Decomposing the latent forest $\mathcal{F}$ splits a cluster into identically distributed subclusters. The structure of $\mathcal{F}$ implicitly records information about each latent. In particular, for a given latent, we define its size $s_{\text{latent}}$ as the number of leaf nodes at or below that latent; its cluster size $s_{\text{cluster}}$ as the number of leaf nodes at or below the root of that latent's tree; and its depth $d_{\text{latent}}$ as the edge length of the path to the latent from the root of its tree. The latents in $\mathcal{F}$ serve multiple roles. Because they correspond to bonds in data space, they indicate that merged subclusters share similar input features. For the same reason, they also identify split points between subclusters. Finally, they provide latent variables used to compute targets.

The latent tree $\mathcal{F}$ records hierarchical relations among subsets of inputs, thereby describing taxonomic hierarchical structure. Hierarchical latent variables associated with subclusters may provide a model for hierarchical context features identified in LLMs [Gurnee et al., 2023, Brill, 2025b]. We discuss taxonomic and compositional hierarchical structure further in Sec. 6.

## 3 Cyclic Coalescent

Directly simulating high-dimensional percolation is intractable. Instead, efficient methods to construct a cluster grow it from a starting site [Leath, 1976] or use invasion percolation [Mertens and Moore, 2018]. However, these methods do not explicitly represent self-similar hierarchical structure. Furthermore, our purpose is not to experimentally probe percolation itself, but to apply the critical mean-field regime as a data model. To that end, we leverage equivalences between percolation and several other mathematical constructions, using these mappings to develop an algorithm that efficiently generates clusters with a hierarchical decomposition equivalent to the model in Sec. 2.2.

A mean-field percolation cluster can be thought of as a size-conditioned Galton-Watson branching process, and mapped to a Brownian excursion [Roch, 2024, Aldous, 1993, Font-Clos and Moloney, 2016]. These mappings imply that critical mean-field percolation clusters have the same distribution as uniform random labeled trees [Aldous, 1991a,b], which are well-studied combinatorial objects [Moon, 1970]. In particular, a deep connection exists between random trees and a merger process known as additive coalescence [Aldous and Pitman, 1998, Pitman, 1999, Aldous, 1999].

In an elegant construction, Pitman [1999] showed that the time-reversal of deleting a random tree's edges uniformly at random is equivalent to merging a forest of smaller trees in an additive coalescent process. At each step, two randomly chosen trees merge with a probability proportional to their total

**Algorithm 1** *Cyclic coalescent*: $O(n\,\alpha(n))$ uniform tree sampler with coalescent decomposition

**Require:** Number of nodes $n$
**Ensure:** Uniform random labeled tree $G$ on $\{1, \ldots, n\}$ and its rooted binary merger tree $B$.
**State.** Blocks form a cyclic order maintained by union-find supporting block lookup, $B$-root lookup, cyclic-successor query, and merge-with-successor in amortized $O(\alpha(n))$.
Arrange $1, \ldots, n$ in a uniformly random cyclic order, with each label its own block.
Initialize $B$ as the forest of leaves $1, \ldots, n$.
**for** $k = 1$ **to** $n - 1$ **do**
    Draw $u \sim \mathrm{Unif}(\{1, \ldots, n\})$; let $A$ be the block of $u$ and $A'$ its cyclic successor.
    Draw $v \sim \mathrm{Unif}(A')$ independently, and add edge $\{u, v\}$ to $G$.
    Let $r_A, r_{A'}$ be the $B$-roots of $A$, $A'$; create a new internal node in $B$ with children $r_A, r_{A'}$.
    Merge $A$ into $A'$, and assign the new internal node as the $B$-root of the merged block.
**end for**
**return** $G,\ B$

size. Our method draws heavily on Pitman [1999], and we review essential results from that work in detail in Appendix E.1. In each step of Pitman's construction, out of $k$ remaining trees with total size $n$, a pair of trees $T_i$ and $T_j$ is picked to merge with probability

$$P(i, j) = \frac{n_i + n_j}{n(k-1)}, \tag{3}$$

where $n_i$ and $n_j$ are those tree's sizes. Next, a node is picked independently and uniformly at random from each of $T_i$ and $T_j$, and an edge added between those nodes. Amazingly, Theorem E.3 states that this process is exactly equivalent to uniform random edge insertion in a random tree. What's more, it is equivalent to the hierarchical procedure described in Sec. 2.2, allowing us to repurpose coalescence as a procedure to build up a hierarchy of latent variables.

We propose a variant of Pitman's construction called the *cyclic coalescent*. As with the standard coalescent construction, the cyclic coalescent uniformly samples random trees. Furthermore, it admits a simple and highly efficient algorithmic implementation that runs in almost linear $O(n\,\alpha(n))$ time. Our construction is based on two key ideas. First, picking a node uniformly at random plays double duty. One random draw simultaneously picks a tree with size-dependent probability and picks a uniform random node from it. Second, we arrange the trees in a fixed random cyclic order. Rather than allowing unrestricted merges among all trees, the chosen tree simply merges with its successor. The initial random cyclic order ensures the distribution's uniformity. Fig. 3 illustrates the cyclic coalescent's operation. We state the construction formally and prove its correctness in Appendix E.2.

We implement the cyclic coalescent using a union-find data structure [Tarjan, 1975]. Pseudocode is shown in Algorithm 1. A union-find algorithm efficiently maintains a partition of subsets, supporting a `find` operation to identify an element's subset and a `union` operation to merge two subsets. Union-find algorithms have wide application for graph problems, including for studying site and bond percolation [Newman and Ziff, 2001]. Our implementation uses union by rank and path halving [Tarjan and Van Leeuwen, 1984]. Each union-find iteration takes amortized $O(\alpha(n))$ time, where $\alpha$ is the extremely slow-growing inverse Ackermann function and is effectively constant for physically realizable $n$. The algorithm runs for $n - 1$ iterations, giving $O(n\,\alpha(n))$ total time complexity.

# 4 Synthetic Dataset

**Cluster distribution** We use an iterative preferential-attachment procedure to generate clusters with a power-law size distribution. Each step increments the dataset size by one. At each step, either a new cluster is created with probability $p_{01}$, or else an existing cluster is chosen with probability proportional to its size $s$. That cluster's size is then increased to $s + 1$. To obtain a distribution consistent with critical mean-field percolation, we set $p_{01} = 1/3$, which we derive in Appendix D. This iterative procedure induces a self-consistent size-independent ordering among data points for a specified random seed, although we do not rely on this property for our experiments here.

**Cluster geometry** For each cluster, we use the cyclic coalescent to generate a random tree and an associated hierarchical latent decomposition. We do this independently for each cluster in the dataset. This results in a forest of data points and a corresponding forest of rooted binary latent trees.

**Embeddings** After generating the graph structures of all clusters, the nodes are embedded as data points in a $d$-dimensional vector space. The dimension $d$ is a hyperparameter that can be set arbitrarily. Each cluster is embedded by first randomly choosing a root node and then iteratively embedding each node's neighbors following a branching random walk. The random choice of the root node and the random generation of the step direction at each node when embedding the graph are both performed hierarchically, following the cluster's latent tree. This embedding procedure yields clusters that have consistent geometrical structure regardless of the number of iterations used in the generation process.

