Enabling Autoregressive Models to Fill In Masked Tokens

Daniel Israel, Aditya Grover, Guy Van den Broeck

University of California, Los Angeles

Abstract

Historically, LLMs have been trained using either autoregressive (AR) or masked language modeling (MLM) objectives, with AR models gaining dominance in recent years. However, AR models are inherently incapable of masked infilling, which is the ability to predict masked tokens between past and future context. In contrast, MLM models suffer from intrinsic computational inefficiencies during both training and inference that hinder their scalability. This work introduces MARIA (Masked and Autoregressive Infilling Architecture), a novel approach that leverages the strengths of both paradigms to achieve state-of-the-art masked infilling performance. MARIA combines a pre-trained MLM and AR model by training a linear decoder that takes their concatenated hidden states as input. This minimal modification enables the AR model to perform infilling while retaining its inherent advantages in terms of faster inference with KV caching. Our results demonstrate that MARIA significantly outperforms existing methods, namely discrete diffusion models, on masked infilling tasks.

1 Introduction

The field of natural language processing (NLP) has witnessed remarkable advancements in recent years, largely driven by the advent of large language models (LLMs) (Zhao et al., 2023) built upon the Transformer architecture (Vaswani, 2017). These models, characterized by their self-attention mechanisms and vast parameter counts, have demonstrated unprecedented capabilities in understanding and generating human-like text.

A critical aspect of LLM training lies in the choice of pre-training objective. Traditionally, two dominant paradigms have emerged: autoregressive (AR) and masked language modeling (MLM). AR models, such as GPT (Achiam et al., 2023), are trained to predict the next token in a sequence, given the preceding context. This left-to-right approach, coupled with causal masking that prevents the model from "seeing" future tokens, enables efficient training and inference. MLM models, exemplified by BERT (Devlin et al., 2019), are trained to predict masked-out tokens in a sequence, leveraging bidirectional context from both past and future tokens.

One notable capability where AR models typically fall short is text infilling (Donahue et al., 2020), the task of predicting missing tokens within a given text span, surrounded by both preceding and subsequent context. While MLM models inherently support infilling due to their bidirectional nature, AR models, with their unidirectional processing, cannot leverage future context for this task. This limitation restricts the applicability of AR models in scenarios where infilling is essential, such as interactive text editing (Lee et al., 2022), code completion (Liu et al., 2020), and structured generation (Xia et al., 2024). Despite the limitations of AR models in handling text infilling, they remain the dominant paradigm for large-scale language modeling due to their superior scalability. AR models benefit from several key advantages that make them more efficient during both training and inference. First, AR models can exploit causal masking to parallelize every next token prediction, enabling faster training on massive datasets across multiple GPUs. This differs from MLM models, which only make predictions for a fixed ratio of masked tokens during training, such as 15 percent in BERT (Devlin et al., 2019). Second, the sequential nature of AR models allows for the use of KV caching at inference time, which significantly reduces the computational cost of attention operations by reusing previously computed embeddings. Significant effort has been dedicated to optimizing the memory and speed of KV caching (Kwon et al., 2023; Zhao et al., 2024; Liu et al., 2024c). Thus, AR models are better suited for real-time applica-

Findings of the Association for Computational Linguistics: EACL 2026, pages 4954–4965

Model	Scalable Training	KV Cached Inference	Supports Mask Infilling
AR	✓	✓	✗
MLM	✗	✗	✓
MARIA	✓	✓	✓

Table 1: **Comparison of different modeling approaches.** We compare the three modelling approaches: Autoregressive (AR), Masked Language Modelling, and our method Masked and Autoregressive Infilling Architecture (MARIA). While AR enjoys more scalable training and computationally efficient inference, it cannot perform masked infilling. Contrarily, MLM can but is less scalable. We argue that our method MARIA inherits the benefits from both approaches.

tions, such as chatbots and virtual assistants, where low-latency responses are critical. These factors contribute to the widespread adoption of AR models in industry and academia, despite their inherent limitations for infilling.

Researchers have explored non-autoregressive paradigms that support text infilling. One such approach is discrete diffusion (Lou et al., 2023), which iteratively refines a noisy input sequence. Discrete diffusion models have shown promise in tasks like text generation and infilling. However, discrete diffusion models are built on the MLM modeling paradigm, making it difficult to scale their training in the same manner as AR models. Furthermore, these models often require numerous refinement steps and do not support KV caching, which can make them less efficient for inference. Given the complementary strengths and weaknesses of AR and MLM models, there is a clear need for a hybrid approach that leverages the best of both paradigms.

In this work, we introduce MARIA (Masked and Autoregressive Infilling Architecture), a novel framework that combines the benefits of AR and MLM models to achieve state-of-the-art performance in text infilling. MARIA integrates a pretrained MLM and AR model by training a linear decoder that takes the concatenated hidden states of both models as input. This minimal modification enables the AR model to perform infilling while retaining its inherent advantages in terms of faster inference with KV caching. Our experiments demonstrate that MARIA significantly outperforms existing methods, including discrete diffusion models, on a variety of text infilling benchmarks. By bridging the gap between AR and MLM paradigms, MARIA offers a new technique for scaling infilling language models. We summarize the advantages of MARIA in Table 1.

