

Zarya: A Hybrid Autoregressive–Masked Diffusion Language Model with Flexible Training and Dual-Mode Inference

Leonid Sinev, Ilya Koziev, and Vladislav Leshchuk



<https://github.com/ai-forever/zarya>

Abstract

Autoregressive language models (ARMs) are constrained by sequential, left-to-right generation, while masked diffusion models (MDMs) enable parallel decoding but suffer from high computational overhead due to the inability to reuse Key-Value (KV) cache and from incoherent generation arising from learning dependencies over an intractable space of token combinations. We introduce ZARYA, a family of hybrid language models that jointly optimizes an autoregressive (AR) objective and a masked-diffusion objective within a single architecture. ZARYA structures training data into variable-size *slots* and employs a curriculum that gradually increases slot granularity, enabling a smooth transition from fine-grained AR learning to coarse-grained diffusion learning. At inference, ZARYA provides two distinct decoding paradigms through a unified interface: (i) MDM sampling with first-hitting denoising, and (ii) *slotted speculative decoding* that interleaves inter-slot diffusion-based selection with intra-slot autoregressive infilling, achieving full KV cache reuse. The training and inference regimes are fully decoupled, allowing a model trained with any configuration to be deployed in either mode. Extensive configurability — including grouped noise patterns (Prefix Completion, Fill-In-the-Prefix, Fill-In-the-Middle), ordered sampling schedules, and noise-level permutation strategies — enables flexible research exploration. We release Zarya models publicly in sizes 0.6B, 1.7B, and 4B, demonstrating performance on standard benchmarks while offering a principled integration of autoregressive and diffusion paradigms.

1 Introduction

Autoregressive models (ARMs) have achieved remarkable success in a wide range of natural language tasks (Brown et al. 2020; OpenAI 2023; Touvron et al. 2023). However, their sequential, left-to-right decoding fundamentally limits inference throughput, preventing parallelization (Cai et al. 2024; Chen et al. 2023). Masked diffusion models (MDMs) offer a compelling alternative by enabling parallel generation through an iterative denoising process without a fixed generation order (T. Li et al. 2026). However, most MDMs suffer from two critical drawbacks: (i) they exclude Key-Value (KV) caching, incurring high computational overhead during inference; and (ii) they learn dependencies over an intractable space of token combinations, leading to incoherent generation (J.-N. Li et al. 2026).

Recent hybrid approaches have sought to bridge these paradigms. Block Diffusion (BD3-LMs) (Arriola et al. 2025) interpolates between AR and MDM by grouping tokens into fixed-size blocks and unmasking them from left to right, but the block size is fixed and lacks flexibility. Esoteric Language Models (ESO-LMs) (Sahoo et al. 2026) fuse AR and MDM paradigms at the loss level, enabling KV caching for MDMs and achieving faster inference than contemporary MDMs. REFUSION (J.-N. Li et al.

2026) elevates parallel decoding from the token level to a higher *slot level*, interleaving inter-slot diffusion-based selection with intra-slot autoregressive infilling, outperforming Qwen3-8B (Yang et al. 2025) on GSM8K (Cobbe et al. 2021) and MBPP (Austin et al. 2021) while being $2.33\times$ faster on average.

We introduce ZARYA, a novel hybrid architecture with the following key contributions:

1. **Training with gradually increasing slot length:** We partition the predicted sequence into fixed-length, consecutive sub-sequences, referred to as slots. The size of the slots gradually increased during training. For each sequence, we randomly mask several slots, reorder the input so that clean slots precede masked ones, and also permute the original order of both masked and clean slots. The model simultaneously learns AR next-token prediction on visible slots for sequential generation, and a denoising loss on the masked slots for context-aware parallel reconstruction.
2. **Training-inference decoupling:** Unlike prior work where the training configuration dictates the inference mode, ZARYA allows any trained model to be deployed in either MDM sampling or slotted speculative decoding mode via a single inference flag, offering higher flexibility. Both modes are fully using KV cache with causal attention masks

2 Related Work

Block Diffusion (BD3-LMs). BD3-LMs interpolate between AR and MDM by grouping tokens into blocks and unmasking them left-to-right (Arriola et al. 2025). However, the block size is fixed, limiting flexibility.

Esoteric Language Models (Eso-LMs). Eso-LMs fuse AR and MDM paradigms using causal attention, enabling exact likelihood computation and KV caching for MDMs (Sahoo et al. 2026). They achieve low perplexities among diffusion models on One Billion Words (LM1B; Chelba et al. 2014) and OpenWebText (OWT; Gokaslan et al. 2019) datasets.