**Targets** We specialize Eq. 2 to a regression task that is a linear function of the latent variables. Specifically, we assign each latent variable $z_i \in \mathcal{F}$ a value $z_i \sim \mathcal{N}(0, 1)$. We record each latent's value and depth. For each point $\mathbf{x}$, we compute its target as the normalized sum of its associated latents' values,

$$y_{\mathbf{x}} = \frac{1}{\sqrt{1 + d_{\mathbf{x}}}} \sum_{i=0}^{d_{\mathbf{x}}} z_i^{(\mathbf{x})}, \tag{4}$$

where $z_i^{(\mathbf{x})}$ is the value of the latent at depth $i$ and $d_{\mathbf{x}}$ is the depth of point $\mathbf{x}$.

Each latent has an expected contribution to the mean-squared error (MSE) proportional to its number of associated data points, given by its size $s_{\text{latent}}$, divided by the expected number of latents per point. Theorem E.3 states a bijection between the time-reversal of edge deletion in a random tree and an additive coalescent process, which implies that the number of merges a random node undergoes equals the number of cuts needed to isolate a node in a random tree. This has expectation $\sqrt{\pi s_{\text{cluster}}/2}$ [Moon, 1970]. We therefore stratify latents by the figure of merit $\text{FOM} \equiv s_{\text{latent}}/\sqrt{s_{\text{cluster}}}$.

# 5 Experiments

**Data generation** We generated two datasets as described in Section 4. The *one-cluster* dataset consists of a single percolation cluster with $2 \times 10^5$ data points. This dataset allows us to study the cluster properties in isolation. Second, a *multi-cluster* dataset with $2 \times 10^6$ data points was generated using the full percolation model. The multi-cluster dataset was filtered to drop clusters with fewer than 500 data points, to ensure that each cluster had sufficient data points for training and validation. Both datasets were stored as input embeddings $\mathbf{X} \in \mathbb{R}^{n \times d}$ and scalar labels $\mathbf{y} \in \mathbb{R}^n$, along with the ground-truth latent features. Full dataset hyperparameters are given in Table 4 in Appendix B.1.

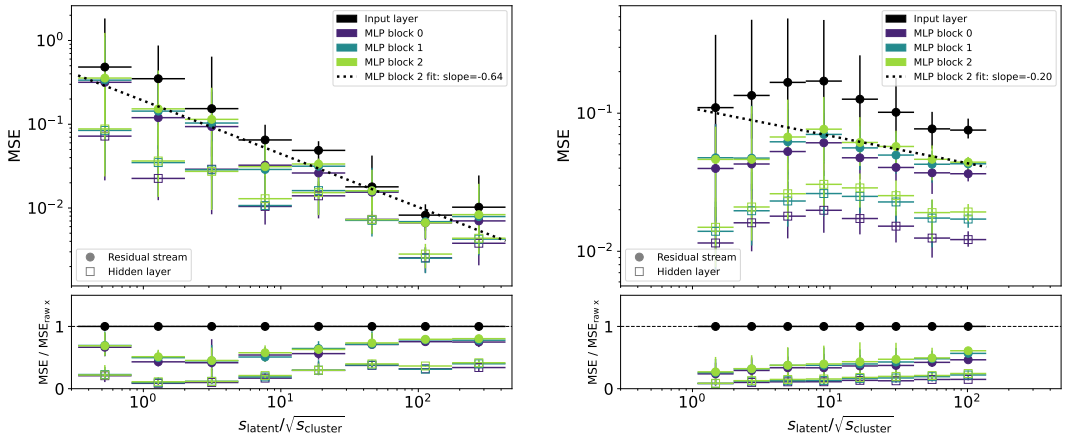
**Neural network training** We trained a residual multilayer perceptron (MLP) on the percolation data. This architecture was chosen to resemble a transformer feed-forward block. Our MLP consists of a linear input projection, $L$ MLP blocks, and a linear output layer. Each MLP block expands by a factor of 4, applies a ReLU activation, and projects back to $d_{\text{model}}$ with a residual connection,

$$\mathbf{x}' = \mathbf{x} + W_2\sigma(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2, \tag{5}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the MLP block's input and output, $W_1$ and $W_2$ are weight matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ are bias vectors, and $\sigma$ denotes the ReLU activation. For the one-cluster dataset we used $d_{\text{model}} = 256$ and $L = 3$ blocks, and for the multi-cluster dataset we used $d_{\text{model}} = 512$ and $L = 3$ blocks. Both models were trained using the AdamW optimizer [Loshchilov and Hutter, 2017] and a cosine learning rate decay schedule. Full hyperparameters are provided in Table 5 in Appendix B.2.

**Model performance** We report MLP performance against two simple baselines, shown in Table 1. Ridge regression performs significantly worse, confirming the task's nonlinearity. We also compute 1-nearest-neighbor (1NN) regression using the full dataset to estimate the best practically achievable performance. The MLP performs comparably to this best-case estimate.

**Linear probes** We trained linear probes [Alain and Bengio, 2016] to assess if MLPs trained to solve the percolation task linearly represent the ground-truth latent variables $z_i$ defined in Eq. 4. Separate probes were trained to regress the values of all latents at each given depth, using all training




(a) (b)

Figure 4: Per-latent linear probe performance for individual latent values $z_i$, showing (a) the one-cluster dataset and (b) the multi-cluster dataset. Dots (squares) mark the median MSE for the residual stream (hidden activations). Error bars show 25th and 75th percentiles. The ratio panels show each layer's per-latent MSE normalized by the MSE of probes trained on the raw input.

points of larger depth. Stratifying by depth fixes a unique latent for each point. We excluded latents with $s_{\text{latent}} < 150$ and corresponding points to have enough training and validation samples.

Linear probes were fit from the activations to the latent value $z_i$. We also fit probes to the partial cumulative sum descending the latent path to $z_i$, with similar results, shown in Appendix F. Probes were computed for depths $0, 5, 10, \ldots, 200$ on the raw input and at each network layer. Probe performance was evaluated using the per-latent MSE, computed separately on the validation set for each latent.

Table 1: Model performance.

| Metric | Model | One-cluster | Multi-cluster |
|---|---|---|---|
| $R^2$ | Ridge | 0.51 | 0.39 |
| | 1NN | 0.94 | 0.86 |
| | MLP | 0.94 | 0.88 |
| MSE | Ridge | 0.20 | 0.59 |
| | 1NN | 0.024 | 0.13 |
| | MLP | 0.026 | 0.12 |

## 6 Discussion

**Linear representations** Fig. 4 shows the probe results. The ground-truth latent variables can be linearly decoded from the network's activations. Per-latent probes trained on MLP activations have consistently lower MSE than on the raw input. The hidden layer activations have lower probe MSE than the residual stream. The ratio plots show that the probe performance gap compared to raw input decreases with increasing FOM. We hypothesize that latents representing coarser subtrees may be easier to decode from the input geometry, requiring less nonlinear computation.

**Power-law trends** The per-latent MSE exhibits an apparent power-law trend as a function of FOM. We visualize this with power-law fits in Fig. 4. In Fig. 4b, the MSE drop at low FOM is an artifact of the minimum cluster size filter, which removes the smallest clusters and suppresses the low-FOM bins. Fig. 5 shows the probe MSE as a function of latent depth. No power-law trend is visible, suggesting that FOM captures the latents' statistical importance more effectively than depth alone. For depths above 100, the global MSE decreases due to a selection bias, as only the few largest clusters have the deepest latents.

