Self-Speculative Decoding in Any-Order and Any-Subset Autoregressive Models

Gabe Guo
Department of Computer Science
Stanford University
gabeguo@stanford.edu

Stefano Ermon
Department of Computer Science
Stanford University
ermon@cs.stanford.edu

Abstract

In arbitrary-order language models, it is an open question how to sample tokens in parallel from the *correct joint distribution*. With discrete diffusion models, the more tokens they generate in parallel, the less their predicted distributions adhere to the originally learned data distribution, as they rely on a conditional independence assumption that only works with infinitesimally small timesteps. We find that a different class of models, any-subset autoregressive models (AS-ARMs), holds the solution. As implied by the name, AS-ARMs can generate tokens in any order, and in parallel. Moreover, AS-ARMs support parallelized joint probability density estimation, allowing them to correct their own parallel-generated token distributions, via our *Any-Subset Speculative Decoding (ASSD)* algorithm. ASSD provably enables generation of tokens from the correct joint distribution, with the number of neural network calls upper bounded by the number of tokens predicted. We empirically verify that ASSD speeds up language generation, without sacrificing quality. Furthermore, we provide a mathematically justified scheme for training AS-ARMs for generation, and show that AS-ARMs achieve state-of-the-art performance among sub-200M parameter models on infilling benchmark tasks, and nearly match the performance of models 50X larger on code generation. Our theoretical and empirical results indicate that the once-forgotten AS-ARMs are a promising direction of language modeling.

1 Introduction

Almost all the SoTA LLMs [Ach+23; Tou+23; Liu+24a; Tea+23] are autoregressive, *i.e.*, they only support left-to-right token generation. As a result, they suffer from two major problems: (1) they must generate tokens one-by-one, which limits their speed; (2) they cannot infill sequences in orders besides left-to-right (unless specialized training strategies are adopted [Bav+22; Fri+22; Roz+23], but these are heuristic and not guaranteed to output the correct structure).

Regarding models that inherently support infilling, there are discrete diffusion models [Aus+21; Cam+22; LME23; Sah+24] and any-order autoregressive models (AO-ARMs) [Yan19; SSE22]. Discrete diffusion models have the benefit of parallel sampling of multiple tokens at a time, which potentially speeds up generation, but at the cost of fidelity to the learned data distribution [LME23]. On the other hand, fast sampling schemes have generally not been explored for AO-ARMs.

We explore the problem of fast parallel sampling from AO-ARMs, without any degradation in output quality. Particularly, we are inspired by speculative decoding techniques, which have accelerated generation in standard autoregressive models by using a draft model to quickly generate multiple tokens without any degradation. Then, the tokens are accepted or rejected based on their fidelity to the joint distribution as evaluated by the oracle model. Interestingly, the outputted distribution is provably the same as would have been obtained from sampling only from the expensive oracle model, while

Structured Probabilistic Inference & Generative Modeling Workshop of NeurIPS 2025, San Diego, CA, USA.

usually requiring fewer neural network forward passes [Che+23; LKM23]. Crucial to speculative decoding are (1) density estimation from the oracle, and (2) a fast draft model. Indeed, AO-ARMs, by design, estimate joint probability density of sequences [Yan19]. Furthermore, AO-ARMs can even act as their own draft models; due to their architectural design and training objective, they can generate tokens in any order and in parallel [Yan19; SSE22].

As such, we propose *Any-Subset Speculative Decoding (ASSD)*, an algorithm that combines speculative decoding with AO-ARMs. ASSD provably generates sequences from the true joint distribution learned by the oracle model. Furthermore, it is mathematically guaranteed to never increase the number of neural network evaluations, which means that it can only speed up generation without losing quality (as we empirically verify). We also provide a mathematically justified training scheme for AO-ARMs. Finally, we show that appropriately trained AO-ARMs achieve state-of-the-art performance among sub-200M parameter models (diffusion and autoregressive) on infilling benchmark tasks, and nearly match the performance of models 50X larger on code generation, while needing less than half the training tokens.

2 Background

Autoregressive Models: Given a text sequence $\mathbf{x} \sim \mathcal{D}$, where $\mathcal{D}$ is a data distribution, autoregressive (AR) models learn

$$p(\mathbf{x}) = \prod_{i=0}^{|\mathbf{x}|-1} p(x_i|x_0, \ldots, x_{i-1}). \tag{1}$$

Crucially, the product rule factorization means that an AR model only needs to learn conditional distributions with support of size $O(S)$, where S is the size of the vocabulary. This factorization also admits a straightforward generation strategy, where token x_i is sampled from $p(\cdot|x_0, \ldots, x_{i-1})$, *i.e.*, the previous tokens are used to produce the next [Ach+23]. This also means that the prompt for an AR model must always be the prefix, *i.e.*, $\mathbf{x}_{0:i}$ – arbitrarily-located prompts are not supported.

Infilling: We consider infilling tasks, where the prompt is not necessarily the prefix, but can be arbitrarily-located. Examples of this are code generation, story completion, and scientific data imputation. Mathematically, this can be formulated as sampling from a joint conditional probability distribution with discrete state space. That is, we wish to sample from $p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$, where $\sigma(i)$ is the (zero-indexed) positional index of the i-th ordered item in the sequence of length N. In other words, σ is a permutation of $\{0, 1, \ldots, N-1\}$, and i is the generation order. So, $\mathbf{x}_{\sigma(<m)}$ represents the prompt tokens, and $\mathbf{x}_{\sigma(\geq m)}$ represents the tokens whose distributions we want to predict. In general, we can have any $\sigma : \{0, \ldots, N-1\} \to \{0, \ldots, N-1\}$, so long as σ is a bijection. As previously established, regular AR models cannot address this task in general, except when $\sigma(i) = i$.

Any-Order Autoregressive Models: Any-order autoregressive models (AO-ARMs) [SSE22; Hoo+21; Yan19] can be seen as collections of $N!$ joint distributions indexed by σ, where σ is the factorization order:

$$\log p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)}) = \sum_{i=m}^{N-1} \log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<i)}; \sigma). \tag{2}$$

That is, the distribution for each token is calculated one at a time, in order of increasing i, and used as conditioning for the next token to be predicted. Concretely, each evaluation of the network produces a probability distribution (for a single token) with support of size $O(S)$, where S is the size of the vocabulary. The probability of each member of the support is explicitly calculated and stored in memory, incurring $O(S)$ memory cost per token.

If trained to optimality, all the joint distributions should be equal. However, except with infinite data and capacity, given different ordering functions α and σ [SSE22],

$$\sum_{i=0}^{N-1} \log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<i)}; \sigma) \neq \sum_{i=0}^{N-1} \log p(x_{\alpha(i)}|\mathbf{x}_{\alpha(<i)}; \alpha). \tag{3}$$

Any-Subset Autoregressive Models: Disambiguating Joint Conditional Calculation: Any-subset autoregressive models (AS-ARMs) are a subclass of AO-ARMs that reduce the number of joint

distributions (σ) learned from $N!$ to 2^N. This puts less training burden on the finite model capacity, while keeping strictly the same expressivity as it relates to conditional joint distributions of the form in Equation 2. To achieve this, AS-ARMs adopt the recursive binary lattice mask decomposition protocol from [SSE22]. The idea underlying this protocol is that we can split every generation task into two parts: the prompt (denoted by $\mathbf{x}_{\sigma(<m)}$) and the tokens that need to be generated (denoted by $\mathbf{x}_{\sigma(\geq m)}$). Mathematically, we want to estimate $p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$, where m is the number of tokens in the prompt (assuming 0-indexing). Now, we make two observations.

Firstly, since we never need to evaluate the density of the conditioning $\mathbf{x}_{\sigma(<m)}$, we have every token attend to every other token within $\mathbf{x}_{\sigma(<m)}$. Secondly, within $\mathbf{x}_{\sigma(\geq m)}$, there is no need to learn all the possible factorization paths, as long as we can get the joint conditional probability $p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$. Taking inspiration from vanilla autoregressive models, we enforce

$$\forall\, i \geq m, j \geq m : \sigma(i) > \sigma(j) \Leftrightarrow i > j. \tag{4}$$

That is, we simply process the masked tokens left to right. Thus, given the location of the prompt, we only have to learn one path to calculate the joint probability of the generation. This also solves the inconsistency problem in Equation 3. (As later shown, this is crucial to Algorithm 1's correctness.)

This reduces the number of queries learned by the model from $N!$ (all possible permutations) to 2^N (all possible mask location selections, times one ordering per mask selection). This makes optimization easier, as noted by [SSE22] and verified in our ablations (Figure 3). In summary, AS-ARMs are a subclass of AO-ARMs incorporating Equation 4's disambiguated ordering strategy. The architecture design is typically the same, but the way we query the architecture is different.

3 Sampling Strategies for Joint Distributions

We want to accurately and efficiently sample from the joint conditional distribution in Equation 2. We assume that we have access to AS-ARMs that explicitly predict single-variable marginals, *i.e.*, next-token prediction under some ordering.

One-Step Sampling from the Joint: Sampling directly from this joint distribution in a single step is typically infeasible, because its support has size $O(S^{N-m})$, where S is the number of tokens in the vocabulary. If we wanted to explicitly calculate the probability of every tuple in the support, the exponential space cost to just store the distribution would quickly exceed memory capacities.

Sequential Sampling via Factorization: We can sample the $\log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<i)})$ one-by-one, using each generated token as the conditioning for the next one. Following Equation 2, we can get samples from the joint conditional distribution by doing this $N - m$ times, once for each $i \in [m, N)$. In (any-order, any-subset) autoregressive models [Ach+23; Tou+23; Yan19; SSE22], there is typically $O(S)$ time cost per-token, so the overall time cost would be $O(S * (N - m))$. Indeed, this is the dominant generation strategy for autoregressive models like GPT [Ach+23], where $\sigma(i) = i$.

Parallel Sampling via Independence Assumption: At another extreme, we can independently sample multiple single-variable marginals in parallel. That is, we predict $\log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<m)})$ for all $i \in [m, N)$. Since $N - m$ generations are done in parallel, computational time cost is only $O(S)$, which is effectively constant with respect to the number of tokens. The limitation is that

$$\sum_{i\in[m,N)} \log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<m)}) \neq \log p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)}), \tag{5}$$

except in the unlikely scenario of true independence, *i.e.*, $\log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<m)}) = \log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<i)})$.

This is analogous to how discrete diffusion models take large discretized timesteps in the reverse CTMC to predict tokens in parallel, even though the predictions at each position are actually generated with a conditional independence assumption that only holds for an infinitesimally small timestep (*i.e.*, one-by-one generation) [LME23; Sah+24].

Best of Both Worlds: Any-Subset Speculative Decoding: We seek a way to combine the runtime benefits of parallel sampling with the fidelity of sequential sampling. That is, can we achieve $O(S)$ time complexity for arbitrary-subset generation, while recovering true samples from $\log p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$? The key insight here is that if we had an oracle model that could evaluate the joint density of a sequence with a singular function evaluation, we could leverage the quick speed

of independent parallel generation and use these samples $\log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<m)})$ as estimates for the true $\log p(x_{\sigma(i)}|\mathbf{x}_{\sigma(<i)})$. Then, via some rejection sampling scheme which uses as an oracle the joint density evaluation of this newly generated sequence, we could keep only the samples that adhere to $\log p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$. In principle, this could take best-case $o(S)$ time (parallelized), if the oracle accepts all the samples, and worst-case $O(S * (N - m))$ time.

