Università Commerciale Luigi Bocconi

Bachelor of Science in
Mathematical and Computing Sciences for Artificial Intelligence

Generative Diffusion for Hierarchical Models of Language

Bachelor of Science Thesis
Tommaso Aiello

Supervisor:
Prof. Saglietti Luca

Academic Year 2024-2025

Contents

1 Introduction: Formal Models of Language and the Random Hierarchy Model

Modern cognitive science traces the roots of human language to an innate neurobiological language faculty. Noam Chomsky's classic "poverty-of-the-stimulus" argument shows that exposure alone is too weak for children to infer the richly hierarchical rules of natural language; some skeletal grammar must be wired into the brain. Mathematics gives us the correct notation for that blueprint: every neurally realized production can be abstracted as a rewrite rule, placing the language faculty squarely in formal-language theory. The Chomsky hierarchy arranges such rule systems by expressive power, and at its sweet spot lie context-free grammars (CFGs) whose tree-shaped derivations line up nicely with both linguistic constituency and neural resource limits.

However, the importance of CFGs lies outside the grammar framework —they exemplify a broader principle of hierarchical, combinatorial structure that shows up in virtually every successful machine-learning domain. Images, texts, and even protein sequences rely on multi-layered feature reuse: characters form words, words form phrases, edges form shapes, shapes form objects. If a dataset lacked that hidden order, learning would require an exponential number of examples, making it practically unfeasible.

This study aims to investigate the representational capacity of the Transformer architecture within these structures, specifically within the context of diffusion processes. We first build on the Random Hierarchy Model (RHM)—a controlled CFG that lets generate synthetic corpora of arbitrary complexity, length, and structure. We are interested in how that complexity affects robustness to noise and the learnability of such structures.

To probe the question, we turn to denoising diffusion models, generative systems that invert a gradual corruption process to recover clean data. Empirically, diffusion models, such as DDPMs, appear to grasp global concepts while refining fine detail, mirroring the top-down constraints of grammatical trees. We implement a Transformer denoiser as an oracle, responsible for learning how to correctly denoise inputs depending on their corruption timestep.

In short, Chomsky's formal grammar and modern diffusion networks converge on the same insight: hierarchical compositional is the secret sauce that makes both human language and high-dimensional learning tractable.

2 Theoretical Background: Context-Free Grammars and Denoising Diffusion

2.1 Historical note: Chomsky's formulation

The grammatical formalisms used throughout this thesis trace back to Noam Chomsky's seminal work in the 1950s. In his notes on the *Logical Structure of Linguistic Theory * and the short paper 'Three models for the description of language' [Chomsky, 1956], Chomsky's follow-up monograph on * Syntactic Structures Chomsky [1957] and his review of Skinner's Verbal Behavior Chomsky [1959] argued that natural language occupies an intermediate spot, * context-free *, or slightly beyond, thus motivating both the empirical study of phrase structure grammars and the formal investigation of their computational properties. The resulting **Chomsky hierarchy** remains a cornerstone of formal-language theory, and provides the scaffolding for the Random Hierarchy Model examined here.

2.2 Formal definition of a grammar

Symbols. A *terminal* is an atomic symbol that may appear in the surface string of the language (the data used, e.g. letters, words, or pixels). A *non-terminal* represents an abstract syntactic category such as `S` (sentence) or `NP` (noun phrase) and is rewritten During a derivation.

Definition 1 (Grammar)**.** A grammar is a quadruple

$$G \;=\; \langle V,\, \Sigma,\, R,\, S \rangle,$$

where

- V is a finite, non-terminal set of non-terminal symbols;
- Σ is a finite set of terminal symbols with $V \cap \Sigma = \emptyset$;
- $S \in V$ is the distinguished *start symbol*;
- R is a finite set of *production rules*.

For any set of symbols V, denote V^* as the set of every possible string, of any length, made out of terminals or non-terminals.
A production rule is an ordered pair $(\alpha, \beta) \in (V \cup \Sigma)^* \times (V \cup \Sigma)^*$, conventionally written $\alpha \to \beta$, where α contains at least one non-terminal.
Collecting all rules yields a multi-valued mapping.

$$f_R : (V \cup \Sigma)^* \;\longrightarrow\; 2^{(V \cup \Sigma)^*}, \quad f_R(\alpha) = \{\beta \mid \alpha \to \beta \in R\}.$$

Derivations and language. Given $\gamma_1, \gamma_2 \in (V \cup \Sigma)^*$ we write $\gamma_1 \Rightarrow_R \gamma_2$ if $\gamma_1 = x\alpha y$, $\gamma_2 = x\beta y$ for some $x, y \in (V \cup \Sigma)^*$ and $\beta \in f_R(\alpha)$. The reflexive–transitive closure is written $\Rightarrow_R^*$. The language generated by G is

$$\mathcal{L}(G) \;=\; \Big\{\, w \in \Sigma^* \;:\; S \Rightarrow_R^* w \Big\}.$$

Illustrative example. Let

$$G = \langle V,\, \Sigma,\, R,\, S \rangle, \quad V = \{S,\, N,\, V\},\; \Sigma = \{\texttt{cats}, \texttt{dogs}, \texttt{run}, \texttt{sleep}\},$$

With the rule set

$$R = \Big\{ S \to N\,V,\; N \to \texttt{cats} \mid \texttt{dogs},\; V \to \texttt{run} \mid \texttt{sleep} \Big\}.$$

Here $f_{\mathsf{CFG}}(N) = \{\texttt{cats}, \texttt{dogs}\}$, etc. Two-sample derivations are

$$S \;\Rightarrow\; N\,V \;\Rightarrow\; \texttt{cats}\,V \;\Rightarrow\; \texttt{cats}\,\texttt{run}, \qquad S \;\Rightarrow\; N\,V \;\Rightarrow\; \texttt{dogs}\,V \;\Rightarrow\; \texttt{dogs}\,\texttt{sleep}.$$

Hence $\mathcal{L}(G) = \{\texttt{cats run}, \texttt{cats sleep}, \texttt{dogs run}, \texttt{dogs sleep}\}$. The corresponding parse tree for the first derivation has root S, children N and V, and leaves `cats run`; a non-terminal in V labels depth is 2 and every internal node.

This formalism—symbols, production mapping, derivation relation, and parse trees provide a precise bridge between linguistic intuition and mathematical formalism.

2.3 Context-free grammars

A **context-free grammar** (CFG) is a grammar whose rules are of the form:

$$A \;\to\; \beta, \qquad A \in V,\; \beta \in (V \cup \Sigma)^*.$$

Every derivation of a terminal string yields a *parse tree* with root S and interior nodes from V. The depth of this tree equals the maximum number of rule applications on any root-to-leaf path.

Why the context-free property is indispensable for bottom-up decoding. To analyze features of reconstructed data, we need each sequence to have no ambiguity in reconstruction. To ensure such conditions, we need each s-tuple at level l to have a unique parent at level *l-1*. This procedure is only well-defined if the parent of a symbol (either terminal or not) depends *solely* on that symbol, not on its neighbors in the level. Formally,

let

$$\pi_\ell : \mathcal{V}_\ell^s \longrightarrow \mathcal{V}_{\ell+1} \quad (\ell = 1, \ldots, L-1)$$

be the mapping that sends an ordered s-tuple of level-ℓ features to its level-$(\ell+1)$ parent. For π_ℓ to exist as a function, the grammar must satisfy

> **Uniqueness of upward mapping.** If two derivations contain the same s-tuple $(\mu_1^{(\ell)}, \ldots, \mu_s^{(\ell)})$ at depth ℓ, then the symbol directly above that tuple is *the same* in both derivations.

A context-free grammar guarantees this property because every rule applies to a *single* non-terminal on the left-hand side, independent of the symbols that precede or follow it in the sentential form. Consequently, π_ℓ is well-defined, and the pipeline

$$\mathcal{V}_1^s \xrightarrow{\pi_1} \mathcal{V}_2 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_{L-1}} \mathcal{C}$$

collapses the entire tree to a unique root label, its level 0 symbol.

2.4 Why Grammars Matter for Hierarchical Data

Natural data are sparse dense sets in $\mathbb{R}^D$; they are *made of parts*. An image pixel belongs to an edge, the edge to a surface, the surface of an object, and the object in a scene. Likewise, a sentence sits in a paragraph, decomposes into clauses, which decompose into phrases, which decompose into words, which decompose into characters – a nested cascade of constituents [Chomsky, 1956] such *recursive*, a grammar neatly captures multi-scale organization: every level is encoded as a non-terminal symbol and every "is-composed-of" relation as a production rule.

The Chomsky hierarchy offers a spectrum of grammatical formalisms, from regular expressions up to unrestricted rewriting systems. context-free Grammars (CFGs) hit a sweet spot: expressive enough to model trees of arbitrary depth yet simple enough for linear-time parsing and exact membership tests. Those properties let us inject inductive bias *and keep experiments tractable.*

Specific Setting To isolate the combinatorics of the problem, we entirely strip away visual and phonetic detail and work with an alphabet of abstract characters. each rule therefore breaks a parent character into s child characters; the choice of children are dictated solely by the production table, not by continuous features. Although minimal, this setup still supports the two hallmarks of real data mentioned above:

- *hierarchy*: repeated application of rules builds an s-ary tree of depth L;

- *compositionality*: high-level "symbols" (classes) are recovered by collapsing siblings bottom-up (Eq. (4)).

Later, we will demonstrate how even this simplistic grammar is rich enough to expose sample-complexity barriers and phase transitions in diffusion denoising, while remaining fully analyzable and keeping computational demands manageable.

[1]

3 The Random Hierarchy Model (RHM)

3.1 Motivation: Abstracting Hierarchy with RHM

Deep vision systems succeed because convolutional layers learn *hierarchically compositional* representations: The random hierarchy model (RHM) [Cagnetta et al., 2024] formalizes this intuition in the purely symbolic setting where every feature is represented by a categorical variable rather than a real-valued vector.

Why keep the abstraction? First, it strips away pixel-level nuisances (brightness, small translations) and focuses attention on the discrete *combinatorics* that make hierarchical data hard to learn. Second, by controlling fan-out s, synonym multiplicity m, and depth L, we can dial the intrinsic entropy of the data without touching network capacity or training protocol, ideally isolating where a denoiser fails. Finally, CFG-generated strings admit an exact membership oracle, letting us measure *grammatical validity* of reconstructed samples—an analogue of "object coherence" in image generation but far easier to score. Indeed, that amounts to looking at the defined rule set bottom-up and looking if each $s-$*pair* has a higher-level synonym (or equivalent symbol), substituting it into the sequence, and seeing what label that leads to.

