TREEPC: POSTERIOR-CONSISTENT AND DEPENDENCY AWARE DECODING FOR FEW-STEP DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (dLLMs) enable parallel generation by unmasking multiple tokens at each denoising step. Existing dLLM efficient decoding methods usually achieve speedups by substantially reducing decoding steps. However, aggressive step reduction forces dLLM to commit far more larger token sets at each step, thereby degrading generation quality. We identify two primary causes of quality degradation in few-step dLLM efficient decoding: (1) ***marginal drift***, where a few-step student's per-position posteriors deviate from those of a fine-step teacher; and (2) ***dependency error***, where independent sampling ignores compatibility among tokens committed within the same step. To fill the identified research gap, we introduce TreePC, a posterior-consistent and dependency-aware decoding framework that first aligns student marginals through Posterior Consistency LoRA (PC-LoRA), then predicts pairwise dependencies from teacher counterfactuals, constructs a counterfactual-dependency maximum spanning tree, and performs parent-conditioned corrections along the tree. A learnable global scale jointly controls the correction budget with edge-specific gates, allowing TreePC to exploit informative dependencies while preserving calibrated posteriors when conditional evidence is weak. During inference, TreePC invokes dLLM backbone model exactly once per diffusion step; its dependency estimator, tree construction, and correction module are lightweight and leave backbone neural function evaluations unchanged. Experimental results on GSM8K, MATH-500, HumanEval, and MBPP demonstrate that TreePC improves task accuracy and reduces marginal and conditional KL divergence while bringing very modest decoding latency. These results establish structured posterior correction as an effective quality-efficiency trade-off for few-step dLLM generation. Code is available at: TREEPC.

1 INTRODUCTION

Autoregressive large language models (LLMs) generate sequences token by token under a left-to-right factorization, imposing a chain of sequentially dependent decoding rounds. In contrast, masked diffusion language models (dLLMs) iteratively reconstruct partially masked sequences and can update multiple token positions within a single denoising step, offering the potential to substantially reduce sequential decoding depth (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025). A common approach of efficient dLLM decoding is therefore to reduce the number of decoding steps and commit more tokens per step for a fixed response length (Wu et al., 2026; Israel et al., 2025a). Formally, let y_t^S be the partially observed sequence at step t, and U_t be the set of candidate positions selected for commitment, with $m_t = |U_t|$. As the total decoding budget decreases, m_t typically grows, requiring more token values to be determined from the same partially observed context. Conventional parallel decoding commits these tokens under a conditionally independent approximation. Specifically, conditioned on y_t^S, the joint distribution of token values at m_t candidate positions in U_t is approximated by the product of m_t position-wise marginal distributions, so the prediction at each position $i \in U_t$ does not condition on the other tokens committed in the same step. Although computationally attractive, this factorization turns reduced sequential depth into a more difficult joint decision: tokens that are individually plausible given y_t^S may not be mutually compatible when committed together.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The quality degradation caused by aggressive parallel decoding stems from two complementary sources. First, ***marginal drift*** occurs when the per-position posterior of a few-step student deviates from that of a fine-step teacher. This perspective is closely related to recent dLLM distillation methods that align compressed students with multi-step models through consistency learning or KL-based distribution matching (Sahoo et al., 2025; Zheng et al., 2026). Let $q_i^T(\cdot \mid y_s^T)$ and $p_i^S(\cdot \mid y_t^S)$ denote the teacher and student posteriors at position i under their aligned decoding states. We quantify their discrepancy by $D_{\mathrm{KL}}\big(q_i^T(\cdot \mid y_s^T) \,\|\, p_i^S(\cdot \mid y_t^S)\big)$. Second, ***dependency error*** persists even when the individual marginals are well calibrated: independently selecting high-probability tokens at multiple positions does not guarantee that their combination has high joint probability, because each prediction ignores the other tokens committed within the same step (Wu et al., 2026; Kang et al., 2026). The two errors therefore capture distinct aspects of few-step decoding. Posterior consistency asks whether each position individually agrees with the teacher, whereas dependency modeling determines whether the selected tokens are jointly compatible. Such incompatibilities arise frequently in structured generation. For example, a mathematical derivation may pair a value with an inconsistent unit, generated code may use a variable in a manner inconsistent with its definition, or a sentence may exhibit subject-verb disagreement or inconsistent coreference. Effective few-step decoding thus requires both accurate marginal posteriors and explicit modeling of same-step token dependencies.

To address these complementary errors, we propose TreePC, a two-stage framework that first calibrates marginal posteriors and then models dependencies among tokens committed in parallel. To mitigate marginal drift, Posterior Consistency LoRA (PC-LoRA) aligns the few-step student's per-position distributions with those of a fine-step teacher, building on the effectiveness of consistency and distribution matching for few-step diffusion language modeling (Sahoo et al., 2025; Zheng et al., 2026). Starting from these calibrated marginals, TreePC addresses dependency error through structured same-step decoding. Motivated by recent evidence that parallel decoding can violate inter-token dependencies (Wu et al., 2026; Kang et al., 2026), a lightweight Dependency Head learns pairwise influences from teacher-generated counterfactuals that reveal a candidate parent token while leaving the child position masked. The predicted scores define a dense weighted graph over m_t candidate positions, which is then reduced to a maximum spanning tree with $(m_t - 1)$ edges. After selecting a root and orienting the tree, a Conditional Correction Head proceeds from root to leaves and uses each committed parent token to correct its child's posterior. TreePC thereby approximates the same-step joint posterior using a tree-structured factorization, where the root follows the calibrated base posterior and every non-root token is conditioned on its selected parent. It is noticeable that such parent-to-child computation is confined to lightweight auxiliary modules. dLLM backbone is evaluated only once per diffusion step, allowing TreePC to model same-step dependencies without additional backbone forward passes. To summarize, our contributions are:

- We introduce TreePC, a unified framework that decomposes the quality degradation of few-step dLLM decoding into two primary sources: **Marginal Drift** and **Dependency Error**, and addresses them through marginal calibration followed by structured correction.

- To mitigate Marginal Drift, we introduce Posterior Consistency LoRA (PC-LoRA), a parameter-efficient adaptation method that aligns the few-step student's per-position marginal posteriors with those of a fine-step teacher.

- To address Dependency Error, we introduce counterfactual dependency supervision and a lightweight Dependency Head for estimating pairwise dependencies among tokens committed within the same step. TreePC compresses the resulting dense dependency graph into a maximum spanning tree and applies parent-conditioned corrections along the tree. A Conditional Correction Head jointly learns a correction scale and instance-specific local gates to control correction magnitude.

- We conduct comprehensive experiments covering downstream task quality, posterior fidelity, dependency recovery, conditional correction effectiveness, and online inference efficiency. Through comparisons under matched decoding budgets, and component-wise ablation studies, we quantify the contribution of each component and identify its limitations.

Table 1: Essential notation used in TreePC. A complete notation table is provided in Appendix C.3.

Notation	Description
$\theta_0,\ \pi_\eta,\ \eta$	Fixed dLLM backbone, the few-step decoder built on it, and its trainable params.
B, B_T, N_{bb}	Student and Teacher decoding budgets; The number of backbone forward passes at each student step.
y_t^S	Partially generated sequence state at diffusion step t.
$U_t,\ m_t$	Candidate positions committed at step t and their number, where $m_t = \lvert U_t \rvert$.
$q_i^T,\ p_i^S$	Teacher posteriors at position i, student posteriors at position i.
$\overline{\mathcal{D}}_{\text{marg}},\ \overline{\mathcal{D}}_{\text{dep}}$	Expected marginal posterior drift and parallel dependency error.
$h_i,\ H_{U_t}$	Hidden state at position i and the hidden-state matrix over U_t.
$w_{ij}^T,\ \widehat{W}_t$	Teacher dependency target for pair (i, j) and the predicted dependency matrix.
$\mathcal{T}_t^\star,\ r,\ \text{pa}_t(i)$	Predicted dependency tree, its root, and the parent of position i.
$\ell_i, \ell_i', r_{ij}, g_{ij}$	Base and corrected logits, logit residual, and local gate.

2 PRELIMINARIES

Marginal posterior drift. Marginal posterior drift refers to the discrepancy between a few-step student's per-position posterior and that of an aligned fine-step teacher, with its formal definition and further discussion deferred to Appendix C.1.

Parallel dependency error. It refers to the error induced by independently sampling the tokens committed within the same diffusion step despite their conditional dependencies, with its formal formulation and theoretical analysis deferred to Appendix C.2 and Proposition G.1.

