Learn from your own latents and not from tokens: A sample-complexity theory

Daniel J. Korchinski *
Institute of Physics
EPFL
Lausanne, VD Switzerland
daniel.korchinski@epfl.ch

Alessandro Favero *
Department of Applied Maths and Theoretical Physics
University of Cambridge
Cambridge, UK
af940@cam.ac.uk

Matthieu Wyart
Department of Physics & Institute of Physics
Johns Hopkins University & EPFL
Baltimore, MD USA & Lausanne, VD Switzerland
matthieu.wyart@epfl.ch

Abstract

Generative models, from diffusion models to large language models, achieve remarkable performance but at a cost in training data orders of magnitude larger than what biological learners require. An alternative paradigm has emerged in which networks are trained to predict their *own* latent representations of related views or masked regions, as in data2vec and JEPA – an idea related to predictive-coding accounts of the cortex. Despite strong empirical results, the theoretical understanding of these methods remains limited. Central questions include: by how much does latent prediction actually improve data efficiency? Is there a benefit to stacking such methods into multi-scale hierarchies? We answer both using as data a tractable probabilistic context-free grammar that captures the compositional structure of natural language and images. Such a grammar generates strings of visible tokens by recursively applying production rules along a tree of hidden symbols of depth L. For such data, supervised or token-level SSL require a number of samples *exponential* in L to recover the latent tree; we prove that latent prediction achieves this with a number of samples *constant* in L, up to logarithmic factors. We confirm this bound with (i) a hierarchical clustering algorithm, (ii) an end-to-end neural network whose predictor-clusterer modules predict their own latents at each level via gradient descent, and (iii) the first sample-complexity analysis of data2vec, which we show implicitly performs hierarchical latent prediction. This suggests that explicit stacking such as H-JEPA is largely redundant.

1 Introduction

Generative models have reached striking empirical success. Diffusion models produce realistic images and video [1–3], while large language models trained by next-token prediction master grammar, world knowledge and reasoning [4–6]. Both rest on a single recipe: predict masked or future fragments of the raw signal at massive scale. Yet this success is bought at a cost biological learners do not pay.

Frontier LLMs are trained on 10^{13}–10^{14} tokens [7, 8], more than five orders of magnitude beyond what a child encounters before reaching adult-level competence [9–11]; state-of-the-art diffusion models likewise rely on billions of images [12]. This gap suggests that token-level pretraining is far

*Equal contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

from optimal, and several hypotheses have been advanced to explain it. One is that biological learning is multimodal and grounded [13, 14]. A second – the one we pursue – is that learning may not be most efficient at the level of raw tokens, but instead in a more abstract latent space [15–17]. Indeed, predicting semantic concepts or continuous latents has been shown to improve data efficiency [18, 19].

An alternative strategy to input reconstruction has gained traction in the last five years: the network is trained to predict its *own* latent representations of related views, masked regions or future signals. These targets are themselves the output of a learned encoder, lifting prediction into increasingly abstract spaces as learning progresses. The idea relates to computational neuroscience, where predictive coding posits that the cortex seeks to predict its own future activity [15, 20, 21]. A growing family of self-supervised methods instantiates this principle, including BYOL [22], DINO [23], data2vec [24] and JEPA [17].

Some advantages of these algorithms over input reconstruction approaches have been discussed in the literature, including the fact that they can be formulated as energy-based methods [17] or are more robust to the presence of irrelevant features in the data [25]. Yet understanding why and by how much these methods improve data efficiency remains a fundamental challenge in both machine learning and neuroscience. Another key question is the putative benefits of stacking such algorithms so as to obtain a multi-scale hierarchy of learned representations. This idea, proposed years ago [17], was recently implemented [26, 27] and appears to lead at best to moderate improvements, for reasons that remain currently unclear.

In this work, we address both questions by focusing on synthetic models of data with a hierarchical latent structure. Probabilistic context-free grammars (PCFGs) are a natural candidate: through recursive production rules they generate data compositionally from a tree of latent variables, and have long been influential as models of language [28, 29] and natural images [30, 31]. General PCFGs are notoriously hard to treat. We therefore work with the Random Hierarchy Model (RHM) [32], a simplified PCFG with fixed tree topology and random production rules. The RHM has produced quantitatively predictive theories, including relating neural scaling laws to the statistics of natural language [33, 34], and a theory of compositional generalization and memorization in diffusion models for language and images [35–38]. The RHM is parameterized by the number of production rules per symbol, m, and the depth of the latent tree, L. Prior work shows that for deep architectures, supervised end-to-end learning requires an order of m^L samples [32, 39], and token-level self-supervised objectives – masked-language modeling and diffusion – require of order m^{L+1} [33, 37, 40] (cf. fig. 1). Both costs grow *exponentially* with the depth L of the latent hierarchy (and thus polynomially in the input dimension, i.e., sequence length). Against this baseline, we establish three results:

1. **Efficient algorithm for representation learning** (Section 3)**.** We show that a hierarchical clustering algorithm recovers the full non-root latent tree from $\widetilde{O}(m^3)$ samples – *independent* of the depth L, and hence exponentially fewer than those required by token-level SSL.[2]
2. **An end-to-end stacked architecture with the same scaling** (Section 4)**.** We propose a new architecture where a stack of small predictor-clusterer modules predicts its own latents at each level. Trained end-to-end with gradient descent, it reaches the m^3 scaling on the RHM.[2]
3. **A theory of data2vec** (Section 5)**.** We show empirically and explain theoretically with reasonable assumptions that data2vec [24] implicitly performs hierarchical latent prediction, again leading to a sample complexity of order m^3. To our knowledge, this is the first sample-complexity analysis of such methods.

These results establish that learning from one's own latents can yield huge sample-complexity gains over token-level objectives. They also show that a single latent-prediction module such as data2vec is in effect already performing multi-scale latent discovery. Such methods may already be implicitly hierarchical, weakening the case for explicit stacking such as H-JEPA.

Related work. The compositional and hierarchical structure of natural data has long been put forward as a key feature making it learnable by neural networks. Deep networks represent hierarchically compositional functions with exponentially fewer parameters than shallow ones [41], yielding nonparametric sample-complexity bounds polynomial in the input dimension for both supervised [42] and self-supervised [43] tasks. These results characterize expressivity and information-theoretic statistics, but do not describe what gradient descent can actually learn from finite data. A complementary line shows that transformers can approximately implement parsing-style inference on

[2] Code and data are available at `github.com/pcsl-epfl/learn-from-latents-not-tokens`.

context-free grammars [44–46], characterizing the algorithm trained models implement, but not its sample complexity. The RHM [32] makes simplifying assumptions – frozen grammatical tree [47, 48] and random production rules [49] – that allow to fill this gap, enabling sample-complexity predictions that match observations in deep nets. Existing RHM analyses cover supervised learning [32] and token-level SSL [33, 37], but not learning from one's own latents, which is the focus of this work.

2 Preliminaries

The Random Hierarchy Model (RHM). The RHM [32] is a grammar whose rules are drawn at random. There are $L+1$ levels: the bottom one is what the learner sees, the L above it are hidden. Each level has its own vocabulary of v symbols. A *rule* says that one symbol may be replaced by a particular ordered tuple of s symbols from the level below. Of the v^s tuples that could appear at a given level, only vm are grammatical, chosen uniformly at random and distributed evenly, m to each of the v symbols above: every symbol has exactly m ways to expand, and no two symbols share an expansion, so that every grammatical tuple identifies its parent unambiguously. In the grammar of Figure 1–**top** ($L=2$, $v=2$, $m=2$, $s=3$), the symbol a can expand as $(0,0,0)$ or $(1,0,1)$, and α as (a,a,b) or (a,b,a). To generate a datum, draw one symbol uniformly at the top, replace it, via one of its m rules chosen at random, by the s children the rule specifies, do the same to each child with that level's rules. After L rounds, the leaves form a string of s^L visible tokens. The m tuples sharing a parent, such as $(0,0,0)$ and $(1,0,1)$, are called *synonyms*: exchanging one for another leaves the sentence grammatical and the derivation above the swap untouched. We denote the root of the tree by $h_1^{(L)}$, its immediate descendants by $h_1^{(L-1)},\dots,h_s^{(L-1)}$, and so on. Level (0) denotes the visible leaf tokens, with $h_i^{(0)}=x_i$. When training neural networks, we randomly sample a grammar instance with RHM parameters (number of symbols v, rules per symbol m, tuple size s, levels L) and then generate a dataset of P sentences from it. We refer the reader to Section B.1 for the formal definition of the RHM grammar.

Correlation-based learning in the RHM. A central observation from Cagnetta et al. [32] is that learning the RHM amounts to learning invariances under the exchange of synonyms. The statistical signal that identifies these synonym classes comes from correlations between a tuple and an external observable. Concretely, let $T^{(\ell)}$ be a tuple of sibling symbols at level ℓ and let Z be an observable elsewhere in the tree. This arrangement is depicted for different learning objectives in Figure 1. One can consider the connected correlation:

$$C_Z(\nu,z) := \mathbb{P}[T^{(\ell)}=\nu, Z=z] - \mathbb{P}[T^{(\ell)}=\nu]\mathbb{P}[Z=z].$$

If two tuples ν,ν' are synonyms, then they have identical correlations with Z: $C_Z(\nu,\cdot)=C_Z(\nu',\cdot)$, whenever Z depends on the tuple only through its parent X.[3] These identical correlations allow ν and ν' to be identified as synonyms for the latent X, but can be obscured by finite-sampling noise. The signal-to-noise ratio of this correlation signal is attenuated by the tree distance between the tuple $T^{(\ell)}$ and the observable Z. Along the path connecting them, each unresolved production rule averages over m synonymous choices, requiring a factor of m more samples to resolve the correlation. For both supervised classification and token-level self-supervision the overall sample complexity is therefore dominated by the longest path correlating a tuple with the target observable as illustrated in Figure 1.

Supervised classification [32, 39]: the observable Z is the class label at the root, and the hardest step is learning the rules mapping level-0 tuples to level-1 latents, requiring a sample complexity $P^{\text{sup}} \asymp m^L$. In contrast to shallow networks, which are cursed by dimensionality, deep networks trained on the supervised RHM classification task were found to saturate this scaling and reconstruct the latent hierarchy in their representations.

Token-level self-supervision [33, 37, 40]: the observable Z is a masked or noised token. In contrast to supervised learning, where the hardest step is the first one, token-level SSL requires only $P^{\text{tok}}_{\ell=1} \asymp vm^3$ samples to learn the first-level latents. Once these low-level latents have been internally reconstructed, the model can use them as coarse context variables. To learn higher levels, the relevant statistics are token–latent correlations: a visible token is correlated with a latent representation of its context

[3] Indeed, conditioned on the parent of the tuple, the rest of the tree is independent of which production rule was used to generate the tuple, and the uniform choice of production rules gives $\mathbb{P}[T^{(\ell)}=\nu]=\mathbb{P}[T^{(\ell)}=\nu']$ within each synonym class.

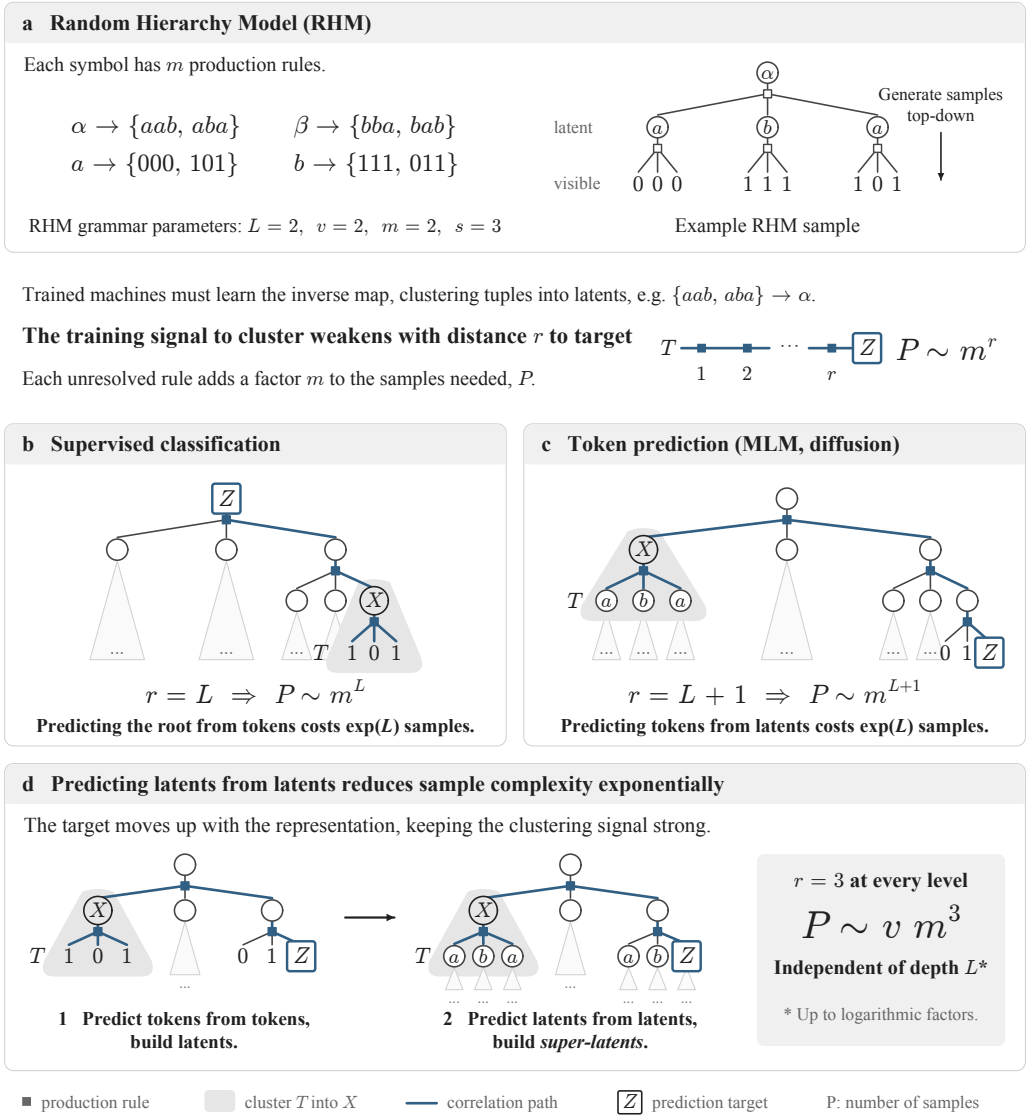


Figure 1: **Learning from latents reconstructs the RHM hierarchy more efficiently than token-level objectives. a.** An example Random Hierarchy Model (RHM) grammar (left) and a generated sample (right), with $L = 2$. The RHM generates tokens by recursively choosing one of m equiprobable production rules per symbol from a fixed grammar of v symbols per level. Each rule expands a symbol into s children; L rounds give s^L visible tokens. Trained on this data, token and root predictors learn to cluster synonymic tuples into latents in order to achieve good performance. **b–c.** For supervised classification (b) and token-level SSL (c), we highlight the hardest latents to learn. These are the latents farthest from the training objective. The path length r counts production rules (squares), each adding a factor m to the samples required to learn those latents. Triangles abbreviate subtrees for visual clarity. **d.** Predicting tokens from tokens builds latents; predicting latents from latents builds super-latents. The distance remains $r = 3$ at every level. Iterating recovers the non-root hierarchy with $\widetilde{O}(vm^3)$ samples under the assumptions of Theorem 1, independent of depth (up to logarithmic factors).