3.2 The Random Hierarchy Model: formal definition

Class labels and vocabularies. Fix a set of class labels $\mathcal{C} = \{1, \dots, v\}$ and L disjoint vocabularies $\mathcal{V}_L, \dots, \mathcal{V}_1$ of *high-* to *low-level* features with homogeneous cardinality $|\mathcal{V}_\ell| = d$. Let $s > 1$ denote the branching rate $m \geq 1$ the number of equivalent, i.e. *synonymic*, rules per feature.

[1] In a context-*sensitive* grammar, the left-hand side of a rule may mention multiple neighboring symbols, so the parent of $(\mu_1^{(\ell)}, \dots, \mu_s^{(\ell)})$ could vary with context; π_ℓ would be multivalued, and bottom-up decoding would be unambiguous.

Top level ($\ell = 1$)**.** Each class label $\alpha \in \mathcal{C}$ generates m distinct s-tuples of high-level features,

$$\alpha \;\mapsto\; \left(\mu_1^{(L)}, \ldots, \mu_s^{(L)}\right), \qquad \mu_i^{(L)} \in \mathcal{V}_L, \tag{1}$$

One of which is selected uniformly.

Recursive levels. For every $\ell = L, L-1, \ldots, 2$ and every feature $\mu^{(\ell)} \in \mathcal{V}_\ell$ produced at the previous step, choose *one* out of m synonymic rules

$$\mu^{(\ell)} \;\mapsto\; \left(\mu_1^{(\ell-1)}, \ldots, \mu_s^{(\ell-1)}\right), \qquad \mu_i^{(\ell-1)} \in \mathcal{V}_{\ell-1}. \tag{2}$$

Leaf level. After $L-1$ recursive applications we reach features $\mu^{(1)} \in \mathcal{V}_1$. The complete expansion, therefore, produces $d = s^L$ terminal symbols, which we one-hot encode to yield an input tensor $x \in \{0,1\}^{d \times v}$.

The generative procedure can be viewed as an s-ary tree with class label α at the root, m synonym alternatives on every internal node, and depth L; see Fig. 1 for an example with $n_c = 2,\ s = 2,\ m = 3,\ L = 3$.

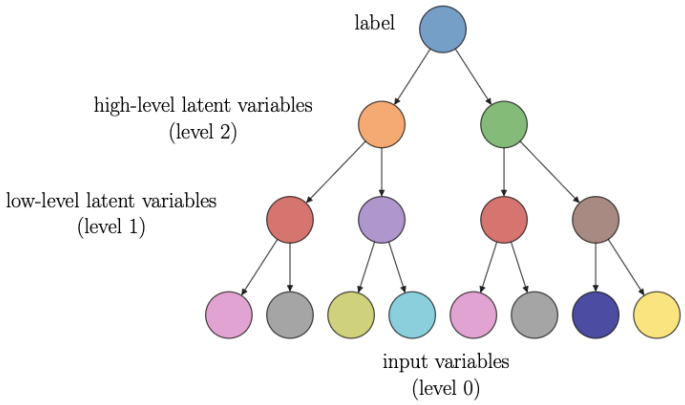


Figure 1: Figure 1: Sketch of the generative hierarchical model we study In the paper, from [Cagnetta et al., 2024]. In this example, depth L = 3 and branching factor s = 2. Different values of the input and latent variables are represented with different colors

Bottom-up label reconstruction. Because every rule rewrites a *single* non-terminal independently of its context, the grammar is context-free; hence, the parent of any s-tuple of features is *unique*. Formally, there exists a family of functions

$$g_\ell : (\mathcal{V}_\ell)^s \longrightarrow \mathcal{V}_{\ell+1}, \qquad \ell = 1, \ldots, L-1, \tag{3}$$

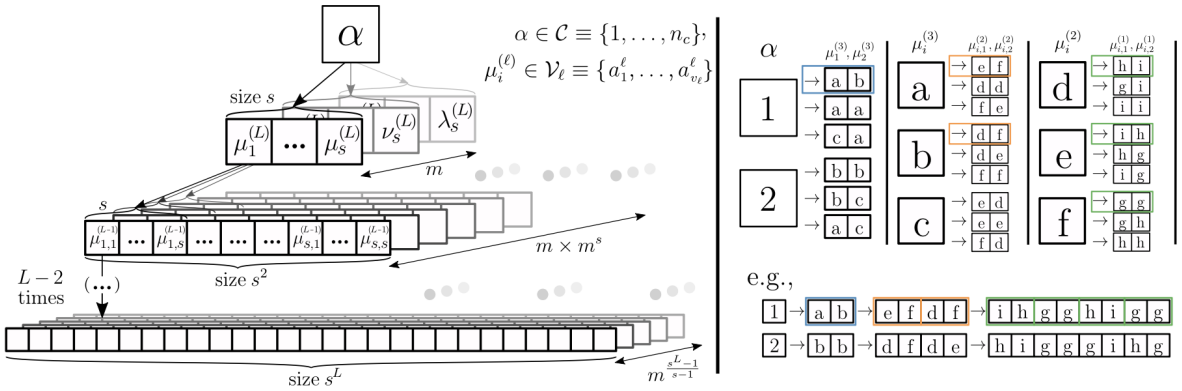


Figure 1. **The Random Hierarchy Model. Left:** Structure of the generative model. The class label $\alpha = 1, \ldots, n_c$ generates a set of m equivalent (i.e., *synonymic*) high-level representations with elements taken from a vocabulary of high-level features $\mathcal{V}_L$. Similarly, high-level features generate m equivalent lower-level representations, taken from a vocabulary $\mathcal{V}_{L-1}$. Repeating this procedure $L-2$ times yields all the input data with label α, consisting of low-level features taken from $\mathcal{V}_1$. **Right:** example of Random Hierarchy Model with $n_c = 2$ classes, $L = 3$, $s = 2$, $m = 3$ and homogeneous vocabulary size $v_1 = v_2 = v_3 = 3$. The three sets of rules are listed at the top, while two examples of data generation are shown at the bottom. The first example is obtained by following the rules in the colored boxes.

Figure 2: Evolution of reconstruction accuracy over diffusion time, from[Cagnetta et al., 2024].

such that collapsing siblings with g_ℓ and repeating $L-1$ times maps the leaf string to its class label

$$\alpha \;=\; g_{L-1} \circ g_{L-2} \circ \cdots \circ g_1\Big(\mu_1^{(1)}, \ldots, \mu_{s^L}^{(1)}\Big). \tag{4}$$

Parameter homogeneity. [Cagnetta et al., 2024] keep s and m *constant* across primary objective of the present work is to examine how the setting changes. The primary objective of the present work is to analyze how the setting varies when lifting this presumably limiting constraint. Restriction and systematically vary fan-out and synonym multiplicity between layers to study how hierarchical non-uniformity affects denoising robustness and sample complexity. The underlying intuition is that a simple tree should be heavily dependent on the "flip" of a single character. Given a sequence generated by a binary tree, consider altering one terminal character to another. That could reflect on the higher-level synonyms, possibly changing it and iterating to higher levels. On the other hand, a more robust tree would need more lower-level alterations of the original sequence to propagate to the parents' nodes.

3.3 Sample Complexity and the Role of Depth

Uniform trees. Consider an RHM in which the branching factor equals the terminal vocabulary size, s, at every level.

Each non-terminal then expands into any of its s children, so the number of distinct leaf

strings grows as s^L for depth L.
Since no substructures repeat, a learner must observe on the order of $\Omega\big(s^L\big)$ examples to cover every derivation path once, the required dataset size therefore scales *exponentially* with depth.

Depth as intrinsic dimension. For an arbitrary grammar, the effective dimension of the leaf manifold is

$$d_{\text{leaf}} \;=\; b_{\text{leaf}}^{L},$$

where b_{leaf} is the average fan-out in the bottom layer. Non-parametric theory tells us that any model treating leaves as independent coordinates needs $\Theta\big(b_{\text{leaf}}^{L}\big)$ labelled samples to attain a fixed generalisation error [von Luxburg and Bousquet, 2004, Bach, 2017]. Hence, depth is the dominant driver of sample complexity.

Full-manifold boundary. If the branching factor m equals the alphabet size s and no lower level synonyms are ruled out, every length-d_{leaf} string is valid. In such a case, the data manifold then fills the whole space, and learning the grammar's structure amounts to learning how to generate a one-hot encoded sequence of given length.
Thus, all experiments will be run in grammar where one lower-level representation is masked out, adding necessary structure. When masking synonyms, we ensure that all symbols appear with the same frequency to avoid any biases.

4 Denoising Diffusion Probabilistic Models

4.1 Forward and Reverse Processes

Denoising diffusion probabilistic models (DDPMs) generate data by *first* corrupting a clean input $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ with a scheduled chain of Gaussian noise injections and *then* training a neural network to run that chain in reverse.
Concretely, the **forward** (noising) process is a Markov chain $(\mathbf{x}_t)_{t=0}^{T}$ defined by

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \;=\; \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \;=\; \mathcal{N}\Big(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\, \beta_t \mathbf{I}\Big), \tag{5}$$

where $(\beta_t)_{t=1}^{T}$ is a variance schedule with $\beta_t \in (0,1)$ chosen so that the marginal variance $\alpha_t \;=\; \prod_{s=1}^{t}(1-\beta_s)$ decays smoothly to zero, implying $\mathbf{x}_T$ converges asymptotically to an isotropic Gaussian. Stacking the Gaussian kernels lets us write any noisy state in closed

form ([Ho et al., 2020]):

$$\mathbf{x}_t \;=\; \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 \;+\; \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{6}$$

where $\bar{\alpha}_t \;=\; \prod_{s=1}^{t} \alpha_s$ making forward sampling immediate.

The **reverse** (denoising) process relies on learning parameter θ so that a neural network delivers mean and variance estimates for the step-by-step Gaussian transition

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \;=\; \mathcal{N}\Big(\mathbf{x}_{t-1};\; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\; \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\Big), \tag{7}$$

In practice, one assumes the variance of $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ to be predefined for any t according to a scheduler constrained to vanish as t approaches ∞. Thus, the only learnable parameter is the mean of the backward conditional distribution.
Thus new data is generated by initializing $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and sampling downward $T \to 0$. While it is possible to jump from 0 to T in the forward noise process, the same is not true for the backward denoising process.