Speculative decoding is one such method that guarantees fidelity to the true distribution. It is mathematically proven to generate samples from the target distribution, and the empirical results show a stark decrease in the number of function evaluations required [LKM23; Che+23]. However, prior to our work, speculative decoding has only been shown to work for traditional GPT-style [Ach+23] autoregressive models.

4 Architectural Design of AS-ARMs

Towards the goal of fast, principled parallel sampling in any order, we seek any-subset autoregressive models (AS-ARMs) that can support our *Any-Subset Speculative Decoding (ASSD)* algorithm (which is fully described in Section 5). To recap, the criteria are: (1) generates arbitrarily-ordered tokens in parallel; (2) evaluates joint density with only one forward pass. We now show how AS-ARMs can be designed to fulfill these criteria. In this section, we will focus on AO-ARM architecture, specifically XLNet [Yan19], as AS-ARMs are architecturally the same as AO-ARMs. [1]

4.1 Parallel Sampling via Arbitrary Positional Queries

Firstly, the architectures of modern weight-tied AO-ARMs [Yan19; SSE22] support parallel sampling. That is, with one function evaluation, we can simultaneously predict in parallel (conditionally independent) distributions for all the masked tokens $\mathbf{x}_{\sigma(\geq m)}$, conditioned on the prompt $\mathbf{x}_{\sigma(<m)}$. This allows the network to act as a quick "draft" model.

To illustrate how this works, we consider the XLNet architecture [Yan19]. It is designed such that we can pass in arbitrary positional queries $\sigma(\geq m)$ of not-yet-predicted tokens to condition on the visible prompt tokens $\mathbf{x}_{\sigma(<m)}$. As implied by the "any-order" moniker, there is no constraint on which positions we query: we could query the leftmost, rightmost, or even a randomly selected unfilled position. Furthermore, the positional queries all attend to the same prompt tokens, but the prompt tokens cannot attend to the positional queries. So, no matter how many positions we query in parallel, each query cannot change the representations of the prompt tokens, and therefore cannot change the outcomes of the other simultaneous queries. This enables parallel sampling. See Figure 1a.

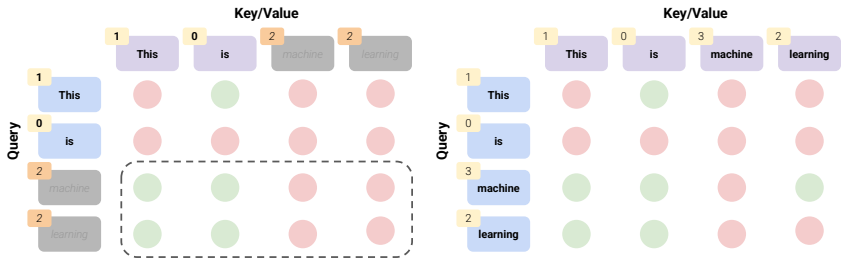
4.2 Quick and Principled Density Estimation via Causal-Like Attention Masking

Another crucial ingredient of speculative decoding is density estimation – this allows the "oracle" model to correct the mistakes of the draft model. As such, discrete diffusion models trained with an ELBO [LME23; Sah+24; DG24] are not readily adaptable to this scheme. However, AO-ARMs [SSE22; Yan19], just like discrete diffusion models, can generate an arbitrary number of tokens in parallel in $O(S)$ time. Furthermore, they are designed to evaluate the true joint density of a sequence, something discrete diffusion models do not currently do.

Care must be taken, however, to pick an architecture that evaluates the joint density of a sequence in one function evaluation, *i.e.*, $O(S)$ time. Recent architectures [SSE22; Hoo+21] take $O(S * N)$ steps, as only logits of masked tokens are predicted at each function evaluation – they are unable to predict the logits of visible tokens.

However, XLNet [Yan19] can calculate the logits for all visible and masked tokens in a single function evaluation with $O(S)$ time. In particular, the attention masks for each token only allow attention (*i.e.*, conditioning) to the preceding tokens in the ordering, such that we can construct a factorization as in Equation 2 [Rad+19]. Then, we process the tokens in parallel with this attention mask, thereby giving us the desired $O(S)$ time. Mathematically,

[1] σ-GPT is another AO-ARM architecture that fulfills these criteria. It is created by adopting an off-the-shelf GPT-style decoder architecture, with two positional encodings concatenated to each token: one for the current position, one for the next position in the permuted order [PCF24]. We focus on XLNet because at the time of writing, it's achieved more widespread community support (*e.g.*, Huggingface availability).



(a) **Parallel Sampling Attention Mask** (b) **Density Estimation Attention Mask**

Figure 1: **Attention Masks in AS-ARMs:** Red means that attention is not allowed, while green means that attention is allowed from the query to the key/value. The numbers on each token represent the generation order, where lower-numbered tokens come first. Panel 1a shows how, by attending to the same conditioning ("This", "is"), the tokens "machine" and "learning" can be generated in parallel. They do not attend to themselves nor each other. Panel 1b shows how we can conduct one-step density estimation on a sequence, with a permuted causal-like attention mask. The mask enforces the order "is", "This", "learning", "machine", where each token only attends to those decoded before it.

$$A_{\sigma(i),\sigma(j)} = \begin{cases} 0 & i \leq j \\ 1 & i > j \end{cases}, \tag{6}$$

where $A_{\sigma(i),\sigma(j)}$ is the masking matrix for the attention maps – 0 means that the query token at index $\sigma(i)$ is *not* allowed to attend to the key/value token at index $\sigma(j)$; 1 means that token $\sigma(i)$ *can* attend to token $\sigma(j)$. Such an attention mask can yield faithful estimates of $\log p(\mathbf{x}_{\sigma(\geq i)}|\mathbf{x}_{\sigma(<i)})$. See Figure 1b, and Appendix D for further discussion.

4.3 Two-for-One Model: Streamlined Drafting

Ideally, we do not want to train a separate draft model, because it takes extra memory on the hardware and additional training cost. Furthermore, the computations from the separate draft model cannot necessarily be re-used for the oracle. If the draft model was the same as the target/oracle model, we would not need to incur an extra memory nor training cost, and could cache the computations from the draft model to accelerate the target calculations.

Luckily, XLNet [Yan19] allows us to predict multiple tokens independently in parallel; we can use this conditionally independent guess (as in Equation 5) for the draft, like described in Section 4.1. We can feed this draft back through the same model (with computations for the already-visible tokens cached) to calculate the oracle density estimates, as in Section 4.2. The draft tokens would then be accepted or rejected based on their fidelity to the oracle density, as described in the next section.

5 How to Modify Speculative Decoding for AS-ARMs?

Now that we have established that there are suitable AS-ARMs architectures, we present *Any-Subset Speculative Decoding (ASSD)*, and prove some of its properties. See Algorithm 1 (in Appendix A). See Appendix B for proofs. In the resampling step, we use the notation $(f(x))_+ = \frac{\max(0,f(x))}{\sum_x \max(0,f(x))}$.

Lemma 1. *The first token speculated in each loop iteration will always be accepted. That is, Line 19's conditional always evaluates to true when $i = n$.*

Theorem 1. *Algorithm 1 requires no more than $N - m$ total function evaluations of $p(\cdot|\cdot)$. That is, there will never be more calls to a neural network than the number of tokens returned on Line 28.*

Based on Lemma 1 and Theorem 1 (see proof), we should always set $k > 2$ (where k is the number of speculated tokens per call to the draft model).

Theorem 2. *Algorithm 1 produces samples from the true joint distribution $p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$.*

We also present a variant of ASSD in Appendix E.6 with a context-derived n-gram as the draft model [Ste+24]. However, this variant does not fulfill Lemma 1.

6 Training and Implementation of Any-Subset Autoregressive Model

6.1 Arbitrary Masking Architecture

As expressed in Equation 6 and Section 4.2, we need a model that allows us to have "causal"-like attention masking in arbitrary orders, such that we can calculate the factorized joint probability in parallel. We use the 110M parameter case-sensitive version of XLNet from Huggingface [Wol+19]. Hyperparameters and datasets are in Appendix E.

6.2 Teacher-Forced Joint Loss

The original XLNet [Yan19] was only trained to predict 85 masked tokens in a sequence length of 512, corresponding to less than 20% of the sequence, which is not ideal for generative modeling tasks [Yan19]. We wish to have a model that can predict tokens almost from scratch, so we finetune the off-the-shelf XLNet model. For our finetuning objective, we maximize the joint conditional probability in Equation 2 with cross-entropy loss:

$$\max_\theta \mathbb{E}_{m\sim f(\cdot),\sigma\sim s(\cdot|m)} \left[\log p_\theta(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})\right], \tag{7}$$

where $f(\cdot)$ is a distribution over integers from $[0, N)$ (*i.e.*, sampling prompt length m), and $s(\cdot|m)$ is a distribution of permutations of integers from $[0, N)$ conditioned on prompt length m (*i.e.*, sampling token ordering σ), where N is the sequence length. The loss has three major components: (1) joint conditional distribution; (2) expectation over token orderings; (3) expectation over prompt lengths.

Joint Conditional Objective: To justify the joint conditional distribution $\log p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$, assume m and σ are fixed. We define a discrete time (absorbing state) Markov chain $\mathbf{x}, \mathbf{x}_{\sigma(<N-1)}, \mathbf{x}_{\sigma(<N-2)}, \ldots, \mathbf{x}_{\sigma(<m+1)}, \mathbf{x}_{\sigma(<m)}$, with time index $t \in \{0, 1, 2, \ldots, N-m-1, N-m\}$, as in the probabilistic graphical model depicted in Figure 2 (Appendix E.1). That is, $X_t = \mathbf{x}_{\sigma(<N-t)}$. To generate data, we follow the time reversal of this Markov chain. To obtain the time reversal, first consider reversing a singular time step, from t to $t-1$. This corresponds to learning

$$\log p_\theta(X_{t-1}|X_t) = \log p_\theta(\mathbf{x}_{\sigma(<N-(t-1))}|\mathbf{x}_{\sigma(<N-t)}) = \log p_\theta(\mathbf{x}_{\sigma(N-t+1)}|\mathbf{x}_{\sigma(<N-t)}). \tag{8}$$

That is, we predict the next token's density. To reverse the whole process, we sum for each timestep:

$$\begin{aligned}\sum_{t=1}^{N-m} \log p_\theta(\mathbf{x}_{\sigma(<N-t+1)}|\mathbf{x}_{\sigma(<N-t)}) &= \log p_\theta(\mathbf{x}_{\sigma(<N)}|\mathbf{x}_{\sigma(<m)}) \\ = \log p_\theta(\mathbf{x}_{\sigma(<m)}, \mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)}) &= \log p_\theta(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)}),\end{aligned} \tag{9}$$

which gives us Equation 7's joint conditional probability. This loss is different than the conditionally independent losses used in [SSE22] and discrete diffusion models [Sah+24; LME23]. Notably, their architectures, due to the lack of causal-like attention masking, could not support joint losses.