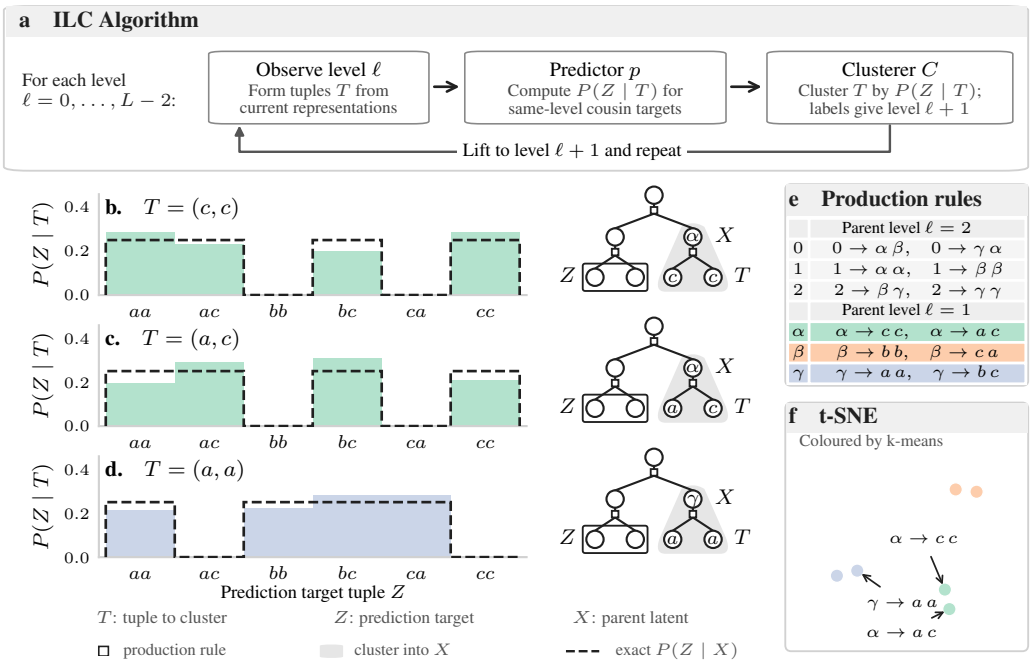


Figure 2: **The Iterative Latent Clustering (ILC) algorithm. a.** At each level, predict cousin targets from tuples, cluster tuples by their prediction vectors, and use the cluster labels as next-level latents. **b–d.** Empirical context vectors $P(Z \mid T)$ (filled histograms) and exact parent-conditioned distributions $P(Z \mid X)$ (dashed outlines) for synonymous tuples (c, c) and (a, c) (**b** and **c** respectively) with shared latent parent α and the same exact predictions; (a, a) (**d**) has parent γ and correspondingly a different context vector. **e.** The fixed production rules generating the example. **f.** t-SNE of the full empirical context vectors, coloured by k-means assignment; arrows identify the three selected tuples. The same colours are used in the histograms and rule table. Example parameters are $P = 500$, $L = 2$, $s = 2$, $v = 3$, and $m = 2$.

through a longer tree path, leading to sample complexity

$$P_\ell^{\text{tok}} \asymp v\, m^{\ell+2}. \tag{1}$$

for level $\ell \geq 1$ latents. Neural networks trained with token-level SSL objectives were empirically shown to saturate this staged scaling: as the dataset size increases, lower-level latents appear first, and higher-level latents become learnable thereafter. The overall sample complexity is gated behind reconstructing latents $\ell = L - 1$ leading to $P_{\text{tok}} \asymp vm^{L+1}$.[4]

3 Recovering the RHM hierarchy by clustering

As discussed in the previous section, the limitation of token-level objectives is that learning latents requires detecting ever weaker latent-token correlations as one moves up the hierarchy. The learn-from-your-own-latent setting removes this bottleneck: once level ℓ has been recovered, both the conditioning object and the prediction target are lifted to level ℓ. The next stage is then again a local synonym-clustering problem, but now with the same statistical strength at every scale. In other words, every level becomes as easy as the first token-level step. Given Equation (1), this suggests that the whole non-root hierarchy should be recoverable from $P \asymp vm^3$ samples. This is illustrated in Figure 1d.

Iterative latent-clustering algorithm (ILC). The preceding section framed learning the RHM as identifying synonyms through their identical correlations with an external observable. Here we cast

[4]Notice that since the root is sampled uniformly and each root symbol has m equiprobable rules, the distribution does not reveal how the vm valid top-level tuples are partitioned into v groups of size m, one group per root symbol. Thus unsupervised learning can recover latents $h^{(1)}, h^{(2)}, \ldots, h^{(L-1)}$ and the support of valid top-level tuples, but not the root labels themselves.

this as a clustering problem. Let T be a level-ℓ tuple with parent X at level $\ell+1$, and let Z be a level-ℓ symbol in a *cousin* tuple of T, i.e., a tuple whose parent is a sibling of X, so that Z and T share their level-$(\ell+2)$ grandparent but not their parent (see Figure 2). Let $e_Z \in \mathbb{R}^v$ be the one-hot encoding of Z. For each grammatical tuple ν, define the *context vector*

$$\phi_\ell(\nu) \;:=\; \mathbb{E}[e_Z \mid T=\nu] \;=\; \big(\mathbb{P}[Z=z \mid T=\nu]\big)_z,$$

i.e., the conditional distribution of the cousin symbol given the tuple.

Each grammatical tuple ν has a unique parent symbol $\mu = h_\ell(\nu)$, and, conditioned on X, the rest of the tree is independent of which of the m rules generated T. Hence, at the population level, the vm context vectors take only v values, the *parent context vectors* $\Phi_\ell(\mu) = \mathbb{P}[Z=\cdot \mid X=\mu]$ (black dashed lines in Figure 2). Synonym classes are identifiable provided these v centers are distinct.

In practice one only has access to the *empirical* context vectors $\widehat{\phi}_\ell(\nu)$, obtained by averaging e_Z over the training samples. They fluctuate around $\Phi_\ell(h_\ell(\nu))$ and are resolved once their sampling noise falls below the separation between centers. Clustering the vm empirical context vectors into v groups then recovers the synonym classes, and the cluster index of a tuple defines its level-$(\ell+1)$ latent.

Algorithm 1 (Iterative Latent Clustering, ILC) operationalizes this procedure. Starting from the visible tokens, at each step ℓ the algorithm takes the decoded level-ℓ sequence, forms its tuples, estimates their empirical context vectors, clusters them, and relabels each tuple by its cluster to obtain level $\ell+1$. This clustering procedure is illustrated graphically in Figure 2: the predictor p estimates context vectors, while the clusterer C maps these vectors to next-level latent labels. Theorem 1 proves that $P \gtrsim vm^3$ samples suffice to reconstruct the hierarchy, up to logarithmic factors, and Figure 3 shows numerically that this threshold is tight.

Theoretical sample complexity. An RHM grammar is *balanced* if, for every level ℓ, every grammatical tuple occurs with probability of order $1/(vm)$. It is *separated* if every pair of parent context vector $\Phi_\ell(\mu), \Phi_\ell(\mu')$ are separated by distance $\gtrsim 1/m$. We refer the reader to Appendix B.3 for the formal statements of these assumptions and their justification in the limit of $v \to \infty$ at fixed $f \in (0,1)$, where $f := mv/v^s$ is the fraction of possible s-tuples allowed by the grammar. For a balanced and separated RHM grammar with $f < 1$, and assuming the existence of a *stable* clustering module (see Appendix B.3), we have:

Theorem 1 (Recovery of the non-root hierarchy; informal)**.** *Fix an RHM grammar that is balanced and separated, and a clustering algorithm stable under perturbations. Then there exists a constant $C > 0$, depending only on the constants in those assumptions and on s, such that the following holds. If*

$$P \geq C\left[vm \log\frac{Lvm}{\delta} + \frac{vm^3}{1-f}\log\frac{Lvm}{\delta}\right], \tag{2}$$

then the iterative latent-clustering algorithm recovers $h^{(1)}, h^{(2)}, \ldots, h^{(L-1)}$ and all production-rule classes below the root with probability at least $1-\delta$, up to independent permutations of the latent labels at each level.

In particular, for fixed f bounded away from 1 and up to logarithmic factors, $P_{\mathrm{ILC}} = \widetilde{O}(vm^3)$.

In Appendix B.3, we give the formal theorem and its proof, which is a level-by-level induction based on concentration of the empirical context vectors.

Numerical validation. We generate samples from the fixed-tree RHM and apply the level-by-level procedure above, estimating the empirical context vectors $\widehat{\phi}_\ell(\nu)$ for observed grammatical tuples and clustering them using k-means (with $k = v$). Figure 3–**left** shows the reconstruction accuracy for $L = 5$, $s = 3$, $v = 8$, and varying m. In the unrescaled plot, the transition to perfect reconstruction shifts to larger sample sizes as m increases. After rescaling the number of samples by vm^3, the curves collapse and reconstruction becomes accurate once P/vm^3 is of order one, confirming the predicted m^3 scaling of the local clustering threshold.

4 Gradient-based Iterative Latent Clustering

In this section, we ask whether the same scaling of the ILC algorithm can be achieved by a differentiable architecture trained end-to-end by gradient descent.

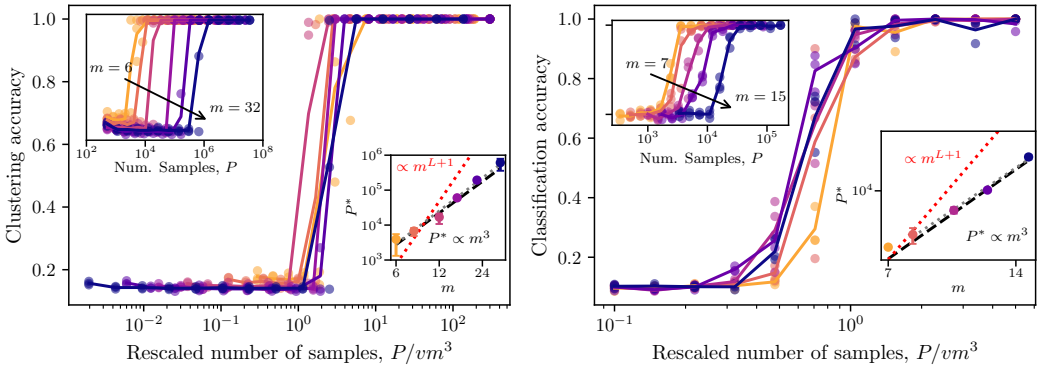


Figure 3: **Recursive clustering recovers the RHM hierarchy at the predicted scale. Left**: Clustering reconstruction accuracy of the iterative clustering algorithm using k-means ($L = 5, s = 3, v = 8$). **Right**: Class reconstruction accuracy of SLC architecture ($L = 4, s = 3, v = 10$) trained end-to-end, evaluated by linear probe on the highest level latents $\widehat{h}^{(L-1)}$. In each panel, the main plot is a *scaling collapse*, effected by rescaling number of samples by predicted sample complexity $P \sim vm^3$. Insets show raw data (top left) and inferred sample complexity (bottom right) P^* for which accuracy reaches 50%. Monte-Carlo fitting $P^* \propto m^\alpha$ (grey-dots) gives $\alpha = 3.1 \pm 0.3$ (ILC) and $\alpha = 2.9^{+0.4}_{-0.3}$ (SLC), see Section F for details.

The Stacked Latent-Clustering (SLC) network. We mirror the recursion of ILC with a stack of $L - 1$ identical modules. Each module is composed of two subnetworks:

Predictor p: $\mathrm{Pred}^{(\ell)}$ takes an s-tuple of level-ℓ tokens and is trained to output a categorical distribution over its cousin tokens; this is the neural analog of the empirical cousin context vector.

Clusterer C: $\mathrm{Clust}^{(\ell)}$ then maps each prediction vector to a soft assignment in a discrete codebook through a contrastive objective: prediction vectors with high similarity are pulled to the same code while dissimilar ones are pushed apart. The number of clusters is not fixed but learned from the data, making it more general than the k-means implementation of ILC.