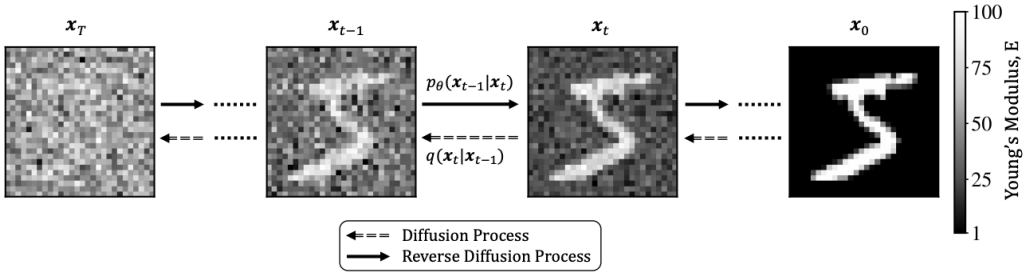


Figure 3: Representation of the diffusion process, (Source:[Vlassis and Sun, 2023]) : Noise is added to destroy the signal of the original distribution. The forward-noising Process q is fixed. A neural network learns the denoising process p.

4.2 Derivation of the Drift Term

The denoising task turns out to be only tractable when conditioned on the previous step:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\Big(x_{t-1};\; \tilde{\mu}_t(x_t, x_0),\; \tilde{\beta}_t\, I\Big).$$

Indeed, using Bayes' rule, we can derive its expression as:

$$
\begin{aligned}
q(x_{t-1} \mid x_t, x_0) &= \mathcal{N}\Big(x_{t-1};\, \tilde{\mu}_t(x_t, x_0),\, \tilde{\beta}_t\, I\Big) \\
&= \frac{q(x_t \mid x_{t-1}, x_0)\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \\
&\propto \exp\left(-\tfrac{1}{2}\left[\tfrac{\|x_t - \sqrt{\alpha_t}\, x_{t-1}\|^2}{\beta_t} + \tfrac{\|x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\, x_0\|^2}{1-\bar{\alpha}_{t-1}} - \tfrac{\|x_t - \sqrt{\bar{\alpha}_t}\, x_0\|^2}{1-\bar{\alpha}_t}\right]\right).
\end{aligned}
$$

Completing the square in x_{t-1} yields:

$$
q(x_{t-1} \mid x_t, x_0) \;\propto\; \exp\left(-\tfrac{1}{2}\left[\left(\tfrac{\alpha_t}{\beta_t} + \tfrac{1}{1-\bar{\alpha}_{t-1}}\right)\|x_{t-1}\|^2 - 2\left(\tfrac{\sqrt{\alpha_t}}{\beta_t} x_t + \tfrac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} x_0\right)^{\top} x_{t-1} + C(x_t, x_0)\right]\right),
$$

so that by matching the Gaussian form, we identify

$$
\begin{aligned}
\tilde{\beta}_t &= \left(\tfrac{\alpha_t}{\beta_t} + \tfrac{1}{1-\bar{\alpha}_{t-1}}\right)^{-1} = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\, \beta_t, \\
\tilde{\mu}_t(x_t, x_0) &= \tilde{\beta}_t\left(\tfrac{\sqrt{\alpha_t}}{\beta_t}\, x_t + \tfrac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\, x_0\right) \\
&= \frac{\sqrt{\alpha_t(1-\bar{\alpha}_{t-1})}}{1-\bar{\alpha}_t}\, x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\, \beta_t}{1-\bar{\alpha}_t}\, x_0.
\end{aligned}
$$

Where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

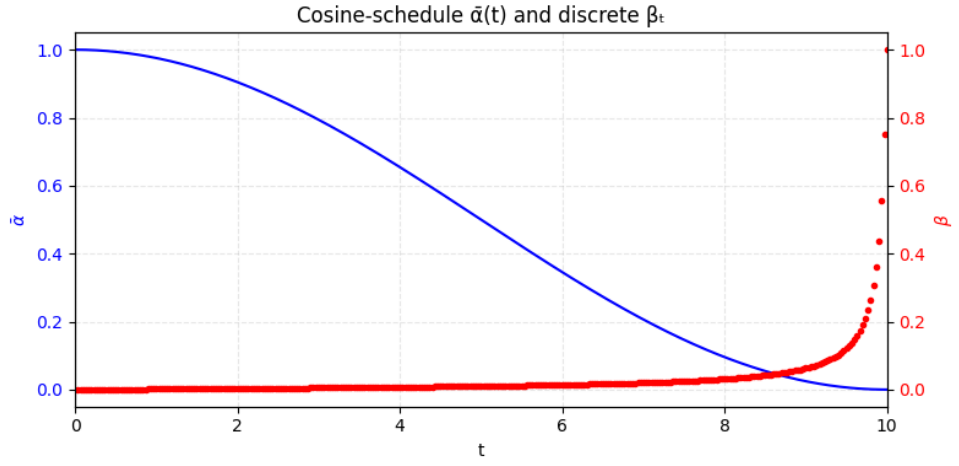


Figure 4: Visualization of $\bar{\alpha}_t$ *and*β_T in the case of a cosine schedule, later employed for training.

4.3 Derivation of the Loss, Variational Lower Bound

The choice of the loss function is nontrivial. To maximize the log-likelihood $p_\theta(x_0 \mid x_1, \ldots, x_T)$ we use the same trick used in the derivation of Variational Lower Bound (E. Jang. A Beginner's Guide to Variational Methods: Mean-Field Approximation).

$$
\begin{aligned}
-\log p_\theta(x_0) &\leq -\log p_\theta(x_0) + D_{\mathrm{KL}}\Big(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0)\Big) \\
&= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(\cdot|x_0)}\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})/p_\theta(x_0)}\right] \\
&= -\log p_\theta(x_0) + \mathbb{E}_q\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} + \log p_\theta(x_0)\right] \\
&= \mathbb{E}_q\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})}\right].
\end{aligned}
$$

Let

$$
\mathcal{L}_{\mathrm{VLB}} = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})}\right] \;\geq\; -\mathbb{E}_{q(x_0)}\Big[\log p_\theta(x_0)\Big].
$$

Moreover:

$$
\begin{aligned}
\mathcal{L}_{\mathrm{VLB}} &= \mathbb{E}_q\Big[\log \tfrac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\Big] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{t=1}^{T} q(x_t \mid x_{t-1})}{p_\theta(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)}\right] \\
&= \mathbb{E}_q\Big[-\log p_\theta(x_T) + \sum_{t=1}^{T} \log \tfrac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(x_T) + \sum_{t=2}^{T} \log \tfrac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} + \log \tfrac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\Big] \\
&= \mathbb{E}_q\Big[\underbrace{\log \tfrac{q(x_T|x_0)}{p_\theta(x_T)}}_{L_T} + \sum_{t=2}^{T} \underbrace{\log \tfrac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}}_{L_{t-1}} - \underbrace{\log p_\theta(x_0 \mid x_1)}_{L_0}\Big] \\
&= \mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}\Big(q(x_T \mid x_0)\|p_\theta(x_T)\Big)}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}\Big(q(x_{t-1} \mid x_t, x_0)\|p_\theta(x_{t-1} \mid x_t)\Big)}_{L_{t-1}} - \underbrace{\log p_\theta(x_0 \mid x_1)}_{L_0}\Big].
\end{aligned}
$$

Thus, we finally notice:

$$\mathcal{L}_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0,$$

where

$$\begin{aligned}
L_T &= D_{\text{KL}}\Big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\Big), \\
L_t &= D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big) \quad \text{for } 1 \le t \le T-1, \\
L_0 &= -\log p_\theta(x_0 \mid x_1).
\end{aligned}$$

4.4 Parameterization of L_t for Training Loss

Recall that we need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process.

$$p_\theta(x_{t-1} \mid x_t) \;=\; \mathcal{N}\Big(x_{t-1};\, \mu_\theta(x_t, t),\, \Sigma_\theta(x_t, t)\Big).$$

We would like to train μ_θ to predict

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\Big(x_t \;-\; \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_t\Big). \tag{8}$$

Because x_t is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict ϵ_t from the input x_t at time step t:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t \;-\; \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\Big).$$

Thus

$$x_{t-1} \;\sim\; \mathcal{N}\Big(x_{t-1}\,;\; \tfrac{1}{\sqrt{\alpha_t}}\Big(x_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\Big),\; \Sigma_\theta(x_t, t)\Big).$$

The loss term L_t is parameterized to minimize the difference from $\tilde{\mu}_t$:

$$\begin{aligned}
L_t &= \mathbb{E}_{x_0,\epsilon}\Big[\, \tfrac{1}{2\,\|\Sigma_\theta(x_t,t)\|_2^2}\, \Big\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\Big\|^2\Big] \\
&= \mathbb{E}_{x_0,\epsilon}\Big[\, \tfrac{1}{2\,\|\Sigma_\theta\|_2^2}\, \Big\| \tfrac{1}{\sqrt{\alpha_t}}\Big(x_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_t\Big) - \tfrac{1}{\sqrt{\alpha_t}}\Big(x_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\Big)\Big\|^2\Big] \\
&= \mathbb{E}_{x_0,\epsilon}\Big[\, \tfrac{(1-\alpha_t)^2}{2\,\alpha_t\,(1-\bar{\alpha}_t)\,\|\Sigma_\theta\|_2^2}\, \|\epsilon_t - \epsilon_\theta(x_t, t)\|^2\Big] \\
&= \mathbb{E}_{x_0,\epsilon}\Big[\, \tfrac{(1-\alpha_t)^2}{2\,\alpha_t\,(1-\bar{\alpha}_t)\,\|\Sigma_\theta\|_2^2}\, \Big\|\epsilon_t - \epsilon_\theta\Big(\sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_t,\; t\Big)\Big\|^2\Big].
\end{aligned}$$

Simplification Empirically, [Ho et al., 2020] found that training the diffusion model works better with a simplified objective that ignores the weighting term:

$$L_{\text{simple}} = \mathbb{E}_{t,x_0,\epsilon_t}\Big[\|\epsilon_t - \epsilon_\theta(x_t,t)\|^2\Big],$$

