π-XLNET: REVISITING XLNET FOR TEXT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In pursuit of efficient inference and flexible generation, text diffusion architectures have gradually moved closer to XLNet-style arbitrary-order models, motivating us to revisit XLNet's potential for training and generation. We find that, under certain conditions, set diffusion and XLNet are nearly equivalent in architecture. To systematically study this architecture, we propose the π-XLNet framework, which treats the training order distribution as a design variable, and introduce the Mallows distribution to continuously control the degree of training disorder. Experiments show that the additional loss introduced by switching evaluation order decays approximately as a power law as training disorder increases. The training cost for π-XLNet to reach the same natural-order loss as the autoregressive baseline is estimated at approximately 3.1–3.3 times that of the baseline, demonstrating its potential to reduce the training cost of masked text diffusion. Furthermore, we propose the HBD-n decoding algorithm, which uses local future information for stochastic branching and backtracking, translating arbitrary-order conditional prediction ability into generation gains through increased inference computation. The method improves the quality–diversity trade-off over natural-order decoding on a self-trained π-XLNet and raises GSM8K accuracy on a public SetDLM model from 64.67% with official SetDiff decoding to 69.14%.

1 INTRODUCTION

XLNet (Yang et al., 2019) is a representative arbitrary-order language model that achieved the then state of the art on multiple text-modeling metrics but reported slow convergence. For arbitrary-order language modeling, σ-GPT (Pannatier et al., 2024) adopts a single-stream architecture and dual positional encodings, while AO-GPT (Xue et al., 2025) further introduces target-position-conditioned adaptive normalization. These works combine disordered and sequential orders with curriculum learning to alleviate training difficulty, without demonstrating whether single-stream architectures outperform two-stream architectures.

Text diffusion introduced D3PM (Austin et al., 2021), connecting diffusion and arbitrary-order generation through token-wise masking, and ARDM (Hoogeboom et al., 2022), further unifying absorbing-state diffusion and arbitrary-order autoregression. However, masked diffusion requires approximately 14–16 times the training compute to match its AR baseline's validation loss (Nie et al., 2025; Sahoo et al., 2026) and also struggles to reuse KV caches.

To enable KV caching, text diffusion architectures are gradually moving closer to XLNet. Block Diffusion (Arriola et al., 2025a) and Fast-dLLM v2 (Wu et al., 2025) begin adopting computational structures close to XLNet's two-stream division of labor; E2D2 (Arriola et al., 2025b) separates context representations from denoising predictions. Set Diffusion (Arriola and Kuleshov, 2026) independently designs a new set-attention mechanism, treats the order distribution as a design variable, and achieves the state-of-the-art speed–quality trade-off among diffusion models.

We prove that set diffusion's architecture is nearly equivalent to XLNet's under certain conditions; set diffusion itself also reports its relation to arbitrary-order models. We therefore adapt this special case into an arbitrary-order model, interpreted as the π-XLNet architecture. For systematic study, we introduce a Mallows distribution centered on natural order, allowing continuous adjustment of training order between natural order and uniformly random permutations.

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.

Experiments show that π-XLNet completes training stably at relatively weak disorder, with loss gradually increasing with disorder. Cross-evaluation further shows that the additional loss from switching evaluation order decays approximately as a power law with increasing training disorder. At low disorder, the optimal data-to-parameter ratio is also close to AR's. We estimate approximately 3.1–3.3 times AR's training cost to match its natural-order test loss, showing π-XLNet's potential to substantially lower masked text diffusion's training cost.

π-XLNet can both accelerate inference and improve decoding quality. Inspired by self-speculative decoding, we design the Hindsight Branching Decoding (HBD-n) tree-search algorithm. On a public SetDLM model, HBD-n raises GSM8K accuracy from official SetDiff decoding's 64.67% to 69.14%; on a self-trained π-XLNet, it improves the quality–diversity trade-off over natural-order decoding. In another unconditional-generation experiment with a public SetDLM, HBD-n and the evaluated AR decoding configuration score 81.30 and 68.33 MAUVE ($\times 100$), respectively. Specifically, we:

1. point out that discrete text diffusion models gradually approach XLNet's architecture, find set diffusion's architecture nearly equivalent to XLNet's, and argue that XLNet's architecture itself merits further study;
2. introduce the Mallows order distribution into arbitrary-order models and systematically study how order-distribution disorder affects model training;
3. find that π-XLNet's additional loss under different evaluation orders decays approximately as a power law;
4. find that π-XLNet needs approximately 3.1–3.3 times AR's training cost to match its natural-order test loss, demonstrating the potential to substantially lower masked text diffusion's training cost;
5. propose the HBD-n tree-search algorithm and show that π-XLNet can implement a tree-search strategy yielding performance gains without additional modules.

2 π-XLNet: A Unified View of Arbitrary-Order Language Models Based on Order Distributions

Set diffusion discusses how its training objective relates to arbitrary-order modeling (Arriola and Kuleshov, 2026). This section further reveals its structural correspondence to XLNet's two-stream architecture and, through π-XLNet, makes training order distributions an explicit model-design dimension.

2.1 Structural Correspondence Between Set Attention and XLNet

XLNet uses content and query streams for arbitrary-order prediction. The content stream receives true tokens and supplies context to subsequent positions; the query stream starts from a shared representation excluding target-token content and outputs the predictive distribution. Under singleton-set, fully masked-input training, set diffusion's clean and noisy branches serve the same respective roles (Yang et al., 2019; Arriola and Kuleshov, 2026).

For a text sequence x of length T and position permutation $z = (z_1, \ldots, z_T)$, order both streams by z and concatenate content then query streams. The two models' attention masks are

$$M_{\text{XLNet}} = \begin{bmatrix} L_{\leq} & 0 \\ L_{<} & 0 \end{bmatrix}, \qquad M_{\text{Set}}^{|S_k|=1} = \begin{bmatrix} L_{\leq} & 0 \\ L_{<} & I \end{bmatrix}.$$

Rows denote query positions and columns key/value source positions; $L_{\leq}$ is a lower-triangular mask including the diagonal, $L_{<}$ a strict lower-triangular mask, and identity matrix I denotes query-stream access to KVs from its own representation.

In both models, the content stream can read content representations at current and preceding permutation positions; the query stream can read those at preceding positions. Their attention connections differ only in query-stream self-KV access. Original XLNet already propagates the query stream's

own state through a residual connection; added self-KV attention further processes existing information without expanding the visible-token range or introducing true content of the target token or later permutation tokens.

Thus, under this setting, set diffusion and XLNet have corresponding two-stream roles and token-information dependencies but different training order distributions. Appendix A gives detailed background and structural figures.

2.2 Order Distributions and the π-XLNet Training Objective

Using this structural correspondence, π-XLNet retains the two-stream roles and query-stream self-attention, directly defining training tasks by an order distribution π. Its attention mask is $M_{\pi\text{-XLNet}} = M_{\text{Set}}^{|S_k|=1}$.

Let $\mathcal{D}$ be the data distribution, θ the model parameters, $\mathfrak{S}_T$ all permutations of positions $[T] = \{1, \ldots, T\}$, and π a probability distribution over $\mathfrak{S}_T$. During training, sample permutation $z \sim \pi$ for text sequence x. The training objective is

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{z\sim\pi}\left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} \mid x_{z_{<t}}, z_t)\right].$$

Here, $x_{z_{<t}}$ denotes permutation-prefix tokens, their permutation order, and original positions; z_t specifies the current prediction position. This definition directly organizes conditional-prediction tasks by position permutations without a diffusion-time variable.

The order distribution π sets the training frequency of each conditional-prediction task. Changing π reweights different contexts with model structure fixed, enabling study of training order's effects on predictive performance and resource requirements.

We retain query-stream self-KV access because it may bring additional performance gains. Replacing I with 0 in the mask's lower-right block recovers original XLNet's attention connections and can be viewed as an attention-sparsified form of π-XLNet. Limited computation prevented multi-random-seed ablation comparisons.

2.3 Relation to XLNet, Autoregressive Modeling, and Set Diffusion

Different order distributions and attention connections correspond to the following settings.