Expectations over Token Orderings and Prompt Lengths: In the objective (Equation 7), we do not make assumptions about the distribution of the prompt length $m \sim f(\cdot)$ nor the distribution of the prompt ordering $\sigma \sim s(\cdot|m)$ (so long as σ follows the decomposition protocol laid out in Section 2). In general, these distributions are task-dependent. For instance, in regular autoregressive tasks, σ would be deterministic, *i.e.*, the identity function. See Appendices E.3 and G for our realizations of $m \sim f(\cdot)$ and $\sigma \sim s(\cdot|m)$.

7 Experiments

Our first experiment (Section 7.1) empirically verifies that ASSD with AS-ARMs indeed preserves the output distribution while being faster, as predicted by our theoretical analysis. Our other experiments (Section 7.2) show that on both natural language and coding benchmarks, AS-ARMs, even with a fraction of the training resources, beat other model classes of comparable size, and are competitive with models orders of magnitude larger. Finally, Sections F and G contain ablations on the impact of number of tokens decoded in parallel, sequence length, KV caching, training strategy, etc.

Sampler	Gen PPL	Entropy	Model NFE	Aux NFE	Time (s)
Sequential	107.9 ± 1.6	7.65 ± 0.01	486.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	18.21 ± 0.00
ASSD (N-Gram)	111.7 ± 2.0	7.64 ± 0.01	$\mathbf{422.0 \pm 0.7}$	422.0 ± 0.7	16.80 ± 0.03
ASSD (Self)	107.6 ± 1.6	7.64 ± 0.01	$\underline{434.1} \pm 0.4$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{16.50 \pm 0.02}$

Table 1: **Comparison of Speculative and Sequential Decoding:** Left-to-right, entries show mean and standard error of generative perplexity (judge: GPT-2 Large), Shannon entropy, number of AS-ARM function evaluations, number of auxiliary draft model calls, and wall clock time. *ASSD (Self)* is from Algorithm 1. We also modify Algorithm 1 with context-derived *N-Grams* [Ste+24] as a draft model (see Appendix E.6). We set $k = 5$ for ASSD.

Model	Size	Code Tokens	Lang. Tokens	Pass @ 1
XLNet-Code	**110M**	**15B**	**0B**	38.59
DiffuLLaMA	6738M	19B	46B	**40.68**

Table 2: **Performance on HumanEval Infilling:** Comparison of code infilling abilities of XLNet finetuned on code versus DiffuLLaMA [Gon+24]. We use HumanEval's single-line infilling task [Bav+22], evaluated by pass@1 (each attempt counts, instead of only the best attempt) on 5165 trials.

7.1 Correctness and Speed of Any-Subset Speculative Decoding

We prompt the model with 640 masked sequences from the WikiText test dataset [Mer+16]. As in training, sequences are packed together into chunks of 512 tokens. We randomly mask out 95% of each sequence, leaving 5% of tokens (uniformly scattered throughout the sequence) as the prompt. We evaluate our finetuned model. See Table 1 for results. Appendix K shows sample outputs.

Empirical Correctness of Distribution: As expected from Theorem 2, the output distribution from speculative decoding is statistically the same as the output distribution from sequential decoding, measured by generative perplexity and entropy.

Speed-Up: Finally, both variants of ASSD (parallel sampling from self as draft, context n-gram [Ste+24] as draft) provide a statistically significant speedup over regular sequential decoding, as predicted by Lemma 1. (When decoding sequentially, the number of calls to the neural network is the same as the number of masked tokens.) ASSD with paralllel sampling is the fastest method, and requires the least total NFEs. ASSD with context n-gram is not that far behind: the intuition is that although it gives lower-quality drafts, it is very cheap. This phenomenon was also observed in similar studies for AR models [Ste+24]. However, only 1.15 tokens get generated per iteration when using context n-gram, as opposed to 2.24 tokens with parallel sampling as draft.

7.2 Infilling Benchmark Tasks

Code Generation: In coding, bidirectional context is crucial, making an appealing use case for AS-ARMs. Table 2 shows that AS-ARMs finetuned on code are competitive with models that are orders of magnitude larger. Furthermore, AS-ARMs use fewer training tokens. We evaluate against DiffuLLaMA as a representative baseline, as other discrete diffusion and autoregressive models of comparable size were shown to be non-competitive [Gon+24]. More details in Appendix E.8.

Natural Language: We also investigate whether AS-ARMs can outperform discrete diffusion models and regular autoregressive models on natural language infilling tasks, in which bidirectional context is key. We follow [Gon+24]'s setup on ROCStories [Mos+16]. Table 3 shows that XLNet is generally better than the baselines, while using the fewest parameters and modest training resources[2]. Off-the-shelf (OTS) XLNet without any finetuning is the best at infilling a single missing sentence. This corresponds to a $\sim 20\%$ masking ratio, which is roughly what it was trained on. Finetuning on a wider distribution of masking ratios as in Equation 7 splits finite model capacity among multiple

[2]To our knowledge, the pretraining details for the public weights of 110M parameter XLNet were never released. If we assume that it uses the same pretraining hyperparameters as those reported for XLNet-Large, it would have been trained on around two trillion tokens [Yan19]. GPT-2 also did not release their full training recipe [Rad+19].

Model	Size	Tokens	Infill 1/5		Infill 3/5	
			ROUGE 1/2/L	***NFE***	***ROUGE 1/2/L***	***NFE***
GPT2-S	127M	n/a[2]	$9.5/0.4/8.7$	10.7 ± 2.8	$13.5/0.6/10.2$	31.8 ± 6.0
SEDD-S	170M	210B	$11.6/0.8/10.7$	32.0 ± 0.0	$16.2/1.3/12.2$	64.0 ± 0.0
MDLM	130M	262B	$11.6/1.1/10.7$	32.0 ± 0.0	$13.3/1.0/10.4$	64.0 ± 0.0
DiffuGPT-S	127M	+130B	$14.0/1.5/13.0$	32.0 ± 0.0	$16.4/\mathbf{2.0}/\mathbf{14.2}$	64.0 ± 0.0
XLNet-OTS	***110M***	2T[2]	$\mathbf{14.4}/\mathbf{1.7}/\mathbf{13.1}$	$\mathbf{8.7} \pm 2.5$	$7.7/0.6/6.4$	$\mathbf{18.5} \pm 6.8$
XLNet-FT	***110M***	***+12B***	$13.1/1.1/12.0$	10.4 ± 2.8	$\mathbf{18.0}/1.4/13.2$	$\underline{30.4} \pm 6.0$

Table 3: **Performance on ROCStories Infilling:** We test on 1871 short stories of five sentences each, and get 5 completions (trials) per story. We compare ROUGE (↑). "Tokens" is the number of training tokens (excluding those for the pretrained initialization). "Infill 1/5" inputs sentences {1, 2, 4, 5} and infills {3}. "Infill 3/5" inputs sentences {1, 5} and infills {2, 3, 4}. We report $\mu \pm \sigma$ NFEs. "+" indicates additional finetuning tokens after the pretrained initialization. See footnote 2.

tasks. Unsurprisingly, when infilling three out of five sentences, our finetuned (FT) XLNet clearly surpasses the off-the-shelf XLNet, and is comparable to DiffuGPT-S.

8 Related Works

Any-Order Autoregressive Models: The defining characteristic of any-order autoregressive models is their ability to estimate the probability density of arbitrary marginal and/or conditional queries. This generalizes vanilla autoregressive models. We primarily base our work off of XLNet [Yan19] and MAC [SSE22]. From XLNet, we adopt the causal attention architecture: crucially, this allows us to estimate probability density in a parallelized single step, *i.e.*, $O(1)$ time [Yan19]. From MAC, we adopt recursive decomposition of queries on a binary lattice, which reduces the permutations needed to be learned from $N!$ to 2^N [SSE22]: this made MAC the first any-subset autoregressive model.

Although XLNet came in 2019, subsequent AO-ARMs [Hoo+21; SO21; SSE22] abandoned causal attention, in favor of full attention with absorbing state tokens to represent "masked" queries. This increases runtime of joint density estimation from $O(1)$ to $O(N)$: to satisfy the factorization rule, each token's density is estimated one-by-one so it conditions only on the preceding tokens.

Discrete Diffusion: Discrete diffusion models data generation as the reversal of an inhomogeneous continuous-time Markov chain (CTMC). Empirically, the best performing CTMC is an absorbing state process. Discrete diffusion can generate tokens in parallel, with arbitary prompt locations. The seminal work is D3PM [Aus+21]. Later, LDR [Cam+22] formalized discrete diffusion under the lens of CTMC theory. More recently, SEDD [LME23], MDLM [Sah+24], DiffuLLaMA [Gon+24], and LLaDA [Nie+25] created discrete diffusion models that rivaled or surpassed autoregressive models.

One limitation of discrete diffusion models is their inability to do joint density estimation. Also, although tokens can be generated in parallel with large discretized timesteps, they are generated under the conditional independence assumption, which may not adhere to the limiting joint distribution. Furthermore, the architectures perform full attention across all tokens, even the masks [LME23; Sah+24]; due to this lack of causal attention masking, it is not straightforward to apply KV-caching.

Speculative Decoding: Speculative decoding is an algorithm developed for autoregressive models that enables quick token generation from an inexpensive draft model. Via some techniques related to rejection sampling, the outputs provably adhere to the joint distribution of the target language model [LKM23; Che+23]. Further related works are in Appendix H.

9 Limitations and Future Works

We noticed performance saturating after ∼10B training tokens. Flash Attention [Dao+22] is also not currently supported. In principle, our AS-ARM training and sampling scheme can be applied to models that were adopted from GPT-style decoders [PCF24] (right now, it's on XLNet encoders), akin to adaptation of such models to discrete diffusion [Gon+24].

References

[Sha48] Claude E Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[UML14] Benigno Uria, Iain Murray, and Hugo Larochelle. "A deep and tractable density estimator". In: *International Conference on Machine Learning*. PMLR. 2014, pp. 467–475.

[Ger+15] Mathieu Germain et al. "Made: Masked autoencoder for distribution estimation". In: *International conference on machine learning*. PMLR. 2015, pp. 881–889.

[Mer+16] Stephen Merity et al. *Pointer Sentinel Mixture Models*. 2016. arXiv: 1609.07843 [cs.CL].

[Mos+16] Nasrin Mostafazadeh et al. "A corpus and cloze evaluation for deeper understanding of commonsense stories". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 839–849.

[LH+17] Ilya Loshchilov, Frank Hutter, et al. "Fixing weight decay regularization in adam". In: *arXiv preprint arXiv:1711.05101* 5 (2017), p. 5.

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[Gok+19] Aaron Gokaslan et al. *OpenWebText Corpus*. http://Skylion007.github.io/OpenWebTextCorpus. 2019.

[Rad+19] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[Wol+19] Thomas Wolf et al. "Huggingface's transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771* (2019).

[Yan19] Zhilin Yang. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *arXiv preprint arXiv:1906.08237* (2019).

[Aus+21] Jacob Austin et al. "Structured denoising diffusion models in discrete state-spaces". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17981–17993.