4.5 Reformulation from Noise to Clean Input

For practical reasons, we prefer to directly work with noisy and denoised inputs, which are algebraically equivalent:

$$\begin{aligned}
x_t = \sqrt{\bar\alpha_t}\,x_0 \implies \epsilon_t &= \frac{x_t - \sqrt{\bar\alpha_t}\,x_0}{\sqrt{1-\bar\alpha_t}},\\
\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\,\epsilon_\theta(x_t,t)\Big) &\implies \epsilon_\theta(x_t,t) = \frac{\sqrt{1-\bar\alpha_t}}{1-\alpha_t}\Big(x_t - \sqrt{\alpha_t}\,\mu_\theta(x_t,t)\Big),\\
L_{\text{simple}} &= \mathbb{E}\Big\|\epsilon_t - \epsilon_\theta(x_t,t)\Big\|^2\\
&= \mathbb{E}\Big[\frac{1}{(1-\alpha_t)(1-\bar\alpha_t)}\Big\|\sqrt{\alpha_t}\Big(x_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\,\epsilon_t\Big) - \sqrt{\alpha_t}\,\mu_\theta(x_t,t)\Big\|^2\Big]\\
&\propto \mathbb{E}\Big\|\tilde\mu_t(x_t,x_0) - \mu_\theta(x_t,t)\Big\|^2.
\end{aligned}$$

Crucially, we will focus on so called *U-turns*: we *interrupt* the diffusion halfway at some $0 < t < T$—yielding a partially noised $\mathbf{x}_t$—and then run the learned p_θ to produce $\hat{\mathbf{x}}_0(t) \sim p_\theta(\hat{\mathbf{x}}_0 \mid \mathbf{x}_t)$. Cagnetta et al. 2023 report a characteristic behavior of partially-noised reconstructed data: low–level texture cues morph continuously for small t, while a sharp, phase-transition-like drop in class-level consistency appears around a critical time t^*; beyond that, even though global semantics flip, the generated image still reuses fine-grained motifs of the original (e.g., colors, edges) inherited from $\mathbf{x}_0$. Although we will operate in a different setting, using sequences of characters rather than images, we expect to observe similar behavior when using the same tree as in the paper.

4.6 the manifold hypothesis and diffusion generative models

The *manifold hypothesis* posits that real-world data occupy a set $\mathcal{M} \subset \mathbb{R}^D$ whose intrinsic dimension $m = \dim(\mathcal{M})$ is much smaller than the ambient dimension
D[Carlsson, 2009, Narayanan and Mitter, 2010, Fefferman et al., 2016]. for images, for instance, D is the number of pixels while m encodes far fewer degrees of freedom, such as lighting, pose, or texture. Diffusion models exploit this structure by first adding isotropic Gaussian noise,

$$q_t(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\Big(\mathbf{x}_t; \sqrt{\bar\alpha_t}\mathbf{x}_0,\, (1-\bar\alpha_t)\mathbf{I}\Big),$$

which pushes samples away from $\mathcal{M}$ toward the full $\mathbb{R}^D$; they then learn a *time-conditioned drift* $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ that drives the reverse stochastic process back onto the manifold [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song et al., 2021]. This drift is proportional to the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, so its shape depends intimately on the particular data manifold encoded by the chosen dataset. different corpora therefore, induce different reverse flows—highlighting that $\mathcal{M}$ is not universal but learned task-by-task.

5 The considered rulesets

Building on the Random Hierarchy Model (RHM) introduced in the previous chapter, we now study how a diffusion denoising process acts on *sequences generated by two trees, which differ in complexity.* We will consider a binary tree, as described in the paper by Cagnetta et al. (2023), and analyze a more structured scenario with a variable branching rate. Moreover, we will study a masked learning scenario in which we artificially penalize prohibited sequences in the loss.

5.1 Why two grammars?

- **Baseline (binary) grammar.** G_{bin} mirrors the Random Hierarchy Model used by [**?**]: every internal node expands into *exactly two children.* The tree is therefore *depth-driven*—complexity grows by adding layers, not fan-out, and serves the purpose of a control condition. However, in these experiments, we lack the computational power necessary to study high-dimensional data. Since the dimensionality itself grows exponentially with the depth of the tree, we will not be able to replicate the *10 Layer Tree* of the Cagnetta et al. work.

- **Variable-branch grammar.** Real-world hierarchies (syntax trees, vision part-trees) rarely branch uniformly. G_{var} relaxes the binary constraint by allowing a location-dependent branching rate $s_\ell \in \{1, 2, 3\}, \quad \ell = 1, \ldots, L-1$, Sampled once at grammar initialization and then held fixed. Specifically, we implement a tree with a high branching rate from the label to the first layer, and then reduce it to a binary tree. We hypothesize that the higher number of synonyms that need to be flipped in the denoising diffusion process will lead to more stable reconstruction in the diffusion, since it would require more than one label to change the parent's label. root (see the percolation argument of §3).

5.2 First Grammar: Binary case

Let

$$\mathcal{C} = \{1, 2\}, \qquad \mathcal{V}_3 = \{a, b, c\},\ \mathcal{V}_2 = \{d, e, f\},\ \mathcal{V}_1 = \{g, h, i\},$$

and denote by $m = 2$ the number of *synonymic* (probability-1) rules per non-terminal. Both grammars are probabilistic CFGs $G = \langle V, \Sigma, R, S \rangle$ with root symbols $V_0 = \mathcal{C}$, terminals $\Sigma = \mathcal{V}_1$, and identical rule weights (all 1.0) so that likelihood depends *solely* on the tree's combinatorics.

$$
\begin{aligned}
1 &\to \{\, ab,\ ac\}, Labels \\
2 &\to \{\, ba,\ bc\}, Labels \\
a &\to \{\, de,\ df\}, Layer2 \\
b &\to \{\, ed,\ ef\}, Layer2 \\
c &\to \{\, fd,\ fe\}, Layer2 \\
d &\to \{\, gi,\ gh\}, Layer3 \\
e &\to \{\, ig,\ ih\}, Layer3 \\
f &\to \{\, hg,\ hi\}, Layer3
\end{aligned}
$$

For example, we could expand as follows:

$$
\begin{aligned}
1 &\to ac \quad (Layer1) \\
a &\to df \quad (Layer2) \\
c &\to fe \quad (Layer2) \\
d &\to gi \quad (Layer3) \\
f &\to hg \quad (Layer3), \textbf{First f} \\
f &\to hi \quad (Layer3), \textbf{Second f} \\
e &\to ig \quad (Layer3)
\end{aligned}
$$

Hence, the final concatenation is

$$gi \mid hg \mid hi \mid ig \;=\; \texttt{gihghiig}\,,$$

Which is one possible allowed sequence of the model? By simple combinatorics, we find that for any ruleset, the number of permitted sequences is

$$\text{Total number of allowed sequences} \;=\; \sum_{\alpha \in \mathcal{C}} \prod_{i=1}^{L} \big(m_i\big)^{\,s^{\,i-1}}$$

Where we sum over all labels $\alpha \in\ C$. In this case, we get

$$\text{Total} \;=\; \underbrace{2}_{\text{roots}} \times \underbrace{2}_{\text{Level 1}} \times \underbrace{4}_{\text{Level 2}} \times \underbrace{16}_{\text{Terminal}} \;=\; 256$$

Out of the space of all possible sequences generated by strings of length 8 of the 3 characters, of cardinality $6561 = 3^8$, we allow about 3.9% to be generated. We will consider

two training scenarios: one where the model sees about half of the possible sequences in training, aiming to make the model correctly generalize to new sequences, and one of complete memorization, where the model is trained to memorize the entire dataset.

5.3 Second Grammar: Variable-branch Case

Let

$$\mathcal{C} = \{1, 2, 3\}, \qquad \mathcal{V}_2 = \{a, b, c, d, e, f\}, \quad \mathcal{V}_3 = \{g, h, i, j, k, l\}, \quad \mathcal{V}_1 = \{u, v, w, x, y, z\},$$

And denote by

$$s = 2, \quad m_1 = 5, \quad m_2 = 2, \quad m_3 = 2$$

The branching rate and synonym multiplicities at layers 1, 2, 3, respectively. All rules carry weight 1.0, so that $G = \langle V, \Sigma, R, S \rangle$ is again a probabilistic CFG whose likelihood depends *only* on its combinatorics.

The production rules are

Layer 1 (root → 2 non-terminals, 5 synonyms each):

$1 \to \{\, ab,\ ac,\ ad,\ ae,\ af \}$
$2 \to \{\, bc,\ bd,\ be,\ bf,\ cd \}$
$3 \to \{\, ce,\ cf,\ de,\ df,\ ef \}$

Layer 2 (each symbol → 2 non terminals, 2 synonyms):

$a \to \{\, gh,\ gi \}, \quad b\{\, hg,\ hi \}, \quad c\{\, ig,\ ih \},$
$d \to \{\, jk,\ jl \}, \quad e\{\, kj,\ kl \}, \quad f\{\, lj,\ lk \},$

Layer 3 (each symbol → 2 terminals, 2 synonyms):

$g \to \{\, uv,\ uw \}, \quad h\{\, vu,\ vw \}, \quad i\{\, wu,\ wv \},$
$j \to \{\, xy,\ xz \}, \quad k\{\, yx,\ yz \}, \quad l\{\, zx,\ zy \}.$

For instance, one derivation might proceed:

$$\begin{aligned} 2 &\to bd \quad \text{(layer 1)} \\ b &\to hi \quad \text{(layer 2)} \\ d &\to jl \quad \text{(layer 2)} \\ h &\to vw \quad \text{(layer 3)} \\ i &\to wu \quad \text{(layer 3)} \\ j &\to xy \quad \text{(layer 3)} \\ l &\to zy \quad \text{(layer 3)} \end{aligned} \quad \Longrightarrow \texttt{vw wu xy zy}.$$

As described in section 5.2, the total number of grammatical sequences is

$$= \sum_{\alpha \in \mathcal{C}} \prod_{i=1}^{L} \left(m_i\right)^{s^{i-1}} = \underbrace{2}_{\text{roots}} \times \underbrace{5}_{\text{Level 1}} \times \underbrace{4}_{\text{Level 2}} \times \underbrace{16}_{\text{Terminal}} = 960$$

The greater vocabulary dimension of $s = 6$ implies the whole space counts $6^8 = 1{,}679{,}616$ possible strings of length 8. Thus, the data manifold covers

$$\frac{960}{6^8} \approx 0.0057\%$$

of the whle configuration space G.
This setting aims to replicate the one described in [Cagnetta et al., 2024]. However, their tree is much deeper ($L = 10$) and uses more labels ($V = 32$). To enable us to tackle the problem from a computational power perspective, and to have a reasonable amount of time to debug the code, we can only afford 2 labels and 3 layers. Indeed, diffusion processes don't require any lower/upper bound on the dimensionality of the input, and don't behave depending on it.