Table 1: Model settings specified by the training order distribution and query-stream self-KV access.

Model setting	Training order distribution	Query-stream self-KV access
Original XLNet full-permutation objective	$\mathrm{Unif}(\mathfrak{S}_T)$	Disallowed
Natural-order π-XLNet	$\delta_{z_{\text{nat}}}$, where $z_{\text{nat}} = (1, \ldots, T)$	Allowed
Singleton-set, fully masked set diffusion	Induced by the position-offset schedule	Allowed

A uniform permutation distribution with sparsified attention recovers XLNet's full permutation language-modeling objective and two-stream attention connections. Original XLNet's partial-prediction strategy additionally selects positions near the permutation's end for training (Yang et al., 2019).

Concentrating the order distribution on the natural permutation yields standard AR's natural-order conditional-prediction task while retaining the two-stream structure. The set-diffusion training special case above can be expressed as a π-XLNet instance with an order distribution induced by the position-offset schedule (Arriola and Kuleshov, 2026).

To systematically study order-distribution effects, the next section introduces a Mallows distribution centered on natural order. It weights permutations by their departure from natural order and continuously controls disorder with one parameter.

3 Training Order and Scaling Behavior

We retain an XLNet-style two-stream architecture and vary training order through the Mallows distribution. We test whether moderate disorder permits conventional optimization settings and data-to-parameter ratios and scaling behavior close to those of AR, while quantifying training cost relative to AR. Experimental settings are in Appendix B.

3.1 The Mallows Order Distribution

We sample training permutations from a Mallows distribution centered on natural order. Keep unselected positions in their original order; with m positions remaining, sample a skip count K from the truncated geometric distribution

$$\Pr(K = k) = \frac{q^k}{\sum_{r=0}^{m-1} q^r}, \qquad k \in \{0, \dots, m-1\}, \quad 0 < q < 1.$$

We then select and remove the $(K+1)$-st remaining position; skipped positions stay in the list. Repetition yields a Mallows-distributed permutation (He et al., 2023).

Parameter q controls the preference for natural order. At $q = 0$, always selecting the earliest remaining position gives natural order; as q increases, later positions become easier to select; as $q \to 1$, the fixed-length permutation distribution approaches uniformity.

We use $d = \frac{q}{1-q}$ to denote the degree of disorder. Under the untruncated geometric distribution, d is the mean number of candidate positions skipped at each step. The actual number of skipped positions in a finite list is affected by truncation, whereas d provides a common comparison scale across order settings.

3.2 Training Disorder and Loss Changes

Appendix B.2 gives the experimental configuration. Figure 1 shows loss curves and final-step losses for different training disorders. All groups share the architecture, training budget, and optimization settings, varying only the training order distribution. All complete scheduled training, with decreasing losses and no divergence over the tested disorder range.

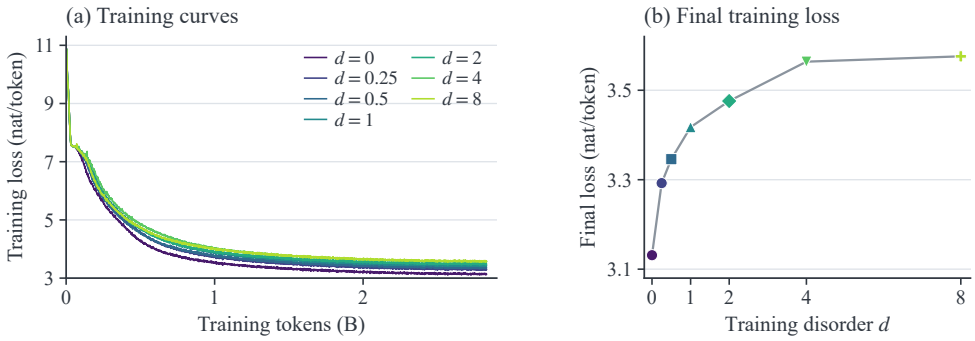


Figure 1: Effect of training disorder on π-XLNet training loss. The left panel shows unsmoothed training-loss curves; the right panel shows the loss recorded at step 2700.

Across the seven settings, final-step loss increases monotonically with disorder d, from 3.1312 at $d = 0$ to 3.5758 nat/token at $d = 8$.

At different training disorders, training losses measure different conditional-prediction tasks and cannot directly support fair performance comparisons. To compare models, we evaluate all at the same evaluation disorder and examine their response to evaluation-order changes.

3.3 Power-Law Decay of the Evaluation-Order Loss Increment

Figure 2 shows cross-evaluation on a held-out test set for models trained with different orders. At fixed evaluation disorder, total loss is not always monotonic in training disorder. For $d_{\text{eval}} = 1$, loss falls from 3.5111 at $d_{\text{train}} = 0.25$ to 3.4107 at $d_{\text{train}} = 2$, then rises to 3.4901 at $d_{\text{train}} = 8$.

Total loss is affected both by each model's natural-order baseline and by the increment induced by switching evaluation order. Let d_{train} and d_{eval} denote training and evaluation disorder, respectively. The loss increment relative to the same model's natural-order baseline is $\Delta\mathcal{L} = \mathcal{L}(d_{\text{train}}, d_{\text{eval}}) - \mathcal{L}(d_{\text{train}}, 0)$.

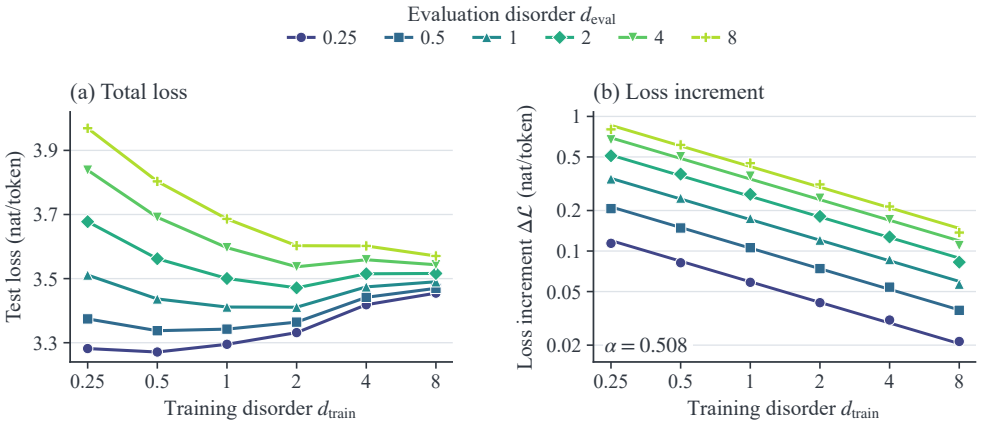


Figure 2: Cross-evaluation across training and evaluation orders. The left panel shows total test loss; the right panel shows the loss increment relative to each model's natural-order baseline on log-log axes. Colors denote evaluation disorder, and solid lines in the right panel are power-law fits with a shared exponent.

The right panel of Figure 2 shows that, at a fixed evaluation disorder, the loss increment decays approximately as a power law as training disorder increases: $\Delta\mathcal{L} \propto d_{\text{train}}^{-\alpha}, \qquad \alpha \approx 0.508.$

The six nonzero evaluation settings yield a common decay exponent close to $1/2$. This relation means that doubling training disorder lowers the additional loss under a fixed evaluation order by approximately 30%.

The joint fit's mean absolute relative error is 2.94%. Holding out a full training-disorder setting yields a mean absolute relative prediction error of 4.61% for its loss increment. The baseline uses each held-out model's measured natural-order loss; this result shows that the relation can predict evaluation-order loss increments reasonably well for training settings excluded from fitting.

Details of the fit and hold-out test are given in Appendix B.2.

3.4 Model Size and Training-Data Ratio

Let N and D denote the numbers of model parameters and training tokens, respectively. At approximately fixed ND, we compare optimal data-to-parameter ratios across training orders, evaluating all losses in natural order. Experimental settings and fitting methods are in Appendix B.3.

For each training setting, quadratic least squares fits natural-order loss against $\log(D/N)$ to obtain the optimal data-to-parameter ratio and its change relative to AR.