[Hoo+21] Emiel Hoogeboom et al. "Autoregressive diffusion models". In: *arXiv preprint arXiv:2110.02037* (2021).

[SO21] Ryan Strauss and Junier B Oliva. "Arbitrary conditional distributions with energy". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 752–763.

[Bav+22] Mohammad Bavarian et al. "Efficient training of language models to fill in the middle". In: *arXiv preprint arXiv:2207.14255* (2022).

[Cam+22] Andrew Campbell et al. "A continuous time framework for discrete denoising models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 28266–28279.

[Cha+22] Huiwen Chang et al. "Maskgit: Masked generative image transformer". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11315–11325.

[Dao+22] Tri Dao et al. "Flashattention: Fast and memory-efficient exact attention with io-awareness". In: *Advances in neural information processing systems* 35 (2022), pp. 16344–16359.

[Fri+22] Daniel Fried et al. "Incoder: A generative model for code infilling and synthesis". In: *arXiv preprint arXiv:2204.05999* (2022).

[SSE22] Andy Shih, Dorsa Sadigh, and Stefano Ermon. "Training and inference on any-order autoregressive models the right way". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 2762–2775.

[Ach+23] Josh Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[Che+23] Charlie Chen et al. "Accelerating large language model decoding with speculative sampling". In: *arXiv preprint arXiv:2302.01318* (2023).

[LKM23] Yaniv Leviathan, Matan Kalman, and Yossi Matias. "Fast inference from transformers via speculative decoding". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 19274–19286.

[Li+23] Raymond Li et al. “Starcoder: may the source be with you!” In: *arXiv preprint arXiv:2305.06161* (2023).

[LME23] Aaron Lou, Chenlin Meng, and Stefano Ermon. “Discrete diffusion language modeling by estimating the ratios of the data distribution”. In: (2023).

[Roz+23] Baptiste Roziere et al. “Code llama: Open foundation models for code”. In: *arXiv preprint arXiv:2308.12950* (2023).

[Tea+23] Gemini Team et al. “Gemini: a family of highly capable multimodal models”. In: *arXiv preprint arXiv:2312.11805* (2023).

[Tou+23] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[Zha+23] Jun Zhang et al. “Draft & verify: Lossless large language model acceleration via self-speculative decoding”. In: *arXiv preprint arXiv:2309.08168* (2023).

[Ank+24] Zachary Ankner et al. “Hydra: Sequentially-dependent draft heads for medusa decoding”. In: *arXiv preprint arXiv:2402.05109* (2024).

[Cai+24] Tianle Cai et al. “Medusa: Simple llm inference acceleration framework with multiple decoding heads”. In: *arXiv preprint arXiv:2401.10774* (2024).

[DG24] Justin Deschenaux and Caglar Gulcehre. “Beyond Autoregression: Fast LLMs via Self-Distillation Through Time”. In: *arXiv preprint arXiv:2410.21035* (2024).

[DWW24] Tianqi Du, Yifei Wang, and Yisen Wang. *Bridging Autoregressive and Masked Modeling for Enhanced Visual Representation Learning*. 2024. URL: `https://openreview.net/forum?id=KUz8QXAgFV`.

[Elh+24] Mostafa Elhoushi et al. “LayerSkip: Enabling early exit inference and self-speculative decoding”. In: *arXiv preprint arXiv:2404.16710* (2024).

[Fu+24] Yichao Fu et al. “Break the sequential dependency of llm inference using lookahead decoding”. In: *arXiv preprint arXiv:2402.02057* (2024).

[Gon+24] Shansan Gong et al. “Scaling Diffusion Language Models via Adaptation from Autoregressive Models”. In: *arXiv preprint arXiv:2410.17891* (2024).

[Liu+24a] Aixin Liu et al. “Deepseek-v3 technical report”. In: *arXiv preprint arXiv:2412.19437* (2024).

[Liu+24b] Sulin Liu et al. “Think While You Generate: Discrete Diffusion with Planned Denoising”. In: *arXiv preprint arXiv:2410.06264* (2024).

[PCF24] Arnaud Pannatier, Evann Courdier, and François Fleuret. “σ-GPTs: A New Approach to Autoregressive Models”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2024, pp. 143–159.

[Par+24] YH Park et al. “Jump your steps: Optimizing sampling schedule of discrete diffusion models”. In: *arXiv preprint arXiv:2410.07761* (2024).

[Sah+24] Subham Sekhar Sahoo et al. “Simple and Effective Masked Diffusion Language Models”. In: *arXiv preprint arXiv:2406.07524* (2024).

[Ste+24] Lawrence Stewart et al. “The N-Grammys: Accelerating Autoregressive Inference with Learning-Free Batched Speculation”. In: *arXiv preprint arXiv:2411.03786* (2024).

[Xu+24] Minkai Xu et al. “Energy-based diffusion language models for text generation”. In: *arXiv preprint arXiv:2410.21357* (2024).

[Zha+24] Yixiu Zhao et al. “Informed correctors for discrete diffusion models”. In: *arXiv preprint arXiv:2407.21243* (2024).

[Zhe+24] Kaiwen Zheng et al. “Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling”. In: *arXiv preprint arXiv:2409.02908* (2024).

[Nie+25] Shen Nie et al. “Large Language Diffusion Models”. In: *arXiv preprint arXiv:2502.09992* (2025).

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and summary are essentially shortened versions of our paper, except without extensive technical details.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The last paragraph contains a list of future directions, which are things our study did not get to address.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: See the appendix for proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have comprehensive details in the appendix. We will also release code upon acceptance.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Data and code will be released upon acceptance. It currently contains identifying information about the authors, so it is not in the submission.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Appendix E.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: See Table 1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See Appendix E.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Experiments were purely computational and conducted based on publicly available data.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: The negative societal impacts are the same as the negative societal impacts of any generative language model: disinformation, unsafe code generation, etc. The positive societal impacts are also generic: accelerated writing efficiency, etc.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: This is a proof-of-concept. The work has not been scaled to a level where it could be harmful.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We build our work upon open-source, and cite all relevant papers.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We have instructions on how to run the code. But again, the code is not released yet, since it contains some identifying author information.

    Guidelines:

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

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

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

    Answer: [No]

    Justification: We don't do human experiments.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: No human experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: The whole paper is about language models.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Any-Subset Speculative Decoding

Due to space limitations, we put the any-subset speculative decoding algorithm here.

Algorithm 1: Any-Subset Speculative Decoding

Input: k: number parallel tokens; N: target sequence length; σ: mapping of decoding order to 0-based positional index; $p(\cdot|\cdot)$: any-order autoregressive model; $\mathbf{x}_{\sigma(<m)}$: prompt tokens

Output: $\mathbf{x}_{\sigma(\geq m)}$: the predicted tokens

1 $n \leftarrow m$ // number of tokens we've already decoded
2 **while** *n < N* **do**
3 $t \leftarrow \min(n + k, N)$
4 // speculate the next t tokens
5 **(Parallelized): for** $i \in [n : t)$ **do**
6 $\tilde{x}_{\sigma(i)} \sim p(\cdot|\mathbf{x}_{\sigma(<n)})$ // sample from partially conditioned distribution
7 $p_{\sigma(i)} \leftarrow p(\tilde{x}_{\sigma(i)}|\mathbf{x}_{\sigma(<n)})$ // get partially conditioned density
8 **end**
9 **if** *n == T - 1* **then**
10 $x_{\sigma(n)} \leftarrow \tilde{x}_{\sigma(n)}$ // accept the proposal
11 **return** $\mathbf{x}_{\sigma(\geq m)}$
12 **end**
13 **(Parallelized): for** $i \in [n : t)$ **do**
14 $q_{\sigma(i)} \leftarrow p(\tilde{x}_{\sigma(i)}|\mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:i)})$ // get ground truth density
15 **end**
16 // rejection sampling
17 **for** $i \in [n : t)$ **do**
18 $r \sim \mathcal{U}[0, 1]$
19 **if** $r < min(1, \frac{q_{\sigma(i)}}{p_{\sigma(i)}})$ **then**
20 $x_{\sigma(i)} \leftarrow \tilde{x}_{\sigma(i)}$ // accept the proposal
21 **else**
22 $x_{\sigma(i)} \sim \left(p(\cdot|\mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:i)}) - p(\cdot|\mathbf{x}_{\sigma(<n)})\right)_+$ // resample
23 **exit** from for loop
24 **end**
25 **end**
26 $n \leftarrow i + 1$ // update number of decoded tokens
27 **end**
28 **return** $\mathbf{x}_{\sigma(\geq m)}$

B Proofs

Line numbers refer to Algorithm 1.

Lemma 1. *The first token speculated in each loop iteration will always be accepted.*

Proof. When $i = n$ on Line 13,

$$q_{\sigma(i)} = p(\tilde{x}_{\sigma(i)}|\mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:i)}) \tag{10}$$
$$= p(\tilde{x}_{\sigma(i)}|\mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:n)}) \tag{11}$$
$$= p(\tilde{x}_{\sigma(i)}|\mathbf{x}_{\sigma(<n)}) \tag{12}$$
$$= p_{\sigma(i)}. \tag{13}$$

So, $r < \frac{q_{\sigma(i)}}{p_{\sigma(i)}} = 1$ on Line 19, and thus, when $i = n$, $\tilde{x}_{\sigma(n)}$ will always be accepted on Line 20. □

Theorem 1. *Algorithm 1 requires no more than $N - m$ total function evaluations of $p(\cdot|\cdot)$. That is, there will never be more calls to a neural network than the number of tokens decoded.*

Proof. The idea underlying this proof is that each iteration of the while loop on Line 2 has two function evaluations. The first function evaluation is on Lines 5 to 8, to speculate tokens. The second function evaluation is on Lines 13 to 15, to evaluate the oracle density.

We then just need to show that at least two tokens will be decoded, making for minimum one token generated per function evaluation.

By Lemma 1, the first token (at $i = n$) is always accepted. So, we can move on to $i = n + 1$.

When $i = n + 1$, regardless of whether we accept or reject $\tilde{x}_{\sigma(n+1)}$, we will still obtain a value for $x_{\sigma(n)}$, whether it is $\tilde{x}_{\sigma(n)}$ on Line 20 or a resampling from the adjusted $()_+$ distribution on Line 22 (which does not require additional calls to $p(\cdot|\cdot)$, as it makes use of already calculated distributions from Lines 6 and 14).

So, on each loop iteration, we are guaranteed to get token values for at least $x_{\sigma(n)}, x_{\sigma(n+1)}$, thereby giving us the requisite two tokens.

On the last loop iteration, if $n = N - 1$, there is an edge case where only one token can be decoded (Line 9). But, as previously shown in Lemma 1, this speculated token is mathematically guaranteed to be accepted, since it was the first and only one to be speculated. So, we can forgo the verification step for this final loop iteration. Thus, to generate this final token, we still only need one function evaluation, maintaining the lower bound of one token generated per function evaluation. □

Theorem 2. *Algorithm 1 produces samples from the true joint distribution* $p(\mathbf{x}_{\sigma(\geq m)}|\mathbf{x}_{\sigma(<m)})$.