2 Related Works

2.1 Discrete Diffusion

Discrete diffusion models have emerged as a promising alternative to traditional autoregressive models for text generation and, notably, text infilling. Inspired by the success of diffusion models in continuous domains like image generation (Ho et al., 2020), these models adapt the diffusion framework to operate on discrete sequences of tokens. In the context of text infilling, discrete diffusion offers several advantages. Its iterative refinement process allows for fine-grained control over the generated text and the ability to tradeoff quality for efficiency. However, as mentioned in the introduction, these models can be computationally expensive during inference due to the multiple refinement steps and the lack of KV caching. They also face challenges in scaling up training compared to autoregressive models. In this paper, we will primarily focus on the work of Scaling Masked Diffusion Model (SMDM) (Nie et al., 2024) and DiffuLlama (Gong et al., 2024), but the space includes many promising works (Sahoo et al., 2024; Liu et al., 2024a,b; Hoogeboom et al., 2021; Ou et al., 2024)

2.2 FIM

AR models can be adapted to perform infilling through a special training process called Fill-in-the-Middle (FIM) (Bavarian et al., 2022), in which the order of the original sequence is changed such that the middle of the sequence is moved to the end and marked with a special FIM token. These FIM models are particularly useful for coding applications (Fried et al., 2023). We make a distinction between FIM and masked infilling. FIM necessitates that the infilled text is a contiguous block, while masked infilling can fill in arbitrary sequences of tokens.

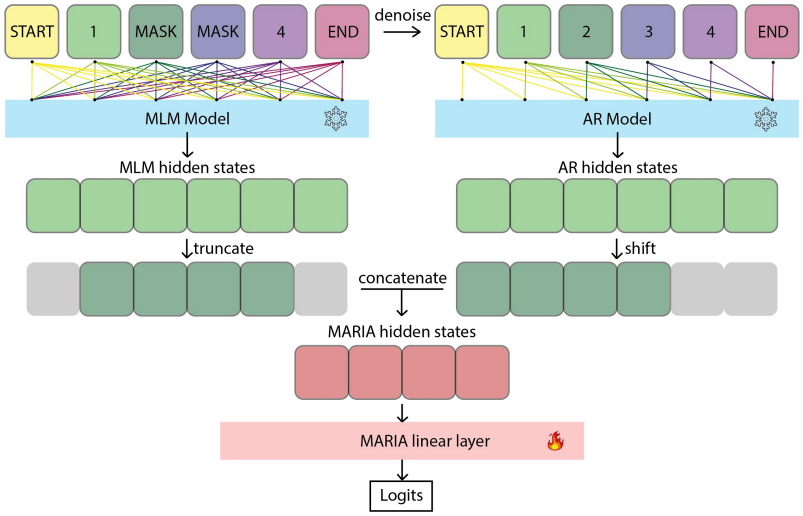

Figure 1: **MARIA architecture and training pipeline.** MARIA takes two frozen pretained models: one MLM and one AR. As input, the MLM recieves the masked inputs and the AR model recieves the denoised inputs. We compute the hidden states under each model and perform truncating and shifting operations to ensure both hidden states model the same tokens. MARIA trains a linear layer to predict the logits of each masked input on the concatenated hidden states. This training scheme models an autoregressive distribution conditioned on unmasked tokens.

2.3 MLM and AR Unification

Notable works to unify MLM and AR modelling include BART (Lewis, 2019). Besides architectural differences, the main distinction between MARIA and BART is that MARIA is applied to existing pretrained MLM and AR models, while BART must be trained end-to-end. Other notable works incorporate together MLM and AR modeling techniques for improved training (Du et al., 2022; Nguyen et al., 2023; Yu et al., 2024), but none are expressly targeting masked infilling as an application. Another related line of work casts infilling as a sequence-to-sequence modeling task. GLM (Du et al., 2022), ILM (Donahue et al., 2020), and SSR (Zhou et al., 2021) are trained end to end, and in this work we emphasize that MARIA is a technique that combines existing pretrained AR and MLM models, utilizing a lightweight linear layer with minimal training.

3 Method

Background. Consider an autoregressive model π_{AR} and masked language model π_{MLM}. The goal of language modeling is to learn some distribution over text sequences of the form $x = (x_1, ..., x_n)$. Given access to a dataset $\mathcal{D} = \{x^1, x^2, ...\}$, autoregressive models are trained to maximize the joint likelihood given by

$$\mathcal{L}_{\text{AR}} = -\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_i \log \pi_{\text{AR}}(x_i \mid x_{<i})\right] \quad (1)$$

Masked language models employ a masking objective that assumes a distribution over masks $\mathcal{M}$, where $m \in \mathcal{M}$ is selection of indices that takes the form $m = \{i_1, i_2, ...\}$. We define the MLM objective below.