The Qwen series emphasizes improving small-model ability through overtraining: continuously increasing training data with a limited parameter count (Qwen Team, 2024). With the Qwen backbone and training settings used here, AR's fitted optimal D/N is 39.29, substantially above the common

reference of about 20 (Hoffmann et al., 2022). Thus, the more meaningful comparison is the change in π-XLNet's optimal ratio relative to AR under the same settings.

Table 2: Fitted optimal data-to-parameter ratios and their changes relative to AR.

Model / training disorder	Fitted optimal D/N	Change relative to AR
AR	39.29	–
π-XLNet, $d = 0$	47.92	+21.96%
π-XLNet, $d = 0.25$	43.76	+11.38%
π-XLNet, $d \approx 0.5$	41.04	+4.46%
π-XLNet, $d = 1$	41.15	+4.74%
π-XLNet, $d \approx 2$	42.43	+7.99%

Over the tested disorder range, π-XLNet's fitted optimal ratios remain close to AR's, without increasing severalfold or showing change magnitudes that increase monotonically with disorder. At $d = 1$, the fitted ratio is about 4.7% above AR's.

3.5 Scaling Behavior and Training Cost

The cross-evaluation shows that greater training disorder can reduce evaluation-order loss increments but raises natural-order loss. Considering losses across training and evaluation disorders and changes in the optimal data-to-parameter ratio relative to AR, we select $d = 1$ ($q = 0.5$) as the representative setting for the cross-scale experiment.

To stay near the common $D/N \approx 20$ training setting, we use $D/N \approx 21$ for both AR and π-XLNet. We train both classes at parameter scales of 135M, 360M, and 1B, evaluate natural-order test loss throughout, and analyze scaling as parameters and training data grow together. Table B6 in Appendix B.4 reports the configurations and numerical results.

At $D/N \approx 21$, both classes' natural-order losses decrease approximately as power laws with scale. Fitting empirical power laws by least squares in log space to the three-scale results gives

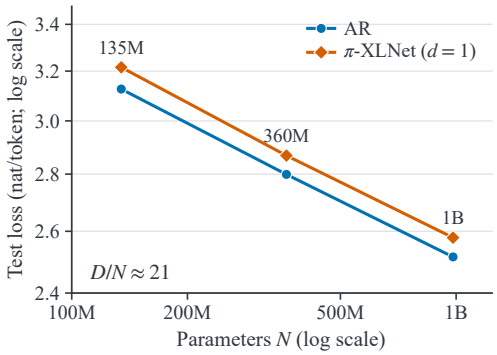


Figure 3: Natural-order test loss of AR and π-XLNet at a fixed data ratio. Both axes use logarithmic scales; markers denote measured results and lines connect the observed points. The training disorder of π-XLNet is $d = 1$.

$$\widehat{\mathcal{L}}_{\mathrm{AR}}(N) = 3.2271 \left(\frac{N}{10^8}\right)^{-0.1095},$$

$$\widehat{\mathcal{L}}_{\pi\text{-XLNet}}(N) = 3.3185 \left(\frac{N}{10^8}\right)^{-0.1111}, \qquad D/N \approx 21.$$

The corresponding loss-fit RMSEs are 0.0031 and 0.0061 nat/token. The decay exponents of the two model classes are both close to 0.11, indicating similar scaling gains along this data-ratio path.

Over their common loss range, the two fits indicate that π-XLNet needs about 25%–28% more parameters and proportionally more training tokens to match AR's natural-order loss. At 2.8 nat/token, for example, fitted parameter counts are about 366M for AR and 462M for π-XLNet, a difference of about 26%.

With 1B models, batch size 4, and original sequence length 2048, measured training step times are 361.9 ms for AR and 724.9 ms for the two-stream model, about 2.00 times as long as AR. Appendix B.4 gives fitting details and complete timing results.

At a fixed data-to-parameter ratio, π-XLNet's time-based training cost to reach the same natural-order test loss is estimated at approximately 3.1–3.3 times that of AR; Appendix B.4 explains the calculation.

Prior masked-diffusion work reports about 14–16 times the training compute to reach its respective AR baseline's validation loss (Nie et al., 2025; Sahoo et al., 2026). Its diffusion likelihood bounds, compute-optimal configurations, and compute-based measure differ from our natural-order test loss, fixed data ratio, and time-based measure; these results therefore serve as a cross-setting cost reference.

4 Decoding via Distribution Transfer

We use arbitrary-order conditional prediction to improve generation quality. Decoding is not limited to remasking and resampling generated content (Wang et al., 2025). Self-speculative decoding uses acceptance and residual correction to transfer parallel draft distributions conditioned on less prefix information to sequential generation distributions conditioned on the full prefix. Inspired by this, we extend the idea to future information: HBD-n tree search uses the sequential generation distribution as a proposal and transfers it to a pseudo-likelihood distribution with local future context.

4.1 Inspiration from Self-Speculative Decoding

Existing self-speculative decoding can be explained from a distribution-transfer perspective: draft tokens already generated are transferred from a proposal distribution to a target distribution while preserving the original tokens as often as possible. Given prefix $x_{\le s}$, for position $j > s$ and candidate token v, the proposal and target distributions are

$$d_j(v) = p_\theta(x_j = v \mid x_{\le s}), \qquad p_j(v) = p_\theta(x_j = v \mid x_{<j}).$$

Subject to the constraint that the output follows p_j, maximizing the probability of retaining the original token gives the acceptance probability and the residual sampling distribution upon rejection:

$$a_j(v) = \min\left\{1, \frac{p_j(v)}{d_j(v)}\right\}, \qquad r_j(v) \propto [p_j(v) - d_j(v)]_+,$$

where $[u]_+ = \max\{u, 0\}$.

The algorithm sequentially verifies the parallel draft, performs residual sampling at the first rejection and discards subsequent draft tokens, then continues generation from the updated prefix (Guo and Ermon, 2025).

4.2 Hindsight Branching Decoding

HBD-n is designed to improve the generation quality–diversity frontier with more inference computation. Our experiments show a concrete instance of diffusion-style conditional prediction repeatedly revising generation paths through branching and backtracking, improving quality while preserving the same empirical token entropy.

Given a finite vocabulary, a prompt $c = x_{1:m}$ of length m, and a maximum total length $L > m$ including the prompt, let nonnegative integer n denote the maximum number of future tokens used. For $m \le t \le L$, define the pseudo-likelihood-like target weight of a prefix as

$$W_n(x_{1:t}) = \prod_{j=m+1}^{t} p_\theta\big(x_j \mid x_{<j}, x_{j+1:\min(j+n,t)}\big), \qquad W_n(c) = 1.$$

For a complete sequence $x = x_{1:T}$, the target weight takes $t = T$, where $m < T \leq L$. EOS denotes the termination token; all sequence summations below range over sequences starting from prompt c and ending when EOS is first encountered or the length limit L is reached.

Using the natural-order AR distribution as the proposal, we transfer the prediction distribution toward one that uses local future context: $p_\theta(x_j \mid x_{<j}) \quad \longrightarrow \quad p_\theta\big(x_j \mid x_{<j}, x_{j+1:\min(j+n,T)}\big)$.

Existing tokens generate future context, which corrects them in turn, creating mutual conditional dependencies. When a correction fails, we undo affected tokens and the future context conditioned on them, then backtrack and expand again. If candidates were only accepted or rejected, a weight ratio above 1 would be truncated and introduce bias. Additional branches therefore make a candidate's expected copy count equal the target weight ratio.

Branch weights are computed from the model's own conditional probabilities, without requiring additional modules. At each step, the proposal probability multiplied by the mean number of expansions should equal the ratio of change in target weight, so we take

$$R(x_{1:t}) = \frac{W_n(x_{1:t})}{W_n(x_{<t})\, p_\theta(x_t \mid x_{<t})} = \prod_{j=\max(m+1,t-n)}^{t-1} \frac{p_\theta(x_j \mid x_{<j}, x_{j+1:t})}{p_\theta(x_j \mid x_{<j}, x_{j+1:t-1})}.$$

The two sampling procedures are given below.