**No layerwise progression** The probe MSE is very similar for all MLP blocks, with the possible exception that the MSE appears sightly lower for MLP block 0 for multi-cluster data in Fig. 4b. Furthermore, no clear pattern is observed for any FOM bin across layers. This lack of trend suggests that the network may not represent latents following an interpretable organization across layers. This

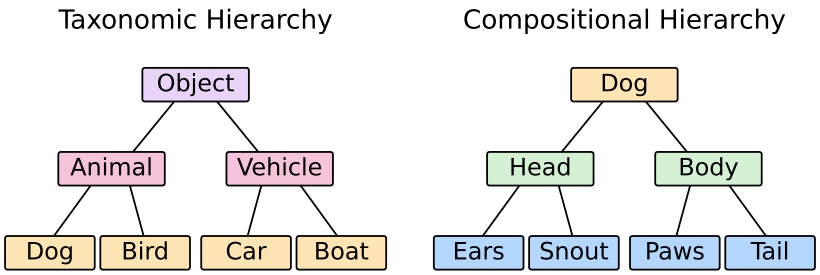



Figure 6: A comparison of taxonomic and compositional hierarchical structures.

hypothesis is supported by the lack of layerwise trend in the cumulative latent sums, shown in Fig. 7 in Appendix F. This suggests that the network does not build up a hierarchical sum across layers.

**Mechanistic interpretability** This work develops a synthetic data model meant to complement empirical studies interpreting LLMs. Because the percolation model has ground-truth hierarchical latent structure, future work could test its validity as a useful data model by investigating whether it causes analogs of SAE pathologies such as feature splitting and absorption [Bricken et al., 2023, Chanin et al., 2024]. In addition, the relationship between data structure and structure in neural activations is poorly understood. A fruitful avenue of further research may be to investigate whether percolation data produces activation structure with features similar to synthetic models [Chanin and Garriga-Alonso, 2026]. We discuss additional related work in Appendix C.

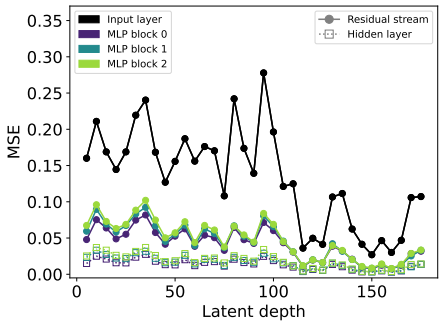



Figure 5: Linear probe recovery of latent values $z_i$ for the one-cluster dataset, stratified by latent depth.

**Hierarchy** The percolation model's latent forest $\mathcal{F}$ describes a taxonomic hierarchy of set inclusion relations among samples. This is a hierarchy distinct from, and complementary to, composition among features, which has been described using synthetic data models [e.g. Cagnetta et al., 2024]. Table 2 and Fig. 6 compare taxonomic and compositional hierarchical relations. In addition, the percolation model naturally maintains an explicit instance/class distinction within a taxonomic hierarchy [Brachman, 1983]. The latents represent classes and data points are instances.

**Limitations** The correlational probing experiments in this work serve as a proof of concept for investigating the percolation model as an interpretability testbed. Causal interventions are needed to validate specific interpretability hypotheses [Meng et al., 2022, Chan et al., 2022].

Table 2: Hierarchical relations.

| | **Taxonomic** | **Compositional** |
|---|---|---|
| Unit | sample | feature |
| Description | set inclusion | part-whole |
| Relation | IS-A | HAS-A |
| Semantics | hyponymy | meronymy |

The percolation model lacks many features of real data. It assumes that the data distribution is populated at random. It does not model compositional relationships among input features. The synthetic dataset we present in this work supports vectorized data with scalar regression labels only. Extending the model to support tokenized data and classification tasks is a direction for future work.

Finally, we present a critical mean-field percolation model. One possible avenue for generalization is to consider percolation beyond the critical regime. In particular, supercritical percolation allowing for clusters with cycles may provide a framework to model overlapping compositional concepts.

# References


M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer, 2014.

I. Alabdulmohsin and A. Steiner. A tale of two structures: Do llms capture the fractal complexity of language? *arXiv preprint arXiv:2502.14924*, 2025.

I. Alabdulmohsin, V. Q. Tran, and M. Dehghani. Fractal patterns may illuminate the success of next-token prediction. *Advances in Neural Information Processing Systems*, 37:112864–112888, 2024.

G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

D. Aldous. The continuum random tree. i. *The annals of probability*, pages 1–28, 1991a.

D. Aldous. The continuum random tree. ii. an overview. *Stochastic analysis (Durham, 1990)*, 167: 23–70, 1991b.

D. Aldous. The continuum random tree iii. *The annals of probability*, pages 248–289, 1993.

D. Aldous and J. Pitman. The standard additive coalescent. *Annals of Probability*, pages 1703–1726, 1998.

D. J. Aldous. Deterministic and stochastic models for coalescence (aggregation and coagulation): a review of the mean-field theory for probabilists. *Bernoulli*, 5(1):3 – 48, 1999.

Z. Allen-Zhu and Y. Li. Physics of language models: Part 1, learning hierarchical language structures. *arXiv preprint arXiv:2305.13673*, 2023a.

Z. Allen-Zhu and Y. Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023b.

A. Atanasov, J. A. Zavatone-Veth, and C. Pehlevan. Scaling and renormalization in high-dimensional regression. *arXiv preprint arXiv:2405.00592*, 2024.

Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439): 509–512, 1999.

M. Baradad Jurjo, J. Wulff, T. Wang, P. Isola, and A. Torralba. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021.

R. E. Bellman. Adaptive control processes. *Adaptive Control Processes A Guided Tour*, 4, 1961.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

L. Bereska and E. Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

J. A. Binnie, P. Dłotko, J. Harvey, J. Malinowski, and K. M. Yim. A survey of dimension estimation methods. *arXiv preprint arXiv:2507.13887*, 2025.

P. Bloem and S. de Rooij. An expectation-maximization algorithm for the fractal inverse problem. *arXiv preprint arXiv:1706.03149*, 2017.

B. Bordelon, A. Canatar, and C. Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, 2020.

R. J. Brachman. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *Computer;(United States)*, 10, 1983.

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. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

A. Brill. Neural scaling laws rooted in the data distribution. *arXiv preprint arXiv:2412.07942*, 2024.

A. Brill. A model for scaling laws of general intelligence. In *ILIAD 2: ODYSSEY*, 2025a.

A. Brill. Representation learning on a random lattice. *arXiv preprint arXiv:2504.20197*, 2025b.

J. Brinkmann, A. Sheshadri, V. Levoso, P. Swoboda, and C. Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4082–4102, 2024.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

B. Bussmann, N. Nabeshima, A. Karvonen, and N. Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

F. Cagnetta and M. Wyart. Towards a theory of how the structure of language is acquired by deep neural networks. *Advances in Neural Information Processing Systems*, 37:83119–83163, 2024.

F. Cagnetta, L. Petrini, U. M. Tomasini, A. Favero, and M. Wyart. How deep neural networks learn compositional data: The random hierarchy model. *Physical Review X*, 14(3):031001, 2024.

F. Cagnetta, H. Kang, and M. Wyart. Learning curves theory for hierarchically compositional data with power-law distributed features. *arXiv preprint arXiv:2505.07067*, 2025.

F. Cagnetta, A. Raventós, S. Ganguli, and M. Wyart. Deriving neural scaling laws from the statistics of natural language. *arXiv preprint arXiv:2602.07488*, 2026.

L. Chan, A. Garriga-Alonso, N. Goldowsky-Dill, R. Greenblatt, J. Nitishinskaya, A. Radhakrishnan, B. Shlegeris, and N. Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. In *AI Alignment Forum*, volume 2, page 19, 2022.