6 Transformer-Based Denoiser: Training Pipeline

6.1 The Transformer Denoiser architecture.

To implement the reverse stage, we train a *Transformer Denoiser* that, for every corruption level $t \in [0, 1]$, learns to map a noised sample $\mathbf{x}_t$ to the fully denoised estimate $\widehat{\mathbf{x}}_0(t)$. Once trained, the network is plugged into the reverse diffusion chain: at each discrete step it provides $\widehat{\mathbf{x}}_0(t_j)$, and the sampler combines that prediction with the current state $\mathbf{x}_{t_j}$ according to Eq. (8), weighting the two terms with time–dependent coefficients fixed by the noise-schedule. In this way, the model supplies a grammar-aware drive term $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ (see 4.1) that steadily steers the chain from pure noise back to the data manifold.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 5: The training and sampling algorithms in DDPM, [Ho et al., 2020]
Note: In this setting, we take a gradient descent step concerning the reconstructed clean input, not the noise. The formulation is completely equivalent, as derived in subsection 4.6.

6.2 Network architecture

The grammar's depth is L; we therefore build a symmetric $\text{Encoder}_1, \dots, \text{Encoder}_L \leftrightarrow \text{Decoder}_1, \dots, \text{Decoder}_L$ stack, so that each layer can collapse one level of the RHM tree. A one-hot slice $|(d = s * n|)$, where n is the length of the bottom layer sequence, is first linearly projected to `embed_dim` $-$ 1 channels, after which we append a sinusoidal positional encoding [Vaswani et al., 2017] to inject the leaf index and a single scalar timestep t (normalized to $[0, 1]$), giving the model explicit access to both *where* the character sits in the sequence and *how much* corruption it currently carries. The resulting tensor passes through the L self-attention encoders, then through the L cross-attention decoders that fuse encoder memory back into the reconstruction stream, and is finally linearly remapped to the original (d, n) logits. This design allows the network to simultaneously learn hierarchical synonym collapses and time-conditioned denoising while keeping the parameter count proportional to the grammar depth L.

6.3 Implementation details.

Synthetic datasets are produced by expanding a code snippet that replicates the RHM implementation of [Cagnetta et al. 2023]: different labels are expanded according to the ruleset, picking lower-level synonyms (and thus production rules) uniformly at random until depth

$$n = \left(\prod_{i=1}^{L} \mathrm{m}_i \right).$$

In the case of a constant branching rate, each string has length $n = m^L$ and each character lives in a vocabulary of size s; we therefore store every example as a torch tensor $\mathbf{x} \in \{0, 1\}^{B \times d \times n}$, where B is the number of samples in the dataset.
After a global shuffle, we sample B independent time steps $t \sim \mathcal{U}(0, 1)$ per example and

corrupt the clean tensor with the forward diffusion operator, yielding the pair $\big(\mathbf{x}_t,\, t\big)$.

training uses the loop sketched below (see `masked_train_for_denoise()` in the repository). The key steps are

1. **data preparation:**

   - randomly split the full dataset into a training set and a validation set. In order to properly test the reverse diffusion of the test set, we need a large batch. Since we can decide how much data to generate, we modify accordingly.
   - wrap each subset in a *DataLoader*.

2. **model, optimizer, and learning-rate scheduling:**

   - use AdamW (a variant of Adam that decouples weight-decay from adaptive moment estimates, effectively performing SGD with momentum and adaptive steps),
   - attach a warm-up schedule so that the learning rate starts small and gradually increases during the first few hundred updates,
   - attach a scheduler that monitors validation loss and reduces the learning rate whenever the loss plateaus,
   - in practice, we fine-tune these optimizer and scheduler parameters (e.g. base learning rate, warm up length, plateau patience) separately for each ruleset or dataset.

3. **loss function:**

   - use standard cross-entropy with label smoothing parameter α, where we vary $\alpha \in [0, 0.2]$. Concretely:

     $$\mathcal{L}_{\mathrm{CE}} \;=\; -\sum_{i=1}^{n} \tilde{y}_i \,\log p_i, \quad \tilde{y}_i \;=\; (1-\alpha)\, y_i \;+\; \frac{\alpha}{n-1}\,(1-y_i),$$

     Where p_i is the model's predicted probability for class i, y_i is the one-hot ground-truth indicator, and α is chosen (for each training scenario) in the interval $[0, 0.2]$.

4. **training loop (for each epoch** $e = 1, \ldots, E$**):**

(a) **training phase:**

- iterate over each mini-batch (x_{noisy}, y_{clean}, t) from the training loader,
- perform a forward pass through the model to produce logits, including positional encoding of the characters through the string and percentage of noise added as a timestep,
- compute cross-entropy loss (with label smoothing α) against the clean targets,
- back propagate gradients and clip their norm at a fixed threshold,
- take an optimizer step, then update the warmup schedule,
- accumulate the batch losses, and at the end, record the average training loss for this epoch.

(b) **validation phase:**

- iterate over each mini-batch in the validation loader,
- compute logits and cross-entropy loss,
- average those losses to obtain the epoch's validation CE,
- pass the validation CE into the scheduler so it can adjust the learning rate if needed,
- log the validation CE.

(c) **reverse-diffusion evaluation (every eval_every epochs):**

- keep the model in evaluation mode,
- create 10 equally spaced buckets over the normalized timestep range $[0, 1]$,
- for each validation batch (x_{noisy}, t):
  - compute a reconstructed "clean" prediction via the reverse-diffusion procedure,
  - turn that reconstruction into a discrete prediction of labels,
  - mark which samples are "legal" versus "illegal" under the grammar rules,
  - for each bucket, count how many examples fall into it and how many of those are illegal,
- after processing all batches, divide illegal counts by total counts per bucket (clamping zeros to avoid division issues), yielding an illegal-probability vector of length 10,
- append (epoch, illegal_prob) to a list and print out the illegal-probabilities for this epoch.

(d) **early stopping and check pointing:**

- maintains the best observed validation CE so far, and a counter of "bad" epochs,
- if the current validation CE improves by at least a small threshold over the best, reset the bad-epoch counter and save the model weights to `best_model.pt`,
- otherwise increment the bad-epoch counter,
- if the bad-epoch counter reaches the patience limit, stop training early.

5. **post-training diagnostics:**

- once training ends (either after E epochs or due to early stop), plot three curves over epoch index:
  (a) training cross-entropy,
  (b) validation cross-entropy,
  (c) accuracy of predicted labels from reconstructing noisy inputs.

This routine is optimized for each rule set. When training on somewhat structured data, we implemented unmasked training, where the mass penalty term in the loss function vanishes. The key point is the number of samples used: For the model to enter a "*memorization phase* we would need d^L number of samples, but we use much less for computational issues.

6.4 Dynamical regimes and speciation time.

[Biroli et al., 2024] shows that the diffusion based on scores evolves through successive regimes *three*. **I) Survival:** For small noise, the trajectory stays confined to its original class manifold. **II) Speciation:** at an intermediate scale, the class label can flip even though low-level features are still partly preserved. **III) Drift:** close to pure noise, every semantic trace is lost and the generative model outputs class labels uniformly at random. They predict that the I↦II crossover (dubbed *speciation*) occurs when

$$\Lambda\, e^{-2t_S} = 1 \quad \Longrightarrow \quad t_S = \tfrac{1}{2}\ln\Lambda,$$

where Λ is the top eigen-value of the *correlation* matrix of the clean data. [2] We will focus solely on this regime transition due to the evident shift in effectiveness of the denoising process occurring at the predicted t_S.

[2] Biroli *et al.* normalize each feature to unit variance before computing eigenvalues. We follow the same convention; raw one-hot covariances would otherwise yield $\Lambda < 1$ and a negative t_S.

7 Training to Memorize

7.1 First Case: Binary tree

To test the correctness of this implementation, we start by checking if the model can denoise by training the transformer model in the scope of memorization, training on a training set of 4 times the number of allowed sequences, $256 * 4 = 1024$, with a test set of 256 more samples. We expect all the samples in the test set to be present in the train set, so the model should be able to properly denoise all inputs. Indeed, we do not expect exact denoising at all timesteps, but only when the data is mildly corrupted. Figure 3 shows the noising process for this rule set, where sequences of length 8 are one-hot encoded in 3 characters.

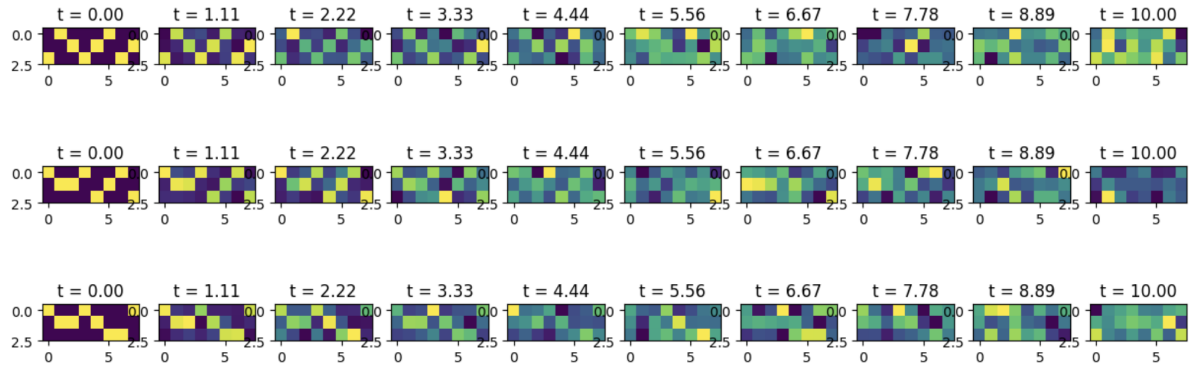


Figure 6: Visualization of the noising process for binary data, $s = 3$, $m^L = 2^3 = 8$. Noising follows a cosine scheduler. The speciation time, corresponding to the passage from reconstructible noise to random Gaussian inputs, occurs at $t_S = 0.582$.