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_{\substack{x\sim\mathcal{D}\\ m\sim\mathcal{M}}}\left[\sum_{i\in m} \log \pi_{\text{MLM}}(x_i \mid x_{\setminus m})\right] \quad (2)$$

We will stipulate that each language model is composed of a function h that embeds inputs into hidden state vectors and a linear weight matrix W used to decode the hidden states into logits.

$$\pi_{\text{AR}}(x \mid \cdot) = \sigma\left(W_1 h_1(x)\right) \quad (3)$$

$$\pi_{\text{MLM}}(x \mid \cdot) = \sigma\left(W_2 h_2(x)\right) \quad (4)$$

where $\sigma(z_i) = e^{z_i}/\Sigma_j e^{z_j}$ is the softmax function. We define $W_1 \in \mathbb{R}^{d_1\times v}$ and $W_2 \in \mathbb{R}^{d_2\times v}$ such that their hidden dimensions d can be different but vocabulary size v are the same.

MARIA Objective. The MARIA architecture can be defined very straightforwardly with a linear layer on the concatenated hidden states of an AR and MLM model.

$$\pi_{\text{MARIA}}(x \mid \cdot) = \sigma(W_3\left[h_1(x); h_2(x)\right]) \quad (5)$$

where $W_3 \in \mathbb{R}^{(d_1+d_2)\times v}$. Finally, we may now define an objective that is both autoregressive and masked. Let $x_{c(i,m)} = \{x_{<i}, x_{>i \cap \setminus m}\}$ define the union of tokens before the index i and all unmasked tokens after i. This objective defines the expected negative log likelihood of an autoregressive distribution conditioned on unmasked tokens.

$$\mathcal{L}_{\text{MARIA}} = -\mathbb{E}_{\substack{x\sim\mathcal{D} \\ m\sim\mathcal{M}}} \left[\sum_{i \in m} \log \pi_{\text{MARIA}}(x_i \mid x_{c(i,m)} \right] \tag{6}$$

Observe that Equation 6 resembles a combination of Equation 1 and 2.

Training. MARIA training can be parallelized in a similar manner as a typical autoregressive Transformer. For a clean input sequence $X_{1:n}$, we consider its masked counterpart $M_{1:n}$. The AR model receives the clean inputs and the MLM model receives the masked inputs such that we compute the hidden state $[h_1(X); h_2(M)]_{1:n}$ concatenated on the sequence dimension. These are then decoded with W_3 to next token logits. Thus, the autoregressive loss is computed over the entire sequence in parallel. This training procedure is best depicted by Figure 1.

As part of our method, we also provide a way to initialize the newly defined MARIA weights W_3. Because we have access to existing weights of pretrained models, namely an autoregressive weights W_1 and masked weights W_2, we can initialize W_3

$$W_3 \leftarrow [W_1/2; W_2/2] \tag{7}$$

Observe that this will output the average of the logits of π_{AR} and π_{MLM}

$$\begin{aligned} &\pi_{\text{MARIA}}(x \mid \cdot) && (8)\\ &= \sigma([W_1/2; W_2/2]\,[h_1(x); h_2(x)]) && (9)\\ &= \sigma((\pi_{\text{AR}}(x \mid \cdot) + \pi_{\text{MLM}}(x \mid \cdot))/2) && (10) \end{aligned}$$

This initialization performs well because the average of logits corresponds to a multiplicative mixture of the two original distributions. This ensemble, known as product of experts (Hinton, 2002), has proven effective in the context of LLMs (Liu et al., 2021). Smart weight initialization leads to faster and better convergence (Samragh et al., 2024), and we demonstrate this with "product initialization" for MARIA in Figure 2.

Inference. A significant advantage of autoregressive (AR) models is their inference efficiency, largely due to KV caching. AR models reuse the computed key and value vectors from all previous steps, avoiding redundant calculations. This makes generation significantly faster. In contrast, bidirectional MLMs, cannot use KV caching for generative infilling. When generating a token to fill a mask, the newly generated token changes the context for all subsequent tokens. This would require a full re-computation of the hidden states for the entire sequence at every single step—a process with quadratic complexity that is prohibitively slow for long sequences.

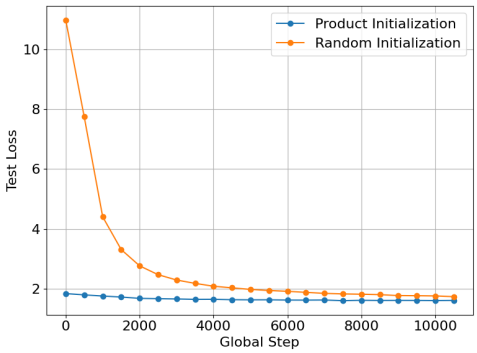

Figure 2: **Comparing evaluation loss curves for two different weight initializations.** Product initialization (ours) is a far better weight initialization than random weight initialization, leading to faster training and better convergence.