D. Chanin and A. Garriga-Alonso. Synthsaebench: Evaluating sparse autoencoders on scalable realistic synthetic data. *arXiv preprint arXiv:2602.14687*, 2026.

D. Chanin, J. Wilken-Smith, T. Dulka, H. Bhatnagar, S. Golechha, and J. Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*, 2024.

V. Costa, T. Fel, E. S. Lubana, B. Tolooshams, and D. Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. *arXiv preprint arXiv:2506.03093*, 2025.

H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

F. Devlin and J. Sanders. Dropout neural network training viewed from a percolation perspective. *arXiv preprint arXiv:2512.13853*, 2025.

S. Eisenstat. Condensation: a theory of concepts. In *ILIAD 2: ODYSSEY*, 2025.

N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

E. Facco, M. d'Errico, A. Rodriguez, and A. Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140, 2017.

F. Font-Clos and N. R. Moloney. Percolation on trees as a brownian excursion: From gaussian to kolmogorov-smirnov to exponential statistics. *arXiv preprint arXiv:1606.03764*, 2016.

L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

J. Garnier-Brun, M. Mézard, E. Moscato, and L. Saglietti. How transformers learn structured data: insights from hierarchical filtering. *arXiv preprint arXiv:2408.15138*, 2024.

S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10(4):041044, 2020.

P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica D: nonlinear phenomena*, 9(1-2):189–208, 1983.

L. Greenspan, D. Berman, A. Brill, R. Jefferson, A. Kolchinsky, J. Lin, A. Mack, A. Maiti, F. E. Rosas, A. Stapleton, et al. Towards worst-case guarantees with scale-aware interpretability. *arXiv preprint arXiv:2602.05184*, 2026.

W. Gurnee, N. Nanda, M. Pauly, K. Harvey, D. Troitskii, and D. Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

M. Hutter. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

R. Karbauskaitė and G. Dzemyda. Fractal-based methods as a technique for estimating the intrinsic dimensionality of high-dimensional data: a survey. *Informatica*, 27(2):257–281, 2016.

H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

P. Leask, B. Bussmann, M. Pearce, J. Bloom, C. Tigges, N. A. Moubayed, L. Sharkey, and N. Nanda. Sparse autoencoders do not find canonical units of analysis. *arXiv preprint arXiv:2502.04878*, 2025.

P. Leath. Cluster shape and critical exponents near percolation threshold. *Physical Review Letters*, 36 (16):921, 1976.

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

S. P. Lehalleur, J. Hoogland, M. Farrugia-Roberts, S. Wei, A. G. Oldenziel, G. Wang, L. Carroll, and D. Murfet. You are what you eat–ai alignment requires understanding how data shapes structure and generalisation. *arXiv preprint arXiv:2502.05475*, 2025.

E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004.

Q. Liang, Z. Liu, M. Ostrow, and I. Fiete. How diffusion models learn to factorize and compose. *Advances in Neural Information Processing Systems*, 37:15121–15148, 2024.

T. Lieberum, S. Rajamanoharan, A. Conmy, L. Smith, N. Sonnerat, V. Varma, J. Kramár, A. Dragan, R. Shah, and N. Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 278–300, 2024.

B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

E. S. Lubana, K. Kawaguchi, R. P. Dick, and H. Tanaka. A percolation model of emergence: Analyzing transformers trained on a formal language. *arXiv preprint arXiv:2408.12578*, 2024.

E. Malach and S. Shalev-Shwartz. Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32, 2019.

A. Maloney, D. A. Roberts, and J. Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

B. B. Mandelbrot. The fractal geometry of nature. *New York*, 1983.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

A. Menon, M. Shrivastava, D. Krueger, and E. S. Lubana. Analyzing (in) abilities of saes via formal languages. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4837–4862, 2025.

S. Mertens and C. Moore. Percolation thresholds and fisher exponents in hypercubic lattices. *Physical Review E*, 98(2):022120, 2018.

E. Michaud, Z. Liu, U. Girit, and M. Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722, 2023.

E. J. Michaud, L. Gorton, and T. McGrath. Understanding sparse autoencoder scaling in the presence of feature manifolds. *arXiv preprint arXiv:2509.02565*, 2025.

G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

J. W. Moon. Counting labelled trees. *Canadian Mathematical Congress*, 1970.

R. Nakamura, R. Tadokoro, R. Yamada, Y. M. Asano, I. Laina, C. Rupprecht, N. Inoue, R. Yokota, and H. Kataoka. Scaling backwards: Minimal synthetic pre-training? In *European Conference on Computer Vision*, pages 153–171. Springer, 2024.

Y. Nam, N. Fonseca, S. H. Lee, C. Mingard, and A. A. Louis. An exactly solvable model for emergence and scaling laws in the multitask sparse parity problem. *Advances in Neural Information Processing Systems*, 37:39632–39693, 2024.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

M. Newman. *Networks*. Oxford university press, 2018.

M. E. Newman and R. M. Ziff. Fast monte carlo algorithm for site or bond percolation. *Physical Review E*, 64(1):016706, 2001.

C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

Z. Pan, S. Wang, and J. Li. Understanding llm behaviors via compression: Data generation, knowledge acquisition and scaling laws. *arXiv preprint arXiv:2504.09597*, 2025.

J. Pitman. Coalescent random forests. *Journal of Combinatorial Theory, Series A*, 85(2):165–193, 1999.

D. D. S. Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American society for Information science*, 27(5):292–306, 1976.

H. Prufer. Neuer beweis eines satzes uber per mutationen. *Archiv derMathematik und Physik*, 27: 742–744, 1918.

S. Roch. *Modern Discrete Probability: An Essential Toolkit*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2024. doi: 10.1017/9781009305129.

A. Sclocchi, A. Favero, and M. Wyart. A phase transition in diffusion models reveals the hierarchical nature of data. *Proceedings of the National Academy of Sciences*, 122(1):e2408799121, 2025.

U. Sharma and J. Kaplan. Scaling laws from the data manifold dimension. *Journal of Machine Learning Research*, 23(9):1–34, 2022.

R. K. Sheth and J. Pitman. Coagulation and branching process models of gravitational clustering. *Monthly Notices of the Royal Astronomical Society*, 289(1):66–82, 1997.

J. Simon, D. Kunin, A. Atanasov, E. Boix-Adserà, B. Bordelon, J. Cohen, N. Ghosh, F. Guth, A. Jacot, M. Kamb, et al. There will be a scientific theory of deep learning. *arXiv preprint arXiv:2604.21691*, 2026.

S. Spigler, M. Geiger, and M. Wyart. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (12):124001, 2020.

D. Stauffer and A. Aharony. *Introduction to percolation theory*. Taylor & Francis, 2018.

R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22 (2):215–225, 1975.

R. E. Tarjan and J. Van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the ACM (JACM)*, 31(2):245–281, 1984.

A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

X. Wang, L. Wang, Y. Wu, et al. An optimal algorithm for prufer codes. *J. Softw. Eng. Appl.*, 2(2): 111–115, 2009.

J. Wentworth and D. Lorell. Natural latents: Latent variables stable across ontologies. *arXiv preprint arXiv:2509.03780*, 2025.

W. A. Woods. What's in a link: Foundations for semantic networks. In *Representation and understanding*, pages 35–82. Elsevier, 1975.

# A Terminology

Because this work ties together methods from several fields, we use field-specific terms in various sections to refer to the same objects. Table 3 translates among them.