The cluster assignments output by C become the input tokens of the level-$(\ell + 1)$ module. To keep the difficulty of the prediction task constant across depth, cluster outputs are tokenized with a softmax. Prediction targets are computed by a teacher network whose weights track an exponential moving average of the student, $W_{\text{teacher}} \leftarrow (1 - \alpha_{\text{ema}})\, W_{\text{teacher}} + \alpha_{\text{ema}}\, W_{\text{student}}$, a common trick to prevent the representation collapse common to self-distilled SSL [50]. Full architectural and training details are given in Appendix C.1.

Results. Figure 3–**right** shows the accuracy of a linear probe trained to recover the top-level latent from the frozen final-layer representations of an SLC network, for $L = 4$, $s = 3$, $v = 10$, and varying m. Rescaling the sample axis by vm^3 collapses the curves, showing that the network reaches the scaling of Theorem 1. Figure 13 shows that the sample complexity for $L \in \{3, \dots, 7\}$ does not vary. Interestingly, inserting a stop-gradient at every module boundary (or even between p and C), so that each level is trained *only* by its own prediction and clustering losses, leaves the vm^3 scaling unchanged (Figure 10). SLC therefore admits a strictly local learning rule, consistent with biologically motivated learning schemes.

5 Analysis of data2vec

In this section, we argue that the iterative clustering mechanism is *implicit* in data2vec [24], a popular SSL method that predicts its own latent representations. We choose data2vec over the closely related JEPA family because it comes with a published recipe for discrete-token inputs. In particular, data2vec uses a student-teacher setup. The student network is fed a partially masked input S; the teacher sees the unmasked input x. At every masked position i, the student is trained to predict the target $Y_i(x)$ given by the average of the last K block activations of a teacher network applied to the unmasked input x, as illustrated in Figure 4–**left**. We follow the data2vec recipe for text, which trains the student with a smooth-L1 loss. Our theoretical analysis idealizes this loss as a squared loss, to

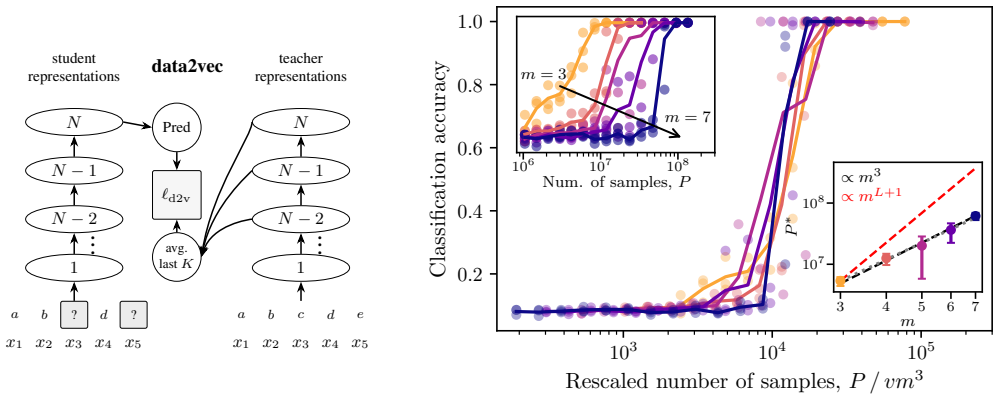


Figure 4: **Data2vec on RHM. Left:** data2vec learns in a teacher-student setting; the student is supervised on reconstruction of pooled teacher representations at masked token positions. **Right:** Root-classification accuracy of probes on pooled final-layer features, as a function of the number of online pretraining samples P. RHM parameters: $v = 16$, $s = 2$, $L = 4$. Insets and main are as in Figure 3, statistical fit $P^* \sim m^\alpha$ (grey dots) yields $\alpha = 2.8^{+0.3}_{-0.4}$.

which it is proportional whenever every per-coordinate residual is below a threshold of order 1.[5] As with SLC, the teacher has the same architecture as the student, and its weights track an exponential moving average of the student's weights, updated after each gradient step.

Empirical sample complexity. We pretrain data2vec on the RHM. We follow the data2vec text recipe: a fraction $p_{\text{mask}} = 0.15$ of positions are masked. Full details are presented in Appendix D. We compare two data-presentation regimes: *online*, where each step draws a fresh batch from the grammar, and *offline*, where a fixed training set of P unique strings is reused across epochs. After pretraining, we freeze the encoder and train a one-hidden-layer MLP probe to predict the root label from mean-pooled final-layer features.

Figure 4–**right** shows the downstream root-classification accuracy of the probes for $L = 4$, $s = 2$, $v = 16$ in the online setting.[6] The learning curves collapse onto a single curve after rescaling the sample axis by vm^3. Root classification on the RHM requires resolving all $L - 1$ non-root latent levels and then identifying the partition of top-level tuples into root classes. As discussed in Section 2, token-level SSL is bottlenecked by learning the level-$(L-1)$ rules from tokens, which costs $\sim vm^{L+1} = vm^5$ samples; an additional $\mathcal{O}(vm)$ labeled examples then suffice to learn the root partition. The observed scaling is hence incompatible with this token-level prediction baseline. We verify this directly by training parameter-matched transformers on the same data with a masked-language-modeling objective. Their sample complexity follows the token-level scaling vm^{L+1} rather than vm^3 (see Appendix D).

Theoretical analysis. We now explain the vm^3 scaling by tying data2vec back to the iterative latent clustering mechanism of Section 3. The argument proceeds inductively by idealizing the teacher EMA as a discrete sequence of phases: within each phase, the teacher is held fixed and the student builds a new higher level of latent variable; between phases the teacher is refreshed to absorb new latents acquired by the student, enriching the target representations of the student. This is precisely the iterative latent clustering mechanism of Section 3, and hence the vm^3 scaling follows.

Assumptions. We formalize the above argument through two assumptions, the first of which we verify numerically in Figure 14.

(A1) Targets carry the learned latents. Let $z_i^{(\ell)} := h^{(\ell)}_{\lceil i/s^\ell \rceil}$ be the level-ℓ ancestor of position i, and let $e_z \in \mathbb{R}^v$ denote the one-hot encoding of a latent symbol z. If the encoder linearly represents the

[5] Replacing the smooth-L1 loss by a pure squared loss leaves the vm^3 scaling unchanged (Appendix D, Figure 16), so this idealization does not affect our conclusions.

[6] The same plots are reported in Appendix D for the offline setting.

latents $h^{(1)}, \dots, h^{(\ell)}$, then the teacher target admits a decomposition

$$Y_i(x) \;=\; F_i(S) \;+\; \sum_{a=0}^{\ell} B_a \, e_{z_i^{(a)}} \;+\; \text{residual},$$

where $F_i(S)$ collects components determined by the visible inputs S and the residual term collects everything else, which is typically nonlinear and not used in our analysis. We require that the linear maps B_a keep different distributions over symbols apart: there is a constant $c_B > 0$, independent of v, m, L, such that for any two probability vectors p, q, $\|B_a\,(p-q)\|_2 \;\geq\; c_B\,\|p-q\|_2$. The residual architecture of the transformer makes the decomposition natural: a feature decodable from one block propagates through the identity path to every later block, and hence appears with a non-zero linear coefficient in the layers entering the top-K teacher average. At initialization, $\ell = 0$ and the only ancestor is x_i itself, so $Y_i(x)$ contains a linear component of the masked one-hot inputs. We verify this assumption numerically in Figure 14.

(A2) Correlation learning. Whenever a correlation between the prediction target and a feature of the visible input becomes detectable above sampling noise, gradient descent extracts that feature. This is the same correlation-learning hypothesis used in previous RHM analyses [33, 37], where it is supported empirically and by one-step gradient calculations.

Phase 0. The population minimizer of the data2vec loss is the conditional expectation $\mathbb{E}[Y_i(x) \mid S]$. By linearity of conditional expectation, in phase 0, predicting $Y_i(x)$ from S reduces, up to the visible component and the non-linear residual, to predicting the masked one-hot e_{x_i} from its visible context. This is exactly the token-level masked-prediction problem analyzed in Section 2: by (A1), B_0 preserves distances between conditional token distributions up to the constant c_B, so local token–token correlations become detectable for $P \gtrsim vm^3$, and by (A2) the network extracts them. The resulting representation collapses synonymous level-0 tuples onto a common image, and the level-1 ancestor becomes linearly decodable from the encoder.

Phase $\ell \geq 1$. Suppose phases $0, \dots, \ell-1$ have produced linearly decodable representations of $h^{(1)}, \dots, h^{(\ell)}$ inside the encoder. After the next teacher update, these latents enter the target, and (A1) gives the new component $\mathbb{E}[e_{z_i^{(\ell)}} \mid S]$ in the optimal predictor. Because levels $1, \dots, \ell$ are decodable, the visible context S can be parsed into level-ℓ symbols away from the masked position. In particular, it contains the level-ℓ cousin tuple T_ℓ, the closest level-ℓ tuple sitting outside the production rule of position i. Conditioning on T_ℓ produces the context vector $\phi_\ell(\nu) = \mathbb{E}[e_{z_i^{(\ell)}} \mid T_\ell = \nu]$ studied in Section 3. The prediction problem at scale ℓ has *lifted* from leaves to level-ℓ tuples, and is now identical to the clustering problem solved by the ILC algorithm of that section, requiring $P \gtrsim vm^3$. When the network extracts this signal, the level-$(\ell+1)$ ancestor $z_i^{(\ell+1)}$ becomes linearly decodable, and the next teacher update carries it into the target.

The induction continues up to $\ell = L-2$. Every phase shares the same vm^3 sample complexity, so the offline sample complexity equals the per-phase threshold, $P_{\mathrm{d2v}} \asymp vm^3$, and depth enters only through the number of phases. Finally, our argument also explains why the teacher–student loop does not collapse. Indeed, the target always contains something the student has not yet extracted (e.g., the masked token at initialization or a new latent after each refresh), so the optimal prediction $\mathbb{E}[Y_i(x) \mid S]$ varies with S and a constant representation is not a minimizer. Novelty then runs out only once the last level has been extracted, possibly leading to collapse at late times.[7]

The encoder internally clusters synonyms. Our theoretical argument predicts that the data2vec encoder collapses synonymous level-ℓ tuples onto a common representation – the very geometric signature of clustering. We test this prediction directly by measuring the *synonym clustering score* $\mathcal{C}_\ell$ of the encoder at level ℓ by comparing the change in its hidden representations under two interventions on the input: a *synonym swap*, which resamples every level-ℓ production rule and thus replaces each level-$(\ell-1)$ tuple by a synonym while leaving higher-level latents unchanged, and a *generic swap* to an unrelated input. Figure 5 reports $\mathcal{C}_\ell$ at each non-root level after the third block of the encoder as a function of pretraining samples P. The scores at all three levels rise from zero

[7] Notice that in continuous time the same holds provided the teacher lags the student by at least the time needed to learn one new level, so that the target still carries novelty.

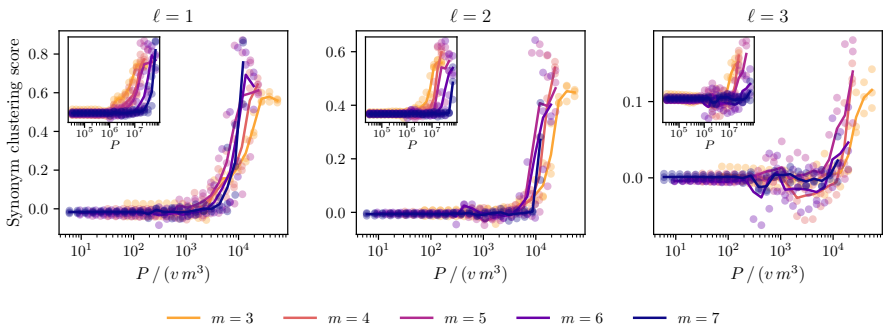


Figure 5: **Data2vec clusters synonyms.** Synonym clustering score $\mathcal{C}_\ell$ of the encoder at $\ell \in \{1, 2, 3\}$, for $L = 4$, $s = 2$, $v = 16$. A positive score means synonyms are mapped closer together than non-synonyms. *Insets:* raw curves vs. P. *Main:* same curves rescaled by vm^3 collapse.

and collapse onto a common curve once the sample axis is rescaled by vm^3. This is direct evidence that data2vec's encoder implements the same recursive clustering mechanism as the ILC algorithm of Section 3, with the same per-level vm^3 sample complexity predicted by the theory.

6 Conclusion

We have shown that learning from one's own latents recovers the full non-root latent tree of the RHM from a number of samples scaling as m^3, exponentially fewer than the m^{L+1} required by token-level self-supervised objectives. We proved this for a hierarchical clustering algorithm, showed empirically that an end-to-end neural network of predictor-clusterer modules reached the same scaling, and argued that data2vec does so by implicitly implementing the same mechanism. The main lesson is that latents at the same level of the hierarchy are far more correlated with each other than they are with raw tokens, so predicting from one's own latents amplifies a signal that token-level prediction dilutes. This gives a quantitative footing to the common intuition that token-level prediction is suboptimal.

Practical implications. Recent work supports that empirical neural scaling laws in language are governed by power-law decays of token correlations with context length [33, 34, 37]. Can generative models trained with own-latent self-supervision systematically beat existing scaling laws, alone or in combination with token-level losses? A useful first step would be a controlled comparison between data2vec and a same-architecture next-token-prediction baseline as the training-set size P is varied: at small P the two should diverge sharply, while at very large P they should converge to the same latent representations. Such a comparison would empirically test our central prediction and, if confirmed, motivate latent-supervised generative models as a route to break current scaling regimes.

Biological plausibility. As we show in Appendix C.2, the predictor-clusterer modules of Section 4 can also reach the m^3 scaling of sample complexity using only quasilocal learning rules – stop-gradients between modules in place of end-to-end backpropagation. In this local learning setting our method is conceptually similar to contrastive predictive coding [51] and CLAPP [20], which both propose biologically plausible self-supervised objectives that predict in latent rather than raw-sensory space. Does the remarkable sample efficiency of the human brain stem from this mechanism? It would be interesting to test whether such bio-plausible architectures can reach an m^3 sample complexity on the RHM simply by learning from their own latents.