Problem Definition. Let π_η denote a few-step decoder with trainable parameters η built on a fixed dLLM backbone θ_0. Given a decoding budget B, at each step t the decoder observes state y_t^S and commits tokens at candidate positions U_t; a fine-step teacher with budget $B_T > B$ is available only for offline supervision. We seek to reduce the following complementary objectives:

$$\begin{aligned} \underset{\eta}{\text{minimize}} \quad & \big(\overline{\mathcal{D}}_{\text{marg}}(\eta), \overline{\mathcal{D}}_{\text{dep}}(\eta)\big) \\ \text{subject to} \quad & \theta = \theta_0, \quad N_{\text{bb}}(\pi_\eta, t) = 1, \quad \forall t \in \{0, \dots, B-1\} \end{aligned} \tag{1}$$

where $\overline{\mathcal{D}}_{\text{marg}}(\eta)$ denotes the expected KL divergence between the decoder's per-position posteriors and the aligned fine-step teacher posteriors, and $\overline{\mathcal{D}}_{\text{dep}}(\eta)$ is the expected KL divergence between the decoder-induced same-step joint distribution over U_t and the ideal joint posterior. N_{bb} denotes the number of dLLM backbone forward passes performed by decoder π_η at diffusion step t, which in this work is restricted to exactly one per step.

3 POSTERIOR-CONSISTENT DEPENDENCY-AWARE FEW-STEP DECODING

Figure 1 summarizes the TreePC framework. PC-LoRA first calibrates the few-step student's marginal posteriors (Section 3.1); counterfactual parent reveals then provide teacher targets for dependency estimation and conditional correction (Section 3.2). A lightweight Dependency Head predicts pairwise dependency scores (Section 3.3), which are compressed into an oriented maximum spanning tree (Section 3.4). The Conditional Correction Head subsequently updates child logits from root to leaves along this tree (Section 3.5). Finally, Sections 3.6 and 3.7 describe TreePC's training procedure and its online decoding with one backbone forward per step, respectively.

3.1 POSTERIOR CONSISTENCY LORA

We first mitigate marginal posterior drift by aligning the few-step student with a fine-step teacher. Both models share the same pretrained dLLM backbone. We freeze its parameters θ and optimize only LoRA parameters ϕ, enabling parameter-efficient adaptation to aggressive decoding budgets.

PC-LoRA follows a two-stage training procedure. We first perform an *off-policy coarse-fine warm-up* using cached teacher trajectories. Coarse and fine states are aligned according to normalized decoding progress and mask compatibility, retaining only positions that remain masked and comparable in both states. This stage provides a stable initialization for posterior alignment.

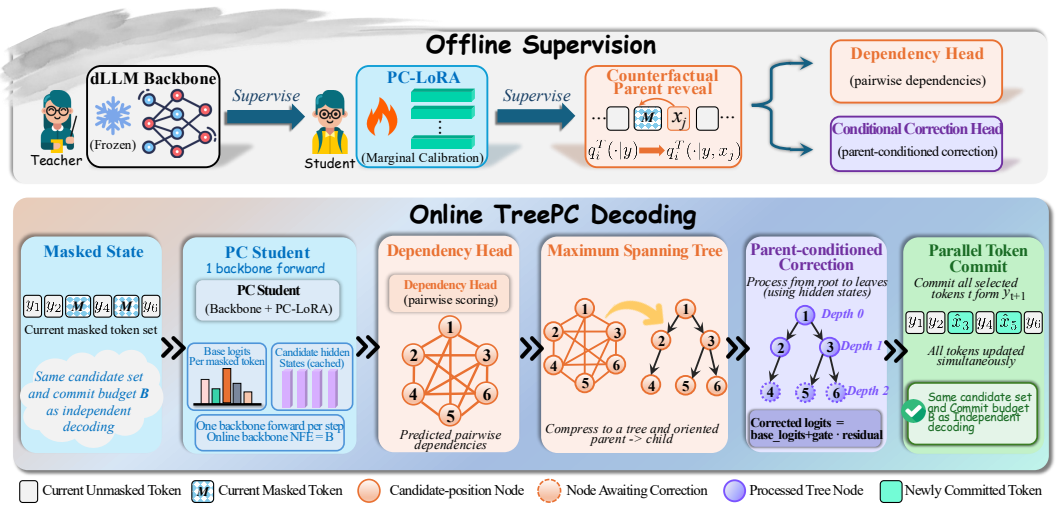


Figure 1: Framework Overview of the proposed TreePC.

The main stage performs *mixed-budget on-policy training*. We sample a student budget from B and generate rollouts with the current student policy. For each student state y_t^S, the frozen teacher either evaluates the same partial state or performs a few refinement steps to obtain an aligned supervision state $y_{s(t)}^T$. Let A_t be the positions that remain masked and comparable in both states. Across both stages, PC-LoRA minimizes the teacher-to-student posterior divergence over these valid positions:

$$\mathcal{L}_{\mathrm{PC}}(\phi) = \mathbb{E}_{t, i \in A_t}\left[D_{\mathrm{KL}}\left(q_i^T\left(\cdot \mid y_{s(t)}^T\right) \,\middle\|\, p_i^S\left(\cdot \mid y_t^S\right)\right)\right] \tag{2}$$

Here, q_i^T and p_i^S denote the teacher and student posteriors at position i, respectively. Only ϕ is updated, while the backbone and teacher remain frozen.

PC-LoRA therefore calibrates the student's per-position marginal posteriors, but does not model dependencies among tokens committed within the same step. The following sections address this remaining dependency error through structured correction.

3.2 Counterfactual Dependency Supervision

PC-LoRA calibrates the per-position marginal posteriors but does not provide direct supervision for dependencies among tokens committed within the same decoding step. After completing PC-LoRA training, we collect states visited by the resulting PC student under its own decoding policy. For each on-policy state y_t^S, the frozen teacher either operates directly on that state or performs a small number of fine-grained refinement steps to obtain a mask-compatible supervision state $y = y_{s(t)}^T$. We construct all counterfactual labels from these final PC-student states, avoiding a distribution mismatch between PC-LoRA and the auxiliary heads.

Consider two distinct candidate positions j and i, interpreted as a prospective parent and child, respectively. We draw S plausible parent tokens $x_j^{(r)} \sim q_j^T(\cdot|y)$ for $r = \{1, ..., S\}$. For each sample, we construct a counterfactual state $y^{j=x_j^{(r)}}$ by revealing $x_j^{(k)}$ at position j while keeping position i masked. We then measure how strongly this controlled parent reveal changes the teacher's posterior at the child position. The resulting directed dependency target is:

$$\widehat{I}_{j\to i}^T(y) = \frac{1}{S}\sum_{r=1}^{S} D_{\mathrm{KL}}\left(q_i^T\left(\cdot \mid y^{j=x_j^{(r)}}\right) \,\middle\|\, q_i^T(\cdot \mid y)\right) \tag{3}$$

If the teacher's marginals and parent-reveal conditionals arise from the same joint posterior, the population counterpart $\widehat{I}_{j\to i}^T(y)$ equals the symmetric conditional mutual information $I_T(X_i; X_j \mid y)$ (Proposition G.2). In real implementation, finite sampling and posterior approximations can make the estimated directions differ. We therefore symmetrize them:

$$w_{ij}^T(y) = \frac{1}{2}\left(\widehat{I}_{j\to i}^T(y) + \widehat{I}_{i\to j}^T(y)\right) \tag{4}$$

A large w_{ij}^T indicates that revealing a token at either position i or j substantially changes the teacher posterior at the other, suggesting an important conditional dependency between the two positions.

3.3 Dependency Head

Counterfactual parent-reveal supervision requires multiple teacher evaluations and is therefore unsuitable for online decoding. We instead train a lightweight Dependency Head, parameterized by ψ, to predict pairwise dependencies directly from the PC student's hidden states. Given candidate positions U with $m = |U|$, one student backbone forward produces $H_U = [h_i]_{i \in U} \in \mathbb{R}^{m \times d}$. The Dependency Head maps these representations to:

$$\widehat{W} = D_\psi(H_U, t) \in \mathbb{R}^{m \times m}_{\geq 0} \tag{5}$$

For each pair (i, j), a shared scorer predicts a nonnegative raw score $\widetilde{w}_{ij} = \mathrm{softplus}(\mathrm{MLP}_\psi(f_{ij}))$, which is symmetrized as $\widehat{w}_{ij} = (\widetilde{w}_{ij} + \widetilde{w}_{ji})/2$, with $\widehat{w}_{ii} = 0$. Consequently, $\widehat{W}$ is symmetric with zero diagonal, and each $\widehat{w}_{ij}$ estimates the counterfactual target w_{ij}^T defined in Section 3.2. The pairwise feature construction and architectural details are provided in Appendix E.

We train the Dependency Head with magnitude regression, pairwise ranking, and symmetry regularization. Regression calibrates the predicted dependency strengths, while ranking preserves their relative ordering. The latter is particularly important because the maximum spanning tree in Section 3.4 depends primarily on the ordering of edge weights rather than their absolute values. Full definitions of the training objectives are deferred to Appendix E.

3.4 Tree Construction via Maximum Spanning Tree

Given the predicted dependency matrix $\widehat{W}$, we construct a complete weighted graph over the candidate positions U and extract its maximum spanning tree:

$$\mathcal{T}^\star = \arg\max_{\mathcal{T} \in \mathfrak{T}(U)} \sum_{(i,j) \in E(\mathcal{T})} \widehat{w}_{ij} \tag{6}$$

where $\mathfrak{T}(U)$ is the set of spanning trees over U. The resulting tree retains the (m-1) edges with the greatest total predicted dependency weight, yielding a sparse and acyclic approximation of the dense dependency graph. It captures the strongest pairwise dependencies but does not explicitly represent discarded edges or higher-order interactions. With coherent teacher conditionals and exact mutual-information weights, this construction yields the KL-optimal tree approximation when its factors match the reference marginals and conditionals (Theorem G.3).