Table 3: Terminology correspondences across fields.

| **Field** | | |
|---|---|---|
| Percolation | site | bond |
| Graphs & trees | node | edge |
| Data model | data point | latent |

# B Model training

## B.1 Data hyperparameters

Table 4 lists the hyperparameters used to generate the one-cluster and multi-cluster datasets described in Sec. 5. For the multi-cluster dataset, a minimum cluster size filter is applied as a post-processing step after generation, where cluster size refers to the number of points per cluster.

Table 4: Data generation hyperparameters.

| **Hyperparameter** | **One-cluster** | **Multi-cluster** |
|---|---|---|
| Mode | `one_cluster` | `distribution` |
| $n$ (dataset size) | $2 \times 10^5$ | $2 \times 10^6$ |
| $d$ (embedding dimension) | 100 | 100 |
| Graph seed | 0 | 0 |
| Embedding seed | 10,000 | 10,000 |
| Value seed | 20,000 | 20,000 |
| *Post-processing* | | |
| Min. cluster size filter | — | 500 |

## B.2 Model hyperparameters

Table 5 lists the full set of hyperparameters for the residual MLP models trained in Section 5. Both models are implemented in PyTorch and share the same architecture and optimisation scheme, differing only in model width and number of training epochs.

Table 5: Neural network training hyperparameters.

| **Hyperparameter** | **One-cluster** | **Multi-cluster** |
|---|---|---|
| $d_{\text{model}}$ | 256 | 512 |
| $N$ (MLP blocks) | 3 | 3 |
| Expansion factor | 4 | 4 |
| Activation | ReLU | ReLU |
| Input dimension | 100 | 100 |
| Output dimension | 1 | 1 |
| Batch size | 1024 | 1024 |
| Epochs | 500 | 2500 |
| Loss | MSE | MSE |
| Optimizer | AdamW | AdamW |
| Learning rate | $10^{-4}$ | $10^{-4}$ |
| Weight decay | 0.01 | 0.01 |
| Scheduler | Cosine | Cosine |
| $T_{\text{max}}$ | 500 | 2500 |
| $\eta_{\text{min}}$ | $10^{-6}$ | $10^{-6}$ |
| Random seed | 42 | 42 |

## C Additional related work

**Fractals in machine learning** Fractals generated using iterated function systems are used to create synthetic training data for image models [Nakamura et al., 2024, Baradad Jurjo et al., 2021, Kataoka et al., 2020]. Bloem and de Rooij [2017] present a expectation-maximization algorithm for fitting a fractal model to data. Malach and Shalev-Shwartz [2019] study the expressivity properties of deep neural networks trained on fractal data distributions, finding that deep but not shallow networks can efficiently express these distributions. Fractal-based methods are used to estimate the intrinsic dimension of data [Grassberger and Procaccia, 1983, Karbauskaitė and Dzemyda, 2016]. Alabdulmohsin et al. [2024], Alabdulmohsin and Steiner [2025] argue that natural language viewed from an information-theoretic time-series perspective exhibits fractal patterns, which may be helpful for detecting LLM-generated texts.

**Synthetic data models** Nanda et al. [2023] apply modular addition as a toy model to investigate progress measures for grokking via mechanistic interpretability. Allen-Zhu and Li [2023b] introduce a synthetic dataset of uniformly distributed discrete knowledge pieces to measure knowledge storage and extraction in transformers. Pan et al. [2025] consider a variant of this model with power-law-distributed knowledge pieces. Liu et al. [2022] find theoretically and empirically that transformers can simulate finite-state automata using non-recurrent hierarchical shortcut solutions. Brinkmann et al. [2024] perform a mechanistic analysis of a transformer trained to perform a symbolic multi-step reasoning task, finding that does so using parallelized depth-bounded recurrent mechanisms.

**Percolation in deep learning** Lubana et al. [2024] apply percolation on a bipartite graph to propose a model of emergent capabilities in transformers. Liang et al. [2024] identify a connection between manifold learning in diffusion models and continuum percolation. Devlin and Sanders [2025] analyze dropout neural network training by considering percolation among the network of weights.

**AI interpretability & alignment** The latent hierarchy proposed by the percolation model may complement research directions that aim to develop a theory of concepts Wentworth and Lorell [2025], Eisenstat [2025]. A data distribution compatible with the percolation model, consisting of a heavy-tailed distribution of sparse low-dimensional clusters, can be used to study scaling laws of general intelligence [Brill, 2025a]. Developing a better understanding of the structure of data is a key element in developing a scale-aware basis for mechanistic interpretability [Greenspan et al., 2026], advancing AI alignment [Lehalleur et al., 2025], and building towards a scientific theory of deep learning [Simon et al., 2026].

## D Cluster statistics

In this section, we derive the cluster creation probability $p_{01}$ that, in a preferential attachment process, yields a power-law distribution consistent with critical mean-field percolation. Our derivation closely follows a master-equation method associated with preferential-attachment models in network science [Price, 1976, Barabási and Albert, 1999, Newman, 2018].

Let $s \geq 1$ denote a cluster's size. We will call a cluster of size $s$ an $s$-cluster. Let $n_s$ be the cluster probability mass function of $s$. For critical mean-field percolation, $n_s \sim s^{-5/2}$ for large $s$. Note that because each $s$-cluster contains $s$ sites, the probability mass function of $s$ obtained by randomly selecting sites rather than clusters is proportional to $sn_s$ rather than $n_s$.

In the following, we derive the sizes used for to generate graphs not embedded in a lattice, and so refer to $s$ as the count of nodes rather than sites.

Let $N$ denote the total number of active nodes that have been generated (i.e., after $N$ iterations), and write $n_s(N)$ to denote the cluster size distribution when there are $N$ active nodes.

We first consider the case $s > 1$. The number of nodes belonging to $s$-clusters after $N$ iterations is

$$Nsn_s(N). \tag{6}$$

Adding a node to an $(s-1)$-cluster increases the number of $s$-clusters by 1. This corresponds to increasing the number of nodes belonging to $s$-clusters by $s$. At each iteration, a node is added to an existing cluster with probability $1 - p_{01}$. If a node is added to an existing cluster, the probability

that it's a node belonging to an $(s-1)$-cluster is $(s-1)n_{s-1}(N)$. All told, the expected number of nodes belonging to $s$ clusters added at each iteration is

$$(1-p_{01})s(s-1)n_{s-1}(N). \tag{7}$$

Similarly, adding a node to an $s$-cluster decreases the number of $s$-clusters by 1, since that cluster would become an $(s+1)$-cluster. This corresponds to decreasing the number of nodes belonging to $s$-clusters by $s$. The expected number of nodes belonging to $s$-clusters removed at each iteration is therefore

$$(1-p_{01})s^2 n_s(N). \tag{8}$$

Putting everything together, we have the master equation for $s > 1$,

$$(N+1)sn_s(N+1) = Nsn_s(N) + (1-p_{01})s(s-1)n_{s-1}(N) - (1-p_{01})s^2 n_s(N). \tag{9}$$

We consider the limit of a large number of iterations, letting $N \to \infty$. Solving Eq. 9 yields the recurrence relation,

$$n_s = \frac{s-1}{s-1+y} n_{s-1}, \tag{10}$$

where $y = (2-p_{01})/(1-p_{01})$.