For this dataset, we expect a speciation time where reconstruction of denoised inputs stops being exact, coinciding with the "generative" region of the process (see *Section 6.4*), of

$$t_S = \tfrac{1}{2}\ln(\Lambda) = \tfrac{1}{2}\ln 5.108 = \approx 0.82.$$

Since all sequences possibly generated by the rule set are fed into the model, the main objective is to see whether the architecture we experiment with is capable of learning the structure of one-hot-encoded data, and whether the diffusion process is correctly implemented.

The loss converges to low values after escaping two plateaus. We observe that denoising occurs primarily for low timesteps with very distinct one-hot columns, converging to random digits for completely noised inputs.

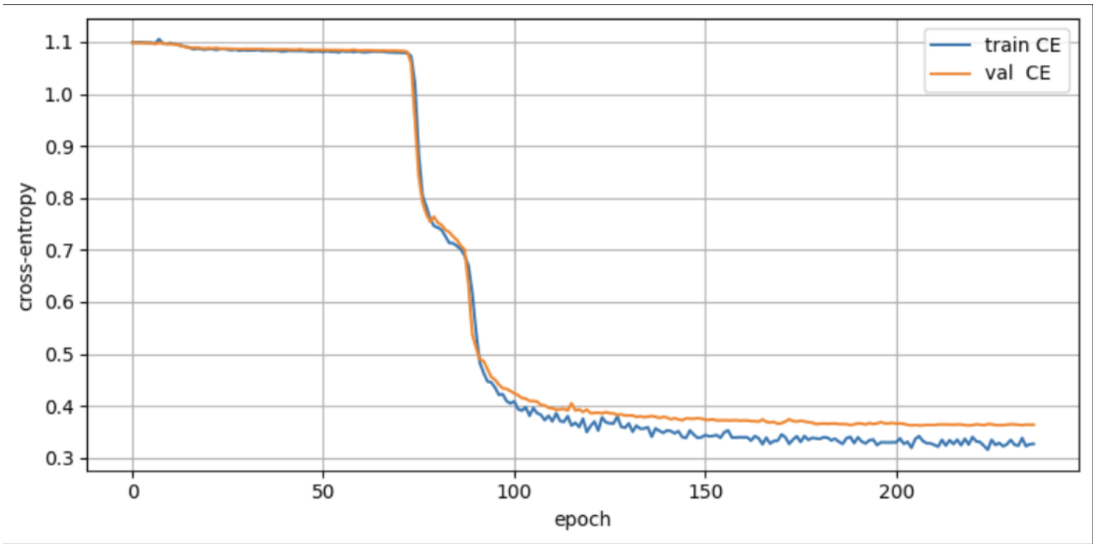


Figure 7: Training curves for the binary tree

Although training occurs explicitly on the transformer denoiser, the convergence of the diffusion process is key to evaluating the model's effectiveness. The restricted number of layers of the tree we implemented is constraining for the informative inner-architecture representation of the vectors across evolution. The choice of the metric that tracks the effectiveness of the diffusion process is, indeed, nontrivial. We favour the "percentage of correctly reconstructed labels", being the metric that we theorize will be affected the most by a more stable, diffused tree.

We examine the U-Turns of the data: noising the test set (unseen by the model) until timestep t and plotting the fraction of correctly reconstructed labels: Recall that in a binary scenario the tree could both reconstruct the correct label, say *1*, the other, *2*, or produce a sequence which is prohibited by the rule set, marked as *0*.

Given that the space of allowed sequences covers about 4% of the whole space, an uninformed transformer where the drive factor is not trained correctly would generate invalid outputs 96% of the time. After extensive training, the model reaches the desired behavior:

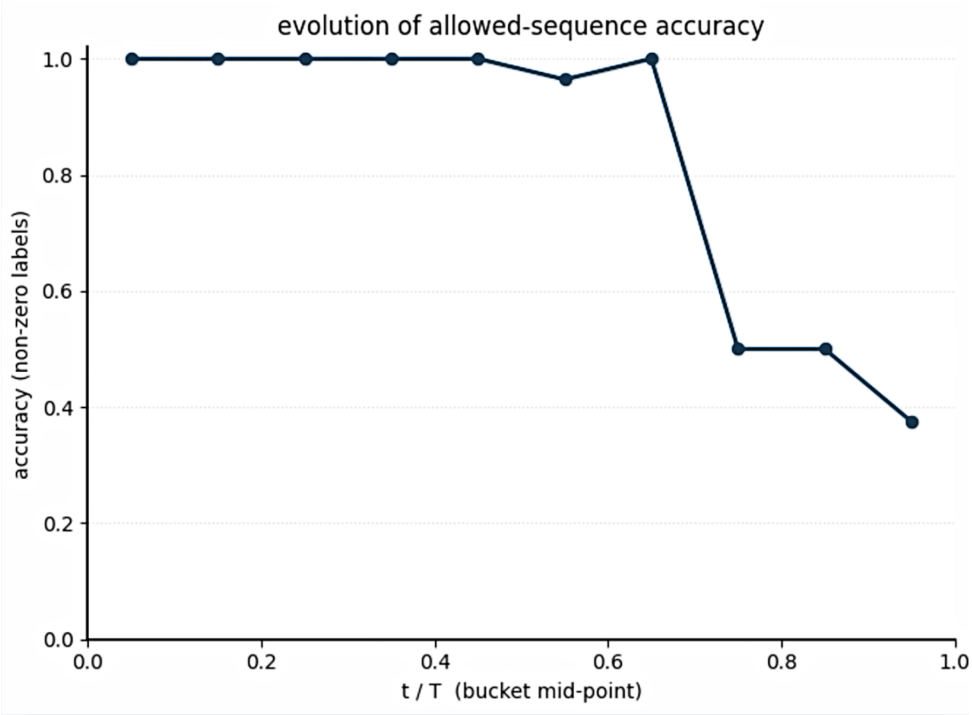


Figure 8: Fraction of denoised sequences which share the same label as the original input: Data gets noised until time t, then the trained model predicts the clean input which gets fed in the reverse diffusion process until timestep 0. We also display the Speciation time $t\_S = 0.82$

A perfectly trained model should reach an accuracy $T = 1$ of 0.5, where the model samples sequences from pure Gaussian noise data uniformly among the two allowed labels. The plot exhibits a sharp loss around the speciation time of the data, where, empirically, we see the phase where the data loses all information from the original input.

7.2 Variable-branch Grammar. Memorization Phase

We train the architecture on a total of 3840 sequences, keeping a test set of size 960 as described in section 5.3

Under memorization, we do not expect perfect recovery at every noise level—only for modest corruption levels where the clean signal still dominates. Figure X visualizes the cosine noise schedule on this grammar's one-hot sequences of length 8 over a 6-symbol alphabet.

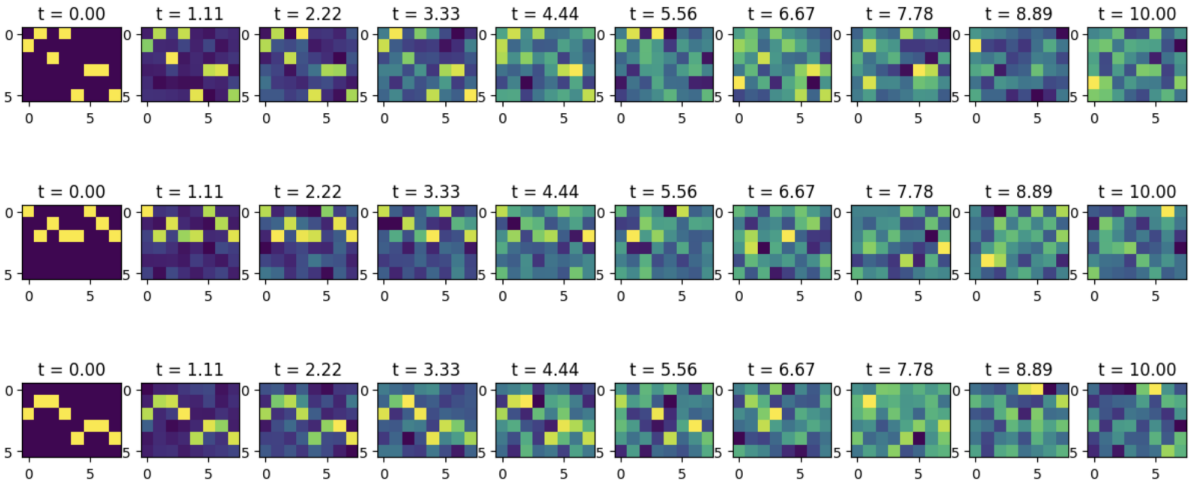


Figure 9: Noising process for the variable-branch grammar ($s = 2$, $|\Sigma| = 6$, length 8), with a cosine schedule.

For this setting we predict a speciation time.

$$t_S \;=\; \tfrac{1}{2}\ln\Lambda \;=\; \tfrac{1}{2}\ln(4.57) = 0.76$$

Where Λ is the top eigenvalue of the one-hot covariance.

Training proceeds exactly as in the binary case (AdamW, warm-up, early stopping, Cosine Annealing scheduler, and cross-entropy with label smoothing). The loss curves in Figure Y show rapid descent after a couple of plateaus, indicating the model indeed memorizes all 960 sequences.

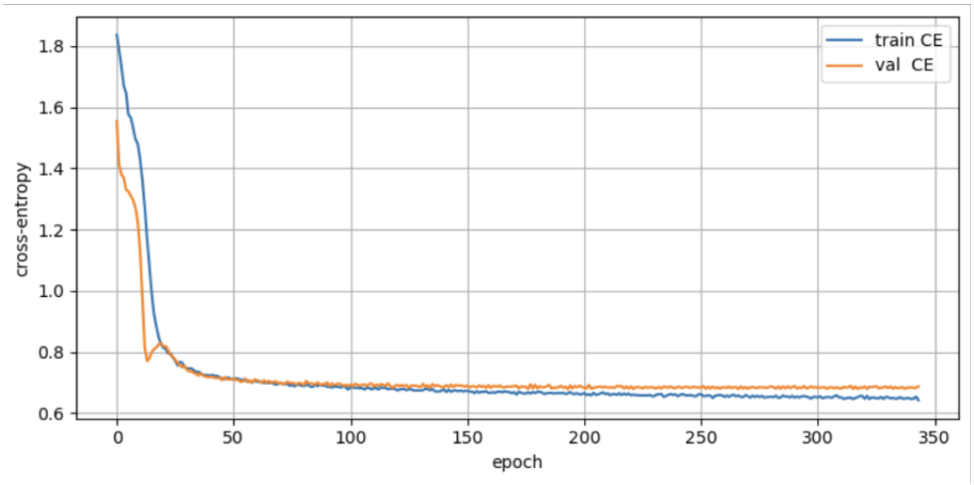


Figure 10: Training and validation cross-entropy for the variable-branch grammar in the memorization regime.