Because $\mathcal{T}^\star$ is initially undirected, we select the candidate position with the highest calibrated confidence as its root:

$$r = \arg\max_{i \in U} \max_{v \in \mathcal{V}} p_i^S(v \mid y_t^S), \tag{7}$$

where $\mathcal{V}$ denotes the vocabulary. Starting from r, breadth-first traversal orients the tree and determines the parent $\mathrm{pa}(i)$ and depth κ_i of each node. The root token is sampled from its calibrated base posterior, while every non-root token is subsequently corrected using its selected parent. Nodes at the same depth can be processed in parallel because they depend only on earlier depths.

3.5 Conditional Correction Head

After orienting the dependency tree, the Conditional Correction Head uses each selected parent token to update its non-root child. It predicts a parent-conditioned residual together with local and global gates that control the correction magnitude, allowing the calibrated base posterior to remain largely unchanged when conditional evidence is weak.

Conditional Residual Direction. For parent j, child i, and selected parent token x_j, we construct a conditional feature and map it through a low-rank transformation:

$$z_{ij} = [h_i; h_j; e(x_j); e_{\mathrm{rel}}(i-j); e_t]\,,\ \Delta h_{ij} = W_{\mathrm{up}}\mathrm{GELU}(W_{\mathrm{down}} z_{ij})\,,\ r_{ij}(v) = e(v)^\top \Delta h_{ij}/\sqrt{d}$$

Here, the feature combines the child and parent hidden states, parent-token embedding, relative position, and diffusion timestep. The resulting residual $r_{ij}(v)$ specifies how the logit of candidate token v should change conditioned on the selected parent.

Local Gate and Learnable Global Scale. Because not every tree edge warrants the same correction, an edge-specific feature z_{ij}^g summarizes the predicted dependency strength, hidden-state similarity, relative distance, and timestep. The local gate is $g_{ij} = \sigma(\mathrm{MLP}_g(z_{ij}^g))$. We further introduce a learnable global scale $l = \sigma(a) \in (0, 1)$, yielding the corrected child logits:

$$\ell_i'(v) = \ell_i(v) + \sigma(a) g_{ij} r_{ij}(v) \tag{8}$$

Thus, r_{ij} determines the correction direction, while g_{ij} and $\sigma(a)$ control its local and global correction magnitudes, respectively.

Correction Training Objective. The supervision target is the teacher conditional posterior after revealing the parent token $q_i^T(\cdot \mid y, x_j)$. By defining the effective correction as: $\delta_{ij} = \sigma(a) g_{ij} r_{ij}$, we can formulate the correction objective as:

$$\mathcal{L}_{\mathrm{corr}} = \mathbb{E}_{(i,j)} \left[D_{\mathrm{KL}} \left(q_i^T(\cdot \mid y, x_j) \,\middle\|\, \mathrm{softmax}\left(\ell_i + \delta_{ij}\right) \right) + \lambda_\delta \left\| \delta_{ij} \right\|_2^2 \right] \tag{9}$$

Regularizing the effective correction avoids scale compensation between the residual and gates and directly limits perturbations to the calibrated posterior. Weak-dependency identity pairs further discourage unnecessary corrections. Complete gate features, posterior-support construction, scale initialization, and additional training details are provided in the Appendix.

3.6 Training Strategy

TreePC is trained sequentially with stage-wise parameter freezing, such that each stage optimizes only its designated module while all previously trained components and the backbone remain fixed.

Stage A: PC-LoRA training. We first warm up PC-LoRA on cached coarse-fine teacher state pairs and then perform mixed-budget on-policy posterior alignment using rollouts from the current student. The resulting PC-LoRA parameters ϕ are subsequently frozen. **Stage B: On-policy supervision construction.** Using the final PC student, we regenerate on-policy states and apply the frozen teacher to construct counterfactual edge targets w_{ij}^T and conditional child posteriors. This ensures that the auxiliary heads are trained on states induced by the calibrated student. **Stage C: Dependency Head training.** The Dependency Head parameters ψ are trained on the cached edge targets using regression and ranking objectives, producing dependency scores suitable for maximum spanning tree construction. **Stage D: Conditional Correction Head training.** Finally, the Correction Head parameters ω and global-scale parameter a are trained against the cached conditional teacher posteriors, first using oracle parent-child pairs and then pairs selected by the learned dependency tree to match the distribution encountered at inference.

3.7 Online TreePC Decoding

Algorithm 1 shows how TreePC conducts online inference after training. At each diffusion step, TreePC performs one PC-student backbone evaluation, constructs a dependency tree over the positions selected by the baseline scheduler, and applies conditional corrections from root to leaves. Tokens at the same tree depth are corrected in parallel, and all selected tokens are written back to the dLLM state simultaneously. TreePC follows a fixed-budget decoding protocol: inference runs for the prescribed NFE budget B, without a learned or confidence-based early-stopping rule.

The root-to-leaf traversal introduces only a logical conditional ordering within each diffusion step. All corrections use cached backbone representations and lightweight auxiliary modules. Therefore, no additional dLLM backbone evaluations are required. Children at the same depth remain conditionally independent given previously selected ancestors and can be processed in parallel.

4 Experimental Evaluation

We aim to answer the following research questions. ***RQ1*:** Can PC-LoRA reduce Marginal Posterior Drift and improve task quality under a fixed NFE budget? ***RQ2*:** Can the counterfactually supervised Dependency Head recover useful edge rankings and dependency-tree structures? ***RQ3*:** Does TreePC improve same-step token compatibility and downstream correctness over other decoding? ***RQ4*:** What latency and memory overhead does TreePC introduce at each decoding step?

4.1 Baselines and Model Variants

We use (I) vanilla LLaDA-8B-Instruct and (II) Dream-7B-Instruct as backbone baselines. We further compare TreePC with: (III) Fast-dLLM, a training-free acceleration method that can be applied to existing backbones. (IV) CD4LM, which combines consistency distillation with adaptive decoding. We align the backbone, prompting format, generation length, evaluation protocol, and hardware whenever supported. Fixed-budget methods are compared at matched NFE, whereas adaptive methods are evaluated using their realized average NFE and task performance.

4.2 Datasets and Evaluation Metrics

Datasets and Tasks. (1) Mathematical reasoning: We evaluate on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021) benchmarks, reporting *exact-match accuracy* as the primary evaluation metric. (2) Code generation: We evaluate on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) benchmarks, reporting *pass@1* as the primary evaluation metric.

Mechanism Evaluation and Efficiency Metrics. We evaluate posterior calibration using Student-Teacher marginal KL and Teacher top-1 agreement. Correction quality is evaluated using conditional KL to the parent-reveal Teacher posterior and the improved-pair rate. Dependency modeling is assessed by Spearman rank correlation, edge overlap, and captured Oracle dependency weight. For comparisons with external decoding baselines, task quality is reported together with the realized average NFE. At matched task quality, efficiency is measured by average NFE, end-to-end decoding latency, and speedup relative to vanilla decoding, while per-step latency, output throughput, and peak GPU memory serve as complementary overhead metrics.

4.3 Implementation Details

Experiments are conducted using four NVIDIA RTX PRO 6000 GPUs with 96GB of memory each. Trajectory collection, counterfactual labeling, and final evaluation are sharded across the four GPUs, whereas PC-LoRA and the two lightweight heads are optimized on a single GPU. We use BF16 computation with PyTorch 2.11.0, CUDA 12.8. We construct a fixed prompt-level manifest using predetermined random seeds. Training, validation, and test prompts are mutually disjoint. Teacher targets computed for test prompts are used only for held-out evaluation. The Teacher uses $B_T \in \{256, 512\}$ decoding steps, from which we retain four aligned states per prompt, while students are evaluated at $B \in \{4, 8, 16, 32, 48, 64, 80, 96, 112\}$. PC-LoRA uses rank 16 and adapts the query, key, value, and output projections. Dependency Head uses a 128-dim projection, and Correction Head uses rank 32 with Top-K=16 posterior support and three counterfactual parent samples. For same-NFE comparisons, all methods share the same prompts, position scheduler, and random seeds.

4.4 Experimental Results and Analysis

RQ1: Posterior Calibration with PC-LoRA. PC-LoRA substantially improves the agreement between the compressed student and its fine step teacher, and the improvement transfers to task quality. Figure 2 (a) shows that mean marginal KL decreases by 66.8% for Dream and 64.5% for LLaDA. Figure 2 (b) further shows that teacher top 1 agreement rises from 63.7% to 90.3% for Dream and from 65.2% to 87.5% for LLaDA. The simultaneous improvement in distributional divergence and token decisions indicates that PC-LoRA corrects posterior drift rather than merely changing prediction confidence. The task curves in Figure 2 (c) and Figure 2 (d) confirm this effect across decoding budgets. At 64 NFE, PC only decoding improves GSM8K accuracy from 51.3% to 67.4% for Dream and from 48.4% to 65.9% for LLaDA. HumanEval pass@1 increases from 36.0% to 55.0% for Dream and from 34.0% to 47.0% for LLaDA. These results answer RQ1 positively and establish calibrated marginals as a stronger basis for dependency correction.