Next, we consider the case $s = 1$. The number of 1-clusters increases by 1 if a new cluster is created, which occurs with probability $p_{01}$. The number of 1-clusters decreases by 1 if a node is added to a 1-cluster, which occurs with probability $(1-p_{01})n_1(N)$. Since each 1-cluster contains 1 node, the equation for $s = 1$ is

$$(N+1)n_1(N+1) = Nn_1(N) + p_{01} - (1-p_{01})n_1(N). \tag{11}$$

Solving Eq. 11 in the limit $N \to \infty$ gives

$$n_1 = \frac{p_{01}}{y(1-p_{01})}, \tag{12}$$

Eq. 10 and Eq. 12 can be combined to give an expression for $n_s$ for general $s$. We can write this expression in terms of Gamma functions and simplify it using the beta function,

$$n_s = \frac{\Gamma(s)\Gamma(y)}{\Gamma(s+y)} \frac{p_{01}}{1-p_{01}} = B(s,y)\frac{p_{01}}{1-p_{01}}. \tag{13}$$

For large $s$, $B(s,y) \approx s^{-y}\Gamma(y)$, giving

$$n_s = \frac{p_{01}}{1-p_{01}} \Gamma\left(\frac{2-p_{01}}{1-p_{01}}\right) s^{-(2-p_{01})/(1-p_{01})}. \tag{14}$$

Equating the exponent to 5/2 gives $p_{01} = 1/3$. Substituting into Eq. 14 and integrating, we obtain the complementary cumulative distribution function (CCDF),

$$\text{CCDF} = \frac{\sqrt{\pi}}{4} s^{-3/2}. \tag{15}$$

# E Cyclic coalescent

The cyclic coalescent construction is underpinned by a close relation between random trees and coalescence. As shown by Pitman [1999], constructing a random (labeled) tree by populating its edges one by one uniformly at random is equivalent to iteratively merging the tree components of a randomly distributed forest in a merger process known as additive coalescence. In this process, tree components merge with a rate proportional to the sum of their sizes. To make this section self-contained, we begin by restating relevant theorems from Pitman [1999]. These are given below with adapted notation as Theorems E.1, E.2, and E.3. An expository presentation of Theorem E.1 can be found in Aigner and Ziegler [2014].

Consider the node set $\{1, \dots, n\}$. Identify a *tree* over $\{1, \dots, n\}$ by a set of $n-1$ edges between nodes, such that every node is connected by at least one edge. Note that this implies that a tree cannot have any cycles or self-edges. Let $\mathcal{T}_n$ be the set of all trees over $\{1, \dots, n\}$, and denote $|\mathcal{T}_n| = T_n$. Call a tree together with the choice of a particular root node a *rooted tree*. Since the root could be any of $n$ nodes, there are $nT_n$ rooted trees over $\{1, \dots, n\}$.

A *forest* over $\{1, \dots, n\}$ is a graph in which each connected component is a tree. Let $\mathcal{F}_{n,k}$ be the set of all forests over $\{1, \dots, n\}$ consisting of $k$ trees. Call a forest together with the choice of a root in each component tree a *rooted forest*. Let $\mathcal{R}_{n,k}$ be the set of all rooted forests over $\{1, \dots, n\}$ consisting of $k$ rooted trees.

## E.1 Prerequisites

**Theorem E.1** ([Pitman, 1999, 1]). *For each rooted forest $r_k \in \mathcal{R}_{n,k}$, the number of rooted trees that contain $r_k$ is $N(r_k) = n^{k-1}$.*

*Proof.* We regard a rooted forest as a directed graph with the edges directed away from the roots. Say that a rooted forest $r$ *contains* another rooted forest $r'$ if the directed graph of $r$ contains the directed graph of $r'$. Call a sequence of rooted forests $r_1, \dots r_k$ a *refining sequence* if $r_i \in \mathcal{R}_{n,k}$ and $r_i$ contains $r_{i+1}$ for all $i$. Let $r_k \in \mathcal{R}_{n,k}$ be a fixed rooted forest. Denote by $N(r_k)$ the number of rooted trees containing $r_k$. Denote by $N^*(r_k)$ the number of refining sequences ending in $r_k$.

First, we count $N^*(r_k)$ starting from a rooted tree and deleting edges. Suppose that $r_1 \in \mathcal{R}_{n,1}$ contains $r_k$. Then $r_1$ has $k-1$ edges that can be deleted in any order to yield a refining sequence from $r_1$ to $r_k$. This means that

$$N^*(r_k) = N(r_k)(k-1)!. \tag{16}$$

Second, we count $N^*(r_k)$ starting from $r_k$ and adding edges. A rooted forest $r_{k-1}$ that refines $r_k$ can be obtained by adding a directed edge from any of the $n$ nodes to any of the $k-1$ roots of rooted trees that do not contain that node. Since there are $n(k-1)$ possibilities at each step, continuing for $k-1$ steps gives

$$N^*(r_k) = n^{k-1}(k-1)!. \tag{17}$$

Equating Eq. 16 and Eq. 17, we obtain $N(r_k) = n^{k-1}$. □

Note that $r_n$ is just the set of $n$ isolated nodes. The number of all rooted trees on $n$ nodes is therefore $N(r_n) = n^{n-1}$. This number is greater than the number of all trees by a factor of $n$.

**Corollary E.1.1** (Cayley's formula). *The number of trees on $n$ nodes is $T_n = n^{n-2}$.*

Cayley's formula is a famous result with many proofs [e.g. Prufer, 1918, Aigner and Ziegler, 2014].

We next straightforwardly apply Theorem E.1 to count unrooted trees.

**Theorem E.2** ([Pitman, 1999, 3]). *For each forest $f_k \in \mathcal{F}_{n,k}$, containing trees with sizes $n_1, \dots, n_k$ such that $\sum n_i = n$, the number of trees that contain $f_k$ is*

$$N(f_k) = \left( \prod_{i=1}^{k} n_i \right) n^{k-2}. \tag{18}$$

*Proof.* Let $f_k \in \mathcal{F}_{n,k}$ be a fixed forest. We count the rooted trees that contain $f_k$ (ignoring edge directions) in two ways. First, there are $n$ ways to choose a tree's root, giving us $nN(f_k)$ rooted trees. Second, choosing a root for each tree component of $f_k$ and applying Theorem E.1 gives us $(\prod_i n_i)\, n^{k-1}$ rooted trees. Equating these expressions yields the above result. □

We now observe a relation between edge deletion in a uniform random tree, an additive coalescent process among trees, and a uniform random forest.

**Theorem E.3** ([Pitman, 1999, 5])**.** *The following three descriptions (i), (ii), and (iii) for the distribution of a sequence of forests $(F_1, \ldots, F_n)$ are equivalent, and imply that, for each $1 \le k \le n$ and for each $f_k \in \mathcal{F}_{n,k}$ with tree components of size $n_1, \ldots, n_k$ in some arbitrary order,*

$$P(F_k = f_k) = \frac{\prod_{i=1}^{k} n_i}{n^{n-k}\binom{n-1}{k-1}} \tag{19}$$

*(i) $F_1$ is a uniform random tree in $\mathcal{T}_n$, and indexing the edges $e_j$ of $F_1$ by a uniform random permutation of $1, \ldots, n-1$, for each $1 \le k \le n$ the forest $F_k$ is derived from $F_1$ by deleting the edges $e_1, \ldots, e_{k-1}$;*

*(ii) $F_n$ is the trivial forest, and for $n \ge k \ge 2$, given $(F_n, \ldots, F_k)$ where $F_k$ has $k$ tree components $T_i, \ldots, T_k$ with sizes $n_1, \ldots, n_k$ such that $\sum_{i=1}^{k} n_i = n$, the forest $F_{k-1} \in \mathcal{F}_{n,k-1}$ is obtained by adding an edge $\{a, b\}$ to $F_k$, chosen according to the following rule: first pick trees $T_i$ and $T_j$, where $1 \le i < j \le k$, with probability*

$$P(i, j) = \frac{n_i + n_j}{n(k-1)}, \tag{20}$$

*then pick $a$ and $b$ independently and uniformly at random from $T_i$ and $T_j$ respectively;*

*(iii) The sequence $(F_1, \ldots, F_n)$ has uniform distribution over the set of all $(n-1)!\, n^{n-2}$ refining sequences of forests $(f_1, \ldots, f_n)$ such that $f_k \in \mathcal{F}_{n,k}$ for every $1 \le k \le n-1$.*

*Proof.* In both descriptions (i) and (iii), the forest $F_k$ is determined by the choice of a tree $t \in T_n$ and a subset of $k-1$ edges of $t$. It follows from Cayley's formula E.1.1 that in both cases there is a total number of $n^{n-2}\binom{n-1}{k-1}$ equally likely choices, so descriptions (i) and (iii) are equivalent. The probability that $F_k = f_k$ equals the number of trees that contain $f_k$ (Eq. 18) divided by the total number, giving the probability shown in Eq. 19.