MARIA overcomes this limitation through the two-stage inference process outlined in Algorithm 1. First, the entire input sequence, including all masked tokens, is processed in a single forward pass by the MLM to compute hidden states for every position. These states capture the bidirectional context and are cached for the entire generation process. Second, MARIA fills the masked tokens autoregressively from left to right. At each step, the AR model processes only the tokens preceding the last filled mask with KV caching. The resulting AR hidden state is then concatenated with the corresponding pre-computed MLM hidden state for the current position. This combined representation is passed to the MARIA linear layer to predict the next token. This process allows MARIA to generate text while conditioning on future context in an efficient manner. As shown in our experiments, the overhead from the initial MLM pass is negligible, allowing MARIA to achieve throughput comparable to a standard AR model.

4 Implementation

A key constraint of MARIA is that the combined AR and MLM models must be trained with the same tokenizer. We make use of two important open-source works that are both trained with a GPT2 (Radford et al., 2019) based tokenizer: ModernBERT (Warner et al., 2024) and OLMo (Groeneveld et al., 2024). We train two models:

- **MARIA 1B**: a model composed of ModernBERT-Large and pretrained OLMo 1B
- **MARIA 7B**: a model composed of ModernBERT-Large and pretrained OLMo 7B

We will refer to these models in this manner throughout the course of the paper.

Our training data is composed of high quality tokens from FineWebEdu (Penedo et al., 2024), a standard pretraining corpus curated for fast convergence and good downstream performance. We randomly mask the data by sampling masking rates from a Beta(2.5, 2.5) distribution, which is more effective than a uniform rate (Shen et al., 2023). To train the MARIA Linear Layer, we initialize the weights as previously described. For MARIA 1B and MARIA 7B respectively, we train for 90000 steps (approximately 30 billion tokens) and 25000 steps (approximately 7 billion tokens). Given the size of FineWebEdu, we complete less than a single epoch, and we evaluate test loss on ten thousand holdout examples. We train at batch size 32 using gradient accumulation with a learning rate of 5-e5 and cosine learning rate schedule. Our training hardware is comprised of 8 NVIDIA 48GB A6000 GPUs connected to a Colfax CX41060s-EK9 4U Rackmount Server with AMD EPYC (Genoa) 9124 processors.

As we will further argue in Section 5, AR models have an advantage at inference time over MLM models with the ability to reuse previous computations through KV caching. Transformers with bidirectional masking cannot cache the computations from previous samples because sampling a new token will change the representations of all existing future tokens. We present a simple KV caching inference algorithm with MARIA in Algorithm 1. This algorithm computes a single forward pass on the MLM model to compute hidden states. After this negligible overhead (see Table 3), we perform standard KV caching just the same as a standard AR model.

Algorithm 1 MARIA KV Cached Inference.

Input: `input_ids,masked_indices`
Output: `input_ids` with infilled `[MASK]`s
`mlm_hs` ←MLM_Model(`input_ids`) {MLM hidden states (computed only once)}
`past_kv` ← None, `prev_idx` ← 0
for `idx` ∈ `masked_indices` **do**
  `ar_in` ←`input_ids[prev_idx:idx]`
  `ar_out` ← `AR_Model(ar_in, past_kv)` {AR model (cache)}
  `past_kv` ← `ar_out.past_kv` {Update cache}
  `ar_hs`←`ar_out.hidden_states`
  `maria_hs`←`Concat(ar_hs, mlm_hs[idx])`
  `logits`←`MARIA_Linear(maria_hs)`
  `token` ← `Sample(logits)`{Fill mask}
  `input_ids[idx]` ← `token`
  `prev_idx` ← `idx`
end for
return `input_ids`

5 Experiments

In this section, we evaluate MARIA in a variety of settings against existing baselines. As a summary of our key findings, we demonstrate the following with MARIA.

- **Superior Perplexity.** MARIA achieves lower perplexity across various masking rates and datasets compared to ModernBERT, SMDM, and DiffuLlama.
- **Efficient Inference.** MARIA offers high throughput by KV caching at inference time. AR decoding with ModernBERT does not scale.
- **High-Quality Samples.** Evaluation using LLM judge based ELO demonstrates that MARIA's generated text is of higher quality than baselines.

5.1 Baselines

We consider three primary baselines to compare our method against. First, we consider ModernBERT. Although ModernBERT is an MLM model, in practice MLM models can be used autoregressively by progressively filling in masks from left to right.