RQ2: Recovery of Pairwise Dependency Structure. The Dependency Head recovers both useful edge rankings and informative tree structure from cached student representations. Across the four benchmarks and two backbones, Figure 3 (a) reports Spearman correlations from 0.74 to 0.85 with the teacher dependency targets. The corresponding maximum spanning trees recover between 73% and 83% of the oracle tree edges. Since tree construction depends primarily on the ordering of edge

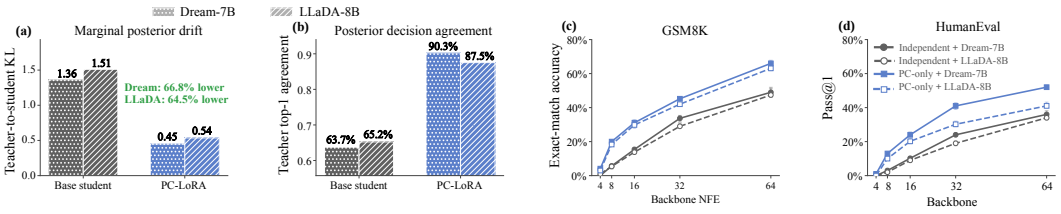


Figure 2: (a) Marginal posterior drift, (b) Posterior decision agreement, (c) and (d) downstream task results on GSM8K and HumanEval.

weights, the strong rank correlation directly supports the learned selection of dependency edges. Figure 3 (b) provides a stricter structural test. Learned trees retain between 75% and 92% of the oracle dependency weight, whereas local trees retain between 42% and 56% and random trees retain between 13% and 28%. The consistent margin over both controls shows that the head captures task relevant relations that cannot be explained by token proximity alone. RQ2 is therefore answered positively for both mathematical reasoning and code generation.

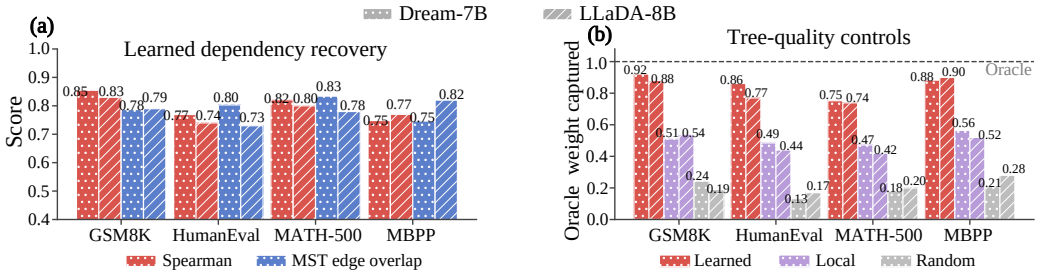


Figure 3: Dependency recovery results. The learned scores are evaluated through rank correlation, maximum spanning tree edge recovery, and captured oracle dependency weight.

RQ3: Effectiveness of Parent-Conditioned Correction. Parent conditioned correction consistently moves child posteriors toward conditional teacher targets. Figure 4 (a) shows conditional KL reductions from 58.8% to 70.4% across all backbone and dataset combinations. The correction improves between 68.0% and 85.2% of evaluated parent and child pairs, indicating useful conditional updates across tasks rather than isolated cases. Figures 4 (b) and (c) further show that full TreePC consistently exceeds independent decoding and PC-only decoding from 8 to 128 NFEs. Table 2 shows comparison with external baselines under similar NFE settings. TreePC achieves the best score on all four benchmarks. This advantage persists despite the fixed and adaptive NFE policies used by the competing baseline models. For example, at 64 NFE, TreePC + Dream backbone gains 72.5 on GSM8K, 39.8 on MATH-500, 55.2 on HumanEval, and 59.6 on MBPP. These consistent gains across posterior, pair, and sequence level evaluations support the premise of RQ3: calibrated marginals and structured conditional correction address complementary error sources.

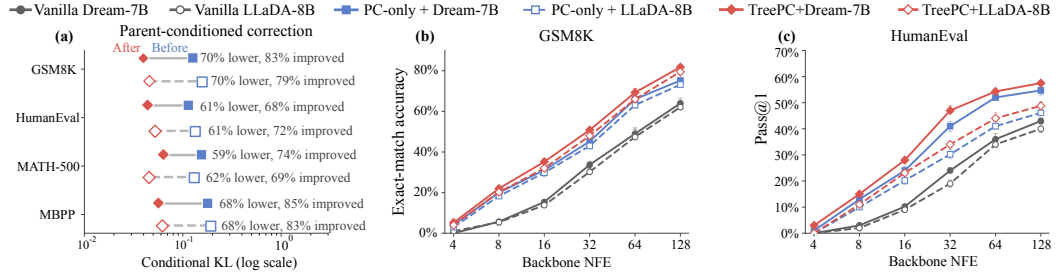


Figure 4: Performance of conditional correction and downstream quality. TreePC is compared with independent and PC-only decoding across both backbones.

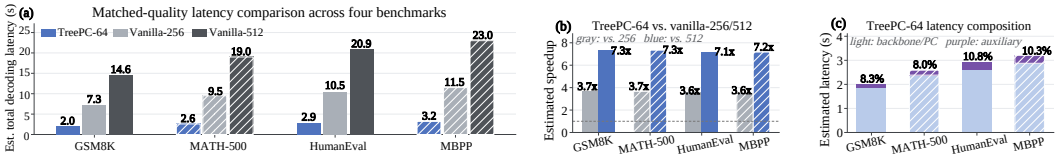


Figure 5: (a) Decoding latency at matched task quality across four benchmarks. (b) and (c) TreePC with 64 backbone evaluations is compared with vanilla decoding at 256 and 512 evaluations.

RQ4: Decoding Efficiency and Inference Overhead. TreePC substantially reduces total decoding latency when methods are compared at matched task quality. Figure 5 compares TreePC at 64 NFE with vanilla decoding at 256 and 512 NFE across all four benchmarks. TreePC requires 2.00 to 3.21 seconds per generation, whereas vanilla decoding requires 7.31 to 11.50 seconds at 256 NFE and 14.62 to 23.01 seconds at 512 NFE. This corresponds to speedups from 3.57 to 3.67 times over vanilla decoding at 256 NFE and from 7.14 to 7.34 times over vanilla decoding at 512 NFE. Figure 5 (c) further shows that the auxiliary modules account for only 8.0% to 10.8% of the estimated TreePC latency. The reduction in backbone evaluations therefore dominates the cost of dependency prediction, tree construction, and conditional correction. These results answer RQ4 at the system level: the auxiliary computation is worthwhile when it enables a 64 step decoder to reach the task quality of substantially finer schedules. The estimates use measured anchors for GSM8K and HumanEval and task family proxies for MATH-500 and MBPP, so they characterize the expected scaling trend rather than direct wall clock measurements. Additional efficiency experiments are in Appendix A.1.

Table 2: Task quality comparison under similar decoding NFE settings. The *score / realized average NFE* metric is reported.

Method	Training	NFE policy	GSM8K Acc. / NFE	MATH-500 Acc. / NFE	HumanEval pass@1 / NFE	MBPP pass@1 / NFE
LLaDA-8B-Instruct						
Vanilla few-step	Training-free	Fixed	47.4 / 64	26.3 / 64	34.0 / 64	22.5 / 64
Fast-dLLM	Training-free	Adaptive	60.6 / 63.7	21.7 / 65.2	28.2 / 68.4	30.6 / 62.8
CD4LM	Distillation	Adaptive	62.5 / 66.3	25.8 / 68.3	33.1 / 63.2	32.9 / 65.6
TreePC	Stage training	Fixed	**67.6** / 64	**32.5** / 64	**47.3** / 64	**38.6** / 64
Dream-7B-Instruct						
Vanilla few-step	None	Fixed	49.7 / 64	16.7 / 64	36.0 / 64	25.0 / 64
Fast-dLLM	None	Adaptive	60.9 / 63.5	28.4 / 69.1	46.3 / 67.8	49.5 / 62.7
CD4LM	Distillation	Adaptive	66.3 / 62.8	33.6 / 63.5	52.5 / 62.0	55.3 / 66.3
TreePC	Stage training	Fixed	**72.5** / 64	**39.8** / 64	**55.2** / 64	**59.6** / 64

Notes: Methods are evaluated within an NFE range of 64±4. For adaptive methods, NFE denotes the measured average number of backbone evaluations.