Proof. The only differences between our algorithm and speculative decoding are:

1. The use of p as its own speculator. This is valid, because speculative decoding is agnostic to the speculator function, as long as it gives probability density estimates.
2. The omission of the verification step when decoding the last token $n = T - 1$ on Line 9. This is justified by Lemma 1, which shows that when $i = n = T - 1$, $\tilde{x}_{\sigma(n)}$ would always meet the acceptance criteria if it were to go through the verification on Line 19.
3. We do not sample an extra token if all of the speculations are accepted at the end of each loop iteration. However, this does not affect the distribution of the other tokens outputted. We skip this extra token sampling because, as shown in Lemma 1, the first token speculated in the next iteration will always be accepted, which "makes up" for the omission of the extra sample.

Other than these inconsequential differences, this algorithm is simply speculative decoding, but with a different ordering than $\sigma = [0, 1, 2, \ldots, N - 1]$. We can permute/sort $\mathbf{x}$ in increasing order of $\sigma(i)$, and define as the result $\mathbf{y}$. Then, our algorithm is mathematically equivalent to speculative decoding with the inputs $\mathbf{y}$ as the sequence, and the draft model the same as the target model. The rest of the proof of correctness of our algorithm is therefore the same as in [LKM23; Che+23]. For completeness, we reproduce their proof in Appendix C, with some additional commentary. □

C Modified Rejection Sampling Correctness

We provide an extended proof of correctness for Theorem 2, as it relates to the modified rejection sampling step in Algorithm 1. Note that the main ideas and layout of this proof are copied from [Che+23] and [LKM23]. We only include it here for the reader's convenience and consistency with our notation.

Proof. WLOG, consider the random variable (token) $x_{\sigma(i)}$, returned by Algorithm 1 (*ASSD*). Consider the iteration where $n \leq i < n + k \leq N$, *i.e.*, the iteration that generates $x_{\sigma(i)}$.

For the algorithm to be correct, we desire that $\mathbb{P}(x_{\sigma(i)} = x) = p(x_{\sigma(i)} = x|\mathbf{x}_{\sigma(<i)})$, where $\mathbb{P}(x_{\sigma(i)} = x)$ is the probability that *ASSD* generates token value x at position $\sigma(i)$, and $p(x_{\sigma(i)} = x|\mathbf{x}_{\sigma(<i)})$ is the probability that regular sequential decoding would generate x at position $\sigma(i)$. In other words, the distribution of *ASSD* is the same as in sequential decoding. Note that $\mathbf{x}_{\sigma(<i)} = \mathbf{x}_{\sigma(<n)} \oplus \tilde{\mathbf{x}}_{\sigma[n:i)}$

in both sequential decoding and our algorithm, because they both use generations from previous iterations (in addition to the prompt) as conditioning for future iterations.

Let $\tilde{x}$ (shorthand for $\tilde{x}_{\sigma(i)}$) be the token value generated from the conditionally independent draft in Line 6. When $x_{\sigma(i)} = x$ is true, there are two mutually exclusive and collectively exhaustive possibilities (from the if-else statement): (1) $\tilde{x}$ was accepted, and $\tilde{x} = x$ (2) $\tilde{x}$ was rejected, and x was the result of the resampling.

$$\begin{aligned}\mathbb{P}(x_{\sigma(i)} = x) &= \mathbb{P}(\tilde{x} \text{ accepted}, \tilde{x} = x) + \mathbb{P}(x_{\sigma(i)} = x, \tilde{x} \text{ rejected}) \\ &= \mathbb{P}(\tilde{x} \text{ accepted } | \tilde{x} = x)\mathbb{P}(\tilde{x} = x) + \mathbb{P}(\tilde{x} \text{ rejected})\mathbb{P}(x_{\sigma(i)} = x | \tilde{x} \text{ rejected})\end{aligned} \tag{14}$$

Analyzing the first term, we look to the "if" (accept) clause on Line 19:

$$\begin{aligned}\mathbb{P}(\tilde{x} \text{ accepted } | \tilde{x} = x)P(\tilde{x} = x) &= \min\left(1, \frac{q_{\sigma(i)}(x)}{p_{\sigma(i)}(x)}\right) p_{\sigma(i)}(x) \\ &= \min\left(p_{\sigma(i)}(x), q_{\sigma(i)}(x)\right)\end{aligned} \tag{15}$$

Here, $q_{\sigma(i)}(x) = p(x_{\sigma(i)} = x | \mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:i)})$, *i.e.*, the oracle density from Line 14; and $p_{\sigma(i)}(x) = p(x_{\sigma(i)} = x | \mathbf{x}_{\sigma(<n)})$, *i.e.*, the speculator density from Line 7.

Regarding the second term, we look at the "else" (reject) clause on Line 21. We analyze each item in the product separately, as the expressions are long:

$$\begin{aligned}\mathbb{P}(\tilde{x} \text{ rejected}) &= 1 - \mathbb{P}(\tilde{x} \text{ accepted}) \\ &= 1 - \sum_{x'} \mathbb{P}(x_{\sigma(i)} = x', \tilde{x} \text{ accepted}) \\ &= 1 - \sum_{x'} \min\left(p_{\sigma(i)}(x'), q_{\sigma(i)}(x')\right) \\ &= \sum_{x'} q_{\sigma(i)}(x') - \sum_{x'} \min\left(p_{\sigma(i)}(x'), q_{\sigma(i)}(x')\right) \\ &= \sum_{x'} q_{\sigma(i)}(x') - \min\left(p_{\sigma(i)}(x'), q_{\sigma(i)}(x')\right) \\ &= \sum_{x'} \max\left(q_{\sigma(i)}(x') - p_{\sigma(i)}(x'), q_{\sigma(i)}(x') - q_{\sigma(i)}(x')\right) \\ &= \sum_{x'} \max\left(q_{\sigma(i)}(x') - p_{\sigma(i)}(x'), 0\right)\end{aligned} \tag{16}$$

From Line 22:

$$\begin{aligned}\mathbb{P}(x_{\sigma(i)} = x | \tilde{x} \text{ rejected}) &= \left(p(x_{\sigma(i)} = x | \mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:i)}) - p(x_{\sigma(i)} = x | \mathbf{x}_{\sigma(<n)})\right)_+ \\ &= \left(q_{\sigma(i)}(x) - p_{\sigma(i)}(x)\right)_+ \\ &= \frac{\max\left(q_{\sigma(i)}(x) - p_{\sigma(i)}(x), 0\right)}{\sum_{x'} \max\left(q_{\sigma(i)}(x') - p_{\sigma(i)}(x'), 0\right)}\end{aligned} \tag{17}$$

Following Equations 16 and 17, we have

$$\mathbb{P}(\tilde{x} \text{ rejected})\mathbb{P}(x_{\sigma(i)} = x | \tilde{x} \text{ rejected}) = \max\left(q_{\sigma(i)}(x) - p_{\sigma(i)}(x), 0\right) \tag{18}$$

Putting it all together back into Equation 14,

$$\begin{aligned}\mathbb{P}(x_{\sigma(i)} = x) &= \min\left(p_{\sigma(i)}(x), q_{\sigma(i)}(x)\right) + \max\left(q_{\sigma(i)}(x) - p_{\sigma(i)}(x), 0\right) \\ &= q_{\sigma(i)}(x) \\ &= p(x_{\sigma(i)} = x | \mathbf{x}_{\sigma(<n)}, \tilde{\mathbf{x}}_{\sigma[n:i)}) \\ &= p(x_{\sigma(i)} = x | \mathbf{x}_{\sigma(<i)})\end{aligned} \tag{19}$$

The last line of Equation 19 is true, because to have gotten to index i in the accept-reject loop (Line 17), we must have accepted $\tilde{\mathbf{x}}_{\sigma[n:i)}$, *i.e.*, $\mathbf{x}_{\sigma[n:i)} = \tilde{\mathbf{x}}_{\sigma[n:i)}$. Therefore, we have shown that *ASSD* gives the same per-token distribution as in sequential decoding. Induction over $i \in [m : N)$ will easily show that *ASSD* gives the correct joint distribution.

□

D Causal-Like Attention Masking

We provide further discussion on the masking introduced in Section 4.2.

In particular, there is a subtlety in attention mechanisms: even if a token does not attend to itself, it still "sees" its own content because it constructs a query representation to compare against the keys. At first glance, this seemingly breaks the ban (Equation 6) on a token attending to itself. The solution is to separate the positional and content information into two streams. The positional stream (which only has positional representations as input) is used to calculate the queries, *i.e.*, row index of the attention map. The content stream (which has token value information) is used to calculate the key/value information, *i.e.*, column index of the attention map. Thus, the evaluation of the positional queries does not "cheat" by looking at the ground truth content at its own position. Conceptually, this can be said to be more like cross-attention than the self-attention common in discrete diffusion models [Aus+21].

In contrast, architectures commonly used for discrete diffusion models [LME23; DG24; Sah+24; Aus+21] and other any-order autoregressive models [SSE22; Hoo+21] do not support this kind of masking. In these architectures,

$$\forall i, j : A_{\sigma(i),\sigma(j)} = 1, \tag{20}$$

which means that every token is allowed to attend to every token including itself when calculating its probability. From a probabilistic graphical modeling perspective, the outputted probabilities very roughly correspond to $\log p(x_{\sigma(i)}|\mathbf{x})$, which would *not* give us principled estimates on the already-visible tokens.

E Additional Experimental Details

E.1 Loss Function Probabilistic Graphical Model

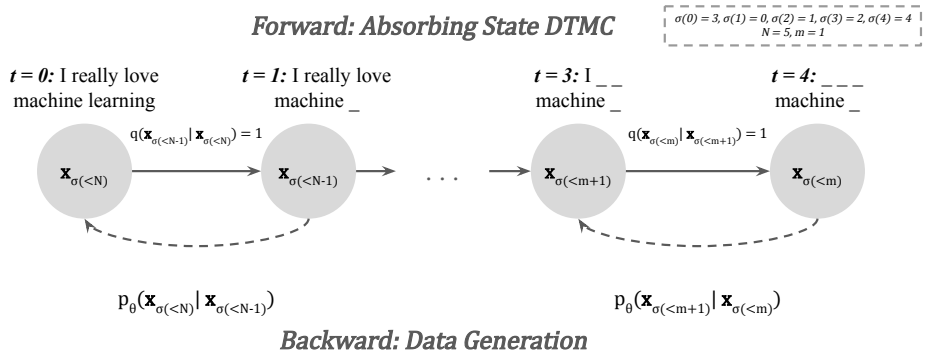


Figure 2: **Probabilistic Graphical Model:** Shows the discrete-time Markov chain for the forward noising process, and its time reversal (*i.e.*, data generation). This justifies Equation 7.

E.2 Dataset

We finetune on the OpenWebText dataset [Gok+19]. Following [Sah+24], we pack the sequences together, and split them into chunks of 512 tokens, based on XLNet's case-sensitive tokenizer with a vocabulary of $32,000$ possible tokens. We have separator tokens to delineate the start of a new document.