Limitations. The RHM is a deliberately simple synthetic language: its context-free grammar has a fixed tree topology and unambiguous, non-recursive production rules. Extending the analysis to (i) variable tree topologies, (ii) recursive production rules that can call themselves, and (iii) explicitly context-dependent rules will bring the theory closer to natural language and images. Although our analysis suggests that stacking JEPAs as in H-JEPA is largely redundant, very recent self-supervised architectures claim explicit hierarchical supervision but are neither stacked JEPAs nor stacked data2vecs, and instead resemble data2vec-style training procedures with multi-scale teachers [52, 53]. These architectures would be interesting to study theoretically.

Acknowledgments and Disclosure of Funding

We thank Francesco Cagnetta, Francesco D'Amico, Ariane Delrocq, Antonio Sclocchi, and Wu Zihan for discussions and feedback on the manuscript, and the members of the Simons collaboration for discussions. D. J. Korchinski acknowledges financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC PDF - 587940 - 2024). A. Favero acknowledges support from the Infosys-Cambridge AI Centre. This work was supported by the Simons Foundation through the Simons Collaboration on the Physics of Learning and Neural Computation (Award ID: SFI-MPS-POL00012574-05), PI Wyart.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Wing Yin Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI Technical Report, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[5] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[6] DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[7] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[9] Jill Gilkerson, Jeffrey A. Richards, Steven F. Warren, Judith K. Montgomery, Charles R. Greenwood, D. Kimbrough Oller, John H. L. Hansen, and Terrance D. Paul. Mapping the early language environment using all-day recordings and automated analysis. *American Journal of Speech-Language Pathology*, 26(2):248–265, 2017.

[10] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[11] Michael C. Frank. Bridging the data gap between children and large language models. *Trends in Cognitive Sciences*, 27(11):990–992, 2023.

[12] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[13] Linda B. Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial Life*, 11(1–2):13–29, 2005.

[14] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022.

[15] Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

[16] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[17] Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

[18] Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. LLM pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*, February 2025. doi: 10.48550/arXiv.2502.08524.

[19] Yuliang Liu, Yunchong Song, Yixuan Wang, Kewen Ge, Alex Lamb, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Next concept prediction in discrete latent space leads to stronger language models. *arXiv preprint arXiv:2602.08984*, February 2026. doi: 10.48550/arXiv.2602.08984.

[20] Bernd Illing, Jean Ventura, Guillaume Bellec, and Wulfram Gerstner. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. In *Advances in Neural Information Processing Systems*, volume 34, pages 30365–30379. Curran Associates, Inc., 2021.

[21] Beren Millidge, Anil Seth, and Christopher L. Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2022.

[22] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent a new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 21271–21284, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6.

[23] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[24] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning*, pages 1298–1312. PMLR, June 2022.

[25] Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. Joint embedding vs reconstruction: Provable benefits of latent space prediction for self-supervised learning. *arXiv preprint arXiv:2505.12477*, 2025.

[26] Lihuan Li, Hao Xue, Shuang Ao, Yang Song, and Flora Salim. HiT-JEPA: A Hierarchical Self-supervised Trajectory Embedding Framework for Similarity Computation. *arXiv preprint arXiv:2507.00028*, June 2025. doi: 10.48550/arXiv.2507.00028.

[27] Abanoub M. Girgis, Ibtissam Labriji, and Mehdi Bennis. Hierarchical JEPA meets predictive remote control in beyond 5G networks. *arXiv preprint arXiv:2602.07000*, January 2026. doi: 10.48550/arXiv.2602.07000.

[28] Taylor L. Booth and Richard A. Thompson. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, 1973.

[29] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[30] Ulf Grenander. *Elements of Pattern Theory*. Johns Hopkins University Press, 1996.

[31] David Mumford. Pattern theory: a unifying perspective. In *First European Congress of Mathematics, Vol. I (Paris, 1992)*, volume 119 of *Progress in Mathematics*, pages 187–224. Birkhäuser, 1994.

[32] Francesco Cagnetta, Leonardo Petrini, Umberto M Tomasini, Alessandro Favero, and Matthieu Wyart. How deep neural networks learn compositional data: The random hierarchy model. *Physical Review X*, 14(3):031001, 2024.

[33] Francesco Cagnetta and Matthieu 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.

[34] Francesco Cagnetta, Allan Raventós, Surya Ganguli, and Matthieu Wyart. Deriving neural scaling laws from the statistics of natural language. *arXiv preprint arXiv:2602.07488*, 2026.

[35] Antonio Sclocchi, Alessandro Favero, and Matthieu Wyart. A phase transition in diffusion models reveals the hierarchical nature of data. *Proceedings of the National Academy of Sciences*, 122(1):e2408799121, January 2025. doi: 10.1073/pnas.2408799121.

[36] Antonio Sclocchi, Alessandro Favero, Noam Itzhak Levi, and Matthieu Wyart. Probing the latent hierarchical structure of data via diffusion models. *Journal of Statistical Mechanics: Theory and Experiment*, 2025(8):84005, August 2025. ISSN 1742-5468. doi: 10.1088/1742-5468/aded6c.

[37] Alessandro Favero, Antonio Sclocchi, Francesco Cagnetta, Pascal Frossard, and Matthieu Wyart. How compositional generalization and creativity improve as diffusion models are trained. In *International Conference on Machine Learning*, pages 16286–16306. PMLR, 2025.

[38] Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Bigger isn't always memorizing: Early stopping overparameterized diffusion models. *arXiv preprint arXiv:2505.16959*, 2025.

[39] Yunwei Ren, Yatin Dandi, Florent Krzakala, and Jason D. Lee. Provable learning of random hierarchy models and hierarchical shallow-to-deep chaining. *arXiv preprint arXiv:2601.19756*, 2026.

[40] Francesco Cagnetta, Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Scaling laws and representation learning in simple hierarchical languages: Transformers versus convolutional architectures. *Physical Review E*, 112(6):065312, 2025.

[41] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but-not-shallow networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[42] Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. *Annals of Statistics*, 48(4):1875–1897, 2020.

[43] Song Mei. U-Nets as belief propagation: Efficient classification, denoising, and diffusion in generative hierarchical models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2404.18444.

[44] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, learning hierarchical language structures. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=mPQKyzkA1K.

[45] Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 16513–16542, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.1029. URL https://aclanthology.org/2023.emnlp-main.1029/.

[46] Jerome Garnier-Brun, Marc Mézard, Emanuele Moscato, and Luca Saglietti. How transformers learn structured data: Insights from hierarchical filtering. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2408.15138.

[47] Elchanan Mossel. Deep learning and hierarchical generative models. *arXiv preprint arXiv:1612.09057*, 2016.

[48] Eran Malach and Shai Shalev-Shwartz. A provably correct algorithm for deep learning that actually works. *arXiv preprint arXiv:1803.09522*, 2018.

[49] Eric DeGiuli. Random language model. *Physical Review Letters*, 122(12):128301, 2019. doi: 10.1103/PhysRevLett.122.128301.

[50] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[51] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*, January 2019. doi: 10.48550/arXiv.1807.03748.

[52] Scott C. Lowe, Anthony Fuller, Sageev Oore, Evan Shelhamer, and Graham W. Taylor. Self-distillation of hidden layers for self-supervised representation learning. *arXiv preprint arXiv:2603.15553*, March 2026. doi: 10.48550/arXiv.2603.15553.

[53] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, March 2026. doi: 10.48550/arXiv.2603.19312.

[54] Jack T Parley, Francesco Cagnetta, and Matthieu Wyart. Deep networks learn to parse uniform-depth context-free languages from local statistics. *arXiv preprint arXiv:2602.06065*, 2026.

[55] Lorenzo Defilippis, Florent Krzakala, Bruno Loureiro, and Antoine Maillard. Optimal scaling laws in learning hierarchical multi-index models. *arXiv preprint arXiv:2602.05846*, 2026.

[56] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[57] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 9912–9924, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6.

[58] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, January 2024. ISSN 2835-8856.

[59] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herve Jegou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, May 2026. ISSN 2835-8856.

[60] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, January 2022.

[61] Timothée Darcet, Federico Baldassarre, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Cluster and predict latent patches for improved masked image modeling. *Transactions on Machine Learning Research*, February 2025. ISSN 2835-8856.

[62] Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, March 2026. doi: 10.48550/arXiv.2603.14482.

[63] Ariane Delrocq, Wu S. Zihan, Guillaume Bellec, and Wulfram Gerstner. Self-supervised local learning rules learn the hidden hierarchical structure of high-dimensional data. *arXiv preprint arXiv:2605.18557*, May 2026. doi: 10.48550/arXiv.2605.18557.

[64] Pierre Quinton and Valérian Rey. Jacobian descent for multi-objective optimization. *arXiv preprint arXiv:2406.16232*, February 2025. doi: 10.48550/arXiv.2406.16232.

[65] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330701.

A Further related work

Approximation- and information-theoretic bounds. Deep networks can represent hierarchically compositional functions with exponentially fewer parameters than shallow ones [41], which translates into nonparametric sample-complexity bounds polynomial in the input dimension for both supervised [42] and self-supervised [43] tasks. These results characterize expressivity and information-theoretic statistics, but do not describe what gradient descent can actually learn from a finite training set.

Supervised learning. Mossel [47] introduced phylogeny-inspired generative models on regular trees, and Malach and Shalev-Shwartz [48] showed that they can be learned by an iterative clustering algorithm. DeGiuli [49] introduced languages with random production rules. The RHM [32] combines both assumptions, leading to models with controllable correlations where sample complexity can be predicted and compared to observations in deep nets. Ren et al. [39] proved rigorously that result for multi-step gradient descent. Parley et al. [54] obtains sample complexity for non-regular trees. Defilippis et al. [55] derive optimal scaling laws for hierarchical multi-index models, a related class of generative models with continuous latents.

Self-supervised learning. A complementary line of work has shown that transformers can approximately implement parsing-style (inside-algorithm) inference on context-free grammars [44–46]. These results interpret the algorithm that trained LLMs implement, but do not characterize the learning mechanism nor its sample complexity. Such analyses do exist for the RHM, but only for next-token prediction [32] and for diffusion [37, 38], not for learning from one's own latents as considered here.

Clustering-based approaches to self-distillation. Our SLC method is similar in spirit to clustering or self-learned pseudo-labeling-based methods for image-representation learning, such as DeepCluster [56] which in turn evolved into SwAV [57], the DINO family of models [23, 58, 59], iBOT [60], and CAPI [61]. Our method adds *explicit* hierarchical clustering – it would be interesting to test to what degree *implicit* hierarchical supervision is present for these clustering-based approaches.

Explicitly hierarchical self-distillation SSL. Schematic realizations of the H-JEPA [17], V-JEPA 2.1 [62], Bootleg [52], and our SLC architectures are depicted in fig. 6. Bootleg and V-JEPA 2.1 are extremely recent, report state of the art performance, and employ multi-loss optimization with one prediction per sampled representation level. Bootleg is schematically very similar to data2vec (cf. fig. 4–**left**), except that in bootleg gradients are computed after averaging losses over representations, whereas in data2vec the averaging is done first over representations. V-JEPA 2.1 meanwhile collates representations from throughout the student network, similar to H-JEPA, but also allows low-level target predictions to be influenced by higher level latents. This high→low prediction path is missing in H-JEPA, but present in SLC, V-JEPA 2.1, and Bootleg. It is important to clarify whether this is related to the fact that, with few exceptions [26, 27], "standard" H-JEPA appears to have found limited application in the literature.

Predictive coding and the RHM. Delrocq et al. [63] recently showed that a backprop-free predictive coding model, CLAPP, can learn the RHM's hierarchical structure with sample complexity comparable to that of backprop. In that work, the model was trained on the RHM in the setting where all production rules exist, such that all sentences belong to the language (corresponding to $f = 1$ in our notations). In that case, self-supervised methods cannot learn the RHM rules, as all inputs are equiprobable and correlations within the input are absent. However, the variation of CLAPP considered in that work uses a contrastive update rule that is gated by an extra learning signal: updates are given a ± 1 sign indicating whether pairs belong to the same top-level class. The sample complexity obtained was that of supervised learning $\sim m^L$. Baseline CLAPP [20] does not rely on this non-local context supervision – it would be interesting to understand whether it learns the RHM rules in sample complexity m^3 when $f < 1$.