Model	Size	Type	Masking Rate				
			0.1	0.3	0.5	0.7	0.9
ModernBert	0.395 B	MLM	2.92	5.79	19.73	136.2	1468
OLMo 1B	1.18 B	AR	22.28	22.13	22.20	22.17	22.62
OLMo 7B	7.3 B	AR	14.93	15.01	14.96	15.00	15.046
SMDM	1.1B	DD	$\leq$ 14.44	$\leq$ 46.36	$\leq$ 118.7	$\leq$ 363.7	$\leq$ 1391
DiffuLlama	6.74 B	DD	$\leq$ 10.36	$\leq$ 30.04	$\leq$ 68.38	$\leq$ 180.5	$\leq$ 599.5
MARIA 1B	1.575 B	MLM + AR	3.10	4.45	7.41	13.80	23.99
MARIA 7B	7.695 B	MLM + AR	**2.82**	**3.85**	**5.94**	**10.11**	**16.30**

Table 2: **Downstream perplexity for various masking ratios.** We evaluate the downstream perplexity, averaging over 5 standard evaluation sets. ModernBERT is computed autoregressively, and we estimate the upper bound perplexity in the discrete diffusion models. MARIA performs the best by inheriting the strengths of its components: OLMo (AR) and ModernBERT (MLM). Based on parameter counts, MARIA presents the most effective way to scale models for masked token infilling.

Surprisingly, MLM models demonstrate considerable in-context learning capabilities when used in this manner (Samuel, 2024). Another necessary baseline for masked infilling are discrete diffusion models, of which we select Scaling Masked Diffusion Model (SMDM) (Nie et al., 2024) and DiffuLlama (Gong et al., 2024). These works execute interesting approaches to for scaling MLM models for discrete diffusion. SMDM analyzes MLM scaling laws and is trained in a compute-optimal manner. DiffuLlama distills an MLM model from an existing AR model, namely LLaMA 7B (Touvron et al., 2023). While these approaches are viable and worthwhile, in the following experiments we shall argue that MARIA is the most pragmatic approach for scaling masked infilling models.

5.2 Downstream Perplexity

Generative models optimize maximum likelihood objectives, and a common way to compare modeling performance is with likelihood on a test set. We compare perplexity, a similar notion, which is defined as the exponentiated average negative log likelihood on some corpus of tokens. We select five standard datasets to evaluate downstream perplexity: WikiText (Merity et al., 2016), LM1B (Chelba et al., 2014), Lambada (Paperno et al., 2016), AG News (Zhang et al., 2016), and ArXiv papers (Clement et al., 2019). Some of the datasets are tokenized for an MLM word level tokenizer, so we detokenize them following standard procedure (Sahoo et al., 2024). Because the context lengths of models differ, we also compute fixed length perplexity on a rolling basis, that is partitioning corpuses of tokens as necessary to fit within a context and summing over the negative log likelihoods for each partition. We compute the perplexity given 5 different masking rates: 0.1, 0.3, 0.5, 0.7, 0.9 (least to most masked); specifically, the goal is to model the randomly masked tokens given the surrounding unmasked context. From the downstream datasets, we subsample 500 examples from each.

Importantly, discrete diffusion models do not admit an exact perplexity. Instead, we compute the negative evidence lower bound (NELBO) though sampling. While it may seem unintuitive to compare exact perplexities with upper bounds, in practice these bounds are tight (Kingma et al., 2023), and these comparisons are widespread in the literature (Ho et al., 2020; Gulrajani and Hashimoto, 2023).

In Table 2, we report the average perplexities for seven models. ModernBERT perplexity is computed using the left to right autoregressive distribution that an MLM model admits by successively unmasking and computing the likelihood from left to right. We also compute the perplexities for regular AR models that cannot condition on future tokens. Finally, the discrete diffusion have upper bound perplexities computed with 512 Monte Carlo samples. These results show that MLM models poorly model heavily noised text. We speculate that for ModernBERT, which was trained at a fixed mask ratio of 0.3 (Warner et al., 2024), performs poorly with higher noise ratios because they are out of dis-

tribution. Meanwhile, AR models cannot condition on future context and therefore demonstrate surprisingly strong performance independent of noising rate. MARIA, which is a mixture of OLMo and ModernBERT, achieves the upside of both models with strong performance in low noise settings, and it stays strong as the noise level increases, similar to the AR models. Of note, performance scales with model size, indicating a straightforward way to scale masked infilling capabilities more efficiently than scaling MLM models.

5.3 Throughput

Efficiency is a crucial reason why AR models are more widely adopted than MLM models. We profile the throughput of each model to better understand how these approaches compare. In light of this, we fix the generation parameters such as number of diffusion steps to 256 steps, which will be later used in infilling experiments. Thus, we can analyze these efficiency results with sample quality results in tandem.

In Figure 3, we measure the throughput in tokens per second on different length inputs with 50 percent masking. We average the throughput over 10 runs, with 2 warm-up runs in the beginning for each model to ensure the GPU is operating maximally. From the results, we observe that MARIA 1B has the best throughput. Table 3 shows that for inputs of length 1024 with 512 masked tokens, MARIA is nearly as fast as its base autoregressive model, and the single ModernBERT forward pass contributes a negligible overhead. Surprisingly, SMDM has worse throughput than larger 7B models. This can be attributed to an expensive classifier free guidance method (which we apply for later results) and miscellaneous implementation details. It is also critical to observe the performance of ModernBERT in Figure 3. Because ModernBERT is an MLM model incapable of KV caching, it is impractical to use for inference. KV caching models will

Model	Latency (s)	Tokens/second
OLMo 1B	5.461 ± 0.021	93.75
MARIA 1B	5.574 ± 0.024	91.85
OLMo 7B	13.405 ± 0.005	38.19
MARIA 7B	13.77 ± 0.005	37.17

Table 3: **Measuring MARIA Overhead**. MARIA is a combination of ModernBERT-large and OLMo, but when we use MARIA at inference time for infilling, the overhead of ModernBERT is negligible.