Algorithm 1 Full-Expansion Aggregate Sampling

Input: prompt c, look-ahead range n, number of expansions M, length limit L

$P \leftarrow$ empty list
repeat M **times**
  $S \leftarrow$ stack containing only c
  while S is nonempty **do**
    $u \leftarrow$ pop the top of S
    $a \sim p_\theta(\cdot \mid u)$
    $v \leftarrow$ append a to u
    $r \leftarrow R(v)$
    $B \leftarrow \lfloor r \rfloor + \mathrm{Bernoulli}(r - \lfloor r \rfloor)$
    repeat B **times**
      if $a = \mathrm{EOS}$ **or** $|v| = L$ **then**
        append v to P
      else
        push v onto S
if P is empty **then**
  return failure
return an entry sampled uniformly from P

Algorithm 2 HBD-n First-Arrival Decoding

Input: prompt c, look-ahead range n, length limit L

$S \leftarrow$ empty stack
loop
  if S is empty **then**
    push c onto S
  $u \leftarrow$ pop the top of S
  $a \sim p_\theta(\cdot \mid u)$
  $v \leftarrow$ append a to u
  $r \leftarrow R(v)$
  $B \leftarrow \lfloor r \rfloor + \mathrm{Bernoulli}(r - \lfloor r \rfloor)$
  repeat B **times**
    if $a = \mathrm{EOS}$ **or** $|v| = L$ **then**
      return v
    push v onto S

Here, $M \geq 1$, and list P retains duplicate entries. All random choices are made independently, and all conditional probabilities involved are positive.

For Algorithm 1, the output distribution conditional on success converges as $M \to \infty$ to $\pi_n(x) = W_n(x)/\sum_y W_n(y)$. The proof is given in Appendix C.1.

The actual decoder (Algorithm 2) uses first arrival with infinite expansion weight at the fixed prompt prefix c, equivalently restarting from c whenever the search stack is empty.

Let N_{end} be the total number of endpoints in one full expansion. Each expansion is finite, and $\mathbb{E}[N_{\mathrm{end}}] = \sum_x W_n(x) > 0$, so

$$\Pr(\text{successful termination}) = \sum_{k=0}^{\infty} \Pr(N_{\text{end}} = 0)^k \Pr(N_{\text{end}} > 0) = 1.$$

The output distribution $\widetilde{\pi}_n$ of first arrival satisfies

$$\|\pi_n - \widetilde{\pi}_n\|_{\text{TV}} \leq \frac{\varepsilon}{1+\varepsilon}, \qquad \varepsilon = \mathbb{E}[N_{\text{end}} - 1 \mid N_{\text{end}} > 0].$$

Here, $\|\cdot\|_{\text{TV}}$ denotes total variation distance; a derivation of the bias bound is given in Appendix C.2.

4.3 Generation Performance

With the same public SetDLM weights, HBD-n raises GSM8K accuracy from 64.67% with official SetDiff decoding to 69.14%. On three RTX 4070 GPUs, complete evaluation takes 3,398.78 s for official SetDiff and 16,699.84 s for HBD-n, 4.91 times as long. Experimental settings and timing conventions are in Appendix D.1.

On a self-trained 1B π-XLNet ($d = 1$), Figure 4 compares HBD-n ($n = 1$) and natural-order AR generation quality and diversity. In an interpolation comparison at the same empirical token entropy, HBD-n lowers generation PPL by about 15%–35%, showing a better quality–diversity trade-off. Experimental settings are in Appendix D.2.

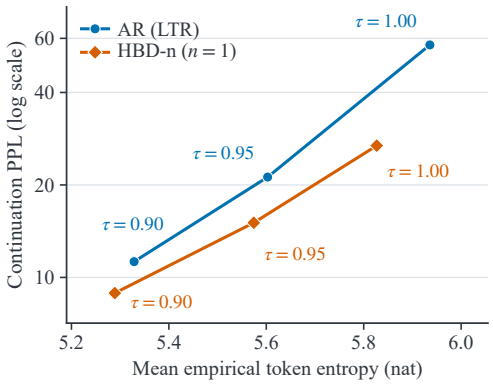


Figure 4: Generation quality and diversity of HBD-n and natural-order AR decoding with the same model. Axes: mean of per-continuation empirical token entropies (horizontal); continuation PPL evaluated by SmolLM2-1.7B (vertical, log scale). τ: sampling temperature; lines connect observed points. Settings: Appendix D.2.

In unconditional generation with a public OWT-SetDLM model, HBD-n ($n = 1$) obtains a MAUVE ($\times 100$) of 81.30 under weaker top-p truncation, higher than the AR value of 68.33 by 12.97 points. The cumulative generation times of natural-order AR (with a self-speculative implementation) and HBD-n are 4,425.10 seconds and 16,725.77 seconds, respectively, with the latter 3.78 times as long. Detailed experimental settings and timing conventions are given in Appendix D.3.

5 Conclusion

We reveal the structural correspondence between set diffusion and XLNet's two-stream architecture, propose π-XLNet with training order distribution as a design dimension, and introduce the Mallows distribution to systematically study training disorder's effects on model training and predictive performance. Experiments show that training disorder controls the trade-off between natural-order prediction performance and adaptability to disordered orders, while evaluation-order loss increments decay approximately as a power law as training disorder increases. π-XLNet's time-based training cost to match AR's natural-order test loss is estimated at approximately 3.1–3.3 times AR's. This shows potential as a lower-training-cost alternative to masked text diffusion.

We further propose HBD-n, using local future information to correct generation paths through stochastic branching and backtracking. Experiments show an improved quality–diversity frontier over natural-order AR decoding with the same model, indicating that disorder-trained conditional prediction can yield practical generation gains. These results motivate renewed exploration of XLNet-style architectures for training efficiency and flexible generation.

AI USE STATEMENT

In this study, we used GPT-6 Astra to assist with code editing, manuscript translation, literature searches, and figure creation.

REFERENCES

[1] Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025a. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. International Conference on Learning Representations (ICLR).

[2] Marianne Arriola, Yair Schiff, Hao Phung, Aaron Gokaslan, and Volodymyr Kuleshov. 2025b. Encoder-Decoder Diffusion Language Models for Efficient Training and Inference. Advances in Neural Information Processing Systems (NeurIPS).

[3] Marianne Arriola and Volodymyr Kuleshov. 2026. Set Diffusion: Interpolating Token Orderings Between Autoregression and Diffusion for Fast and Flexible Decoding. International Conference on Machine Learning (ICML).

[4] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured Denoising Diffusion Models in Discrete State-Spaces. Advances in Neural Information Processing Systems (NeurIPS).

[5] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big—Data-Centric Training of a Small Language Model. arXiv preprint arXiv:2502.02737.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168.

[7] Aaron Gokaslan and Vanya Cohen. 2019. OpenWebText Corpus. Dataset.

[8] Gabe Guo and Stefano Ermon. 2025. Reviving Any-Subset Autoregressive Models with Principled Parallel Sampling and Speculative Decoding. arXiv preprint arXiv:2504.20456.

[9] Jimmy He, Tobias Müller, and Teun W. Verstraaten. 2023. Cycles in Mallows Random Permutations. Random Structures & Algorithms, 63(4):1054–1099.

[10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. An Empirical Analysis of Compute-Optimal Large Language Model Training. Advances in Neural Information Processing Systems (NeurIPS).

[11] Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. 2022. Autoregressive Diffusion Models. International Conference on Learning Representations (ICLR).

[12] Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025. Scaling Up Masked Diffusion Models on Text. International Conference on Learning Representations (ICLR).

[13] Arnaud Pannatier, Evann Courdier, and François Fleuret. 2024. σ-GPTs: A New Approach to Autoregressive Models. arXiv preprint arXiv:2404.09562.

[14] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2024. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track.

[15] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers. Advances in Neural Information Processing Systems (NeurIPS).

[16] Qwen Team. 2024. Qwen2.5 Technical Report. arXiv preprint arXiv:2412.15115.

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. OpenAI technical report.