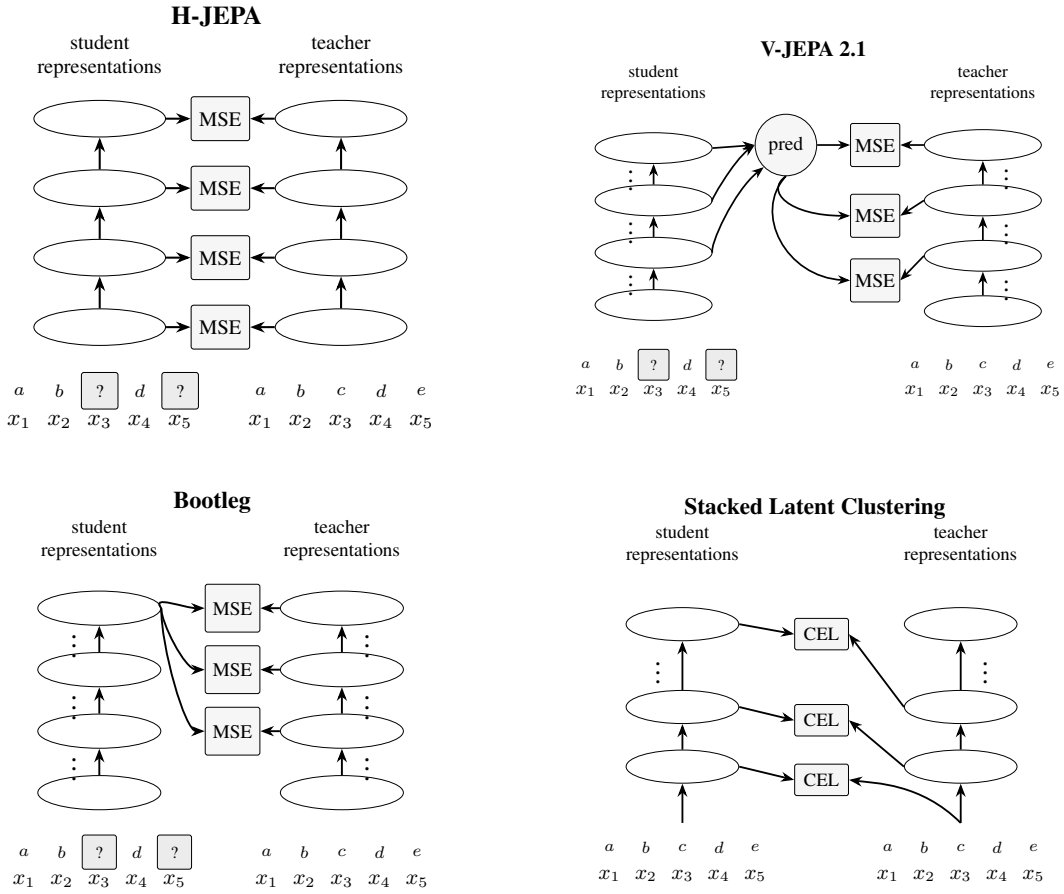


Figure 6: **Top left:** Structure of losses and targets for naïve H-JEPAs. **Top right:** Structure of losses and targets in V-JEPA 2.1. **Bottom left:** Structure of losses and targets for Bootleg. **Bottom right:** A schematic implementation of Stacked Latent Clustering, in which high-level representations are grounded in predicted lower-level representations.

Algorithm 1: Iterative Latent Clustering (ILC) — see Figure 7 for a graphical representation.

Input: P samples $x^{(1)}, \dots, x^{(P)}$, RHM parameters L, s, v, and a clustering module $\mathsf{Cluster}_v$.
Output: estimated non-root hierarchy $\widehat{h}^{(1)}, \dots, \widehat{h}^{(L-1)}$.

Initialize: $\widehat{h}_i^{(0)} = x_i$.

for $\ell = 0, 1, \dots, L-2$ **do**

Form all level-ℓ tuples $\widehat{T}_u^{(\ell)}$.

Estimate the support of grammatical tuples $\widehat{\mathcal{S}}_\ell$ by the observed tuple set.

For every $\nu \in \widehat{\mathcal{S}}_\ell$, count its incidence as $N(\nu) = \sum_{p=1}^{P} \mathbf{1}\{\widehat{T}_\ell^{(p)} = \nu\}$ and compute its empirical cousin context vector

$$\widehat{\phi}_\ell(\nu) := \frac{1}{N(\nu)} \sum_{p=1}^{P} e_{\widehat{Z}_\ell^{(p)}} \mathbf{1}\{\widehat{T}_\ell^{(p)} = \nu\},$$

where $\widehat{T}_\ell^{(p)}$ is a fixed tuple at level ℓ in sample p, and $\widehat{Z}_\ell^{(p)}$ is a fixed level-ℓ element in a cousin tuple (i.e. sharing the $\ell+2$ grandparent with $\widehat{T}_\ell^{(p)}$).

} Predictor, p

Cluster the context vectors: $\widehat{\mathcal{S}}_{\ell,1}, \dots, \widehat{\mathcal{S}}_{\ell,v} = \mathsf{Cluster}_v\left(\{\widehat{\phi}_\ell(\nu) : \nu \in \widehat{\mathcal{S}}_\ell\}\right)$.

Define the next-level latent label by the cluster identity:

$$\widehat{h}_u^{(\ell+1)} = a \qquad \text{if} \qquad \widehat{T}_u^{(\ell)} \in \widehat{\mathcal{S}}_{\ell,a}.$$

} Clusterer, C

B Proof of Iterative Latent Clustering recovery

Here, we provide formal definitions for the random hierarchy model and of the ILC algorithm. We then prove the recovery guarantee for ILC, which is stated informally in Section 3.

B.1 The random hierarchy model

The RHM [32] is a probabilistic context-free grammar on a fixed regular tree. The tree has depth L, branching factor s, and vocabularies $\mathcal{V}_0, \mathcal{V}_1, \dots, \mathcal{V}_L$, all of size v. Level 0 is visible and levels $1, \dots, L$ are latent. At level ℓ, there are $s^{L-\ell}$ variables $h_1^{(\ell)}, \dots, h_{s^{L-\ell}}^{(\ell)}$, with $h_u^{(\ell)} \in \mathcal{V}_\ell$. The visible sequence is $x_i = h_i^{(0)}$, $i = 1, \dots, s^L$.

For each level $\ell = 0, \dots, L-1$, a grammar instance is sampled by choosing vm distinct tuples from $\mathcal{V}_\ell^s$ uniformly without replacement and partitioning them into v labeled groups of size m, one group $\mathcal{R}_{\ell,a}$ for each parent symbol $a \in \mathcal{V}_{\ell+1}$. A rule $\nu = (\nu_1, \dots, \nu_s) \in \mathcal{R}_{\ell,a}$ means that parent a may generate the tuple of children ν. The set of grammatical tuples at level ℓ is $\mathcal{S}_\ell := \bigsqcup_{a \in \mathcal{V}_{\ell+1}} \mathcal{R}_{\ell,a}$, so $|\mathcal{S}_\ell| = vm$. We write $f := m/v^{s-1}$ for the fraction of valid s-tuples, since vm out of the v^s possible tuples are grammatical. Because the rule map is injective, the grammar is unambiguous: each grammatical tuple has a unique parent. We denote it by $\mathrm{par}_\ell(\nu) = a \iff \nu \in \mathcal{R}_{\ell,a}$. Two grammatical tuples $\nu, \nu' \in \mathcal{S}_\ell$ are called *synonyms* if they have the same parent. The map par_ℓ is the map h_ℓ used in the main text. We write $\Phi_\ell(a)$ for the parent context vector associated with parent value $a \in \mathcal{V}_{\ell+1}$, so that $\phi_\ell(\nu) = \Phi_\ell(\mathrm{par}_\ell(\nu))$.

Generation proceeds top-down. First $h_1^{(L)} \sim \mathrm{Unif}(\mathcal{V}_L)$. Then, recursively, if $h_u^{(\ell+1)} = a$, the child tuple $T_u^{(\ell)} := \left(h_{(u-1)s+1}^{(\ell)}, \dots, h_{us}^{(\ell)}\right)$ is sampled uniformly from $\mathcal{R}_{\ell,a}$.

B.2 Explicit clustering algorithm

Below, we present the formal ILC clustering algorithm, initially presented schematically in fig. 2.

B.3 Proof of recovery guarantee

This appendix proves the recovery guarantee for Iterative Latent Clustering. Throughout the proof, probabilities are taken over fresh samples from a fixed grammar instance, unless an expectation over grammars is explicitly denoted by $\mathbb{E}_{\mathcal{G}}$.

B.3.1 Formal assumptions

We use the following two assumptions. The first is a typical-event assumption on the random grammar. The second is an algorithmic stability assumption on the clustering module.

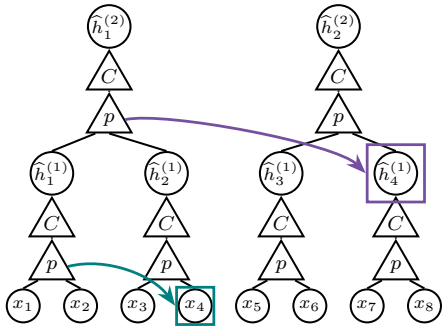


Figure 7: **Graphical representation of Algorithm 1 for $L = 3$.** Predictor p implements steps 3-5, and clusterer C constructs next-level latents. Highlighted prediction targets are as in Figure 1d.

Assumption 1 (Balanced and separated grammar)**.** *There exist constants $c_{\mathrm{bal}}, C_{\mathrm{bal}}, c_{\mathrm{sep}} > 0$, independent of v, m, L, such that for every level $\ell = 0, \dots, L-2$, every grammatical tuple $\nu \in \mathcal{S}_\ell$ satisfies*

$$\frac{c_{\mathrm{bal}}}{vm} \le \mathbb{P}[T = \nu] \le \frac{C_{\mathrm{bal}}}{vm},$$

and the parent context vectors satisfy

$$\min_{a \neq b} \|\Phi_\ell(a) - \Phi_\ell(b)\|_2 \ge c_{\mathrm{sep}} \frac{\sqrt{1-f}}{m}.$$

Assumption 2 (Stable clustering module)**.** *Let $\{z_\nu : \nu \in \mathcal{S}_\ell\} \subset \mathbb{R}^v$ be points with true centers $\{\Phi_\ell(a) : a \in \mathcal{V}_{\ell+1}\}$. Suppose that*

$$\|z_\nu - \Phi_\ell(\mathrm{par}_\ell(\nu))\|_2 \le \varepsilon$$

for all $\nu \in \mathcal{S}_\ell$, and that

$$\|\Phi_\ell(a) - \Phi_\ell(b)\|_2 \ge \Delta$$

for all $a \neq b$. If $\varepsilon \le \Delta/8$, then $\mathsf{Cluster}_v(\{z_\nu\}_{\nu \in \mathcal{S}_\ell})$ returns the true partition $\mathcal{S}_\ell = \bigsqcup_{a \in \mathcal{V}_{\ell+1}} \mathcal{S}_{\ell,a}$, up to a permutation of labels.

Assumption 1 is a high-probability random-grammar event. Balancedness follows from asymptotic uniformity of single-symbol marginals in the limit $v \to \infty$ at fixed $f \in (0,1)$, see Lemma C.1 of [54]. Indeed, if $\nu \in \mathcal{S}_{\ell,a}$, then ν is generated with probability $1/m$ conditional on its parent a. Therefore

$$\mathbb{P}[T^{(\ell)} = \nu] = \frac{1}{m}\mathbb{P}[h^{(\ell+1)} = a] \sim \frac{1}{vm}.$$

The separation scale is the latent-latent correlation scale. Indeed, writing

$$\psi_\ell(\nu) := \phi_\ell(\nu) - \mathbb{E}[e_Y],$$

the connected covariance column is

$$C_\ell(:, \nu) = \mathbb{P}[T = \nu]\psi_\ell(\nu).$$

For $\ell = 0$, T is a grammatical visible s-tuple and Y is a visible token in a sibling branch, with lowest common ancestor two levels above the leaves. This is precisely the token–tuple covariance computed in Cagnetta and Wyart [33]. In our notation, their calculation gives, for a fixed visible value y and a grammatical tuple ν,

$$\mathbb{E}_{\mathcal{G}}\left[C_0(y, \nu)^2 \,\middle|\, \nu \in \mathcal{S}_0\right] \asymp \frac{1-f}{v^3 m^4}.$$

Summing over the v possible values of Y yields the squared column norm

$$\mathbb{E}_{\mathcal{G}}\left[\|C_0(:, \nu)\|_2^2 \,\middle|\, \nu \in \mathcal{S}_0\right] \asymp v \cdot \frac{1-f}{v^3 m^4} = \frac{1-f}{v^2 m^4}.$$

The same estimate holds at every latent level by self-similarity of the RHM. Indeed, fix $\ell > 0$ and consider the sequence of level-ℓ variables

$$H^{(\ell)} := \big(h_1^{(\ell)}, \dots, h_{s^{L-\ell}}^{(\ell)}\big).$$

Marginally, $H^{(\ell)}$ is generated by an RHM of depth $L - \ell$ with the same local parameters (s, v, m), using the production rules at levels $\ell, \ell + 1, \dots, L - 1$. In this truncated RHM, the level-ℓ variables play the role of visible tokens. Our statistic $C_\ell(:, \nu)$ is therefore exactly the level-0 token–tuple covariance of this truncated model: T is a terminal s-tuple, Y is a terminal child in a sibling branch, and their lowest common ancestor is again two levels above the terminal scale. Since the random rule ensemble has the same law at every level, the same covariance calculation gives

$$\mathbb{E}_{\mathcal{G}}\big[\|C_\ell(:, \nu)\|_2^2 \,\big|\, \nu \in \mathcal{S}_\ell\big] \asymp \frac{1 - f}{v^2 m^4},$$

uniformly for $\ell = 0, \dots, L - 2$. Since

$$C_\ell(:, \nu) = \mathbb{P}[T = \nu]\psi_\ell(\nu)$$

and, on the balancedness event, $\mathbb{P}[T = \nu] \asymp 1/(vm)$, this implies

$$\mathbb{E}_{\mathcal{G}}\big[\|\psi_\ell(\nu)\|_2^2 \,\big|\, \nu \in \mathcal{S}_\ell\big] \asymp \frac{(1 - f)/(v^2 m^4)}{1/(v^2 m^2)} = \frac{1 - f}{m^2}.$$

Thus $\sqrt{1 - f}/m$ is the natural scale of the centered context-vector signal.

To turn this into a separation statement, we need to assume that different parent context vectors have generic relative positions, as expected for independent random production rules. In particular, under a random-direction heuristic, overlaps are smaller than squared norms by a factor $v^{-1/2}$, so pairwise distances have the same scale as the individual norms.

B.3.2 Theorem and proof

Theorem 2 (Recovery of the non-root hierarchy)**.** *Assume the balancedness and separation conditions above, and assume the clustering is stable under perturbations as described. Then there exists a constant $C > 0$, depending only on the constants in those assumptions and on s, such that the following holds. If*

$$P \geq C \left[vm \log \frac{Lvm}{\delta} + \frac{vm^3}{1 - f} \log \frac{Lvm}{\delta} \right], \tag{3}$$

then the iterative latent-clustering algorithm recovers $h^{(1)}, h^{(2)}, \dots, h^{(L-1)}$ and all production-rule classes below the root with probability at least $1 - \delta$, up to independent permutations of the latent labels at each level.