To confirm memorization, we once again ran the forward process on each held-out example to time t, ran the reverse diffusion chain back to $t = 0$, reconstructed the label

bottom-up, and checked whether the original label was recovered. Figure 10 plots the fraction of correctly reconstructed labels versus t. As expected, accuracy remains near 1.0 for $t < t_S$, then it exhibits a sharp drop around the specification time and decays until $T = 1$.

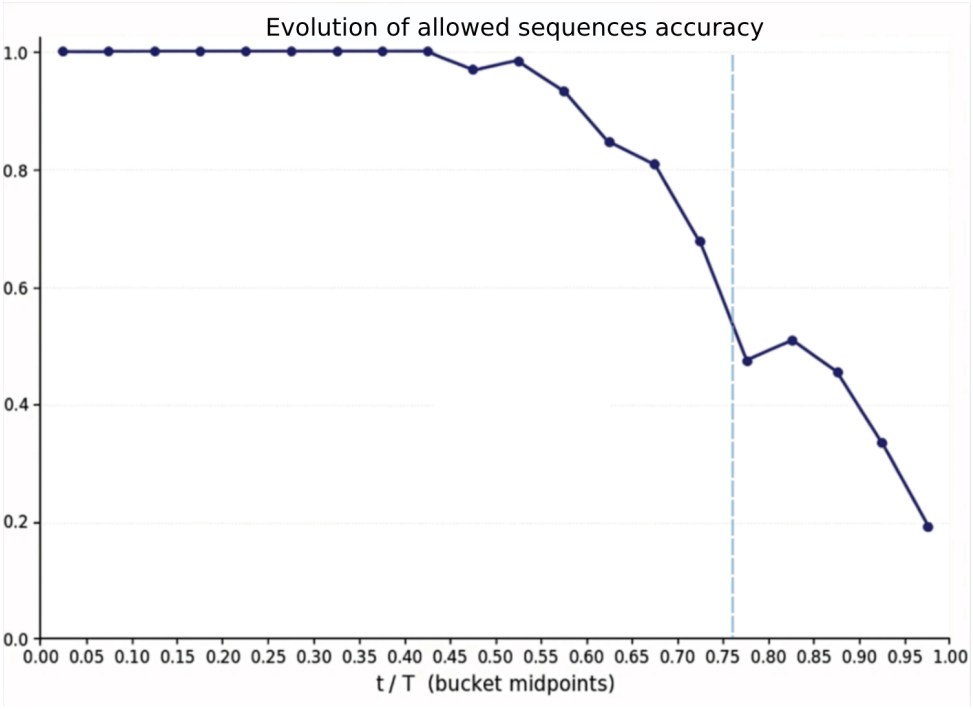


Figure 11: Label recovery rate on held-out sequences under increasing noise for the variable-branch grammar.

A perfectly trained model should reach a $T = 1$ accuracy of 0.33 since this grammar has 3 labels, each with the same number of synonyms at any level l. This suggests an imperfection in training, where the model still reconstructs a portion of the prohibited sequences. This is likely explained by the almost negligible portion of the whole space covered by the possible sequences, and by the reduced sample size of the training. All in all, we predict this issue stems from the limited number of training and testing samples.

7.3 Motivation and Interest

The motivation of this study is to highlight how reconstruction accuracy varies for different grammars, which range in complexity and, thus, stability under noise. Notably, [Cagnetta et al., 2024] conducts a thorough research in a deep tree, where it is possible to informatively keep track of the evolution of the internal state of the model, with its performance varying more smoothly.

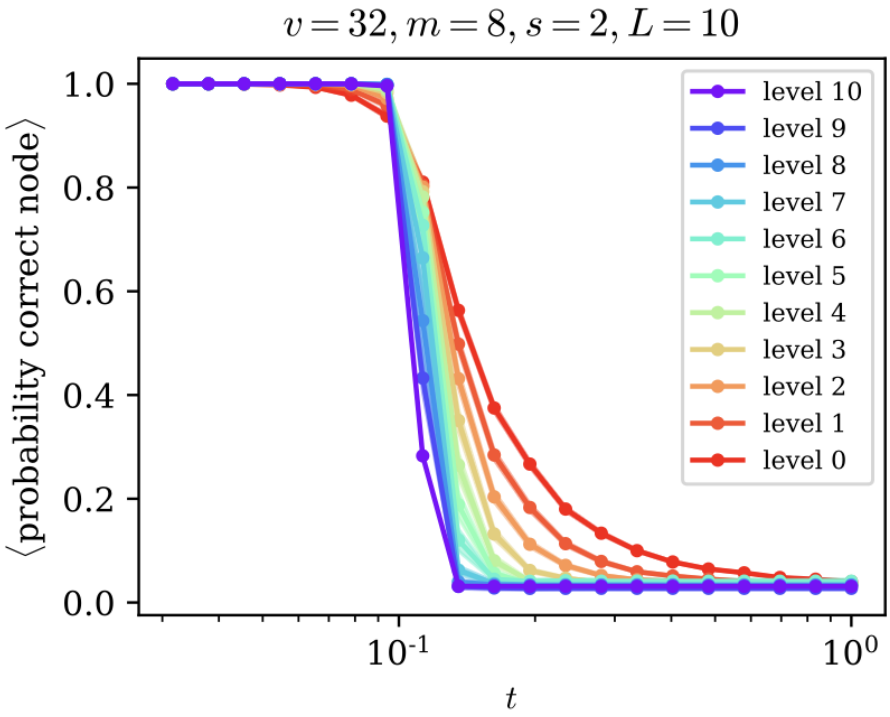


Figure 12: (Label recovery rate on held-out sequences under increasing noise for the variable-branch grammar, from [Cagnetta et al., 2024].

Figure 12 illustrates the layerwise reconstruction accuracy. Shallow layers exhibit a gradual decline in accuracy as noise levels increase. In contrast, deeper layers undergo a steep drop, culminating in a phase transition between perfect reconstruction and random behavior.

Importantly, these accuracy curves are not *empirical* estimates derived from a trained neural network but *exact* probabilities computed by the *Belief Propagation* algorithm [Mossel et al., 2016]. Belief propagation leverages the grammar's tree structure to concatenate conditional probabilities and propagate them in a bottom-up pass. Consequently, training a model in this setting corresponds to learning the best possible approximation of these exact belief-propagation predictions.

7.4 The Key Result

The computational limitations of this study's setting necessitate smaller RHM-generated datasets, implying that the corresponding architectures must also be tuned down to match the problem's scale. By observing only the model's logits (and inferring labels via context-free grammar decoding), we uncover a pattern. In the simpler binary grammar—replicating the setup of [Cagnetta et al., 2024]—the phase transition at the logits level is abrupt and mimics the mentioned paper's results: below the critical noise threshold, the network behaves as a near-perfect denoiser, whereas beyond that threshold it

effectively samples uniformly over the allowed sequences (see *Figure 8*). In contrast, the variable-branching grammar exhibits a smoother transition: between the extremes of perfect reconstruction and random sampling lies an intermediate regime of gradual logit decay (see *Figure 11*). This suggests that the grammar's branches fundamentally influence the stability of transformer-based denoising under increasing noise perturbations.

8 Learning to Generalize

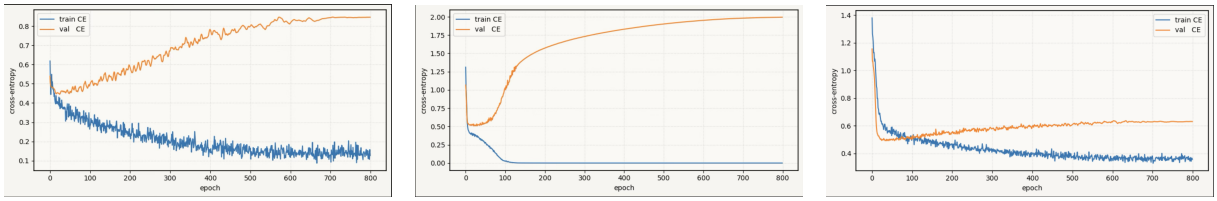
One of the transformer's primary features is its ability to encode hierarchical structure via attention heads. To assess whether this inductive bias yields true generalization rather than mere memorization, we train on a restricted subset of valid sequences and evaluate on a held-out set that is twice as large as the full grammar.

We begin with two hypotheses:

- when trained on a fraction of the grammar's sequences, the model should still perfectly denoise its training set at low to moderate corruption levels, recovering clean one-hot representations;
- Moreover, a correctly trained oracle will reconstruct novel valid strings it has never seen while rejecting sequences that violate the grammar.

For this experiment, we deliberately constrain the training set while maintaining an extensive test set size to ensure reliable metrics. Specifically, to minimize overlap and prevent data leakage, we generate exactly n allowed sequences (256 for the binary grammar and 960 for the variable-branch grammar) and split each set evenly into training and test halves. This approach emphasizes the model's generalization capabilities while maintaining an informative evaluation across varying noise levels.

Interestingly, the observed behaviour is different than the memorization case: Using the same Transformer model implemented for memorization, the model does not learn to denoise at any rate of corruption, but only to generate one-hot encoded data sampled from the space of possible s-characters' strings of length l. After scaling down the architecture with proper hyperparameter tuning, the training can escape plateaus and output one-hot-encoded data; however, it never enters a phase where both training and test losses decrease simultaneously. Any successful escape from a stall phase in training coincides with the beginning of overfitting, which appears to be unavoidable across all recorded training attempts.