[18] Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukic. 2026. Scaling Beyond Masked Diffusion Language Models. arXiv preprint arXiv:2602.15014.

[19] Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Remasking Discrete Diffusion Models with Inference-Time Scaling. Advances in Neural Information Processing Systems (NeurIPS).

[20] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2025. Fast-dLLM v2: Efficient Block-Diffusion LLM. arXiv preprint arXiv:2509.26328.

[21] Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. 2025. Any-Order GPT as Masked Diffusion Model: Decoupling Formulation and Architecture. arXiv preprint arXiv:2506.19935.

[22] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. Advances in Neural Information Processing Systems (NeurIPS).

A Background

A.1 Autoregressive Factorization and Arbitrary-Order Modeling

Let the text sequence be $x = (x_1, \ldots, x_T)$ and the set of positions be $[T] = \{1, \ldots, T\}$. Here, θ denotes model parameters; write $x_{a:b} = (x_a, \ldots, x_b)$, which is empty when $a > b$, and let an empty product equal 1. A standard autoregressive language model factorizes the sequence probability in natural order:

$$p_\theta(x) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}).$$

The chain-rule factorization of probability does not require natural order. For any permutation of positions $z = (z_1, \ldots, z_T)$, we can likewise define

$$p_\theta(x \mid z) = \prod_{t=1}^{T} p_\theta(x_{z_t} \mid x_{z_{<t}}, z_t),$$

where $x_{z_{<t}}$ denotes the tokens in the permutation prefix, their permutation order, and their original positions, while z_t specifies the current prediction position. What changes here is the probability factorization order; the original positions of tokens in the text remain unchanged.

An arbitrary-order language model learns conditional predictions under different factorization orders through shared parameters. XLNet's permutation language modeling objective can be written as

$$\mathcal{L}_{\mathrm{XLNet}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{z\sim\mathrm{Unif}(\mathfrak{S}_T)}\left[\sum_{t=1}^{T}\log p_\theta(x_{z_t} \mid x_{z_{<t}}, z_t)\right],$$

where $\mathcal{D}$ is the data distribution and $\mathfrak{S}_T$ is the set of all position permutations. The expression above describes the full permutation language modeling objective; the original XLNet implementation also uses a partial-prediction strategy that predicts only positions toward the end of the permutation (Yang et al., 2019).

A.2 XLNet's Two-Stream Attention

XLNet's two-stream structure can be described by concatenating sequence representations and specifying their attention mask. Let h_i and g_i denote the representations of the content stream and query stream, respectively; let e be the token embedding function and w a shared learnable vector. The initial representations of the two streams are

$$h_i^{(0)} = e(x_i), \qquad g_i^{(0)} = w.$$

Target-position information is introduced through relative positional encoding (Yang et al., 2019, Sec. 2.3 and Appendix A.2).

Given a permutation $z = (z_1, \ldots, z_T)$, arrange the two streams in permutation order and concatenate them along the sequence dimension:

$$U_z^{(0)} = \big[\underbrace{e(x_{z_1}), \ldots, e(x_{z_T})}_{\text{content stream}}, \underbrace{w, \ldots, w}_{\text{query stream}}\big].$$

This concatenation and reordering provide a unified description of the computational structure of the two streams; positional encodings retain the tokens' positions in the original text.

The content-stream position z_t can read content representations at preceding positions and the current position in the permutation; the query-stream position z_t can read only content representations at preceding positions. Ignoring cross-segment memory, let rows be query positions and columns be key/value source positions, with 1 denoting an allowed attention access. The mask is

$$M_{\mathrm{XLNet}} = \begin{bmatrix} L_{\leq} & 0 \\ L_{<} & 0 \end{bmatrix}, \qquad (L_{\leq})_{ts} = \mathbf{1}[s \leq t], \quad (L_{<})_{ts} = \mathbf{1}[s < t].$$

Thus, the upper-left block is a causal mask including the diagonal, the lower-left block is a strict causal mask excluding the diagonal, and the two right-hand blocks are zero, meaning that attention KVs for neither stream come from the query stream.

The query stream still constructs its Query from its own representation and passes its state through the residual connection. In Figure A1, the orange diagonal overlaid in the lower-right block marks this residual path (official XLNet implementation).

A.3 Masked Diffusion and Conditional Subset Prediction

Masked diffusion constructs a forward process by progressively masking tokens and trains a model to recover the original content from corrupted sequences. Let $S \subseteq [T]$ be the set of visible positions at a given step. The model predicts an unknown position $i \notin S$ as

$$p_\theta(x_i \mid x_S, i),$$

where x_S contains the visible tokens and their positions. We omit time conditioning that a particular model may use. At generation time, the model predicts unknown positions from currently visible content and then updates the sequence according to a sampling rule. Absorbing-state discrete

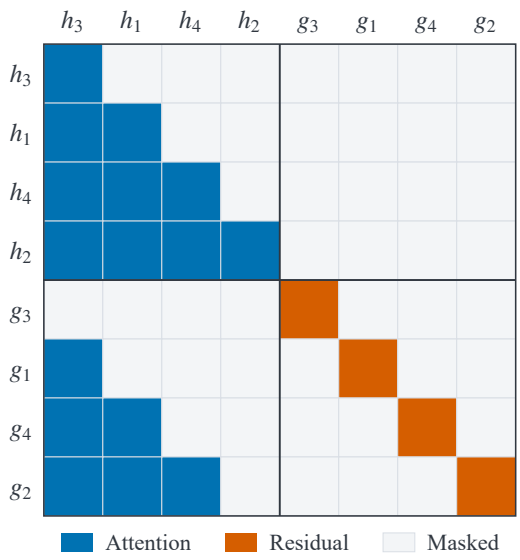


Figure A1: XLNet attention mask and query-stream residual path under permutation $z = (3, 1, 4, 2)$. Rows and columns correspond to representations at the current layer and the previous layer, respectively. Blue indicates allowed attention access; orange marks only the query stream's residual propagation and does not indicate permitted attention to its own KVs. The content-stream residual path is not shown separately.

diffusion provides a probabilistic modeling foundation for this masked generation process and its connection to autoregressive modeling (Austin et al., 2021).

From the perspective of conditional prediction, arbitrary-order modeling and masked diffusion both predict unknown tokens from visible tokens. For a permutation language model, the visible set is determined by the permutation prefix, namely $S_t = \{z_1, \ldots, z_{t-1}\}$; for masked diffusion, it is determined by the noise process and sampling state. This relation between their conditional prediction tasks provides a starting point for further comparison of their training distributions and computational structures.

A.4 Order Construction and Attention Masks in Set Diffusion

Set diffusion constructs a generation order through position-dependent random reveal times. Under its position-offset schedule, each position i corresponds to an interval $[a_i, a_i + \omega]$, whose starting points are ordered in the natural text order. Independently sampling

$$R_i \sim \mathrm{Unif}[a_i, a_i + \omega], \qquad z = \mathrm{argsort}(R_1, \ldots, R_T),$$

produces a generation permutation. The window width ω controls the overlap among position intervals and thereby adjusts the natural-order bias of the order distribution. Continuous reveal times correspond to singleton-token sets; discretizing reveal times and merging positions revealed at the same time produces a general ordered set sequence $\sigma = (S_1, \ldots, S_K)$ (Arriola and Kuleshov, 2026, Sec. 4).