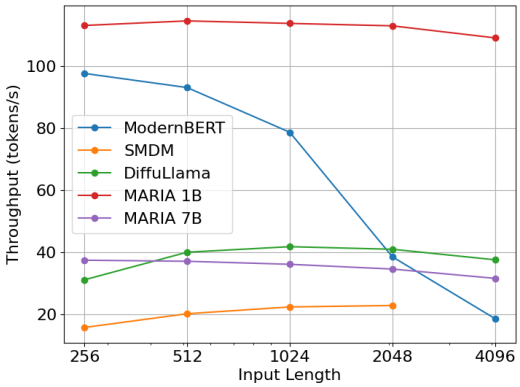

Figure 3: **Throughput over input length.** We show the throughput in tokens per second for sequences of given lengths at 0.5 masking rate. MARIA 1B exhibits the best performance, and MARIA-7B is comparable to DiffuLlama 7B. Decoding ModernBERT autoregressively is extremely inefficient at scale, and therefore is impractical in many circumstances.

have an inference runtime $O(n^2)$ in the sequence length, and without caching this runtime is $O(n^3)$. Though we include decoding ModernBERT autoregressively in the experimental benchmarks, poor efficiency at scale makes it severely impractical. Though discrete diffusion models cannot KV cache, they can unmask multiple tokens at each iteration. Thus, we see that DiffuLlama and MARIA 7B have similar throughputs. However, we shall show in the following section that MARIA achieves much better quality for similar efficiency.

5.4 Sample Quality

To evaluate sample quality, we adopted the same setting as before using 1000 samples total from the downstream datasets previously described (200 samples each). The task is to infill a random 50 percent of the text for each. However, to ensure comparable masked sequences in light of different tokenizers, we mask 50 percent words by replacing them with the mask string (i.e. `[MASK]`), ensuring that every model is given the same task. We define a word to be an alphanumeric string with spaces at the beginning and end.

We set the inference time hyperparameters to the respective values that achieved the best results for DiffuLlama and SMDM. For DiffuLlama, it uses nucleus sampling (Holtzman et al., 2020) and temperature scaling of 0.9 each. For SMDM, it applies classifier guidance scaling of 2 with greedy sampling. In all of the following experiments, we

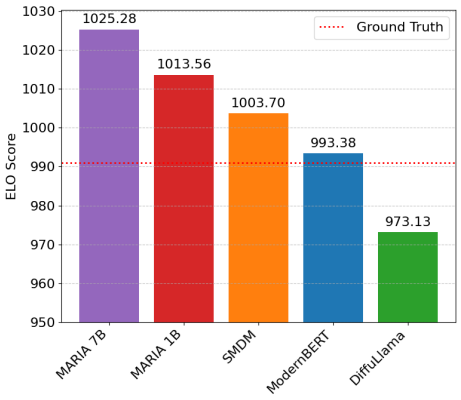

Figure 4: **ELO scores for masked infilling.** We perform infilling on downstream data with words masked 50 percent. Using GPT4o-mini as a judge we compute the ELO scores for each model respectively. MARIA 7B and 1B have the highest rating ELO rating under the Bradley-Terry model.

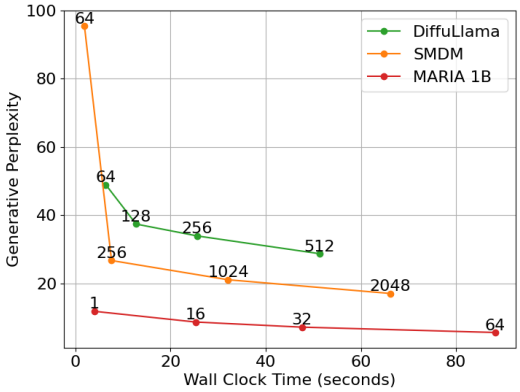

Figure 5: **Scaling test time compute for unconditional generation.** We compare our simulated annealing inference approach for MARIA to our baseline discrete diffusion methods. MARIA 1B using simulated annealing effectively trades-off quality (as measured by generative perplexity) and with compute (measured in wall clock time).

use 256 denoising steps. For ModernBERT and MARIA models, we decode greedily.

We assess sample quality using an ELO system judged by GPT-4o mini (Achiam et al., 2023). We create 1000 random fixtures and prompt GPT to give a score for each text "based on coherence, fluency, and style". For ELO scoring, a higher score is a win (1), lower score is a loss (0), and even score is a tie (0.5). We then calculate the ELO through logistic regression using the Bradley-Terry model, the same method as ChatBot Arena (Chiang et al., 2024). This method ensures that match order does not influence the final score, which is a problem with iteratively computing online ELO. We employ standard hyperparameters of scale 400, base 10, and inital rating of 1000.