In particular, up to logarithmic factors, $P_{\mathrm{ILC}} = \widetilde{O}\left((1 - f)^{-1} vm^3\right)$.

Synonyms have identical context vectors. We first record the property that makes clustering possible.

Lemma 1 (Synonyms have identical context vectors)**.** *If $\nu, \nu' \in \mathcal{S}_\ell$ satisfy $\mathrm{par}_\ell(\nu) = \mathrm{par}_\ell(\nu')$, then $\phi_\ell(\nu) = \phi_\ell(\nu')$.*

Proof. Let A be the level-$(\ell + 1)$ parent of the branch containing T, let B be the level-$(\ell + 1)$ parent of the sibling branch containing Y, and let G be their shared level-$(\ell + 2)$ parent. The local graphical structure is $G \to (A, B)$, $A \to T$, and $B \to Y$. By unambiguity, observing $T = \nu$ determines $A = \mathrm{par}_\ell(\nu)$. By the context-free Markov structure, conditioned on A, the rest of the tree is independent of which production rule was used to generate T. Hence, if $\nu \in \mathcal{S}_{\ell,a}$,

$$\mathbb{P}[Y = y \mid T = \nu] = \mathbb{P}[Y = y \mid A = a].$$

The right-hand side depends on ν only through its parent a. Therefore all tuples in the same synonym class have the same context vector. □

Concentration of empirical context vectors. Fix a level ℓ and a grammatical tuple $\nu \in \mathcal{S}_\ell$. Let N_ν be the number of samples in which the chosen cousin tuple equals ν, and let $Y_1, \ldots, Y_{N_\nu}$ be the corresponding target values. Conditional on $T = \nu$, these are independent draws from the conditional law of Y, and the empirical context vector

$$\widehat{\phi}_\ell(\nu) = \frac{1}{N_\nu} \sum_{q=1}^{N_\nu} e_{Y_q}$$

is the empirical mean of N_ν independent one-hot vectors with mean $\phi_\ell(\nu) = \mathbb{E}[e_Y \mid T = \nu]$.

Lemma 2 (Context-vector concentration)**.** *There is a constant $C_0 > 0$ such that, for any $\eta \in (0, e^{-1}]$,*

$$\mathbb{P}\left[\|\widehat{\phi}_\ell(\nu) - \phi_\ell(\nu)\|_2 > C_0 \sqrt{\frac{\log(1/\eta)}{N_\nu}} \,\middle|\, N_\nu \right] \le \eta.$$

Proof. Condition on $N_\nu = n \ge 1$. Define

$$U_q := e_{Y_q} - \phi_\ell(\nu), \qquad \bar{U} := \frac{1}{n} \sum_{q=1}^{n} U_q.$$

Then $\mathbb{E}[U_q] = 0$ and

$$\widehat{\phi}_\ell(\nu) - \phi_\ell(\nu) = \bar{U}.$$

Since the U_q's are independent and mean zero, the cross terms vanish, and

$$\begin{aligned} \mathbb{E}\|\bar{U}\|_2^2 &= \mathbb{E}\left\| \frac{1}{n} \sum_{q=1}^{n} U_q \right\|_2^2 \\ &= \frac{1}{n^2} \sum_{q=1}^{n} \mathbb{E}\|U_q\|_2^2 \\ &\le \frac{1}{n}, \end{aligned}$$

where we used

$$\mathbb{E}\|e_Y - \phi_\ell(\nu)\|_2^2 = 1 - \|\phi_\ell(\nu)\|_2^2 \le 1.$$

Thus,

$$\mathbb{E}\|\bar{U}\|_2 \le n^{-1/2}.$$

Now consider the function

$$F(Y_1, \ldots, Y_n) := \|\bar{U}\|_2.$$

Changing one observation changes $\bar{U}$ by at most $\sqrt{2}/n$, and hence changes F by at most $\sqrt{2}/n$. By McDiarmid's bounded-differences inequality,

$$\mathbb{P}[F - \mathbb{E}F > t] \le \exp(-nt^2).$$

Taking

$$t = \sqrt{\frac{\log(1/\eta)}{n}},$$

we get, with probability at least $1 - \eta$,

$$F \le \frac{1}{\sqrt{n}} + \sqrt{\frac{\log(1/\eta)}{n}}.$$

Since $\eta \le e^{-1}$, we have $\log(1/\eta) \ge 1$, so the first term is absorbed into the second. Thus, for a constant C_0,

$$\|\widehat{\phi}_\ell(\nu) - \phi_\ell(\nu)\|_2 = \|\bar{U}\|_2 \le C_0 \sqrt{\frac{\log(1/\eta)}{n}}$$

with probability at least $1 - \eta$. □

Proof of the theorem. At each level ℓ, let T_ℓ be the level-ℓ tuple to be clustered and Z_ℓ the level-ℓ target in the sibling branch. For sample p, denote the corresponding true variables by $T_\ell^{(p)}$ and $Z_\ell^{(p)}$

We now prove Theorem 2.

Proof of Theorem 2. We first define a good event using the true latent variables. For every level $\ell = 0, \dots, L-2$ and every grammatical tuple $\nu \in \mathcal{S}_\ell$, let

$$N_{\ell,\nu} := \sum_{p=1}^{P} \mathbf{1}\{T_\ell^{(p)} = \nu\}$$

be the number of samples in which the true level-ℓ cousin tuple equals ν, and let

$$\widetilde{\phi}_\ell(\nu) := \frac{1}{N_{\ell,\nu}} \sum_{p=1}^{P} e_{Z_\ell^{(p)}} \mathbf{1}\{T_\ell^{(p)} = \nu\}$$

be the corresponding oracle empirical context vector.

By balancedness,

$$\mathbb{P}[T_\ell = \nu] \geq \frac{c_{\mathrm{bal}}}{vm}.$$

Thus $N_{\ell,\nu}$ is binomial with mean at least $c_{\mathrm{bal}} P/(vm)$. A Chernoff bound and a union bound over all $(L-1)vm$ pairs (ℓ, ν) imply that, if

$$P \gtrsim vm \log \frac{Lvm}{\delta},$$

then, with probability at least $1 - \delta/2$,

$$N_{\ell,\nu} \gtrsim \frac{P}{vm}$$

simultaneously for all ℓ and ν.

On this count event, Lemma 2, applied with $\eta = \delta/(2Lvm)$, and another union bound give

$$\max_{\ell,\nu} \|\widetilde{\phi}_\ell(\nu) - \phi_\ell(\nu)\|_2 \lesssim \sqrt{\frac{vm}{P} \log \frac{Lvm}{\delta}}$$

with probability at least $1 - \delta/2$. Therefore, if

$$P \gtrsim \frac{vm^3}{1-f} \log \frac{Lvm}{\delta},$$

then, with probability at least $1 - \delta$,

$$\|\widetilde{\phi}_\ell(\nu) - \phi_\ell(\nu)\|_2 \leq \frac{1}{8} c_{\mathrm{sep}} \frac{\sqrt{1-f}}{m}$$

for all ℓ and all $\nu \in \mathcal{S}_\ell$. Call this simultaneous event $\mathcal{E}$.

We now show that on $\mathcal{E}$, the algorithm succeeds deterministically. At level 0, the variables are visible tokens, so $\widehat{h}^{(0)} = h^{(0)}$. Suppose inductively that level ℓ has been recovered exactly, up to a permutation of labels. Then the recovered level-ℓ tuples are the true level-ℓ tuples, up to relabeling. Consequently, the empirical context vectors computed by the algorithm are exactly the oracle vectors $\widetilde{\phi}_\ell(\nu)$, up to permutations of tuple labels and output coordinates. These permutations preserve distances.

Thus, on $\mathcal{E}$, every empirical context vector lies within

$$\frac{1}{8} c_{\mathrm{sep}} \frac{\sqrt{1-f}}{m}$$

of its true parent center. By separation,

$$\min_{a \neq b} \|\Phi_\ell(a) - \Phi_\ell(b)\|_2 \geq c_{\mathrm{sep}} \frac{\sqrt{1-f}}{m}.$$

The stable clustering assumption therefore implies that the clustering step recovers the true synonym partition

$$\mathcal{S}_\ell = \bigsqcup_{a \in \mathcal{V}_{\ell+1}} \mathcal{S}_{\ell,a}$$

up to a permutation of labels. Assigning each tuple to its cluster recovers level $\ell + 1$, again up to a permutation of labels.

Iterating this deterministic induction from $\ell = 0$ to $L-2$ recovers $h^{(1)}, \dots, h^{(L-1)}$ and all production-rule classes below the root. Since $\mathcal{E}$ holds with probability at least $1 - \delta$, the theorem follows.

The required sample size is

$$P \gtrsim vm \log \frac{Lvm}{\delta} + \frac{vm^3}{1-f} \log \frac{Lvm}{\delta}.$$

Thus, up to logarithmic factors, $P \gtrsim vm^3/(1-f)$ suffices for recovery. □

C Stacked latent-clustering details and additional results

This appendix gives implementation details and additional diagnostics for the SLC network introduced in Section 4. We use the RHM notation of Section 2: true latent variables are denoted by $h_i^{(\ell)}$, while learned SLC tokens are denoted by $\widehat{h}_i^{(\ell)}$.

C.1 SLC architecture details

We encode RHM input strings of size s^L into one-hot feature vectors of dimension $\mathbb{R}^{s^L \times d_h}$, where $d_h \gg v$ is the dimension into which we tokenize learned latents. This initializes the level-0 tokens as $\widehat{h}_i^{(0)} = x_i$. We use a single dimension d_h for the cluster codebook and for the token vocabulary passed between SLC modules: each clusterer assigns labels in $\{1, \dots, d_h\}$, these labels become the tokens for the next module, and predictors therefore output categorical distributions over d_h token identities.

A SLC model for an RHM of depth L is comprised of $L-1$ SLC modules. Module ℓ reduces the learned level-ℓ sequence to an encoding of the level-$(\ell+1)$ latents. For a module at level ℓ, the tensor shapes are

$$\begin{aligned}
\widehat{h}^{(\ell)} &\in \mathbb{R}^{s^{L-\ell} \times d_h}, \\
x_u^{(\ell)} := \big(\widehat{h}^{(\ell)}_{(u-1)s+1}, \dots, \widehat{h}^{(\ell)}_{us}\big) &\in \mathbb{R}^{s \times d_h}, \\
\widehat{\phi}_u^{(\ell)} = \mathrm{Pred}^{(\ell)}(x_u^{(\ell)}) &\in \mathbb{R}^{s \times (s-1) \times s \times d_h}, \\
\mathrm{vec}\, \widehat{\phi}_u^{(\ell)} &\in \mathbb{R}^{s^2(s-1)d_h}, \\
q_u^{(\ell+1)} = \mathrm{Clust}^{(\ell)}(\mathrm{vec}\, \widehat{\phi}_u^{(\ell)}) &\in \Delta^{d_h - 1}, \\
\widehat{h}^{(\ell+1)} &\in \mathbb{R}^{s^{L-\ell-1} \times d_h}.
\end{aligned}$$

Here $u = 1, \dots, s^{L-\ell-1}$ indexes level-ℓ patches, and d_{h_2} is the hidden width of the CNN layers below. The dimensions of $\widehat{\phi}_u^{(\ell)}$ enumerate, respectively, the possible position of the input patch within its grandparent, the target cousin patch, the target token within that patch, and the d_h token identity. This task is easiest for $d_h = v$, as clustering is architecturally enforced, and is hardest for $d_h \geq mv$ (when synonyms can be uniquely encoded without clustering). Unless otherwise noted, we will always use $d_h = mv$.

A SLC module is composed of a predictor and a clusterer. This arrangement is depicted in Figure 8.

Predictor. The predictor operates on an s-tuple patch and predicts all $s(s-1)$ tokens with the same grandparent (cf. Figure 9 for a visualization). Using all cousin targets increases the available signal and improves prefactors, although it does not change the asymptotic sample-complexity scaling. Because the same patch can occupy any of the s positions under its grandparent and we use weight sharing across positions, the predictor outputs $s^2(s-1)$ distributions; for each observed patch, only

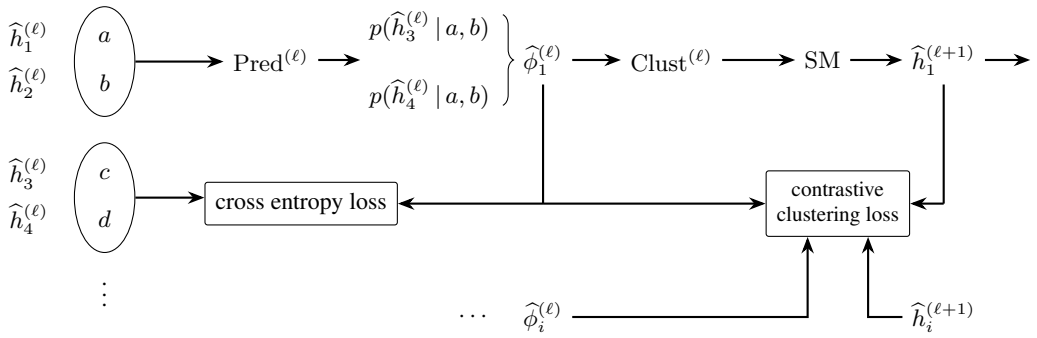


Figure 8: **Schematic of a single SLC module, with the layer-wise prediction and clustering losses.**

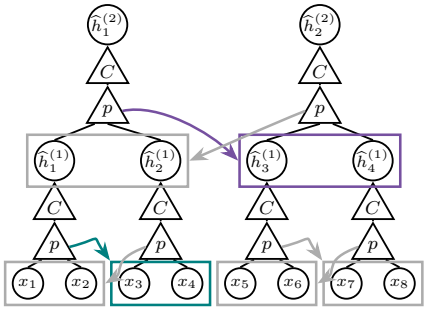


Figure 9: **Multiple predictions improve signal**. We train the predictor and clusterer modules with weight sharing, and train cross-entropy prediction at all possible positions and cousin targets.

the $s(s-1)$ distributions matching its actual position enter the loss. The predictor is implemented in terms of convolutional layers, as

$$\widehat{\phi}^{(\ell)}(x) = \text{Pred}^{(\ell)}(x) = (\text{SM} \circ \text{CNN}_3 \circ A \circ \text{CNN}_2 \circ A \circ \text{CNN}_1)(x) \tag{4}$$

where $A = \text{ReLU} \circ \text{BN}$ denotes the activation function, $\text{CNN}_1 : \mathbb{R}^{s \times d_h} \to \mathbb{R}^{d_{h_2}}$ is a stride s 1d convolutional layer, while $\text{CNN}_2 : \mathbb{R}^{d_{h_2}} \to \mathbb{R}^{d_{h_2}}$ and $\text{CNN}_3 : \mathbb{R}^{d_{h_2}} \to \mathbb{R}^{s \times (s-1) \times s \times d_h}$ are both stride 1. The softmax operation SM acts on the last dimension so that $\sum_{a=1}^{d_h} \widehat{\phi}^{(\ell)}(x)_{r\rho ta} = 1$. This last dimension is therefore the marginal probability distribution over neighbouring tokens, conditioned on the input patch position r, target cousin-patch index ρ, and target token position t.