Set diffusion arranges the clean sequence and noisy sequence according to the set order and concatenates them along the sequence dimension:

$$U_\sigma^{(0)} = \big[\underbrace{e(x_{S_1}), \ldots, e(x_{S_K})}_{\text{clean branch}}, \underbrace{e(\widetilde{x}_{S_1}), \ldots, e(\widetilde{x}_{S_K})}_{\text{noisy branch}}\big],$$

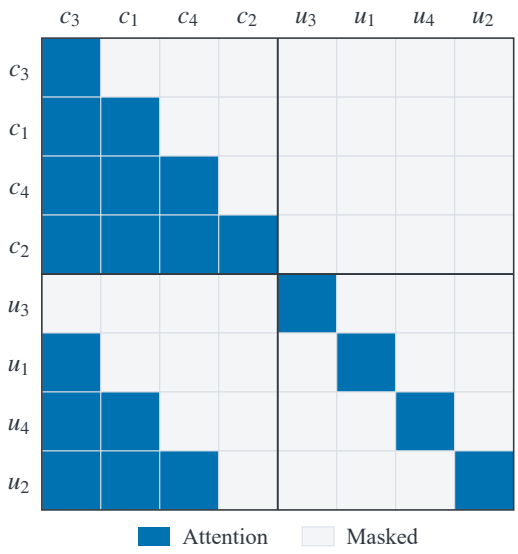


Figure A2: Attention mask for set diffusion with singleton sets and permutation $z = (3, 1, 4, 2)$. c_i and u_i denote the clean and noisy branches, respectively. Blue indicates allowed attention access; the blue diagonal in the lower-right block indicates the noisy branch's self-attention. Residual connections are not shown separately.

where $\widetilde{x}_{S_k}$ denotes the noisy tokens in set S_k, while original positional information remains unchanged. Let c_i and u_i be the hidden representations of the two branches, respectively. Their attention mask is

$$M_{\text{Set}} = \begin{bmatrix} M_{\text{SC}} & 0 \\ M_{\text{OSC}} & M_{\text{SD}} \end{bmatrix}.$$

The upper-left block allows the clean branch to read clean representations in the current and preceding sets; the lower-left block allows the noisy branch to read clean representations in preceding sets; and the lower-right block allows noisy representations within the same set to attend to each other. Thus, the three nonzero blocks are a block lower-triangular, a strict block lower-triangular, and a block-diagonal mask, respectively (Arriola and Kuleshov, 2026, Appendix K).

The SW-SetDLM configuration actually used in the paper employs singleton sets $S_k = \{z_k\}$ during training and sets all inputs to the noisy branch to mask tokens. The second stream is then initialized with the embedding of the mask token, and set attention reduces to

$$M_{\text{Set}}^{|S_k|=1} = \begin{bmatrix} L_{\leq} & 0 \\ L_{<} & I \end{bmatrix}.$$

Compared with XLNet, the two left blocks are the same; the difference is the identity matrix in the lower-right block: the noisy branch can read KVs produced by its own representation (Arriola and Kuleshov, 2026, Sec. 5).

The information-flow structure of the set-diffusion model is therefore very close to XLNet's; the only difference is whether the query stream can attend to its own representation.

B Training Experiments and Scaling Analysis

B.1 General Training and Evaluation Settings

The experiments use the Qwen architecture as the backbone. The design and evaluation of the three experiment groups are as follows, and all losses are reported in nat/token.

Table B1: Training-experiment overview.

Experiment	Model and training order	Model and data budget	Evaluation setting
Order-distribution sweep	π-XLNet, $d \in \{0, 0.25, 0.5, 1, 2, 4, 8\}$	135M, 2.8312B training tokens	2,000 sequences from a held-out test set; cross-evaluation under seven orders
Model and data ratio	AR and π-XLNet; the latter uses $d \in \{0, 0.25, \approx 0.5, 1, \approx 2\}$	Hidden dimension 576, depth 12–42 layers, approximately fixed ND	2,000 sequences from a held-out test set; uniform natural-order evaluation
Cross-scale comparison	AR and π-XLNet with $d = 1$	135M, 360M, 1B, $D/N \approx 21$	2,000 sequences from a held-out test set; uniform natural-order evaluation

Here, N is the number of model parameters and D is the number of training tokens; M and B denote 10^6 and 10^9, respectively.

Each of the three experiment groups uses a fixed set of 2,000 held-out test sequences. We refer to the mean per-token negative log-likelihood on this set as test loss below.

The 135M models in the order sweep and cross-scale comparison come from different training batches.

B.2 Order Sweep and Loss-Increment Fitting

The detailed configuration for the order-distribution sweep is below. All training settings use the same model configuration, data manifest, and training budget.

Let $a = d_{\text{train}}$, $b = d_{\text{eval}}$, and

$$\Delta\mathcal{L}_{ab} = \mathcal{L}(a, b) - \mathcal{L}(a, 0).$$

Using the 36 observations with $a, b \in \{0.25, 0.5, 1, 2, 4, 8\}$, we fit

$$\widehat{\Delta\mathcal{L}}_{ab} = F_b a^{-\alpha}, \qquad \min_{\alpha, \{F_b > 0\}} \sum_{a,b} \left[\log \Delta\mathcal{L}_{ab} - \log F_b + \alpha \log a\right]^2 .$$

All observations are equally weighted. The six evaluation orders share exponent α and have separate amplitudes F_b. The case $a = 0$ does not enter the log-space fit; $b = 0$ is used only to construct the natural-order baseline for each model. The fit gives $\alpha = 0.508394$, with amplitudes below.

Mean absolute relative error is computed in the original loss-increment space:

$$\text{MAPE} = \frac{1}{36} \sum_{a,b} \frac{|\widehat{\Delta\mathcal{L}}_{ab} - \Delta\mathcal{L}_{ab}|}{\Delta\mathcal{L}_{ab}}.$$

The in-sample error is 2.94%. In the hold-out test, each round removes the six increments associated with one training disorder, refits on the other 30 observations, and predicts the six removed increments; the mean absolute relative error over the six rounds is 4.61%. In every round, the measured natural-order loss of the held-out model is still used to construct the baseline, so the test evaluates evaluation-order loss increments.

Table B2: Order-distribution sweep training configuration.

Configuration	Setting
Training data	FineWeb-Edu sample-10BT pool (Penedo et al., 2024)
Tokenizer	HuggingFaceTB/cosmo2-tokenizer
Model parameter scale	135M
Transformer layers	30
Hidden / feed-forward dimension	576 / 1536
Attention heads / key-value heads	9 / 3
Sequence length	2048
Global batch size	512 sequences
Training steps	2700
Total training tokens	2,831,155,200
Peak learning rate	3×10^{-3}
Learning-rate schedule	200-step linear warmup, then cosine decay to 10% of peak
Weight decay	0.01
Gradient-norm clipping threshold	1.0
Random seed	42
Natural-order control	$d = 0$, retaining the π-XLNet two-stream architecture
Training metric	Batch loss recorded during training
Test metric	Mean per-token negative log-likelihood under the specified evaluation order

Table B3: Amplitudes of the power-law fit for loss increments.

Evaluation disorder b	F_b
0.25	0.059101
0.5	0.105170
1	0.171305
2	0.255022
4	0.342640
8	0.424460

B.3 Model–Data-Ratio Experiments and Fitting

Under approximately fixed ND, we compare the optimal data-to-parameter ratios of different training orders. The experimental settings are below, and all losses use uniform natural-order evaluation.

Table B4: Model–data-ratio experiment configuration.

Configuration	Setting
Control model	AR
π-XLNet training parameter q	$0,\ 0.2,\ 0.333,\ 0.5,\ 0.667$
Corresponding training disorder d	$0,\ 0.25,\ \approx 0.5,\ 1,\ \approx 2$
Hidden dimension	576
Model depth	$12,\ 18,\ 24,\ 30,\ 36,\ 42$
Corresponding training tokens (B)	$5.3792,\ 4.1419,\ 3.3659,\ 2.8312,\ 2.4432,\ 2.1496$
Total training configurations	36
Parameter-count convention	Includes embedding parameters; shared parameters are counted once

Each training setting contains six depth configurations. Using actual parameter counts, let $u = \log(D/N)$ and fit natural-order loss by equally weighted quadratic least squares:

$$\widehat{\mathcal{L}}(u) = b_2 u^2 + b_1 u + b_0, \qquad \left(\frac{D}{N}\right)_* = \exp\left(-\frac{b_1}{2b_2}\right).$$

All settings give $b_2 > 0$. The ratio change relative to AR is the ratio of the two fitted optimal ratios minus one. Fitting uses full-precision observations; coefficients and errors are as follows.

Table B5: Fitting coefficients and errors for optimal data-to-parameter ratios.