5 Conclusion

This work introduced TreePC, a posterior-consistent and dependency-aware framework for mitigating two complementary failure modes of few-step diffusion language model decoding: marginal posterior drift and incompatibility among tokens committed in parallel. TreePC first uses Posterior Consistency LoRA (PC-LoRA) to align a few-step student with a fine-step teacher. It then derives pairwise supervision through counterfactual parent reveals, predicts dependency strengths from cached student representations, compresses the resulting graph into a maximum spanning tree, and applies gated parent-conditioned logit corrections from root to leaves. This design preserves one dLLM backbone forward pass per diffusion step while introducing structured dependence through lightweight auxiliary modules. Experiments on representative mathematical reasoning and code generation benchmarks support the proposed approach: PC-LoRA improves marginal posterior alignment, the Dependency Head recovers meaningful teacher-induced dependency structure, and parent-conditioned correction moves child posteriors closer to the teacher's conditional targets. TreePC shows superior performance in final-answer accuracy, and tree construction and correction add measurable latency, particularly at very small decoding budgets. The results therefore position TreePC as a concrete framework for studying dLLM efficient few-step decoding and point to promising directions for achieving robust and consistent end-to-end improvements.

AI USE STATEMENT

Generative AI tools are used solely to assist with manuscript drafting and language editing, including improving clarity, grammar, and fluency. They are not used to formulate the research ideas or theoretical claims, design the methodology or experiments, implement the proposed method, or process and interpret the results. The authors reviewed and revised all AI-assisted text and take responsibility for the accuracy, originality, and final content of this paper.

REFERENCES

Hasan Amin, Yuan Gao, Yaser Souri, Subhojit Som, Ming Yin, Rajiv Khanna, and Xia Song. Consistent diffusion language models. *arXiv preprint arXiv:2605.00161*, 2026.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. *arXiv preprint arXiv:2509.25188*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Tianqi Chen, Shujian Zhang, and Mingyuan Zhou. DLM-One: Diffusion language models for one-step sequence generation. *arXiv preprint arXiv:2506.00290*, 2025.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968. doi: 10.1109/TIT.1968.1054142.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-1763.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion LLMs via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*, 2025b.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Koo, Nam Ik Cho, et al. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. In *International Conference on Learning Representations*, volume 2026, pp. 89900–89938, 2026.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. *arXiv preprint arXiv:2511.05664*, 2025.

Ian Li, Zilei Shao, Benjie Wang, Rose Yu, Guy Van den Broeck, and Anji Liu. Breaking the factorization barrier in diffusion language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yihao Liang, Ze Wang, Hao Chen, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Emad Barsoum, Zicheng Liu, and Niraj K. Jha. CD4LM: Consistency distillation and adaptive decoding for diffusion language models. *arXiv preprint arXiv:2601.02236*, 2026.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-Cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848. PMLR, 2024.

Xiaocheng Lu, Hualei Zhang, Shuhan Guo, Jie Zhang, Xiaoyi Pang, Jian Liu, Haoxi Li, Bohai Gu, Haoxuan Che, Jingcai Guo, and Song Guo. OPTD: On-policy transition distillation with consistency-guided adaptive compression for few-step diffusion language models. *arXiv preprint arXiv:2608.02942*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184, 2024. doi: 10.52202/079017-4135.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52584–52619. PMLR, 2025.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, 4(1):66–82, 1960. doi: 10.1147/rd.41.0066.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Chengzhi Mao, Hao Wang, Vladimir Pavlovic, and Dimitris N. Metaxas. Few-step diffusion language models via trajectory self-distillation. *arXiv preprint arXiv:2602.12262*, 2026.

Haoyang Zheng, Xinyang Liu, Cindy Xiangrui Kong, Nan Jiang, Zheyuan Hu, Weijian Luo, Wei Deng, and Guang Lin. Ultra-fast language generation via discrete diffusion divergence instruct. In *The Fourteenth International Conference on Learning Representations*, 2026.

A Additional Experimental Results

A.1 Additional decoding efficiency experiments (RQ4)

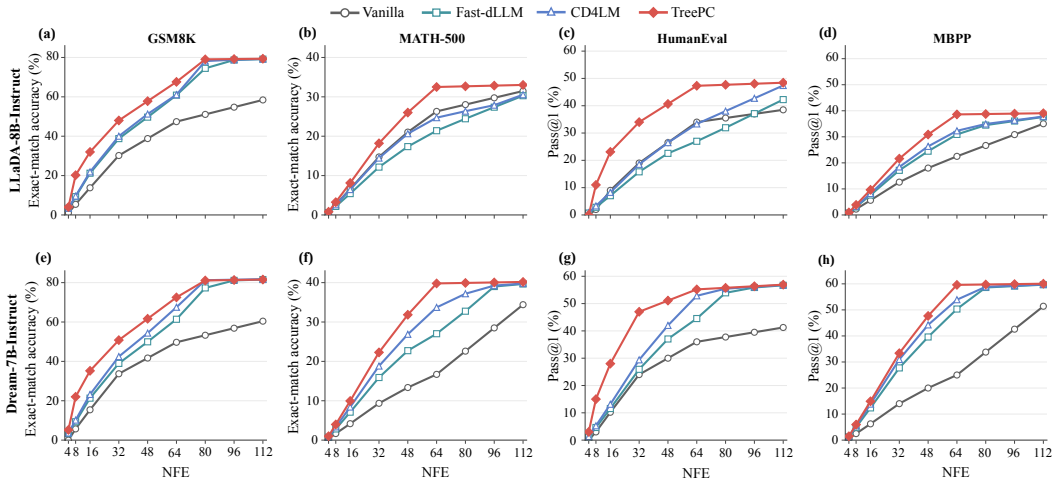


Figure 6: Task-quality versus NFE Pareto curves for competing baseline models across LLaDA-8B-Instruct and Dream-7B-Instruct backbones and four benchmarks.

Figure 6 complements the latency analysis by examining task quality across decoding budgets. TreePC uses fixed budgets $\{4, 8, 16, 32, 48, 64, 80, 96, 112\}$, whereas Fast-dLLM and CD4LM use adaptive decoding. For plotting, their realized mean NFEs across evaluation samples are mapped to these nominal budgets only when they fall within the respective tolerances $\{\pm 0.1, \pm 0.2, \pm 0.2, \pm 0.3, \pm 0.3, \pm 0.4, \pm 0.4, \pm 0.5, \pm 0.5\}$. These coordinates therefore represent approximately matched computation rather than identical realized NFE. TreePC generally achieves higher quality at comparable NFE across both backbones and all four benchmarks, particularly at low to intermediate budgets. On Dream-7B-Instruct, TreePC reaches approximately 40% accuracy on MATH-500 at 64 NFE, whereas Fast-dLLM and CD4LM approach this level at larger budgets. On LLaDA-8B-Instruct, TreePC attains approximately 47% pass@1 on HumanEval at 64 NFE, comparable to CD4LM near 112 NFE. At larger budgets, gaps narrow on GSM8K and several Dream-based tasks as competing methods approach similar performance. These trends support TreePC's ability to preserve quality under aggressive step reduction, while indicating that its advantage varies with task and budget.

B Related Work

B.1 Large-Scale Diffusion Language Modeling

Discrete diffusion language models generate sequences through iterative denoising rather than left-to-right factorization, creating the possibility of updating multiple token positions in parallel. Recent advances have substantially improved the modeling quality and scalability of this paradigm. SEDD formulates discrete diffusion through ratio estimation and demonstrates a favorable trade-off between sample quality and neural function evaluations (Lou et al., 2024). Masked diffusion language models simplify the training objective while supporting flexible semi-autoregressive sampling (Sahoo et al., 2024). Subsequent work scales diffusion language modeling either by adapting pretrained autoregressive models, as in DiffuLLaMA (Gong et al., 2025), or by training large-scale masked diffusion models such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025). Nevertheless, parallel prediction does not automatically translate into low-latency generation: high-quality decoding still often requires many backbone evaluations, whereas aggressively reducing the number of denoising steps forces more tokens to be committed in each iteration and amplifies errors caused by premature or incompatible predictions.

B.2 Efficient dLLM Decoding

Existing efficient dLLM decoding approaches can be broadly organized into three categories: (1) Adaptive token commitment and remasking; (2) Computation reuse across denoising steps; and (3) Dependency-aware parallel decoding.

Adaptive token commitment and remasking. This approach improves the allocation of a fixed decoding budget by determining which positions are sufficiently reliable to commit at each step. ReMDM permits previously decoded tokens to be remasked, restoring iterative correction when early commitments prove unreliable (Wang et al., 2025). Fast-dLLM uses a confidence threshold to reveal multiple high-confidence tokens in parallel (Wu et al., 2026), while KLASS measures the stability of successive token distributions through KL divergence and commits predictions that remain stable across iterations (Kim et al., 2025). Moving beyond manually designed confidence rules, Learn2PD trains a lightweight filter to predict whether each token is ready to be committed and additionally detects when decoding can terminate early (Bao et al., 2025). These methods primarily improve the choice and timing of token commitments without changing the underlying factorized token posterior.

Computation reuse across denoising steps. The second approach reduces the cost of each decoding step by reusing intermediate representations. Fast-dLLM introduces an approximate block-wise KV cache for bidirectional dLLMs, allowing historical representations to be reused alongside confidence-aware parallel decoding (Wu et al., 2026). dLLM-Cache instead exploits the observation that most token representations vary only modestly between adjacent denoising steps and selectively refreshes cached features according to their measured stability (Liu et al., 2025). Such caching methods are complementary to reducing the number of decoding steps: they lower the computation required by individual backbone evaluations but do not directly correct the errors introduced by committing many tokens simultaneously.