As shown in Figure 4, the MARIA models score the highest ELO ratings, with MARIA 7B and 1B attaining the top scores. In the ELO rating system, every difference of 400 corresponds to a 10x improvement in winning odds. From these results, we infer that the win probability of MARIA 7B against SMDM and DiffuLlama are 53.1% and 57.4%. Though these differences are not drastic, in practice it is difficult to achieve large differences in win rate if the LLM judge is insufficient to adequately differentiate between texts. Interestingly, the LLM judges the generated texts of four models as higher quality than the ground truth clean text. This may be a consequence of greedy decoding producing more likely text than the source text.

5.5 Test Time Scaling

In Appendix A, we describe an algorithm to apply infilling towards test-time scaling. In short, repeatably resampling subsequences of text, while gradually lowering the sampling temperature is an effective method to iteratively improve text. This inference strategy is called simulated annealing. In Figure 5, we measure the generative perplexity of 200 unconditional samples according to Llama3 8B (Grattafiori et al., 2024). We show that for MARIA 1B, simulated annealing is an effective and efficient way to generate higher quality samples, converging faster than both DiffuLlama and SMDM. MARIA 7B with simulated annealing is far slower to converge than MARIA 1B, and it is omitted to avoid plot scaling issues.

6 Conclusion

MARIA is a hybrid approach, unifying AR and MLM models, that has demonstrated significant improvements in masked token infilling. It achieves lower perplexity scores and outperforms existing methods in both quality and efficiency. Future research directions include inference speed optimizations and using fine-tuned models for domain-specific infilling.

7 Limitations

We have not applied modern AR inference optimizations to MARIA, such as PagedAttention (Kwon et al., 2023), as it is beyond the scope of

this work. Domain specific applications of infilling are also beyond our scope, for example coding (Fried et al., 2022) and DNA imputation (Schiff et al., 2024).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *Preprint*, arXiv:2207.14255.

Dimitris Bertsimas and John Tsitsiklis. 1993. Simulated annealing. *Statistical science*, 8(1):10–15.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. *Preprint*, arXiv:1312.3005.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. 2019. On the use of arxiv as a dataset. *Preprint*, arXiv:1905.00075.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. *Preprint*, arXiv:2103.10360.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. 2023. Incoder: A generative model for code infilling and synthesis. *Preprint*, arXiv:2204.05999.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2022. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2024. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, and 1 others. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Ishaan Gulrajani and Tatsunori B. Hashimoto. 2023. Likelihood-based diffusion language models. *Preprint*, arXiv:2305.18619.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Preprint*, arXiv:2006.11239.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *Preprint*, arXiv:1904.09751.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. 2021. Autoregressive diffusion models. *CoRR*, abs/2110.02037.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. 2023. Variational diffusion models. *Preprint*, arXiv:2107.00630.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Mina Lee, Percy Liang, and Qian Yang. 2022. Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities. In *Proceedings of the 2022 CHI conference on human factors in computing systems*, pages 1–19.

Mike Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. *arXiv preprint arXiv:2105.03023*.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. 2024a. Discrete copula diffusion. *arXiv preprint arXiv:2410.01949*.

Fang Liu, Ge Li, Yunfei Zhao, and Zhi Jin. 2020. Multi-task learning based pre-trained language model for code completion. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 473–485.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. 2024b. Think while you generate: Discrete diffusion with planned denoising. *Preprint*, arXiv:2410.06264.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2024c. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2023. Discrete diffusion language modeling by estimating the ratios of the data distribution.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Anh Nguyen, Nikos Karampatziakis, and Weizhu Chen. 2023. Meet in the middle: A new pre-training paradigm. *Preprint*, arXiv:2303.07295.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2024. Scaling up masked diffusion models on text. *Preprint*, arXiv:2410.18514.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2024. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *Preprint*, arXiv:2406.03736.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *Preprint*, arXiv:1606.06031.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Preprint*, arXiv:2406.17557.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Preprint*, arXiv:2406.07524.

Mohammad Samragh, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. 2024. Scaling smart: Accelerating large language model pre-training with small model initialization. *arXiv preprint arXiv:2409.12903*.

David Samuel. 2024. Berts are generative in-context learners. *Preprint*, arXiv:2406.04823.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *Preprint*, arXiv:cs/0306050.

Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. 2024. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv preprint arXiv:2403.03234*.

Tianxiao Shen, Hao Peng, Ruoqi Shen, Yao Fu, Zaid Harchaoui, and Yejin Choi. 2023. Film: Fill-in language models for any-order generation. *Preprint*, arXiv:2310.09930.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. 2023. Long horizon temperature scaling. In *International Conference on Machine Learning*, pages 31422–31434. PMLR.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, and 1 others. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*.