Training setting	b_2	b_1	b_0	Loss-fit RMSE
AR	0.071290	-0.523409	4.129345	0.0086
π-XLNet, $d = 0$	0.041089	-0.317986	3.777187	0.0151
π-XLNet, $d = 0.25$	0.074822	-0.565479	4.334958	0.0034
π-XLNet, $d \approx 0.5$	0.066258	-0.492246	4.133352	0.0045
π-XLNet, $d = 1$	0.111820	-0.831342	4.803388	0.0077
π-XLNet, $d \approx 2$	0.090798	-0.680594	4.603616	0.0045

RMSE is computed in the original loss space in nat/token; coefficients in the table are rounded only for display.

B.4 Cross-Scale Fitting, Timing, and Cost Estimation

Table B6: Cross-scale experiment results.

Nominal size	Layers / hidden dimension	Training tokens (B)	Steps	Peak learning rate	AR loss	π-XLNet loss
135M	30 / 576	2.8312	2700	3×10^{-3}	3.1264	3.2157
360M	32 / 960	7.5497	4800	3×10^{-3}	2.7990	2.8683
1B	36 / 1536	20.9715	10000	1.5×10^{-3}	2.5150	2.5787

Using natural-order losses at the three scales for AR and π-XLNet with $d = 1$, we fit with equal weights

$$\log \widehat{\mathcal{L}} = \log A - \beta \log(N/10^8).$$

The fitting uses actual rather than nominal parameter counts:

Table B7: Actual parameter counts in the cross-scale experiment.

Nominal size	AR parameter count	π-XLNet parameter count
135M	134,518,848	134,519,424
360M	361,826,176	361,827,136
1B	981,585,408	981,586,944

For AR, $A = 3.227091$ and $\beta = 0.109483$; for π-XLNet, $A = 3.318476$ and $\beta = 0.111071$. After transforming predictions back to the original loss space, the respective RMSEs are 0.0031 and 0.0061 nat/token.

For a target loss ℓ within the commonly covered range, we calculate required parameter counts for the two model classes from

$$N(\ell) = 10^8 \left(\frac{A}{\ell}\right)^{1/\beta}$$

and then take their ratio to obtain the equal-loss parameter requirement. The training-cost multiplier is calibrated using the per-step time ratio $724.9/361.9$ measured at 1B, and training time is approximately extrapolated with ND along the fixed data-ratio path. At fixed D/N, increasing the parameter count by a factor of 1.25–1.28 increases the training-token count by the same factor. Extrapolating training time with ND therefore gives the estimated time-based training-cost multiplier

$$\frac{724.9}{361.9} \times (1.25\text{–}1.28)^2 \approx 3.1\text{–}3.3.$$

The results above are estimates combining measured relative per-step time with empirical fitting at three scales.

We benchmark the 1B AR and two-stream models on NVIDIA A100 40GB GPUs. Both use BF16 precision, batch size 4, and original sequence length 2048. The measured results are below.

Table B8: Per-step training time, throughput, and memory of 1B models.

Benchmark target	Model size	Batch size	Original sequence length	Per-step time (ms)	Raw-token throughput (K tokens/s)	Peak allocated memory (GiB)
AR	1B	4	2048	361.9	22.6	22.3
Two stream	1B	4	2048	724.9	11.3	37.0

Throughput is computed from $4 \times 2048 = 8192$ raw tokens per step; query representations in the two-stream model are not counted again as tokens. The per-step time ratio is $724.9/361.9 \approx 2.003$, and the main text uses approximately 2.00 times. This measurement calibrates relative time at the same scale; equal-loss training cost is estimated by combining it with the scaling fit.

C Theoretical Supplement for HBD-N

C.1 Convergence of the Aggregate-Sampling Distribution

Let $C(x)$ be the number of occurrences of $x = x_{1:T}$ in one full expansion. By the unbiasedness of randomized rounding and iterated conditional expectation,

$$\begin{aligned}\mathbb{E}[C(x)] &= \prod_{t=m+1}^{T} p_\theta(x_t \mid x_{<t})\, R(x_{1:t}) \\ &= \prod_{t=m+1}^{T} \frac{W_n(x_{1:t})}{W_n(x_{<t})} = W_n(x).\end{aligned}$$

Let $N_{\text{end}} = \sum_x C(x)$ be the total number of endpoints, and let the counts in the k-th independent expansion be $C_k(x), N_{\text{end},k}$. By the strong law of large numbers and bounded convergence, the output distribution of aggregate sampling conditional on success satisfies

$$\begin{aligned}\rho_M(x) &= \mathbb{E}\left[\frac{\sum_{k=1}^{M} C_k(x)}{\sum_{k=1}^{M} N_{\text{end},k}} \,\middle|\, \sum_{k=1}^{M} N_{\text{end},k} > 0\right] \\ &\xrightarrow[M\to\infty]{} \frac{\mathbb{E}[C(x)]}{\mathbb{E}[N_{\text{end}}]} = \frac{W_n(x)}{\sum_y W_n(y)} = \pi_n(x).\end{aligned}$$

C.2 Bias Bound for the First-Arrival Distribution

Let Y be the first endpoint reached in one full expansion under the traversal order above. The two algorithms have the same random process before reaching this endpoint; hence

$$\widetilde{\pi}_n(x) = \sum_{k=0}^{\infty} \Pr(N_{\text{end}} = 0)^k \Pr(N_{\text{end}} > 0, Y = x) = \Pr(Y = x \mid N_{\text{end}} > 0),$$

whereas

$$\pi_n(x) = \frac{\mathbb{E}[C(x)]}{\mathbb{E}[N_{\text{end}}]} = \frac{\mathbb{E}[C(x) \mid N_{\text{end}} > 0]}{\mathbb{E}[N_{\text{end}} \mid N_{\text{end}} > 0]} = \frac{\mathbb{E}[C(x) \mid N_{\text{end}} > 0]}{1+\varepsilon}.$$

Since $C(x) \geq \mathbf{1}_{\{Y=x\}}$ in a nonempty expansion,

$$\begin{aligned}\|\pi_n - \widetilde{\pi}_n\|_{\text{TV}} &= \frac{1}{2(1+\varepsilon)} \sum_x \left|\mathbb{E}\big[C(x) - \mathbf{1}_{\{Y=x\}} \mid N_{\text{end}} > 0\big] - \varepsilon \widetilde{\pi}_n(x)\right| \\ &\leq \frac{\mathbb{E}[N_{\text{end}} - 1 \mid N_{\text{end}} > 0] + \varepsilon}{2(1+\varepsilon)} \\ &= \frac{\varepsilon}{1+\varepsilon}.\end{aligned}$$

D Generation Experimental Settings and Supplementary Results

D.1 GSM8K Evaluation Settings and Timing

Both decoding methods use EMA weights of the public model `kuleshov-group/setdlm-gsm8k-smax8`. The evaluation uses all 1,319 questions in the GSM8K (Cobbe et al., 2021) test set in a 0-shot setting, generating once for each question. Both groups use the same prompt:

```
<|im_start|>user
Please reason step by step, and put your final answer
within $\boxed{}$.  {question}<|im_end|>
<|im_start|>assistant
```

Table D1: GSM8K decoding and evaluation configuration.

Configuration	Official SetDiff	HBD-n
Number of test questions	1,319	1,319
Maximum number of added tokens	1,024	1,024
Model context length	1,024	1,024
Look-ahead range n	–	1
Sampling settings	Token sampling disabled, temperature 0	Temperature 0.90, top-p 0.95
Repetition penalty	Not separately set	1.0
Generation strategy	Add noise after prediction, linear reveal	First-arrival HBD-n
Diffusion steps	1,024	–
Generation block size	1,024	–
Maximum window size	4	–
Confidence threshold	0.9	–
First-hit sampling / confidence remasking / confidence-interval remasking	All disabled	–
GPUs / number of parallel processes	3 RTX 4070 GPUs / 3	Same
Batch size per process	1	1
Precision / attention backend	FP16 / SDPA	Same
Random seed	1234	1234

The official configuration enables KV caching, does not align inputs to generation blocks, and uses a maximum noise-process block size of 8. It uses GSM8K answer regular-expression matching and repeated-token detection as stopping rules; task stop strings are `Question:`, `</s>`, and `<|im_end|>`. HBD-n uses `<|im_end|>` as EOS. When visible context and target position exceed the model window, HBD-n retains the most recent 1,023 visible tokens together with their absolute RoPE positions and rebuilds KVs at every query. All 1,319 HBD-n outputs end with EOS; none reaches the maximum of 1,024 added tokens.