E.3 Mask Distribution

We are interested in generating text from near-scratch, so we train a model where $m \sim \mathcal{U}[0.01 * N, 0.10 * N]$. To get σ, we first sample $\sigma_{\text{pre}} \sim \mathcal{U}(S_N)$, where S_N represents all the possible permutations of the integers from $[0, N)$. But, remembering Section 2's efficient masking protocol, we sort each of $\sigma_{\text{pre}}(< m)$ and $\sigma_{\text{pre}}(\geq m)$ in ascending order to get σ. This eliminates the ambiguity in paths the model has to learn to calculate a given joint, while still maintaining which positions are in the prompt and which need to be predicted.

We also use a similar low-discrepancy sampler as [Sah+24] to reduce variance among prompt lengths within a batch.

E.4 Training Hyperparameters

In finetuning XLNet, we use maximum learning rate 10^{-4} and batch size 320 (16 per device, 4 accumulations, 5 devices). We have linear learning rate warmup for 5000 steps, and linear learning rate decay for $70,000$ additional steps, making for a total of $2.4 * 10^7$ samples seen. (Notably, this is far fewer than the $2.56 * 10^8$ samples that DiffuGPT-S saw [Gon+24].) We start at 15% masking rate, then linearly increase the minimum masking rate to 90% and the maximum masking rate to 99% over 5000 steps.

We train on NVIDIA RTX A6000 Ada devices, which have 48 GB of RAM each. Training lasts around four days. We calculate validation loop metrics every 500 steps on 64 samples. We use the AdamW [LH+17; KB14] optimizer.

We also tried training for more epochs, but we found that performance saturated early on.

E.5 Metrics

We are focused on generation quality. To that end, we have two metrics.

To quantify how likely the generated outputs of our model are, we calculate generative perplexity, where lower values are better. It is calculated as

$$\text{PPL}_{\text{gen}} = \left(e^{\sum_i \log q(x_i|\mathbf{x}_{0\ldots i-1})}\right)^{-1/N}, \tag{21}$$

where q is an oracle language model (we use GPT-2 Large [Rad+19]), and $\mathbf{x}$ is a generated sequence.

To quantify the diversity in tokens, we calculate Shannon entropy [Sha48], where higher values are better. The formula is

$$H(\mathbf{x}) = -\sum_{x_i \in \mathbf{x}} \log_2(p(x_i))p(x_i), \tag{22}$$

where $p(x_i) = \frac{|\{j|\mathbf{x}_j = x_i\}|}{|\mathbf{x}|}$, *i.e.*, the frequency of a token in the sequence.

Typically, there is a trade-off between generative perplexity and Shannon entropy – highly repetitive sequences of common words like "a" have low generative perplexity, but also low entropy. Random sequences of gibberish have high entropy and high generative perplexity. We want to generate sequences with high entropy and low generative perplexity.

E.6 Speculative Decoding Variants

We compare sequential decoding to two variants of ASSD: the first is described in Algorithm 1. The second uses context-based n-grams, which were initially proposed for left-to-right speculative decoding [Ste+24], although they easily fit into our framework. To adopt it to arbitrary order, we replace the speculator $p(\cdot|\cdot)$ in Lines 5-8 of Algorithm 1 with a context-based n-gram model $c(\cdot|\cdot)$, as in Algorithm 2. There, $c(a|b)$ is the probability over the partially decoded sequence that a bigram starting in b ends in a, as in Equation 23:

$$c(a|b) = \frac{\sum_{i, x_i \neq \texttt{MASK}, x_{i+1} \neq \texttt{MASK}} \mathbb{1}[\mathbf{x}_{i,i+1} = (a, b)]}{\sum_{j, x_j \neq \texttt{MASK}} \mathbb{1}[x_j = b]}, \tag{23}$$

We initialize $c(\cdot|\cdot)$ by sweeping over the prompt, then update it iteratively, as the sequence is decoded.

Algorithm 2: Speculation with Context-Based N-Grams

Input: same as Algorithm 1
Output: see Algorithm 1

for $i \in [n : t)$ **do**
  if $x_{\sigma(i)-1} \neq \texttt{MASK}$ **then**
    $x_{\text{cond}} \leftarrow x_{\sigma(i)-1}$
  else
    $x_{\text{cond}} \leftarrow \tilde{x}_{\sigma(i)-1}$
  end
  $\tilde{x}_{\sigma(i)} \sim c(\cdot|x_{\text{cond}})$ // sample from partially conditioned distribution
  $p_{\sigma(i)} \leftarrow c(\tilde{x}_{\sigma(i)}|x_{\text{cond}})$ // get partially conditioned density
end

Proposition 1. *When $i \geq 1$, Algorithm 2 always sets x_{cond} to a valid non-*`MASK` *value.*

Proof. We conduct strong induction on $i \geq m$, where m is the given prompt length.

For a given value of i, we have two cases:

1. $x_{\sigma(i)-1} \neq \texttt{MASK}$. This case is trivial.
2. $x_{\sigma(i)-1} = \texttt{MASK}$. This reduces to showing that $\tilde{x}_{\sigma(i)-1} \neq \texttt{MASK}$. Firstly, there exists some j such that $\sigma(j) = \sigma(i) - 1$, which trivially means $\sigma(j) < \sigma(i)$. From Equation 4, this is equivalent to saying $j < i$. By the inductive hypothesis, $\tilde{x}_{\sigma(j)} \neq \texttt{MASK}$ should have been speculated for all $j < i$.

□

We note that context-based n-grams lack the guarantees of Lemma 1, so they could increase the total number of function evaluations (AS-ARMs + n-gram draft) beyond the sequence length. However, n-grams are typically cheap to evaluate, which could make up for the increased function evaluations.

E.7 Infilling Benchmark

We run 5 trials over the dataset of 1871 stories, making for 9355 samples per masking level (short or long) per model. The models compared are: GPT-2 [Rad+19], MDLM [Sah+24], SEDD [LME23], DiffuGPT [Gon+24], XLNet-OTS (off-the-shelf) [Yan19], and XLNet-Finetuned (finetuned by us). We do not compare against models like LLaDA [Nie+25] because they have access to significantly more training budget than us, not to mention their orders-of-magnitude larger model size.

For the XLNet models, we use ASSD. We set $k = 15$ for our speculative decoding. Without speculative decoding on XLNet, "Infill 1/5" requires 10.9 ± 2.9, and "Infill 3/5" requires 32.4 ± 6.2 evaluations. For GPT, we use sequential decoding, since it cannot be its own speculator. Following [Gon+24], we only give GPT the left conditioning, as it is not straightforward to give it rightward conditioning without instruction-tuning. This experiment can be run on a 16GB GPU.

E.8 Code Generation

We finetune XLNet on 14.7 billion tokens from Starcoder's Python data [Li+23]. This corresponds to $75,000$ training steps with a batch size of 384 (16 per device, 4 accumulation steps, 6 GPUs) of 512 tokens each. We started from learning rate 0, warmed up for $20,000$ steps to learning rate $1.2 * 10^{-4}$, then scheduled to decay linearly for $63,333$ steps (which corresponds to $32 * 10^6$ total samples seen), although the run crashed slightly earlier than this. We start from a 5% masking rate, then warm up over $20,000$ steps to a 15% minimum masking rate, and a 99% maximum masking rate, sampled uniformly.

Since the XLNet tokenizer does not support whitespaces, we replace whitespace tokens with special characters during training (\n : <cls>, \t : <sep>, "_ _" : <unk>, "_ _ _" : <pad>), where _ is the

space. When generating code, we reverse the special character mapping and truncate the leading space from the generated lines, since the tokenizer by default inserts a leading space to each word.

We evalute on the HumanEval single-line infilling task on 1033 Python test cases, following [Gon+24]. We generated 5 completions for each test case, for a sample size of 5165. We evaluate with the pass@1 metric, *i.e.*, we count the failure or success of each of the five completions for a test case (rather than taking the best out of the five completions).

For the baseline, we compare the publicly available DiffuLLaMA [Gon+24] model. We find that the results from [Gon+24] are actually under-reported. When manually inspecting the outputs, we found that there were often generations that were correct, except that the number of leading spaces was off by one. Our investigation suggests that the LLaMA 7B tokenizer also seems to have an issue with counting prefix spaces, etc. Since this is not a problem with model capacity, we relaxed the evaluation, and manually inserted the ground truth indentation (*i.e.*, correct number of leading spaces) to DiffuLLaMA's outputs. After doing so, the pass@1 rate increased from about 16% to 40%.

This experiment can be run on a 16GB GPU.

F Additional Results

F.1 Any-Subset Speculative Decoding: Comparison of Finetuned and Off-the-Shelf

Sampler	Gen PPL	Entropy	NFEs	Time (s)
Sequential	59.08 ± 5.61	3.119 ± 0.065	486.0 ± 0.0	18.04 ± 0.00
Speculative	59.12 ± 5.26	3.206 ± 0.064	247.3 ± 1.8	9.36 ± 0.07
Difference	$+0.08\%$	$+2.78\%$	-49.12%	-48.09%

Table 4: **Comparison of ASSD (Algorithm 1) and Sequential Decoding in Off-the-Shelf Model:** The entries show mean and standard error of generative perplexity (judge: GPT-2 Large), Shannon entropy, number of network function evaluations, and wall clock time. Metrics are calculated over 640 decoded WikiText sequences of length 512, where 95% is randomly masked out. We set $k = 5$ in speculative decoding.

See Table 4. This shows extended results from Table 1, with the off-the-shelf model from Huggingface.

While the generative perplexity of the off-the-shelf model is much lower than the generative perplexity of our finetuned model, this comes at the cost of very low entropy. Indeed, when we manually inspected the outputs, we found that the off-the-shelf model generated highly repetitive, nonsensical sequences of a few common words (see Appendix J). On the other hand, the finetuned model generated sentences that were, for the most part, coherent both semantically and syntatically (see Appendix K).

Additionally, the off-the-shelf model gains a larger boost in runtime from speculative decoding. Examining the outputted sequences, it again appears to be due to its highly repetitive output distributions, as these would be easier to speculate with a mean field model.

F.2 How Many Tokens to Speculate?

We have ablations on the effect of k in Table 5. Generally, our selected value of $k = 5$ was a close-to-optimal choice. The difference in perplexity and entropy, according to t-tests, is also not statistically significant, as predicted by Theorem 2 (which does not depend on k). Results in the below tables are calculated from 750 sequences, where we infill 95% of the tokens per sequence of length 128 and 256.

As we had predicted in Section 5, a choice of $k = 2$ slowed down the algorithm, since the algorithm, no matter what, is guaranteed to generate at least two tokens (with two function evaluations: speculate and verify) per loop iteration. So, if only two tokens are speculated, there could never be a speedup over the sequential capabilities, and in fact, the overhead time from running speculative decoding would slow it down.

k	2	3	4	5	6	8	10	15	20
Length: 128									
Speedup (%)	-4.0	6.7	8.8	8.9	9.1	8.7	8.4	9.0	9.0
PPL p-Value	0.66	0.37	0.22	0.49	0.12	0.41	0.48	0.22	0.83
Entropy p-Value	0.14	0.20	0.73	0.48	1.00	0.28	0.76	0.52	0.17
Length: 256									
Speedup (%)	-2.7	6.6	8.4	8.8	9.3	8.9	8.4	8.5	8.2
PPL p-Value	0.31	0.03	0.32	0.13	0.47	0.74	0.77	0.39	0.78
Entropy p-Value	0.43	0.60	0.49	0.15	0.17	0.48	0.56	0.95	0.22

Table 5: **Tuning k:** Generally, ASSD is not sensitive to the value of k (number of tokens speculated per call to draft model), so long as it is at least 4.