We train the predictor with a cross-entropy prediction loss. Fix a grandparent index $g \in \{1, \ldots, s^{L-\ell-2}\}$ and an input patch position $r \in \{1, \ldots, s\}$ within that grandparent, so that $u = (g-1)s + r$ and the input patch is $x_u^{(\ell)}$. For a target cousin patch position $r' \neq r$, let $\rho_r(r') \in \{1, \ldots, s-1\}$ denote the index of r' among the patch positions excluding r. The level-ℓ token at target position t inside target patch r' has absolute index

$$i(g, r', t) := ((g-1)s + r' - 1)s + t.$$

For this input patch, the prediction loss is

$$\mathcal{L}_{\text{pred}}(g, r) = -\sum_{r' \neq r} \sum_{t=1}^{s} \sum_{a=1}^{d_h} \widehat{h}^{(\ell)}_{i(g,r',t),a} \log\left(\widehat{\phi}^{(\ell)}(x_u^{(\ell)})_{r,\rho_r(r'),t,a}\right). \tag{5}$$

The predictor loss for module ℓ is the mean of this quantity over all grandparent indices g and input patch positions r.

Clusterer. The clusterer in turn assigns soft cluster labels to each patch's prediction vector – this amounts to assigning a learned level-$(\ell+1)$ latent label.

$$q^{(\ell+1)} = \text{Clust}^{(\ell)}(\widehat{\phi}^{(\ell)}) = (\text{SM} \circ \text{CNN}_5 \circ A \circ \text{CNN}_4)(\widehat{\phi}^{(\ell)}) \tag{6}$$

where $\widehat{\phi}^{(\ell)}$ denotes the flattened output of $\mathrm{Pred}^{(\ell)}$, $\mathrm{CNN}_4 : \mathbb{R}^{s^2(s-1)d_h} \to \mathbb{R}^{d_{h_2}}$ and $\mathrm{CNN}_5 : \mathbb{R}^{d_{h_2}} \to \mathbb{R}^{d_h}$ are stride-1 CNNs. The soft code $q^{(\ell+1)}$ is the learned token $\widehat{h}^{(\ell+1)}$ passed to the next module. *A priori* we do not specify that $d_h = v$, so the clusterer could choose to instead assign each synonym of a patch into its own cluster. Indeed, for finite data, differing but synonymous patches will imply different contexts due to finite sampling effects. We address this by using a contrastive clustering loss that penalizes assignments of sufficiently similar predictions into different clusters.

Let $q_i = \mathrm{Clust}^{(\ell)}(\widehat{\phi}_i^{(\ell)})$ denote the soft cluster assignment for prediction vector $\widehat{\phi}_i^{(\ell)}$, where the index i is taken to be over patch positions and batches, and we drop the $^{(\ell)}$ superscript for brevity. The predictions are then centred, so that $\widehat{\phi}_i^{(\ell)} \to \widehat{\phi}_i^{(\ell)} - \langle \widehat{\phi}_j^{(\ell)} \rangle_j$. The similarity S_{ij} between two predictions is the scaled cosine similarity

$$S_{ij} = \frac{1}{2}\left(1 + \frac{\widehat{\phi}_i^{(\ell)} \cdot \widehat{\phi}_j^{(\ell)}}{||\widehat{\phi}_i^{(\ell)}||_2 ||\widehat{\phi}_j^{(\ell)}||_2}\right) \tag{7}$$

of these centred predictions. The clustering loss is then

$$\mathcal{L}_{\mathrm{clust}} = \langle \mathcal{L}_{\mathrm{sim}}(q_i, q_j, S_{ij}) + \lambda_{\mathrm{sep}} \mathcal{L}_{\mathrm{sep}}(q_i, q_j, S_{ij}) \rangle_{ij} + \lambda_{\mathrm{sparsity}} \langle \mathcal{L}_{\mathrm{sparsity}}(q_i) \rangle_i \tag{8}$$

where the $\langle \cdot \rangle_{ij}$ indicates an average over patch positions and batches. The similarity loss component

$$\mathcal{L}_{\mathrm{sim}}(q_i, q_j, S_{ij}) = (q_i - q_j) \cdot (q_i - q_j)\, \mathrm{ReLU}(S_{ij} - S_m) \tag{9}$$

serves to penalize predictions that are sufficiently similar (more than the margin S_m) but assigned to different clusters (i.e. with non-zero Δq) – this loss ensures that clusters consist of similar predictions. Alone, of course, the model could simply cluster ALL predictions into a single cluster. Therefore, we include a separation loss

$$\mathcal{L}_{\mathrm{sep}}(q_i, q_j, S_{ij}) = q_i \cdot q_j (1 - S_{ij}) \tag{10}$$

which penalizes dissimilar (i.e. large $1 - S_{ij}$) predictions that are assigned into overlapping clusters (i.e. $q_i \cdot q_j > 0$). Finally, we introduce

$$\mathcal{L}_{\mathrm{sparsity}}(q_i) = -q_i \cdot q_i \tag{11}$$

which tends to maximize the L_2 norm of the cluster assignment and therefore the sparsity (since $||q_i||_1 = 1$ due to the softmax). Preliminary experiments demonstrated that this sparsity reward improves training stability and downstream interpretability.

A general caveat of employing contrastive losses is that memory costs can explode. Employed naïvely, the first SLC module would compute $s^{2(L-1)}$ comparisons, each of which involves a potentially costly dot-product of a $s^2(s-1)d_h$ vector. In practice for each batch, we select a random subset of N_{compare} patch positions and batches on which to compute the clustering loss for memory constraints.

Training protocol and stabilizers. To keep the prediction problem at depth ℓ comparable to the prediction problem at depth 0, we tokenize clusterer outputs by means of a softmax operation, $\mathrm{SM}(X, T) = \exp(X/T)/\sum_i \exp(X_i/T)$. For $T = 0$, this is hard label assignment. Preliminary experiments found that $T < 1$ can improve convergence, but here we report results for $T = 1$.

The clustering codebook dimension d_h is another important stabilizer. If the space of possible labels is large, each input can be assigned a unique label; this is a failure to cluster, because synonyms of a latent are not assigned the same symbol in the next layer. The input vocabulary for layer $\ell + 1$ then increases by a factor m, making the prediction task for layer $\ell + 1$ intractable. This problem is exacerbated by noisy or finite sampling, where synonymous inputs do not produce identical predictions. This can be resolved by an architectural bottleneck, i.e. setting $d_h = v$, as is done for the ILC. For most natural data, this dimension is unknown *a priori*; to simulate these conditions we set the label dimension to $d_h > mv$ (the hardest case) and instead use the contrastive loss to set the scale of attraction between similar predictions.

To mitigate representation collapse, we use a teacher-student framework, in which latent targets are obtained from a teacher network whose weights are simply the exponentially lagged weights of the student. After each step of gradient descent, the teacher weights $W^{(T)}$ are updated towards the student weights $W^{(S)}$ with $W^{(T)} \leftarrow (1 - \alpha_{\mathrm{ema}})W^{(T)} + \alpha_{\mathrm{ema}} W^{(S)}$, where $1/\alpha_{\mathrm{ema}}$ sets the exponential moving average timescale. The prediction loss therefore uses teacher representations to

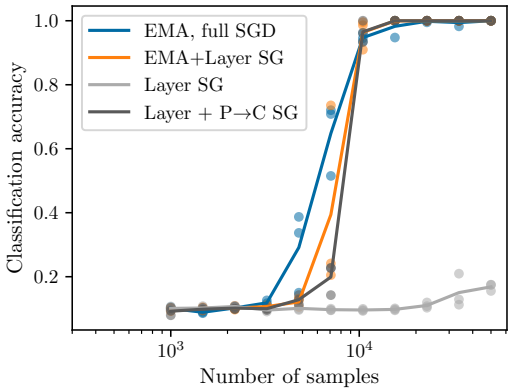


Figure 10: **Local learning suffices to learn the RHM.** We compare baseline training (blue) to increasingly local learning rules. "Layer SG" in legend indicates stopgradients between SLC layers, "$P \to C$ SG" indicates stopgradient between the predictor and clusterer submodules. EMA in legend indicates exponential-moving-average teacher is used as a target. RHM parameters are: $L = 4, v = 10, m = 10, s = 3$. Solid lines are average over three independent realizations.

obtain targets $x_{:}$, which are evaluated against student predictions. The clustering loss meanwhile, computes the similarity S_{ij} using the teacher's predictions and subsequently uses the student's clustering assignments q_i, q_j. As an alternative to the teacher-student framework, we highlight in Section C.2 that collapse can also be overcome by introducing stop-gradients between SLC modules, effectively making learning entirely local, reminiscent to predictive coding. Finally, we perform Jacobian descent, computing the gradient for each loss independently, and aggregate the gradients by means of the UPGrad algorithm. This selects an update in the dual-cone of the gradients to avoid conflicting updates [64], which we found in preliminary experiments reduced the need for careful hyperparameter selection.

Unless otherwise specified, SLC models are trained using AdamW and Jacobian descent (via TorchJD and UPGrad [64]) and following hyperparameters: $\alpha_{\text{ema}} = 0.015$, $\text{lr} = 3 \times 10^{-3}$, $\text{wd} = 10^{-3}$, $d_{h_2} = 150$, batch size 32, clustering dimension $d_h = 128$, $S_m = 0.8$, $\lambda_{\text{sep}} = 0.5$, $N_{\text{compare}} = 300$, and $\lambda_{\text{spars}} = 10^{-2}$. These were optimized by hyperparameter sweeps conducted at $L = 5$ for $v = 16$, $m = 8$, $s = 3$ using Optuna's Tree-Parzen-Estimator sampler and Hyperband pruner [65].

We use a validation set of 2000 samples to evaluate the model throughout training. When training classifiers on the final SLC representation, we use early stopping to select the best pretraining checkpoint and classification checkpoint based on this validation set performance, but report test set performance on a held-out test set of 2000 samples. In Figure 13 the test sets are smaller, with 1000 samples.

C.2 Training with local rules

In Figure 10, we ablate the global training machinery while also testing increasingly biologically plausible training conditions. We first prevent gradient flow between SLC layers (denoted "Layer SG" for layerwise stop-gradient), then disable the exponential moving average (EMA) teacher network, and finally disable gradient flow between the prediction and clustering submodules. These changes remove the long-range backpropagating error signal and the biologically implausible time-delayed duplicate teacher network. When disabling EMA but allowing gradients to flow from the clustering loss through to the predictor, we observe representation collapse. We credit this to the clustering loss overpowering the prediction loss: trivial predictions are easy to cluster perfectly at the expense of prediction accuracy. Preventing gradient flow between prediction and clustering networks prevents this failure mode, showing that SLC can still learn when updates are constrained to local signals.

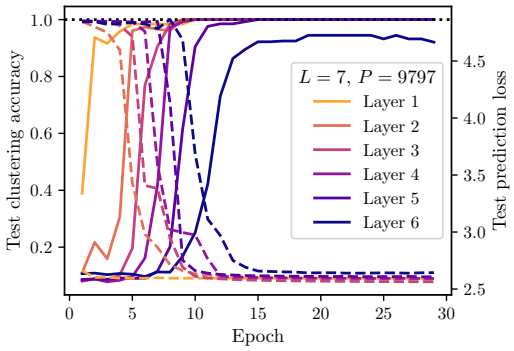


Figure 11: **Training proceeds layer-wise.** As each layer's clustering improves, the subsequent layer's prediction loss decreases, enabling that subsequent layer to then cluster.

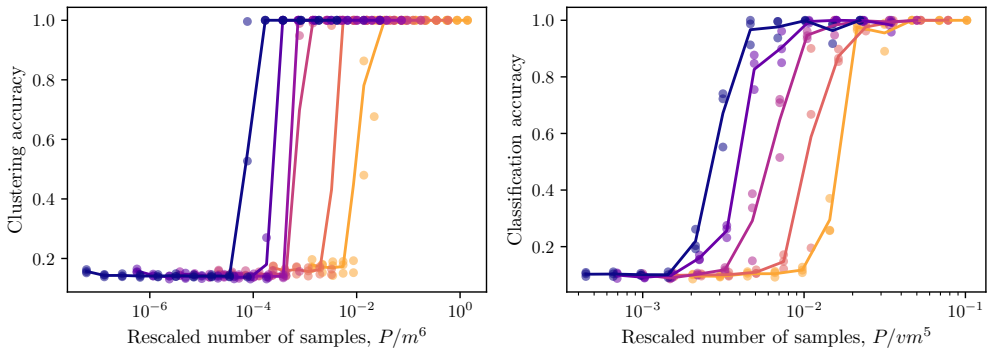


Figure 12: **Attempted curve collapse with token based SSL scaling.** Data are as in Figure 3, but with the m^{L+1} scaling that holds for token level SSL. The collapse of accuracy vs. training sample is significantly degraded.