Next, we show that description (ii) is equivalent to (i). The process is Markov, so it suffices to consider the conditional probability that $F_{k-1} = f_{k-1}$ given $F_k = f_k$. One approach is as follows [Sheth and Pitman, 1997]. Using Bayes' rule,

$$P(F_{k-1} = f_{k-1} \mid F_k = f_k) = \frac{P(F_{k-1} = f_{k-1})}{P(F_k = f_k)} P(F_k = f_k \mid F_{k-1} = f_{k-1}). \tag{21}$$

From Eq. 19 and the fact that in (i), $f_k$ is obtained from $f_{k-1}$ by deleting one of $n - (k-1)$ edges uniformly at random, we have

$$\begin{aligned} P(F_{k-1} = f_{k-1} \mid F_k = f_k) &= \frac{n^{n-k}\binom{n-1}{k-1}(n_i + n_j)\prod_{l \notin \{i,j\}} n_l}{n^{n-(k-1)}\binom{n-1}{k-2}\prod_l n_l} \frac{1}{n-(k-1)} \\ &= \frac{(n-(k-1))(n_i+n_j)}{n(k-1)n_i n_j} \frac{1}{n-(k-1)} \\ &= \frac{n_i + n_j}{n(k-1)} \frac{1}{n_i} \frac{1}{n_j}. \end{aligned} \tag{22}$$

This conditional probability is satisfied by the procedure in description (ii). □

### E.2 Cyclic coalescent

We now build on these prior results to show that a distinct construction, the cyclic coalescent, is equivalent to the descriptions in Theorem E.3. This implies that running the cyclic coalescent to completion produces a uniform random tree. Moreover, such an algorithm can be implemented in almost linear time, as discussed in the main text.

**Theorem E.4** (Cyclic coalescent)**.** *The following description for the distribution of a sequence of forests $(F_1, \dots F_n)$ is equivalent to the descriptions in Theorem E.3.*

*$F_n$ is the trivial forest, and arranging the tree components (nodes) of $F_n$ in a cycle indexed by a uniform random permutation of $1, \dots, n$, for $n \geq k \geq 2$ and given $(F_n, \dots, F_k)$ where $F_k$ has $k$ tree components $T_i, \dots, T_k$ with sizes $n_1, \dots, n_k$ such that $\sum_{i=1}^{k} n_i = n$, the forest $F_{k-1} \in \mathcal{F}_{n,k-1}$ is obtained by adding an edge $\{a, b\}$ to $F_k$, chosen according to the following rule: first pick the node $a$ uniformly at random from $F_k$, then, denoting the tree containing this first-chosen node as $T_i$ and its successor tree as $T_j$, pick the node $b$ uniformly at random from $T_j$.*

*Proof.* Consider the conditional probability $F_{k-1} = f_{k-1}$ given $F_k = f_k$, where $f_{k-1}$ differs from $f_k$ only by adding the edge $\{a, b\}$. The edge $\{a, b\}$ can be added iff $a \in T_i$ and $b \in T_j$ or $b \in T_i$ and $a \in T_j$. There are $k$ trees, so both arrangements occur with equal probability $1/(k-1)$. Then because the first-chosen node is uniformly distributed among $F_k$ and among $T_i$, and the second-chosen node is uniformly distributed among $T_j$, the total probability to choose $\{a, b\}$ is

$$\begin{aligned} P(F_{k-1} = f_{k-1} \mid F_k = f_k) &= \frac{1}{k-1}\frac{n_i}{n}\frac{1}{n_i}\frac{1}{n_j} + \frac{1}{k-1}\frac{n_j}{n}\frac{1}{n_j}\frac{1}{n_i} \\ &= \frac{n_i + n_j}{n(k-1)}\frac{1}{n_i}\frac{1}{n_j}. \end{aligned} \tag{23}$$

This matches the conditional probability given by Eq. 22. □

**Corollary E.4.1.** *The cyclic coalescent after $n-1$ steps produces a uniform random tree in $\mathcal{T}_n$.*

This follows from description (i) in Theorem E.3.

**Corollary E.4.2.** *The cyclic coalescent after $k$ steps produces a uniform distribution over forests $f_k \in \mathcal{F}_{n,k}$.*

This follows from description (iii) in Theorem E.3.

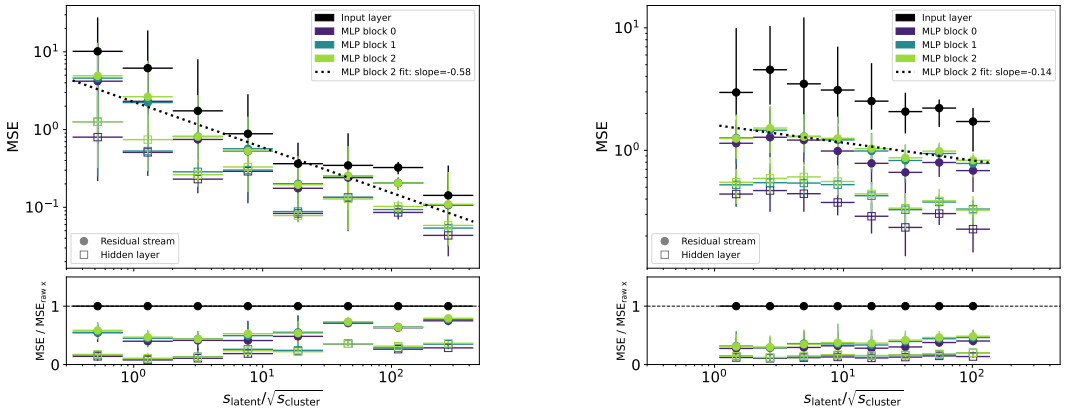
## F Additional probing experiments

Fig. 7 shows the results of linear probing experiments on the partial cumulative sums of latent values.

## G Additional validation plots

Fig. 8 shows the complementary cumulative distribution function (CCDF) for the multi-cluster dataset described in Sec. 5. The fitted slope is consistent with the theoretical expectation of $3/2$ as derived in Eq. 15.

Fig. 9 shows plots validating the correctness of the cyclic coalescent algorithm. A well-known bijection, called a Prüfer sequence [Prufer, 1918], exists between the set of $n^{n-2}$ labeled trees on $n$ nodes and the set of sequences of length $n-2$ on the labels 1 to $n$. This bijection permits a powerful check of uniformity tractable at small tree sizes. Fig. 9a confirms that trees of size $n = 6$ generated by the cyclic coalescent have a uniform distribution when coded as Prüfer sequences. As another check, Fig. 9b validates that a large tree of size $10^6$ has the theoretically expected degree distribution $1 + \text{Poisson}(1)$.