Dependency-aware parallel decoding. The third family addresses the conditional-independence approximation underlying simultaneous token prediction. Adaptive Parallel Decoding combines dLLM marginals with sequence-level scores from a small auxiliary autoregressive model and dynamically adjusts the degree of parallel commitment (Israel et al., 2025b). CoDD replaces the fully factorized output distribution with a lightweight probabilistic inference layer that captures dependencies among concurrently generated tokens (Li et al., 2026). TreePC takes a different approach by learning same-step dependencies from teacher-supervised counterfactual influence, compressing the resulting dependency graph into a maximum spanning tree, and performing parent-conditioned posterior correction along the inferred tree. It preserves the candidate set and commit budget of the corresponding independent decoder, thereby separating dependency correction from position scheduling, while requiring no additional dLLM backbone evaluations.

B.3 Distillation and Posterior Consistency for Few-Step dLLM Decoding

Training-based acceleration seeks to adapt diffusion language models directly to large denoising transitions rather than relying exclusively on inference-time scheduling. The most representative methods are: (1) Teacher-Student trajectory distillation, (2) Consistency-based few-step learning.

Teacher-student distillation. These methods train a student model to replace several fine-grained teacher updates with one or a few transitions. For continuous diffusion language models, DLM-One aligns a one-step student's outputs with the score function of a pretrained multi-step teacher in continuous embedding space (Chen et al., 2025). For masked dLLMs, trajectory self-distillation trains a few-step student to imitate trajectories generated by the same model under a substantially finer decoding schedule (Zhang et al., 2026). OPTD further addresses the distribution mismatch of offline trajectory supervision by collecting states from the few-step student's own policy and using a frozen teacher to determine how aggressively future transitions can be compressed (Lu et al., 2026). These approaches transfer multi-step generative behavior at the level of scores, state transitions, or complete trajectories.

Consistency-based few-step learning. This approach encourages predictions to remain consistent across different points or paths of the diffusion process. The Diffusion Duality introduces discrete consistency distillation for uniform-state diffusion, enabling a student to approximate multi-step generation with substantially fewer evaluations (Sahoo et al., 2025). CD4LM combines discrete-space consistency distillation with confidence-adaptive decoding, training the model to produce stable predictions under long denoising jumps (Liang et al., 2026). Consistent Diffusion Language Models formulate discrete consistency through stochastic posterior bridges and train a path-invariant denoiser without requiring a separately pretrained teacher (Amin et al., 2026). Collectively, these methods improve few-step decoding by aligning predictions across diffusion states rather than supervising only the final generated sequence.

C Preliminaries

C.1 Marginal Posterior Drift

Let y_t^S denote the few-step student state at decoding step t, and let y_s^T be its aligned state from a fine-step teacher. Because the teacher and student follow different decoding schedules, their partially generated sequences may not contain the same masked positions. We therefore define A_t as the set of valid aligned positions that remain masked and comparable in both states. For each $i \in A_t$, let $q_i^T(\cdot \mid y_s^T)$ and $p_i^S(\cdot \mid y_t^S)$ denote the teacher and student posterior distributions over the vocabulary, respectively. We define the average marginal posterior drift as:

$$\mathcal{D}_{\text{marg}} = \frac{1}{|A_t|} \sum_{i \in A_t} D_{\text{KL}}\big(q_i^T(\cdot \mid y_s^T) \,\big\|\, p_i^S(\cdot \mid y_t^S)\big)\,. \tag{10}$$

A larger $\mathcal{D}_{\text{marg}}$ indicates that the few-step student assigns insufficient probability mass to tokens supported by the fine-step teacher. Comparing only final tokens or top-1 predictions is inadequate for characterizing this discrepancy: two posteriors may produce the same most likely token while differing substantially in confidence, uncertainty, and the probability assigned to plausible alternatives.

C.2 Parallel Dependency Error

Consider a partially observed state y and a set U of positions to be committed within the same decoding step. Let $X_U = \{X_i : i \in U\}$ denote their token-valued random variables. Conventional parallel decoding samples from the per-position marginals under the conditionally independent approximation:

$$P_{\text{ind}}(X_U \mid y) = \prod_{i \in U} p_i^S(x_i \mid y) \tag{11}$$

Let $P^*(X_U \mid y)$ denote the unknown ideal joint posterior over these positions. We conceptually define the parallel dependency error as:

$$\mathcal{D}_{\text{dep}} = D_{\text{KL}}(P^*(X_U \mid y) \,\|\, P_{\text{ind}}(X_U \mid y)) \tag{12}$$

This error can remain large even when every marginal posterior is individually accurate, since tokens with high marginal probabilities need not form a jointly compatible assignment. Directly estimating $\mathcal{D}_{\text{dep}}$ is intractable because $P^*(X_U \mid y)$ is unknown and its support grows exponentially with $|U|$.

C.3 NOTATIONS

Table 3: Core notation used throughout TreePC. Superscripts T and S denote teacher and student quantities, respectively.

Notation	Meaning
θ, θ_0	dLLM backbone parameters and their fixed pretrained value; the backbone remains frozen as $\theta = \theta_0$ during TreePC training.
π_η, η	Few-step decoder and its trainable parameters $\eta = (\phi, \psi, \omega, a)$, optimized in separate stages.
ϕ, ψ, ω, a	Components of η: ϕ, ψ, ω are parameters of PC-LoRA, the Dependency Head, and the Conditional Correction Head, respectively; a is the trainable scalar parameter of the global correction scale.
B, B_T	Online decoding budgets of the few-step student and fine-step teacher, with $B < B_T$.
t, i, j, v	Diffusion-step index, child position, parent position, and candidate vocabulary token, respectively.
y_t^S	Partially generated sequence state at diffusion step t.
$y_{s(t)}^T$	Fine-step teacher state aligned with student step t.
U_t, m_t	Candidate positions committed at step t and their number $m_t = \lvert U_t \rvert$.
A_t	Positions that remain masked and comparable in the aligned student and teacher states.
X_{U_t}	Token-valued random variables at the candidate positions U_t.
$\mathcal{V}, L, d$	Vocabulary, sequence length, and backbone hidden dimension.
q_i^T, p_i^S	Teacher posterior and PC-LoRA student posterior at position i.
p_i^S	Calibrated student posterior before tree-conditioned correction.
P_{ind}, P^*	Conditionally independent same-step distribution and the unknown ideal joint posterior, respectively.
$\mathcal{D}_{\text{marg}}, \mathcal{D}_{\text{dep}}$	Marginal posterior drift and parallel dependency error at a decoding state.
$\overline{\mathcal{D}}_{\text{marg}}(\eta), \overline{\mathcal{D}}_{\text{dep}}(\eta)$	Corresponding errors averaged over decoding states and diffusion steps.
$N_{\text{bb}}(\pi_\eta, t)$	Number of dLLM backbone forward passes performed by decoder π_η at step t.
$S, x_j^{(k)}, y^{j=x_j^{(k)}}$	Number of sampled parent tokens, the k-th parent-token sample, and the counterfactual state obtained by revealing that sample at position j.
$\widehat{I}_{j\to i}^T, w_{ij}^T$	Teacher-directed dependency target and its symmetric pairwise counterfactual dependency target.
h_i, H_U	Hidden state at position i and the matrix of hidden states at all candidate positions.
u_i, f_{ij}	Projected node representation and pairwise feature used by the Dependency Head.
$\widehat{w}_{ij}, \widehat{W}$	Predicted dependency weight for pair (i, j) and the resulting symmetric dependency matrix.
$\mathcal{T}^\star, \mathfrak{T}(U), E(\mathcal{T})$	Maximum spanning tree, set of all spanning trees over U, and edge set of a tree.
$r, \text{pa}(i), \kappa_i$	Selected tree root, parent of node i, and depth of node i in the oriented tree.
$e(x), e_{\text{rel}}, e_t$	Token embedding, relative-position embedding, and diffusion-step embedding.
$z_{ij}, \Delta h_{ij}, r_{ij}(v)$	Parent-conditioned feature, predicted hidden-state residual, and resulting logit residual for token v.
z_{ij}^g, g_{ij}	Edge-specific gate feature and local correction gate.
$l = \sigma(a), \delta_{ij}$	Global correction scale constrained to $(0, 1)$ and effective correction $\delta_{ij} = l\, g_{ij} r_{ij}$.
ℓ_i, ℓ_i'	Base and corrected logits at child position i.

Table 3 summarizes the core notation used throughout TreePC, covering teacher and student posteriors, dependency-tree construction, conditional correction, and trainable parameters. Superscripts T and S distinguish teacher and student quantities, respectively. When discussing a single diffusion step, we omit the time index from step-local quantities.

D ONLINE TREEPC DECODING ALGORITHM

Algorithm 1 Online TreePC Decoding Inference.