C.3 Training dynamics

In Figure 11 we report the layer-wise accuracy and prediction loss of the different SLC modules. We evaluate the accuracy of module ℓ by taking the learned hard labels $\widehat{h}_i^{(\ell+1)}$ (with the $T = 0$ hard label assignment) and comparing them with the ground-truth RHM latents $h_i^{(\ell+1)}$. We identify an optimal mapping between the assigned labellings and the true latents with the Kuhn-Munkres "Hungarian" algorithm, and report the accuracy attained with such a mapping.

C.4 Independence of sample complexity with L

In Figure 12 we check that the sample complexity scaling for ILC and SLC better supports m^3 rather than m^{L+1}. Furthermore, in Figure 13 we validate the sample complexity $P \sim m^3$ for systems with varying depth, up to $L = 7$.

D Further results on data2vec

All data2vec runs use the hyperparameters of Table 1 unless stated otherwise.

Figure 14–**left** shows the synonym clustering score for encoder layers to latents at different depths. At the root level $\ell = L = 4$ the score stays ≈ 0 in every layer, as expected since there is no signal from which the network can learn the root latent. For latents at level $\ell < L$, we find clustering throughout the middle layers of the network. This indicates that clustering occurs throughout the early layers of data2vec. Furthermore, the fact that for latent ℓ encoder layer $\ell - 1$ is the first to exhibit clustering of the synonyms indicates that latents are constructed layer-by-layer. Deeper layers exhibit stronger

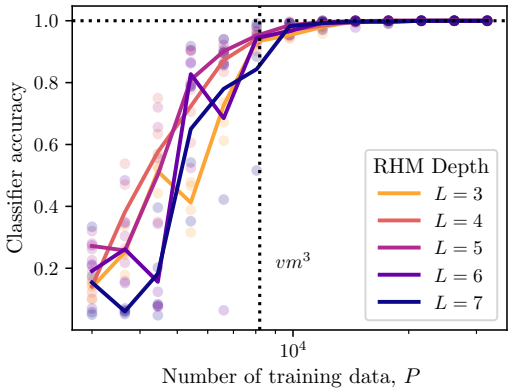


Figure 13: **The SLC model exhibits a depth-independent sample complexity.** Accuracy of linear probes trained to classify the root latent $h_1^{(L)}$ from the final SLC representation. RHM parameters are $s = 3$, $m = 8$, and $v = 16$, with depth L indicated by the legend. The pre-trained models were trained for 30 epochs, and the classifiers for the same duration and on the same dataset. Solid lines are an average over three independent RHM instantiations.

clustering of low-level latents, up to layer 2. This inversion is reminiscent of an encoder-decoder architecture, and mimics what is observed in U-Net architectures [37].

Figure 14–**right** tests the presence of linear traces of latents in the teacher target for different m and ℓ. This data collapses with vm^3, confirming that the transition to all latents being linearly encoded in the teacher occurs at the predicted vm^3 scaling.

Offline setting. In Figure 15 we test the scaling collapse for data2vec instances trained in the offline setting, thereby decoupling the number of gradient descent steps from the number of original observed samples. We find vm^3 provides a good collapse.

Squared loss. Our analysis idealizes the smooth-L1 loss of data2vec as a squared loss. With $\beta = 4$, the smooth-L1 loss equals $r^2/(2\beta) = r^2/8$ for each per-coordinate residual $|r| < \beta$ and grows only linearly beyond. To check that the linear tails play no role, we repeat the online runs of Figure 4 with the pure squared loss $\frac{1}{8}\sum_k r_k^2$, keeping everything else fixed. Figure 16–**left** shows that the curves again collapse with vm^3 and that $P^* \propto m^3$, as with the smooth-L1 loss.

Token-level baseline. We train the same transformer on the same online RHM data ($v = 16$, $s = 2$, $L = 4$, $m = 3, \ldots, 7$, one seed) with a masked-language-modeling (MLM) objective: single tokens are masked independently with probability 0.15, and a linear vocabulary head is trained with cross-entropy on the masked positions only. Figure 16–**right** shows that the root-classification curves collapse when P is rescaled by $vm^{L+1} = vm^5$, and that P^* follows m^{L+1} rather than m^3, as predicted for token-level objectives.

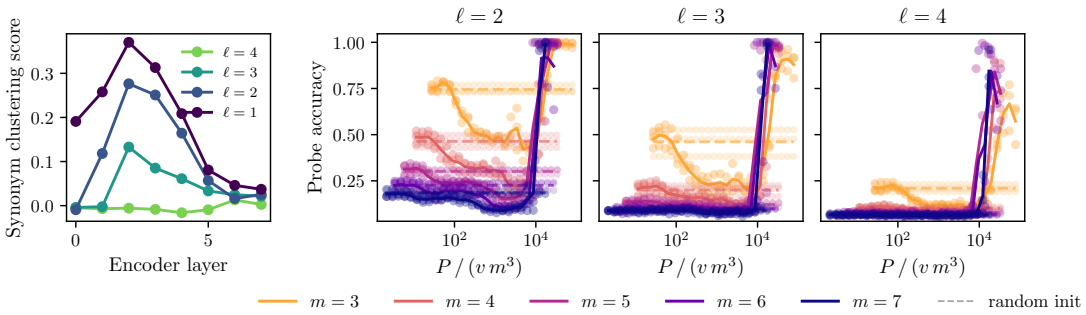


Figure 14: **Left:** post-training synonym clustering score per encoder layer and RHM level ℓ ($m = 6$). Values above 0 indicate that synonymous level-$(\ell - 1)$ tuples are clustered closer than non-synonymous ones. **Right:** linear-probe accuracy for the latent $h^{(\ell)}$ from the teacher target, vs. rescaled sample size $P/(vm^3)$, for $\ell = 2, 3, 4$. Curves collapse with vm^3 for every ℓ.

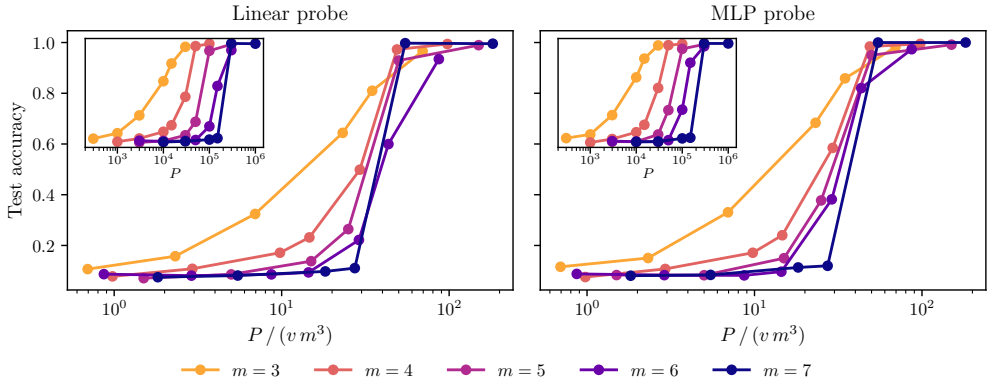


Figure 15: **Offline data2vec pretraining.** Linear and MLP probe test accuracy on the classification task as a function of the pretraining training-set size P. Each point is averaged over three seeds. Main axes: curves rescaled by vm^3. Insets: same data, raw P.

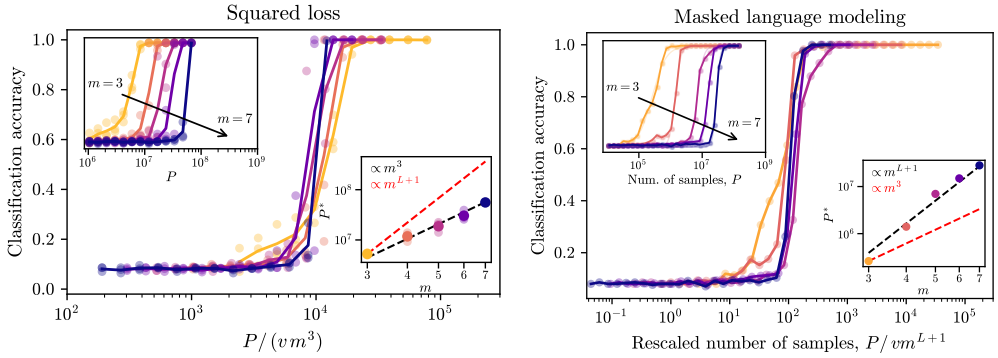


Figure 16: **MSE loss and MLM baseline.** Root-classification accuracy of probes on mean-pooled final-layer features (online, $v = 16$, $s = 2$, $L = 4$). **Left:** data2vec with a pure squared loss in place of smooth-L1. Curves collapse with vm^3, as with the smooth-L1 loss. **Right:** the same transformer trained with masked-language modeling. Curves collapse with the token-level scaling $vm^{L+1} = vm^5$. Notice that in the online regime P is tied to the number of gradient steps, so prefactors also reflect optimization and are not comparable across objectives. Insets: raw curves vs. P (top left) and the P^* at which accuracy reaches 50%, with m^3 and m^{L+1} guides (bottom right).

Table 1: Data2vec hyperparameters.

Group	**Hyperparameter**	**Value**
Encoder architecture	hidden dim d_{model}	2048
	attention heads n_h	32
	head dim	64
	encoder layers n_l	8
	FFN dim d_{ff}	$8192 = 4\,d_{\text{model}}$
	non-linearity	GELU
	dropout	0.1
	positional embeddings	learned, $T \times d_{\text{model}}$
	token embeddings	learned, $V_{\text{tok}} = v + 2$ ids
	LayerNorm placement	pre-LN
data2vec objective	mask probability	0.15
	mask span length	1
	top-K teacher layers averaged	$K = 4$
	per-layer LayerNorm of targets	yes
	global LayerNorm of targets	no
	regression head depth	1 linear layer
	loss	smooth L1, $\beta = 4$
	teacher EMA decay μ	0.99 (constant; no annealing)
Optimization	optimizer	AdamW
	(β_1, β_2)	$(0.9, 0.98)$
	weight decay	0.01 (excl. bias / LayerNorm)
	learning rate	$\eta = 10^{-4}$ (constant)
	LR schedule	none (no warmup, no decay)
	gradient clipping	global L2 norm ≤ 1.0
	batch size B	512
	training steps T_{steps}	$262\,144$
Probes	pooling	mean over positions
	linear probe	$\texttt{Linear}(d_{\text{model}}, v)$
	MLP probe	$d_{\text{model}} \to 2d_{\text{model}} \to v$, GELU, dropout 0.1
	probe optimizer	Adam, lr 10^{-3}
	probe steps per eval	2000 (each: linear, MLP)

E Compute budget

SLC experiments were conducted on individual H100 nodes. The SLC experiments in Figure 3 took approximately 10 H100 hours. The L scaling experiment reported in Figure 13 cost approximately 100 H100 hours, as the amount of computations scales linearly with input data size which consists of s^L tokens (and therefore exponentially in depth). Figure 10 cost approximately 3 H100 hours. We do not have an estimate for the Optuna hyperparameter sweeps or preliminary experiments.

The data2vec experiments cost approximately 1,000 H100 hours.

F Power-law fitting

To extract power-law fits of the form $P^*(m) \sim m^\alpha$, we perform monte-carlo sampling. For each value of m, we estimate the empirical average and standard-deviation of $P^*(m)$ from N simulations. In each monte-carlo, we then draw N iid samples from Gaussian distribution with that average and standard deviation, for each value of m and perform a least-squares fit in log-log space. The distribution of fitted α after 300 Monte-Carlo iterations provides the 2σ uncertainty ranges reported in the captions of Figures 3 and 4. The SLC fits are conducted over m=8-15, excluding m=7 as it has pre-asymptotic (small-m) factors that bias the fit; including that data results in a lower estimate $\alpha = 2.71 \pm 0.15$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main contributions of the work are presented in the abstract, and again summarized in the introduction. The scope is theoretical and test are on synthetic data, explicitly acknowledged in the intro and abstract.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations and paths forward at the end of the conclusions section, under the limitations header.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In our informal statements of theorems or arguments, we nonetheless provide an informal statement of assumptions (e.g. large v limit at fixed f). In the appendix, all assumptions are formally set out in relation to the proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We introduce the new SLC method, and as such include an extensive appendix detailing its architecture, as well as a discussion of the *motivation* for those architectural choices. Data2vec is widely available on public repos. We provide an explicit algorithim for realizing ILC, which could be implemented by any practitioner. Repositories already exist for efficiently sampling from the RHM.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a link (https://github.com/pcsl-epfl/learn-from-latents-not-tokens) to a public repository containing all code needed to reproduce the main experimental data, as well as summary statistics necessary to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide these details in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report averages over three independent realizations for most experiments, and indicate with transparent scatter plots directly the underlying data entering into those averages. For exponent inference, we detail the boostrap procedure used to estimate the error bars.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include a compute budget section in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics and do not find ourselves in conflict with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: Our work contributes to effectiveness machine learning and AI broadly. Whether this will have broad positive or negative societal impact remains to be seen, but we do not see specific societal harms deriving from our work. Therefore, we offer no discussion of this in the manuscript.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our trained models and work have no obvious safety risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [N/A]

Justification: We use no existing assets for which a license needs to be included.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We are not releasing new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use LLM coding assistants and agents for managing routine code implementation and rote data analysis. Central routines and model implementations were done by the authors. LLMs were used as assistive tools for checking proofs and formalizing mathematical statements. Theoretical results were authored and verified by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.