F.3 Effect of Sequence Length

Length	Speedup (%)	PPL p-value	Entropy p-value
128	8.94	0.49	0.48
256	8.84	0.13	0.15
512	9.44	0.71	0.96
640	10.45	0.23	0.24
768	10.59	0.80	0.35
896	10.90	0.94	0.48
1024	11.00	0.78	0.50

Table 6: **Effect of Sequence Length on Speedup:** Set $k = 5$, infill 95% of the sequence.

Interestingly, we find out that as we increase the sequence length, the speedup becomes more pronounced. The differences in output distribution from sequential decoding are not statistically different, as predicted by Theorem 2. See Table 6.

F.4 KV-Cache

We generally found that KV caching was not helpful for smaller model sizes and sequence lengths. But, it does help as we scale up the problem size. When using a 340M parameter XLNet (off-the-shelf, as we did not have the resources to retrain) to infill 128 tokens out of sequence length 1024 (as opposed to 110M parameter on sequence length 512), we observe that KV caching yields a 50% speedup in model inference time. See Table 7.

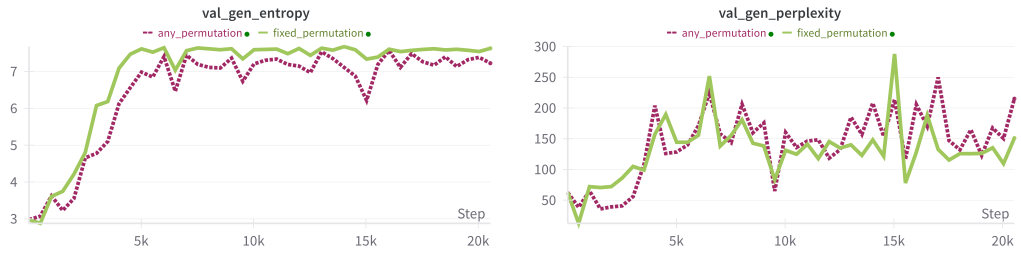
Caching	Time (s)
KV Cache	0.0706 ± 0.0006
No Cache	0.1407 ± 0.0004

Table 7: **KV Cache Benefits:** Average inference time of the model (per parallel speculation NFE), averaged over 100 trials (KV cache, versus not using KV cache). Results on 340M-parameter XLNet to infill 128 out of 1024 tokens.

G Ablation

G.1 Mask Decomposition Ablation

Figure 3 shows the ablation of mask decomposition protocol described in Section 2. The entropy (generation diversity) is consistently better when training with the recursive binary mask decomposition protocol, while the generative perplexity is about the same.



(a) **Entropy** (↑) (b) **Generative Perplexity** (↓)

Figure 3: **Fixed (Recursive Binary Lattice) Versus Any Permutation Mask Decomposition:** Validation loop metrics on generated sequences from each training strategy. The curves shown are for models trained with an effective batch size of 96 across four NVIDIA RTX A4000 devices. Each validation iteration has 36 sequences of length 512 tokens.

(a) **Entropy** (↑) (b) **Generative Perplexity** (↓)

Figure 4: **Narrow ($1\% \to 10\%$) Versus Wide ($1\% \to 85\%$) Prompting Rates:** Validation loop metrics on generated sequences from each training strategy, as it relates to the distribution of prompt lengths in the train set. The curves shown are for models trained with an effective batch size of 320 across five NVIDIA RTX 6000 Ada devices. Each validation iteration has 64 sequences of length 512 tokens from OpenWebText, where the task is to infill 95% of the masked sequence given a 5% prompt.

G.2 Impact of Masking Distribution

Figure 4 shows the effect of the realization of the distribution of prompt (*i.e.*, masking) length from Section E.3. Since the validation task is to infill 95% of a masked sequence, it is expected that training the model exclusively on shorter prompt lengths would be better than training the model on a mixture of long and short prompt lengths (which would dilute its capacity). Indeed, we see that, with respect to generative perplexity, the model trained with $m \sim \mathcal{U}[0.01, 0.10]$ outperforms the model trained with $m \sim \mathcal{U}[0.01, 0.85]$, where m is the percentage of the source text that is given as prompt (*i.e.*, *un*masked). On entropy, the short prompt training strategy performs marginally better at low training steps, but the gap closes as training continues.

H Additional Related Works

H.1 Any-Order Autoregressive Models

There are some works from the pre-transformer era [Vas+17] that deal with the problem of arbitrary density estimation, namely NADE [UML14] and MADE [Ger+15]. AO-ARMs also bear some resemblance to generative, transformer-based, masked image models [Cha+22; DWW24].

H.2 Discrete Diffusion Models

There is also work on improving sampling. EDLM [Xu+24] learns a partition function to allow parallel sampling, while ensuring that the generated tokens adhere to a desired joint distribution. DDPD [Liu+24b] learns a planner network to optimize the step size taken with uniform diffusion models. JYS [Par+24] also works on optimizing the step schedule, although their method involves an optimization problem over (part of) the training dataset. [Zha+24] work on predictor-corrector methods and introduce the $k-$Gillespie sampling algorithm. SDTT [DG24] distilled discrete diffusion models to achieve comparable generative performance with fewer sampling steps. However, [Zhe+24]'s work revealed a numerical precision error with discrete diffusion sampling that leads to unintentional low-temperature sampling.

H.3 Speculative Decoding

Simple draft models include context-based and model-based n-grams [Ste+24]. Lookahead Decoding also relies upon parallel generation of n-grams, and does not need to train an additional draft model [Fu+24]. Hydra [Ank+24] and Medusa [Cai+24] augment the target language model with lightweight heads to quickly predict additional tokens. The advantage of this approach, like ours, is that at least two tokens are guaranteed to be accepted on each loop iteration: the first token comes from the base model; if the next token is rejected, it is resampled from a combination of the proposal and oracle distributions. There are also works [Zha+23] that, similar to ours, use the same model for drafting and verification; one such work is LayerSkip [Elh+24]. This is known as self-speculative decoding. A difference is that these methods skip model layers, while we skip inputs.

The only work that we know of that uses a speculative decoding-like algorithm for non-left-to-right models is σ-GPT [PCF24]. However, their algorithm actually violates Theorem 1 and Theorem 2, meaning that it is not theoretically guaranteed to produce tokens from the correct target distribution, and can slow down sampling. See Appendix I for details.

I Comparison to σ-GPT

Algorithm 3: Token-based rejection sampling, reprinted from σ-GPT [PCF24]

Input: T: minimum target length, y: any-order autoregressive model, N_o: number of orderings to sample, $\mathbb{X}$: prompt of length t_0

1 $t \leftarrow t_0$
2 **while** $t < T$ **do**
3 In parallel, compute distribution conditioned on prompt $p(x_i|\mathbb{X}), \forall i \in t, \ldots, T$
4 In parallel, sample at every position $\tilde{x}_i \sim p(x_i|\mathbb{X}), \forall i \in t, \ldots, T$
5 Draw N_o random order σ and in parallel, compute all logits $q\left(x_i|\mathbb{X}, \tilde{x}_{\sigma(<i)}\right), \forall i \in t, \ldots, T$
6 In parallel sample $T - t$ variables $u_i \sim \mathcal{U}[0, 1], \forall i \in t, \ldots, T$ from a uniform distribution.
7 In parallel, compute the acceptance decision $a_i = u_i < \min\left(1, \frac{q\left(\tilde{x}_i|\mathbb{X}, \tilde{x}_{\sigma(<i)}\right)}{p(\tilde{x}_i|\mathbb{X})}\right)$ for every order.
8 Select the order that accepts the most tokens before seeing a first rejection.
9 Keep that order and add the a accepted tokens before the first rejection to the prompt.
10 Set $t = t + a$
11 **end**

σ-GPT is a work that superficially seems to have a similar sampling algorithm as ours for any-order autoregressive models [PCF24]. However, as we will see, their algorithm (Algorithm 3) actually has subtle yet critical mistakes that lead to violations of Theorem 1 and Theorem 2, removing the theoretical guarantees.

I.1 Ambiguous Factorization Order and Joint Distribution

Firstly, their algorithm does *not* provably give the correct joint distribution, because they randomly sample multiple factorization orders (Lines 5, 8) at each draft-accept cycle, given a prompt. This

would re-introduce the consistency problem in Equation 3, *i.e.*, different orderings give different joint distributions. Thus, it is unclear what the true target distribution they are trying to match actually is, violating Theorem 2.

Another consequence of the multiple factorization orders sampled is that the number of NFEs could exceed the number of tokens eventually accepted, if the multi-order oracle evaluations all reject the speculated tokens. So, Theorem 1 is violated, which means that this scheme could potentially slow down sampling.

I.2 Lack of Resampling Step

Furthermore, their algorithm also does not have a resampling step in case of rejection. This decreases the number of tokens each iteration is guaranteed to accept from two to one (see Theorem 1's proof). This means that their algorithm is *not* mathematically guaranteed to reduce the number of function evaluations, and can in theory increase it: in the case that only the first conditionally independent token from the draft is accepted, the function evaluation of the oracle did not lead to an extra token being accepted. This violates Theorem 1.

Furthermore, the lack of resampling potentially once again violates Theorem 2's guarantee that the returned distribution will match the joint. Using the notation and ideas from Appendix C, they have the "accept" (first) term in Equation 14, but they do not have the "reject" (second) term to balance the probability out. In fact, their Line 7 cannot even be considered to be proper rejection sampling, because it lacks the proper normalization for the decision threshold [LKM23].

I.3 Any-Order Speculative Decoding Addresses Problems

In contrast, by enforcing [SSE22]'s recursive binary lattice mask decomposition, we ensure that given a prompt, there is *only one* correct path to calculating the joint conditional probability of the missing tokens. This makes it clear in our Algorithm 1 what the target distribution actually is, and our output provably matches that distribution (Theorem 2). Furthermore, we have resampling in Line 22. Combined with the single-path evaluation, this mathematically guarantees that we never increase the number of function evaluations above the number of masked tokens (Theorem 1).

J Off-the-Shelf Model Outputs

Text comes from the WikiText dataset [Mer+16]. This is from the default XLNet model provided on Huggingface (*not* our finetuned), which was only trained to predict around 20% of tokens.