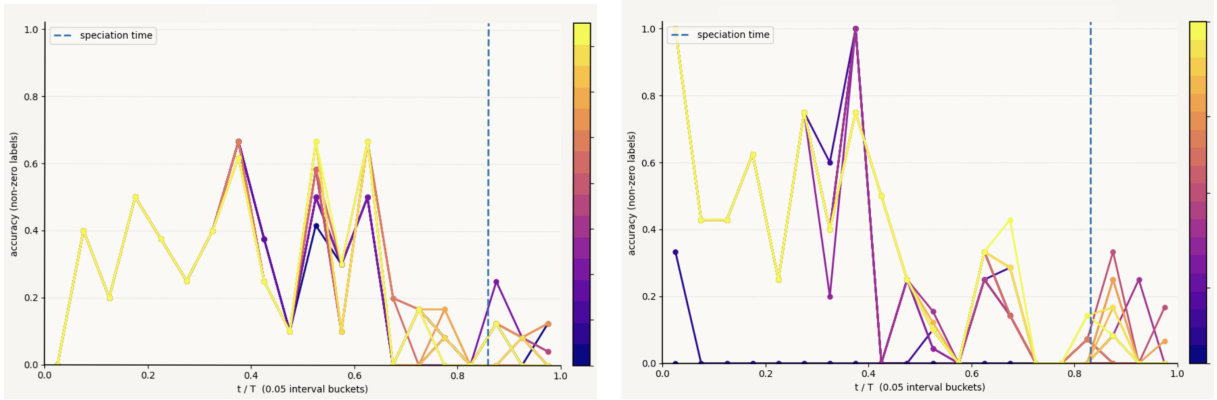
(a) (b) (c)

Figure 13: Training in the generalization phase. (a) Full-size Transformer model; (b) Half-size embedding and feed-forward dimension for the binary ruleset; (c) Half-size embedding and feed-forward dimension for the Variable-Branch ruleset.

The model's behavior under generalization training diverges sharply from the previously examined case. In the overfitting region, it still manages to reconstruct one-hot sequences from pure noise.

Still, the fraction of valid outputs converges to the background density of allowed sequences in the whole length space—about 3.9% for the binary grammar and 0.3% for the variable-branch grammar. As a result, at t = 1, the reverse diffusion effectively samples uniformly at random, showing no bias toward the seen examples.

At intermediate noise levels, reconstruction accuracy becomes erratic: the model cannot reliably recover lightly corrupted inputs near 100% as it did in the memorization setting. *Figure 14* displays this chaotic behaviour, exploiting the lack of significant difference between the rule sets in this regime.



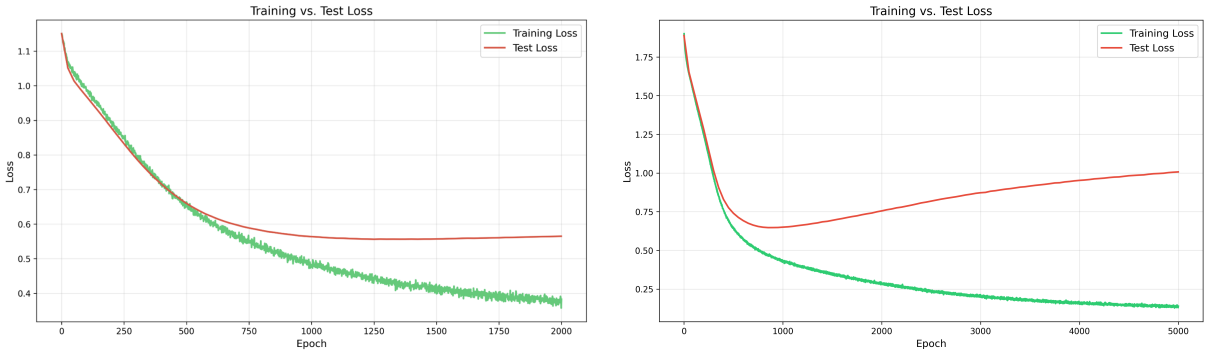
(a) (b)

Figure 14: Evolution of the percentage of correctly reconstructed labels per timestep bucket over epochs. (a) Binary case; (b) Variable-Branch case.

8.1 External validation and Observation

To validate the correctness of this implementation, given its underwhelming behavior, we replicate this setting in a more robust and well-established project [Moscato, 2025]. After adapting the project's pipeline to the seeded data used in previous experiments and

copying the seeds of all previously cited experiments, we replicate the architecture and hyperparameters, then train on the two datasets.



(a) (b)

Figure 15: Training and Testing Accuracy in [Moscato, 2025] implementation of RHM diffusion; (a) Binary case; (b) Variable-Branch case.

Training reveals a lack of a generalization phase, where the two tracked losses decay simultaneously.

Both of our initial hypotheses have been disproven in this restricted scenario. We formulate two possible explanations:

- The first is that the training pipeline remains suboptimal, both in its hyperparameter configuration and in the Transformer's architectural design.
- The second is that the convergence is fundamentally impeded by the low number of samples in this low-dimensional regime.

Despite exhaustive hyperparameter sweeps, experiments with varying inner dimensionality, and application of overfitting countermeasures, the model's training and denoising behaviors changed only marginally. We do not exclude the possibility that further optimization may yield improvements in both combating overfitting and achieving healthier loss drops. However, a more compelling interpretation is that a fundamental lower bound on sample size prevents the model from ever reaching the desired phase of generalization. While the formulation and derivation of denoising diffusion processes are untied from input dimensionality, the performance of machine learning models depends critically on the dimensionality of their data. This discrepancy may be a crucial factor in the convergence behavior of our pipeline.
Moreover, the extreme sparsity of the allowed sequence space, occupying only a tiny fraction of the whole combinatorial domain, possibly imposes a hard lower bound on sample complexity.

The model must therefore both memorize the sequences it has seen and infer the grammar's latent structure in the form of a "drive" term that guides diffusion back onto the valid data manifold (see Section 4.6). In such a hostile setting, this task may be untractable.

9 Conclusions and Outlook

9.1 Key achievements

This work focused on developing a compact but complete sandbox for studying hierarchical data through the lens of denoising diffusion. By The Random Hierarchy Model to a Transformer-based denoiser, we verified that diffusion can faithfully reproduce training examples in a strictly controlled CFG setting and, in the process, reproduced the survival–speciation–drift transition predicted by recent theory. Although the models did not attain complete generalisation, the experiments clarified how branching topology and sample size jointly govern reconstruction behaviour. These findings, modest in scale, nonetheless provide a transparent foundation on which deeper or more expressive Follow-up studies could be built.

9.2 Limitations

While this study provides valuable insights into the interaction between hierarchical grammars and diffusion models, several methodological and practical constraints warrant careful consideration. These limitations not only bound the interpretation of our current results but also suggest important directions for future research to build upon this foundation. We summarize the limitations encountered in this study:

1. **Shallow Tree Structures:** Our experiments were necessarily confined to grammar depths of $L3$ due to computational constraints. This shallow hierarchy fails to capture the full complexity of real-world hierarchical systems, where deeper trees (L 10) are common in applications ranging from natural language syntax parsing to computer vision object decomposition.

2. **Restricted Vocabulary Spaces:** The small terminal alphabet sizes (|s| 6) used in our experiments create an artificially sparse data manifold that may distort several aspects of the learning process. On one hand, the limited vocabulary makes memorization deceptively easy, as the model can store all valid sequences with relatively few parameters. On the other hand, the extreme sparsity, together with the low dimensionality (with valid sequences sometimes representing $< 0.01\%$ of possible combinations), may exaggerate the challenges of generalization.

3. **Model Capacity Constraints:** To match our relatively small-scale datasets, we employed downscaled Transformer architectures with reduced embedding dimensions and fewer attention heads. While this ensured proportional model sizes, it may have created an artificial bottleneck on the network's ability to develop rich hierarchical representations.

4. **Limited Metrics involved** Our primary metric of label recovery accuracy, while straightforward to measure, may not fully capture subtle aspects of hierarchical reconstruction. More sophisticated metrics could track: (a) layer-wise reconstruction fidelity as in Cagnetta et al., or (b) grammatical validity of partial reconstructions at intermediate noise levels.

9.3 Future directions

While the present results already reveal several intriguing phenomena, they also point to several clear avenues for further extension and deeper analysis:

Deeper and broader grammars: Scaling to larger L, richer vocabularies, or heterogeneous branching factors would help in testing whether the soft-decay regime persists, possibly enlightening where the sample-complexity threshold may lie. Inspecting the inner representation of a model across all settings, varying in structure, could lead to interesting insights about the convergence and regimes of Denoising Diffusion Probabilistic Models.

Alternative Denoisers: In a more Machine Learning fashion, it would be interesting to implement different denoising structures (Convolutional Networks, Adversarial Networks) and see how the behaviour across layers varies when considering the same grammars and sample sizes.

Analytical lower bounds: This proposal would also link well to the study of the three denoising regimes (*Section 6.4*) across various models, layers, or both simultaneously. If the predicted phase boundaries line up across the three axes there could be evidence that the regimes are universal properties of score-based recovery, not quirks of a particular architecture or dataset.

In summary, this thesis establishes a reproducible sandbox for probing the interplay between hierarchical grammar structure and diffusion-based generation. While true generalisation eluded our current scale, the methodology and diagnostic tools developed here lay the groundwork for systematic exploration of deeper trees, larger models, and refined objectives.

References

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *arXiv preprint arXiv:2402. 18491*, 2024.

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.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46 (2):255–308, 2009.

Noam Chomsky. Three models for the description of language. *Information and Control*, 2(3):227–254, 1956.

Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

Noam Chomsky. A review of b. f. skinner's behavior. *Language*, 35(1):26–58, 1959.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

Emanuele Moscato. rhm-diffusion: A diffusion-based sequence modeling repository. `https://github.com/emanuele-moscato/rhm-diffusion/tree/final`, 2025. GitHub repository; accessed 2025-06-18.

Elchanan Mossel, Joe Neeman, and Allan Sly. Belief propagation, robust reconstruction and optimal recovery of block models. *The Annals of Applied Probability*, 26(4), August 2016. ISSN 1050-5164. doi: 10.1214/15-aap1145. URL `http://dx.doi.org/10.1214/15-AAP1145`.

Hariharan Narayanan and Sanjoy K Mitter. Sample complexity of testing the manifold hypothesis. In *Advances in Neural Information Processing Systems*, volume 23, pages 1786–1794, 2010.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Advances in Neural Information Processing Systems*, volume 28, pages 2256–2264, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

Nikolaos N. Vlassis and WaiChing Sun. Denoising diffusion algorithm for inverse design of microstructures with fine-tuned nonlinear material properties. *Computer Methods in Applied Mechanics and Engineering*, 413:116126, August 2023. ISSN 0045-7825. doi: 10.1016/j.cma.2023.116126. URL `http://dx.doi.org/10.1016/j.cma.2023.116126`.

Ulrike von Luxburg and Olivier Bousquet. Distance-Based classification with lipschitz functions. *Journal of Machine Learning Research*, 5:669–695, 2004.