Xinru Yu, Bin Guo, Shiwei Luo, Jie Wang, Tao Ji, and Yuanbin Wu. 2024. Antlm: Bridging causal and masked language models. *Preprint*, arXiv:2412.03275.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification. *Preprint*, arXiv:1509.01626.

Siyan Zhao, Daniel Israel, Guy Van den Broeck, and Aditya Grover. 2024. Prepacking: A simple method for fast prefilling and increased throughput in large language models. *arXiv preprint arXiv:2404.09529*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Wangchunshu Zhou, Tao Ge, Canwen Xu, Ke Xu, and Furu Wei. 2021. Improving sequence-to-sequence pre-training via sequence span rewriting. *Preprint*, arXiv:2101.00416.

A Simulated Annealing

In general, the primary inference use case for MARIA is autoregressive text infilling. While MARIA is specifically trained for that task, which is conditioned on existing text to fill in masked tokens, we also propose a method to sample unconditionally. A desirable feature of this generative model is to enable iterative refinement of text such that more computation leads to better samples. Discrete diffusion has this property for the number of denoising steps, and while it is possible to use MARIA directly as a discrete diffusion model, it is undesirable because autoregressive sampling at each time step is slow, and discrete diffusion only unmasks a small number of tokens at a time, remasking most samples at every iteration. Thus, we propose using MARIA as a generative model with an inference strategy inspired by simulated annealing (Bertsimas and Tsitsiklis, 1993). We can describe the process as follows:

1. Sample from the base AR model at temperature 1.
2. Using MARIA, resample a fixed percentage of tokens autoregressively at temperature T.
3. Repeat the process for some number of iterations, annealing T from 1 to 0.

This inference strategy is a way to optimize over the joint likelihood of a sequence, and it is an improvement over standard greedy sampling because it is non-myopic (Shih et al., 2023). Thus, this procedure gives a principled way to scale test-time compute using a model that can infill masked tokens.

More formally, we are sampling from the following distribution:

$$p\left(x^{(i)}\right) \propto \sum_{\substack{x^{1:i-1} \\ m^{1:i-1}}} \prod_{j=1}^{i} \pi_{\text{MARIA}}\left(x^{(j)} \mid x^{(j-1)}, m^{(j-1)}; t^{(j-1)}\right)$$

where $x^{(i)}$ is the sequence at step i, $t^{(j)}$ is a temperature at step j defined according to a temperature schedule, $m^{(k)}$ is a mask at step k according to a masking schedule, and $\pi_{\text{MARIA}}(\cdot\,; t)$ denotes the autoregressive MARIA distribution temperature scaled by t.

B Representations

Representation	Accuracy
ModernBERT	0.642 ± 0.002
MARIA 1B	0.714 ± 0.002
MARIA 7B	0.735 ± 0.002

Table 4: **Representation learning for part-of-speech tagging.** We demonstrate that MARIA representations produce higher accuracy when used to predict parts-of-speech. This indicates that the concatenated AR and MLM hidden states of MARIA contain more information than MLM alone.

Representation learning is a key motivation behind training Transformers with an MLM objective. We aim to analyze MARIA through a representation learning perspective to offer insight into why combining MLM and AR models can improve performance. Specifically, we study the token level representations by measuring performance on part-of-speech tagging. The part-of-speech tagging task has a history in NLP (Manning, 2011), and we use the CoNLL-2003 dataset (Sang and Meulder, 2003). We train a linear classifier on representations from ModernBERT, MARIA 1B, and MARIA 7B on 10000 sentence examples with POS labels that can belong to 48 different classes. We train for 10 epochs with a learning rate of 1e-4. As Table 4 shows, part-of-speech tagging accuracy increases with MARIA 1B and further increases with MARIA 7B. These results are somewhat expected because MARIA hidden states are much larger in dimension: ModernBERT has dimension 1024, MARIA 1B has dimension 3072, and MARIA 7B has dimension 5120. These results confirm that AR representations contain information that MLM representations do not due to scale.

C Model Size Ablation

In our main experiments, we train MARIA with ModernBERT-Large. Here with run an ablation on the perplixity scores training MARIA with the weaker ModernBERT-Base. We still observe strong performance, but predictably worse than with perplexity results with ModernBERT-Large;

Table 5: Comparison of MARIA models with varying noise ratios.

Noise Ratio	MARIA (OLMo 1B + ModernBERT-Base)
0.1	3.53
0.3	5.03
0.5	7.95
0.7	13.34
0.9	21.98

D LLM as a Judge

We provide the full prompt used in our LLM-as-a-judge framework.

Our prompt: *"We would like to request your feedback on the quality of text. You will be provided two passages of text which can differ by few or many words. Please rate each text based on coherence, fluency, and style. Penalize poor syntax, incorrect grammar, and excessive repetition. Each text receives an overall score on a scale of 1 to 10, where a higher score indicates higher quality. Please first output a single line containing only two values indicating the scores for Texts 1 and 2, respectively. The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment."*