(a) (b)

Figure 7: Per-latent linear probe performance for partial cumulative sums of latent values $z_i$, showing (a) the one-cluster dataset and (b) the multi-cluster dataset. Dots (squares) mark the median MSE for the residual stream (hidden activations). Error bars show 25th and 75th percentiles. The ratio panels show each layer's per-latent MSE normalized by the MSE of probes trained on the raw input.

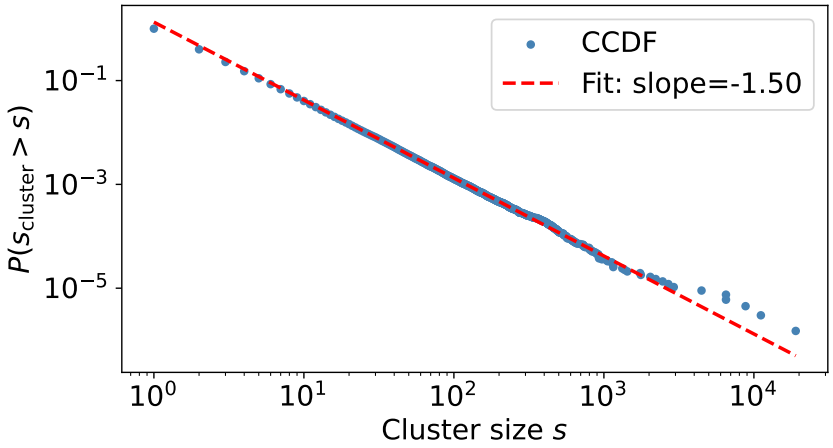
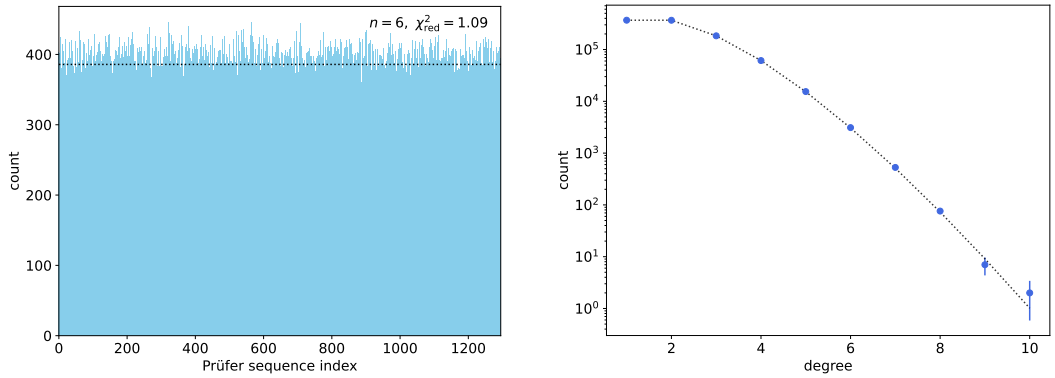



Figure 8: Cluster size distribution. A maximum-likelihood power-law fit to clusters with $s \geq 35$ is shown as a dashed red line. Each cluster (not data point) is counted once in the distribution.

In Fig. 10, we show timing measurements to benchmark the empirical performance of the cyclic coalescent algorithm implemented in Python. As shown in Fig. 10a, the measured time to generate a tree of size $n$ scales slightly superlinearly, with a measured slope of about 1.1, rather than 1 as expected for a linear-time algorithm. This observation is consistent with $O(n \log n)$ scaling. We believe that the theoretical time complexity is obscured in practice by details of the Python implementation, rather than a fundamental issue. Specifically, the tree is constructed by iteratively inserting edges into adjacency dictionaries associated with each node. As the tree is built up using random-access edge insertions, an extraneous logarithmic slowdown may result due to cache misses among these heap-allocated dictionaries.

To see this, we compare the performance to two baselines with known $O(n)$ time complexity. First, we compare to `random_labeled_tree()` from the `networkx` library [Hagberg et al., 2008], which constructs a random tree from a Prüfer sequence using the optimal $O(n)$ algorithm from [Wang et al., 2009]. Second, we compare to a trivial baseline with transparently linear algorithmic time complexity, shown below as the function `build_random_graph()`.

```
import networkx as nx
import numpy as np
```



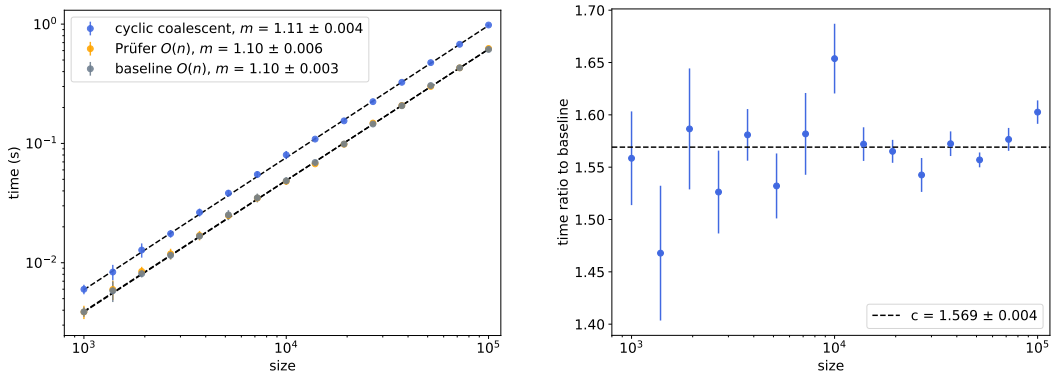

(a) Distribution over Prüfer sequences.

(b) Degree distribution.

Figure 9: Correctness validation for the cyclic coalescent. Panel (a) shows that 500,000 trees generated with the cyclic coalescent have a uniform distribution over the 1296 Prüfer sequences for size $n = 6$. Panel (b) shows the degree distribution of a tree of size $10^6$. Measured counts are shown in blue with Poisson error bars. The empirical counts match the theoretical distribution, shown with a black dotted line.

```python
def build_random_graph(n: int, rng: np.random.Generator):
    G = nx.empty_graph(n)
    edges = rng.integers(0, n, size=(n - 1, 2))
    for u, v in edges:
        G.add_edge(int(u), int(v))
    return G
```

As can be seen in Fig. 10a, the slopes of all three methods are consistent within error. The reported error bars are underestimates, as they do not incorporate variation in measured wall-clock time from run to run. In Fig. 10b, we show the ratio of the measured time of the cyclic coalescent to the `build_random_graph()` baseline. The ratios appear roughly constant and have no apparent trend, consistent with equivalent to linear performance. Running the cyclic coalescent takes about $60\%$ longer than a pure graph-construction baseline. This is expected because the cyclic coalescent constructs an additional binary coalescence tree containing $O(n)$ nodes alongside the random tree.




(a) Absolute time required to generate one cluster.

(b) Relative time required to generate one cluster.

Figure 10: Performance validation for the cyclic coalescent. Panel (a) shows the time required to generate a random tree as a function of tree size for the cyclic coalescent and two $O(n)$ baseline methods, described in the text. The data points show the mean and standard deviation over 20 trials. The black dashed lines show linear fits. The slopes are consistent among the methods. Panel (b) shows the performance ratio of the cyclic coalescent to the linear baseline `build_random_graph()` as a function of tree size. No clear trend is apparent. The black dashed line shows a constant fit.