Original Text

<sep><cls></s> = Robert Boulter =<sep><cls></s><sep><cls></s> Robert Boulter is an English film , television and theatre actor . He had a guest @-@ starring role on the television series The Bill in 2000 . This was followed by a starring role in the play Herons written by Simon Stephens , which was performed in 2001 at the Royal Court Theatre . He had a guest role in the television series Judge John Deed in 2002 . In 2004 Boulter landed a role as " Craig " in the episode " Teddy 's Story " of the television series The Long Firm ; he starred alongside actors Mark Strong and Derek Jacobi . He was cast in the 2005 theatre productions of the Philip Ridley play Mercury Fur , which was performed at the Drum Theatre in Plymouth and the Menier Chocolate Factory in London . He was directed by John Tiffany and starred alongside Ben Whishaw , Shane Zaza , Harry Kent , Fraser Ayres , Sophie Stanton and Dominic Hall .<sep><cls></s> In 2006 , Boulter starred alongside Whishaw in the play Citizenship written by Mark Ravenhill . He appeared on a 2006 episode of the television series , Doctors , followed by a role in the 2007 theatre production of How to Curse directed by Josie Rourke . How to Curse was performed at Bush Theatre in the London Borough of Hammersmith and Fulham . Boulter starred in two films in 2008 , Daylight Robbery by filmmaker Paris Leonti , and Donkey Punch directed by Olly Blackburn . In May 2008 , Boulter made a guest appearance on a two @-@ part episode arc of the television series Waking the Dead , followed by an appearance on the television series Survivors in November 2008 . He had a recurring role in ten episodes of the television series Casualty in 2010 , as " Kieron Fletcher " . Boulter starred in the 2011 film Mercenaries directed by Paris Leonti .<sep><cls></s><sep><cls></s> = = Career = =<sep><cls></s><sep><cls></s><sep><cls></s> = = = 2000 – 2005 = = =<sep><cls></s><sep><cls></s> In 2000 Boulter had a guest @-@ starring role on the television series The Bill ; he portrayed " Scott Parry " in the episode , " In Safe Hands " . Boulter starred as " Scott

Prompt

_ the television_ _ _
_ _ _ _ _ _ _ _ _ _ _ in_ _ _ _ _ _ _ _ _ _ ,_ _ _ _ _ _ _ _ Court_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ "_ and_ _ _ _ _
He_ _ _ _ 2005_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ Plymouth_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ Stanton and_ _ _ _ _ _ _ _ _ _ _ _ _ starred_ _ _ _ _
play Citizens_ _ _ _ _ _ _ _ _ _ _ a_ Josie_ _ _
_ ._ key_ _ _ _
_ _
_ Merc_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ =_ _ _ _ _ _ _ _ _ _ _ _ =_ _
the_ _ _ _ _ _ _ portrayed_ starred_ _ _ _ _

Off-the-Shelf, Sequential Decoding

300 300 300 300<eop> Ethnic or Ethnic Ethnically Ly Americans or Others or Others Beyond
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly the television Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly in Ly Ly Ly Ly Ly Ly Ly North), Ly Ly Ly Ly Ly
Ly Ly Ly Court Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly" Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly and Ly Ly Ly Ly Ly He Ly Ly Ly Ly 2005 Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Plymouth Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Stanton
and Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly starred Ly Ly Ly Ly Ly play Citizens Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly a Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Josie Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly. Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Lykey Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Merc Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly = Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly = Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly the Ly Ly Ly Ly Ly Ly Ly portrayed Ly Ly Ly Ly Ly Ly Ly
Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly Ly; Ly Ly starred Ly Ly Ly Ly Ly

Off-the-Shelf, Speculative Decoding (Algorithm 1)

series and led starred starred starred starred starred starred starred starred starred starred
starred starred starred starred starred starred starred starred starred starred starred starred
starred starred starred starred starred starred starred starred starred starred starred starred
starred and in the television major starred starred starred starred starred starred starred
starred starred when the on in became being the title characters into scene as filming the, and
too happen which came. The Screen Court Movie""""1996)(were cast they are booked cast
to be cast moondicly on set as the movie upon movies too news faint as of taking pres "".
They cast to appear on set as of other celebrities the her shee sheeee heee and and is casting
as the Hee Hee she 2005 """"Uh Uh Uh Uh Uh Uh Uh, Plymouth listed on set the first here
Hee hee she is casting as they are casting to appear as the scheduled here Hee hee he is cast
as previously its scheduled her to appear as Stanton and into the Las Vegas during draft her
seen her can be cast or starred as the backup for the play Citizens Las Vegas are casting ankle
to appear assed or a the next scheduled Las Vegas Las ("""""" shortly before after the next
scheduled Las Vegas Las Vegas Las "" Josie Las Vegas Las ""jinohlah,x Vegas Las Vegas
Las Vegas Las Vegas Las Vegas Las Vegas "." Las Vegas Las Vegas Las Vegas Las Vegas Las
Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegaskey Las Vegas Las
Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas
Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las
Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas
Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las
Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Las
Vegas Las Vegas Las Merc Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas
Las Vegas Las Vegas Las Vegas = Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las
= Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las
Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas the Las Vegas Las Vegas Las
Vegas Las portrayed Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las
Vegas Las Vegas Las Vegas Las Vegas Las Vegas Las Vegas starred Las Vegas Las Vegas Las

K Finetuned Model Outputs

Text comes from the WikiText dataset [Mer+16]. Samples correspond to Table 1, with our model trained from $90\% \rightarrow 99\%$ masking rate, as described in Appendix E.4.

Original Text

<cls></s> AFI ’s 10 Top 10 - # 6 Sports Film<sep><cls></s><sep><cls></s> = = Legacy = =<sep><cls></s><sep><cls></s> In the decades since its release , The Hustler has cemented its reputation as a classic . Roger Ebert , echoing earlier praise for the performances , direction , and cinematography and adding laurels for editor Dede Allen , cites the film as " one of those films where scenes have such psychic weight that they grow in our memories . " He further cites Fast Eddie Felson as one of " only a handful of movie characters so real that the audience refers to them as touchstones . " TV Guide calls the film a " dark stunner " offering " a grim world whose only bright spot is the top of the pool table , yet [with] characters [who] maintain a shabby nobility and grace . " The four leads are again lavishly praised for their performances and the film is summed up as " not to be missed . "<sep><cls></s> Paul Newman reprised his role as Fast Eddie Felson in the 1986 film The Color of Money , for which he won the Academy Award for Best Actor in a Leading Role . A number of observers and critics have suggested that this Oscar was in belated recognition for his performance in The Hustler . In 1997 , the Library of Congress selected The Hustler for preservation in the United States National Film Registry as " culturally , historically , or aesthetically significant . " Carroll and Rossen ’s screenplay was selected by the Writers Guild of America in 2006 as the 96th best motion picture screenplay of all time . In June 2008 , AFI released its " Ten top Ten " — the best ten films in ten " classic " American film genres — after polling over 1 @,@ 500 people from the creative community . The Hustler was acknowledged as the sixth best film in the sports genre .<sep><cls></s> The Hustler is credited with sparking a resurgence in the popularity of pool in the United States , which had been on the decline for decades . The film also brought recognition to Willie Mosconi , who , despite having won multiple world championships , was virtually unknown to the general public . Perhaps the greatest beneficiary of the film ’s popularity was a

Prompt

_ <sep>_ ,_ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ direction_ _ _ _ _ _ _ _ _ _ _ _ _ editor_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ our_ _
_ the_ _ _ _ _ _ _ _]_ _ _ _
_ _ _ _ _ _ _ and_ _ _ _ _ _ _ _ _ _ _ lavish_ _ _ _ _ and_ film_ _ _ _ _ _ not_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ role_ _
_ _ _ for_ _ _ _ _ _ _ _ _ _ _ _ _ _ Library_ _
_ FI_ _ _ _ _ _ Ten_ _
_ _ _ _ _ _ _ ten_ _
_ _ _ _ Hus_ also_ _ _ Willie_ _ _ _ _ _
_ ’_ _ _ _

Finetuned, Sequential Decoding

available to all user accounts. 3. Feel free to throw us a few questions and development suggestions.<sep><cls></s> GENEVA, Switzerland – The paper will not draft further stories, and we will, after all, ignore the party of journalistic freedom, which includes the right not to evaluate claims against anyone, except a dissenting direction from the scientific community. I assure you that we will continue the editor-in-chief role as a quarterly magazine where there will be no exemptions for writers of a particular viewpoint. This will lead our competition to host an active website, WMSU, which will host an automated competition about WMSU as well as to develop accredited methods of talking, researching and writing editorial articles in Swedish ("HF") and English (Roban Xismen in German). History [edit] Flawed motives for the art of the book have been debated much.[2] Until 2001, in ten of our 10 articles on 2001 and '02, we have used scientific forms, lavish rigor and style, and fake film. Since then, it has not happened to us. ARandom House, in cooperation with Wikipedia, consists in role-playing games and fantasy series literary novels with a unique approach to artistic media, including "Jeressande" by Audiom et les Hamerswein. See also [edit] Table of contents of articles for 2001-11 The intensive research effort of the Fabian Society and the Library and Museum of Great Britain on the RMA 1000 series includes more than nine dazzling articles, 50 "classics" and other pieces that garnered a mention from the Swedish press, an astute academic, sergeant d'Hoskin's Service, an item of art on the rise for WMSU in the 2002 World FI World Games.[3] Ten or more previous articles in February contributed to a ten-years archive by the L.B. Law Foundation of knowledge-urgent WB02 publications data. In 2005, Wercd et al. took the first step toward finishing "Moswart Huslas", which was published in the 1976 edition. In 2007, Munsry et al. repeated "White People's Rescue" also published events by Willie Miller and other Al Jazeera fresh and original voices. In a 2003 paper in the sociology of journalism, David Blabas reporteds, "The so called 'white men's

Finetuned, Speculative Decoding (Algorithm 1)

it learned to leave the visit." Read the folksy post—in full here...<sep><cls></s> CSI: Drayton, Part I Sir Ken & Honi Pratt Expedition Expeditions, Ltd Research & Development Canada U.S. History and Culture 1,200 Games History Remfacile Rerail will provide nine series of digital direction, redesign and multi-distribution services, leading original biographer and editor David Tatnow appears in the final stages of return service. "It's almost as fast as walking, getting the basics off our shelves," Chief Creative Officer Tony Paschus said. "We envy Historic Stadium, Ottawa and the story we can tell with access to our digital resource. The behind-the-scenes humblings cost the taxpayers more than $14 million last year and were the costliest part of CSI's entire qualification process isn't the only aspect of our financial problem [yet]. Players will have access to the most advanced titles, and collectible characters of any franchise. But they will be lavished with additional digital content and new film releases, in things that will not cut through Rui's new, toned down, strong-arm role—for example, when, in those spacesuits and used as rival generals, she can have dramatic conversations with everyone on her team. (Each character continues to be introduced during missions.) UK flags are symbolic of support for this transition from digital play to entertainment. Credit: Beaujorie Library Release date: 2015-01-25 Format: Remastered digital Download framerate: 57 kbps (a range of 1.2 to 54 kbps) Thumbnail up to 1920x1080 (16 in., 5 notches at 40.1°C) Planting Function: BFI 16 00 | Twelve Tens of catchests Producers: tenthreekorea-reports.gl.ca SUBSCRIBE TO Northeastern University's CSI: Drayton News & Games Tour: 51 Place (including Canada), Alberta (including Canada) Huston University University College of Art and Design MacAskill University of Waterloo (Note: Gamers 3 and under are welcome to sign up) See also: All About Willie Lewis<sep><cls></s> Posing for AC/DC's album's tour in October, D'Angelo Brakes of British Columbia proceeded with a classic 'DC' mixta