Require: Initial masked state y_0^S; decoding budget B; trained PC student $F_{\theta,\phi}$; Dependency Head D_ψ; Conditional Correction Head C_ω; global-scale parameter a; tree-confidence threshold τ

Ensure: Completed sequence y_B^S use scheduler $\mathcal{S}$, with all target positions committed within B steps

for $t = \{0, \dots, B-1\}$ **do**
  $(\ell_t, H_t) \leftarrow F_{\theta,\phi}(y_t^S, t)$ ▷ One backbone evaluation
  $U_t \leftarrow \mathcal{S}(y_t^S, \ell_t, t)$ ▷ Same candidate set and commit budget as the baseline
  $m_t \leftarrow |U_t|$
  $p_i^S \leftarrow \mathrm{softmax}(\ell_i)$ for all $i \in U_t$
  $\widehat{W}_t \leftarrow D_\psi(H_{U_t}, t)$
  if $m_t > 1$ **then**
    $\mathcal{T}_t^\star \leftarrow \mathrm{MST}(\widehat{W}_t)$
    $G_t \leftarrow \sum_{(i,j)\in E(\mathcal{T}_t^\star)} \widehat{w}_{ij}$
  else
    $G_t \leftarrow 0$
  end if
  if $m_t \le 1$ **or** $G_t < \tau$ **then**
    for all $i \in U_t$ **in parallel do**
      $\widehat{x}_i \sim p_i^S$
    end for ▷ PC-only fallback
  else
    $r \leftarrow \arg\max_{i\in U_t} \max_{v\in\mathcal{V}} p_i^S(v)$
    $\mathrm{pa}_t, \kappa_t \leftarrow \mathrm{Orient}(\mathcal{T}_t^\star, r)$
    $\widehat{x}_r \sim p_r^S$
    for $\kappa = 1, \dots, \max_{i\in U_t} \kappa_t(i)$ **do**
      for all $i \in U_t$ such that $\kappa_t(i) = \kappa$ **in parallel do**
        $j \leftarrow \mathrm{pa}_t(i)$
        $(g_{ij}, r_{ij}) \leftarrow C_\omega(h_i, h_j, \widehat{x}_j, \widehat{w}_{ij}, i-j, t)$
        $\ell_i' \leftarrow \ell_i + \sigma(a) g_{ij} r_{ij}$
        $\widehat{x}_i \sim \mathrm{softmax}(\ell_i')$
      end for
    end for
  end if
  $y_{t+1}^S \leftarrow \mathrm{Commit}(y_t^S, U_t, \widehat{x}_{U_t})$ ▷ Single simultaneous state update
end for
return y_B^S

Algorithm 1 summarizes TreePC inference with all learned parameters fixed. At each step, a single backbone evaluation provides the base posteriors and hidden states for the positions selected by scheduler $\mathcal{S}$. When $m_t > 1$ and the total predicted tree weight G_t meets threshold τ, decoding proceeds along the maximum spanning tree, with parent-conditioned corrections applied in parallel at each non-root depth; otherwise, tokens are sampled independently from the calibrated base posteriors. The selected tokens are committed simultaneously, and $\mathcal{S}$ ensures that all target positions are filled within the prescribed budget B.

E DETAILED DEPENDENCY HEAD

The counterfactual supervision described above is expensive to compute online because it requires multiple teacher evaluations for every candidate pair. We therefore train a lightweight Dependency Head to predict these pairwise dependencies directly from the PC student's hidden states. Given a candidate set U with $m = |U|$, a single PC student backbone forward produces the candidate

representations $H_U = [h_i]_{i \in U} \in \mathbb{R}^{m \times d}$. The Dependency Head, parameterized by ψ, maps these representations to a symmetric nonnegative edge-weight matrix $\widehat{W} \in \mathbb{R}^{m \times m}_{\geq 0}$.

To reduce pairwise computation, we first project each candidate representation into a lower-dimensional space:

$$u_i = W_p h_i, \quad W_p \in \mathbb{R}^{d_p \times d}, \quad d_p \ll d. \tag{13}$$

For each ordered pair of distinct candidate positions (i, j), we construct the pairwise feature

$$f_{ij} = [u_i; u_j; |u_i - u_j|; u_i \odot u_j; e_{\mathrm{rel}}(i-j); e_t], \tag{14}$$

where $e_{\mathrm{rel}}(i-j)$ encodes the relative position between the two candidates and e_t represents the current diffusion timestep. A shared multilayer perceptron predicts the corresponding raw edge weight: $\widetilde{w}_{ij} = \mathrm{softplus}\,(\mathrm{MLP}_\psi(f_{ij}))$ The softplus activation ensures nonnegative dependency scores. Because the raw predictions may depend on the ordering of the two input nodes, we explicitly symmetrize the output and clear its diagonal: $\widehat{W} = \frac{1}{2}\left(\widetilde{W} + \widetilde{W}^\top\right), \quad \widehat{w}_{ii} = 0$. Let $\mathcal{E}_U = \{(i,j) : i < j,\ i,j \in U\}$ denote the set of undirected candidate edges, and let w^T_{ij} be the counterfactual dependency target defined in Section 3.2. We train the Dependency Head using a combination of magnitude regression, pairwise ranking, and raw-score symmetry:

$$\begin{aligned}\mathcal{L}_{\mathrm{edge}} = \frac{\lambda_{\mathrm{reg}}}{|\mathcal{E}_U|} \sum_{(i,j)\in\mathcal{E}_U} \mathrm{Huber}\left(\widehat{w}_{ij}, w^T_{ij}\right) + \lambda_{\mathrm{rank}}\mathcal{L}_{\mathrm{rank}} \\ + \frac{\lambda_{\mathrm{sym}}}{|\mathcal{E}_U|} \sum_{(i,j)\in\mathcal{E}_U} |\widetilde{w}_{ij} - \widetilde{w}_{ji}|.\end{aligned} \tag{15}$$

To define the ranking term, let $\mathcal{R}_U$ contain sampled pairs of edges (e, e') satisfying $w^T_e > w^T_{e'}$. We use the pairwise logistic objective

$$\mathcal{L}_{\mathrm{rank}} = \frac{1}{|\mathcal{R}_U|} \sum_{(e,e')\in\mathcal{R}_U} \log\left[1 + \exp\left(-\left(\widehat{w}_e - \widehat{w}_{e'}\right)\right)\right]. \tag{16}$$

The regression term calibrates the predicted dependency magnitudes, whereas the ranking term preserves their relative ordering. This distinction is important because the subsequent maximum spanning tree is determined primarily by the ordering of edge weights rather than their absolute values. Consequently, we evaluate the Dependency Head using both magnitude-based errors and rank- or tree-recovery metrics.

F Detailed Training Strategy

TreePC is trained sequentially to ensure that each module receives supervision from the state distribution it encounters at inference time.

Stage A: PC-LoRA training. PC-LoRA adopts a two-phase training. In the initial phase, an off-policy warm up is performed using coarse-fine state pairs extracted from cached teacher trajectories. This phase accounts for approximately 20% of the PC-LoRA training updates and provides a stable initialization for posterior alignment. Then, in the second phase, we conduct mixed-budget on-policy training by uniformly sampling $B \in \{4, 8, 16, 32, 48, 64, 80, 96, 112\}$ and generating rollouts with the current student. For each visited state, the frozen teacher either provides posterior targets directly or performs a small number of fine-grained refinement steps before supervision. The rollout buffer is refreshed periodically to track the evolving student distribution. After this stage, the parameters ϕ of PC-LoRA are frozen.

Stage B: On-policy supervision construction. Using the trained PC-LoRA student model, we regenerate on-policy states and apply the frozen teacher to construct counterfactual supervision for the auxiliary heads. In particular, parent-reveal evaluations provide the edge targets w^T_{ij} for dependency estimation and the conditional posteriors required for correction. Reconstructing this cache from the final PC student avoids a distribution mismatch with states produced by the original backbone or earlier student checkpoints. The offline cache retains only information required for subsequent training,

including state tokens, mask ratios, timesteps, candidate positions, aligned hidden representations, top-K posterior probabilities.

Stage C: Dependency Head training. We train the Dependency Head parameters ψ on the cached counterfactual edge targets w_{ij}^T. Model selection is performed on held-out states using both edge-weight regression and ranking-based metrics, reflecting the dependence of maximum spanning tree construction on relative edge ordering.

Stage D: Conditional Correction Head training. Finally, we train the Correction Head parameters ω and the global-scale parameter a using the cached conditional teacher posteriors. Training begins with oracle parent-child pairs derived from teacher dependency targets, providing reliable conditional supervision. The head is then adapted at a smaller learning rate using pairs selected by the learned dependency tree, thereby matching the parent-child distribution encountered during TreePC inference.

G Theoretical Analysis of TreePC

Fix an aligned student–teacher state pair (y_t^S, y) and a nonempty candidate set $U \subseteq A_t$, with $m = |U|$. Write $p_i^S = p_i^S(\cdot \mid y_t^S)$ and $q_i^T = q_i^T(\cdot \mid y)$. Let $P^*(X_U \mid y)$ be a reference joint posterior whose marginals are q_i^T. Assume full support on the finite vocabulary, ensuring finite KL divergences. This reference joint is an idealization; neural-teacher predictions need not satisfy the parent-reveal consistency condition introduced below. The results characterize posterior approximation at a fixed state.