HBD-n does not further restrict the number of search or backtracking steps; it terminates when EOS is generated or the maximum generation length is reached.

Evaluation uses lm_eval 0.4.9 and GSM8K task version 3.0, reporting strict-match exact-match accuracy. Duplicate questions produced by distributed evaluation are counted only once, for 1,319 questions in total. Official SetDiff and HBD-n solve 853 and 912 questions correctly, respectively.

Table D2: GSM8K accuracy and complete evaluation time.

Method	Strict accuracy	Total evaluation time (s)	Relative time
Official SetDiff	64.67%	3,398.78	1.00×
HBD-n, $n = 1$	69.14%	16,699.84	4.91×

Total time is the actual elapsed time from task launch to exit, including model loading, generation, scoring, and result saving. Both groups use the same parallel configuration on three GPUs, and the time ratio is 4.9135. The two methods use their respective sampling and stopping settings above.

D.2 Generation Quality and Diversity (Figure 4)

Using the same weights of a self-trained 1B π-XLNet ($d = 1$, trained for 10,000 steps), we compare natural-order AR with HBD-n ($n = 1$). At temperatures 0.90, 0.95, and 1.00, the two methods

each generate 300 continuations, for 1,800 continuations in total. The six groups use the same 300 prompts from FineWeb-Edu sample-100BT; each prompt has 50 tokens and each continuation has 1,998 tokens. EOS is masked during generation, which stops at a fixed length.

The generation tokenizer is `HuggingFaceTB/cosmo2-tokenizer`; the random seed is 42. BF16, SDPA, and KV caching are used, with batch size 8. Neither method uses top-p or top-k truncation. HBD-n caches conditional probabilities that have already been queried and does not further restrict the number of model queries or search transitions.

We use SmolLM2-1.7B (Ben Allal et al., 2025) as an external language model and compute continuation negative log-likelihood with FP32 and SDPA. Prompts and continuations are tokenized separately and then concatenated, with no extra special tokens added; only the continuation portion, including its first token, is counted. Let m_i be the number of scored tokens in continuation i and ℓ_i its total negative log-likelihood. Then

$$\mathrm{NLL} = \frac{\sum_i \ell_i}{\sum_i m_i}, \qquad \mathrm{PPL} = \exp(\mathrm{NLL}).$$

Thus, PPL is calculated from the token-weighted NLL of all continuations.

Empirical token entropy uses continuation token IDs from the generation tokenizer. Let $f_i(v)$ be the relative frequency of token v in continuation i. Then

$$\overline{H} = \frac{1}{300} \sum_{i=1}^{300} \left[-\sum_v f_i(v) \log f_i(v) \right].$$

This metric is calculated for each continuation and then averaged, excludes the prompt, and uses natural logarithms throughout. The six groups yield the following results.

Table D3: Measured generation-quality and diversity results for Figure 4.

Decoding method	Temperature	NLL (nat/token)	PPL	Empirical token entropy (nat)
Natural-order AR	0.90	2.4210	11.2570	5.3287
Natural-order AR	0.95	3.0547	21.2153	5.6028
Natural-order AR	1.00	4.0450	57.1125	5.9358
HBD-n, $n = 1$	0.90	2.1852	8.8925	5.2890
HBD-n, $n = 1$	0.95	2.7122	15.0625	5.5743
HBD-n, $n = 1$	1.00	3.2902	26.8492	5.8267

The same-entropy comparison is performed over the entropy interval covered by both methods. We linearly interpolate NLL as a function of empirical token entropy for each method, then exponentiate the interpolated NLL to obtain PPL. If the same-entropy NLL difference is $\Delta = \mathrm{NLL}_{\mathrm{AR}} - \mathrm{NLL}_{\mathrm{HBD}}$, the PPL reduction is $1 - \exp(-\Delta)$.

Table D4: Interpolation comparison at the same empirical token entropy.

Aligned empirical token entropy	AR NLL	HBD-n NLL	PPL reduction
5.3287	2.4210	2.2585	15.00%
5.5743	2.9889	2.7122	24.17%
5.8267	3.7207	3.2902	34.98%

Interpolation uses full-precision data and does not extrapolate outside the observation interval. The approximately 15%–35% in the main text corresponds to the three same-entropy comparison points above; Figure 4 shows the six measured points and their connecting lines, with a logarithmic vertical axis.

D.3 MAUVE Evaluation

This experiment uses the public model `kuleshov-group/owt-setdlm-smax32`. HBD-n ($n = 1$) uses the first-arrival strategy, performs lazy depth-first expansion through remaining branch

weights, and backtracks when a branch is exhausted. The AR baseline uses a self-speculative implementation with draft length 3.

Table D5: Generation configuration for the MAUVE experiment.

Configuration	AR	HBD-n ($n = 1$)
Number of generated texts	1,000	1,000
Prompt	BOS only	BOS only
Maximum total length	1,024 tokens, including BOS	Same
Stopping condition	Native EOS or length limit	Same
Temperature τ	0.90	0.90
top-p	0.90	0.95
Repetition penalty	1.0	1.0
Batch size	4	4
Base random seed	1234	1234
Precision / attention backend	FP16 / SDPA	FP16 / SDPA
Mean generated length (excluding BOS)	704.32 tokens	662.93 tokens

All generated results are included in scoring, with no filtering by entropy or text quality and no additional EOS truncation. Both the proposal distribution and correction probabilities of HBD-n apply the temperature and top-p transformations. The top-p values of the two configurations differ; this experiment therefore provides generation-quality evidence at the decoding-configuration level rather than a single-factor algorithmic ablation at fixed sampling parameters.

Reference documents are drawn from the final 100,000 documents of the OpenWebText (Gokaslan and Cohen, 2019) training split, using random seed 20260701, in five groups of 1,000 documents. Sampling is without replacement within each group, while groups may overlap; documents remain independent and are not concatenated or packed. Both methods use exactly the same reference documents.

MAUVE compares the distributions of generated and reference texts in the GPT2-large (Radford et al., 2019) feature space (Pillutla et al., 2021). Before scoring, `<|fim_middle|>` and `<|endoftext|>` are removed. The feature model reads at most 1,024 tokens from each document and uses FP16 and batch size 4. Scoring uses mauve-text 0.4.0, with the number of clusters determined automatically and equal to 100 for all groups; feature reduction retains 90% explained variance, and K-means uses 5 restarts with at most 500 iterations each. The MAUVE scaling factor is 5, and the divergence curve uses 25 discrete points.

Table D6: MAUVE scores for each reference group.

Reference group / clustering seed	AR MAUVE ($\times 100$)	HBD-n MAUVE ($\times 100$)
1 / 1234	66.7691	87.0507
2 / 1235	66.3366	81.0962
3 / 1236	67.1885	80.1921
4 / 1237	68.7386	77.6217
5 / 1238	72.6083	80.5374
Mean $\pm$ population standard deviation	68.3282 ± 2.2884	81.2996 ± 3.1129

For each method's fixed 1,000 generated texts, the reported population standard deviation of five scores reflects variation from changing only the reference-document group and clustering seed; it is neither the standard deviation of independent generation repetitions nor a confidence interval.

Generation time is the sum of the actual batch generation times across GPUs, excluding model loading, text decoding, scoring, and result saving. Both groups use RTX 4070 GPUs and batch size 4, with the final incomplete batch run using its actual number of samples.

The cumulative generation-time ratio of HBD-n to AR is 3.7797. This measure sums generation time across GPUs and is not equivalent to the wall-clock waiting time from task start to completion.

Table D7: Cumulative generation time in the MAUVE experiment.

Method	Number of generated texts	Cumulative generation time (s)
Natural-order AR (self-speculative implementation)	1,000	4,425.10
HBD-n	1,000	16,725.77