G.1 Exact KL Decomposition of Marginal Drift and Dependency Error

For independent decoding, let $P_{\mathrm{ind}}(X_U \mid y_t^S) = \prod_{i\in U} p_i^S(x_i)$. Define the restricted marginal drift as $\mathcal{D}_{\mathrm{marg},U} = m^{-1}\sum_{i\in U} D_{\mathrm{KL}}(q_i^T \| p_i^S)$. The pure dependence in the reference joint is its total correlation (Watanabe, 1960):

$$\mathrm{TC}_T(U \mid y) = D_{\mathrm{KL}}\left(P^*(X_U \mid y) \,\middle\|\, \prod_{i\in U} q_i^T(X_i \mid y) \right) \tag{17}$$

Proposition G.1 (Exact decomposition)**.** *Under the preceding setup,*

$$\begin{aligned} D_{\mathrm{KL}}\big(P^*(\cdot \mid y) \,\big\|\, P_{\mathrm{ind}}(\cdot \mid y_t^S)\big) &= \mathrm{TC}_T(U \mid y) + \sum_{i\in U} D_{\mathrm{KL}}(q_i^T \| p_i^S) \\ &= \mathrm{TC}_T(U \mid y) + m\mathcal{D}_{\mathrm{marg},U} \end{aligned} \tag{18}$$

Proof. Insert the product of reference marginals into the log likelihood ratio:

$$\log \frac{P^*(x_U \mid y)}{\prod_{i\in U} p_i^S(x_i)} = \log \frac{P^*(x_U \mid y)}{\prod_{i\in U} q_i^T(x_i)} + \sum_{i\in U} \log \frac{q_i^T(x_i)}{p_i^S(x_i)} \tag{19}$$

Taking expectations under $P^*(\cdot \mid y)$ yields the result, since each term in the sum depends only on X_i, whose marginal is q_i^T. □

When $U = A_t$, $\mathcal{D}_{\mathrm{marg},U}$ is exactly the marginal drift in Appendix C.1. At a fixed reference state, PC-LoRA targets the marginal term, while even perfect marginal alignment leaves $\mathrm{TC}_T(U \mid y)$ unless the reference tokens are independent. The quantity called $\mathcal{D}_{\mathrm{dep}}$ in Appendix C.2 is the full joint-to-product KL and therefore includes marginal mismatch; the pure dependency component is $\mathrm{TC}_T(U \mid y)$.

G.2 Counterfactual Dependency and Conditional Mutual Information

Assume additionally that the teacher's parent-reveal posteriors are conditionals of the same reference joint: for every distinct $i, j \in U$ and parent token x_j, $q_i^T(\cdot \mid y^{j=x_j}) = P^*(X_i = \cdot \mid y, X_j = x_j)$. This coherent-teacher assumption connects the score in Equation 3 to conditional mutual information. Here $I_T(X_i; X_j \mid y)$ denotes mutual information under the fixed conditional distribution $P^*(\cdot \mid y)$.

Proposition G.2 (Population counterfactual score)**.** *Under the coherent-teacher assumption,*

$$\mathbb{E}_{X_j \sim q_j^T}\left[D_{\mathrm{KL}}\big(q_i^T(\cdot \mid y^{j=X_j}) \,\big\|\, q_i^T(\cdot \mid y)\big)\right] = I_T(X_i; X_j \mid y) = I_T(X_j; X_i \mid y) \tag{20}$$

Proof. Let q_{ij}^T be the (i,j) marginal of $P^*(\cdot \mid y)$. The expectation equals:

$$\begin{aligned} &\sum_{x_j, x_i} q_j^T(x_j \mid y) q_i^T(x_i \mid y, x_j) \log \frac{q_i^T(x_i \mid y, x_j)}{q_i^T(x_i \mid y)} \\ &\quad = \sum_{x_i, x_j} q_{ij}^T(x_i, x_j \mid y) \log \frac{q_{ij}^T(x_i, x_j \mid y)}{q_i^T(x_i \mid y) q_j^T(x_j \mid y)} = I_T(X_i; X_j \mid y) \end{aligned} \tag{21}$$

which is unchanged when i and j are exchanged. □

With independent parent samples and exact posterior evaluation, $\widehat{I}_{j\to i}^T$ is unbiased for this mutual information and converges almost surely as $S \to \infty$; the symmetrized target w_{ij}^T has the same expectation. The population score is symmetric. Practical asymmetry in Section 3.2 can arise from finite sampling, inconsistent neural conditionals, or top-K and numerical approximations. Symmetrization produces undirected targets but cannot ensure coherence; without it, these scores remain counterfactual influence proxies.

G.3 Chow-Liu Optimality of the Maximum Spanning Tree

The mutual-information identity connects the tree construction in Section 3.4 to the classical Chow–Liu projection (Chow & Liu, 1968). Under the same coherent reference, take any spanning tree $\mathcal{T} \in \mathfrak{T}(U)$, select a root r, and orient its edges away from r. Its ideal tree distribution is:

$$Q_{\mathcal{T}}^*(x_U \mid y) = q_r^T(x_r \mid y) \prod_{i \in U \setminus \{r\}} q_i^T(x_i \mid y, x_{\mathrm{pa}(i)}) \tag{22}$$

Theorem G.3 (Chow-Liu projection)**.** *For any fixed tree $\mathcal{T}$, $Q_{\mathcal{T}}^*$ minimizes $D_{\mathrm{KL}}(P^* \| Q)$ among distributions Q factorizing over that tree, and*

$$D_{\mathrm{KL}}(P^*(\cdot \mid y) \,\|\, Q_{\mathcal{T}}^*(\cdot \mid y)) = \mathrm{TC}_T(U \mid y) - \sum_{(i,j) \in E(\mathcal{T})} I_T(X_i; X_j \mid y). \tag{23}$$

Consequently, the KL-optimal tree structure is any maximum spanning tree with edge weights $I_T(X_i; X_j \mid y)$.

Proof. Write H_T for entropy under $P^*(\cdot \mid y)$. Expanding the tree factorization gives:

$$\begin{aligned} D_{\mathrm{KL}}(P^* \| Q_{\mathcal{T}}^*) &= -H_T(X_U \mid y) + H_T(X_r \mid y) + \sum_{i \neq r} H_T(X_i \mid X_{\mathrm{pa}(i)}, y) \\ &= \sum_{i \in U} H_T(X_i \mid y) - H_T(X_U \mid y) - \sum_{i \neq r} I_T(X_i; X_{\mathrm{pa}(i)} \mid y) \end{aligned} \tag{24}$$

Summing out leaves verifies normalization, and the entropy difference equals total correlation. Replacing any ideal factor adds a nonnegative marginal or conditional KL, as shown below, proving optimality for a fixed tree. Since total correlation is independent of $\mathcal{T}$, the optimal structure maximizes the sum of edge mutual informations. □

To connect this ideal projection to TreePC, write $\widehat{p}_i^S(\cdot \mid x_j) = \mathrm{softmax}(\ell_i + \sigma(a) g_{ij} r_{ij})$, suppressing the fixed student state y_t^S; similarly, write $q_i^T(\cdot \mid x_j) = q_i^T(\cdot \mid y, x_j)$. The sampled tree distribution is $\widehat{Q}_{\mathcal{T}}(x_U \mid y_t^S) = p_r^S(x_r) \prod_{i \neq r} \widehat{p}_i^S(x_i \mid x_{\mathrm{pa}(i)})$. For a tree and root fixed before token sampling, as in Algorithm 1, the same expansion gives, with fixed state arguments suppressed:

$$\begin{aligned} D_{\mathrm{KL}}(P^* \| \widehat{Q}_{\mathcal{T}}) = \; &\mathrm{TC}_T(U \mid y) - \sum_{(i,j) \in E(\mathcal{T})} I_T(X_i; X_j \mid y) + D_{\mathrm{KL}}(q_r^T \| p_r^S) \\ &+ \sum_{i \neq r} \mathbb{E}_{X_{\mathrm{pa}(i)} \sim q_{\mathrm{pa}(i)}^T}\left[D_{\mathrm{KL}}\big(q_i^T(\cdot \mid X_{\mathrm{pa}(i)}) \,\big\|\, \widehat{p}_i^S(\cdot \mid X_{\mathrm{pa}(i)})\big)\right] \end{aligned} \tag{25}$$

The expectation uses the reference parent marginal. To verify the identity, split the log ratio as $\log(P^*/\widehat{Q}_{\mathcal{T}}) = \log(P^*/Q^*_{\mathcal{T}}) + \log(Q^*_{\mathcal{T}}/\widehat{Q}_{\mathcal{T}})$ and marginalize each factor under P^*.

The three terms separate residual dependence, root mismatch, and conditional approximation error: PC-LoRA calibrates base marginals, the Dependency Head selects edges, and the Correction Head targets conditional KL. Proposition G.1 covers PC-only fallback. Root choice leaves the ideal projection error unchanged but can affect the learned mismatch terms. Chow-Liu optimality requires coherent population mutual information and exact factors; learned weights and constrained corrections do not guarantee global KL optimality for the implemented TreePC.