ReFusion. REFUSION introduces slot-level parallel decoding, elevating generation from tokens to fixed-length slots (J.-N. Li et al. 2026). It interleaves inter-slot diffusion-based selection with intra-slot autoregressive infilling, reordering newly generated slots ahead of remaining masks after each iteration. This design unlocks full KV cache reuse and reduces learning complexity from an intractable token combination space to a manageable slot-level permutation space. However, REFUSION inference is tied only for prefix completion tasks.

3 Zarya Architecture

3.1 Model Backbone and Configuration

ZARYA wraps a Qwen3 backbone (Yang et al. 2025) with a custom `Zarya` class registered with Hugging Face’s `AutoModel` and `AutoConfig` systems. We initialize ZARYA from the Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B checkpoints, respectively, and fine-tune it for 1 epoch on a diverse 32M-sample instructional SFT dataset (approximately 37.7B tokens) covering mathematics, coding, and general instruction-following tasks. We release ZARYA in three sizes (see Table 1).

Table 1: Model architecture of Zarya models

Model	Layers	Heads (Q / KV)	Tie Embedding	Hidden Size	Intermediate Size	Context Length
Zarya-0.6B	28	16 / 8	Yes	1024	3072	2K
Zarya-1.7B	28	16 / 8	Yes	2048	6144	2K
Zarya-4B	36	32 / 8	Yes	2560	9728	2K

Because the Qwen3 backbone retains causal attention, masked positions cannot attend to future masked positions. Thus, the diffusion objective used by ZARYA is a causal masked-reconstruction objective rather than the fully bidirectional masked-token objective commonly used in masked diffusion language models. The reordering of visible and masked positions ensures that all masked positions can attend to the visible prefix while preserving the causal attention pattern and KV-cache compatibility.

The architecture is parameterized with diffusion-specific hyperparameters, which extends common AR model parameters:

- α_0 (default 0.25) and ϵ (default 0.001): parameters of the linear noise schedule $\alpha_t = \alpha_0(1 - t)$;
- $\lambda = \text{diffusion_loss_proportion}$ (default 0.5): weighting between MDM and AR losses;
- `sequential_shuffle` / `diffusion_shuffle`: control slot/token shuffling for each phase;
- `ordered_sampling`: monotonically increases p_{mask} left-to-right;
- `grouped_noise` and `max_span_length`: enable Prefix Completion, Fill-in-the-Prefix, Fill-in-the-Middle, and random-span masking patterns;
- `noise_sorting`: reorders tokens by mask/unmask state before the forward pass.

4 Training

4.1 Slotted Training

When `slotted_training=True`, the `forward_process()` transforms each batch as follows (this regime is heavily inspired by J.-N. Li et al. (2026)):

1. **Slot partitioning:** Each answer is split into slots of size `slot_size` (from `slot_size_set`, e.g., for released model checkpoints [2, 4, 8, 16, 32, 64]).
2. **Mask sampling:** For each example in batch, sample mask probability (p_{mask}) uniformly. This determines what fraction of slots will be masked (treated as the diffusion task).
3. **Slot assignment:**
 - *AR slots:* tokens remain visible; the model predicts the next token within each slot, yielding $\mathcal{L}_{\text{seq}}$.

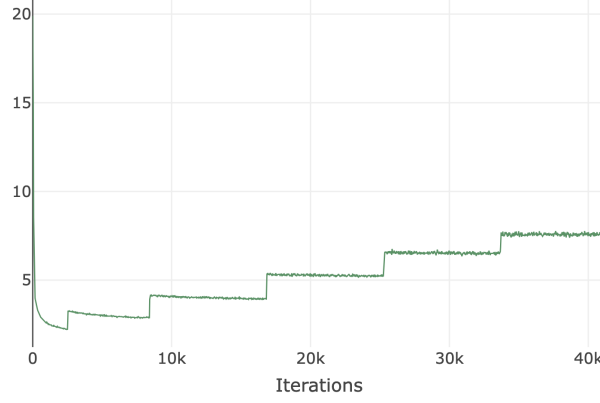


Figure 1: Loss curve during training of Zarya-4B model. Step-by-step increasing of slot size causes increase of loss too

- *MDM slots*: all tokens are replaced with the special `<mdm_mask>` token; the model reconstructs the original tokens, yielding $\mathcal{L}_{\text{dif}}$. Each token inside slot assigned p_{mask} value to be used in per-token normalization as $\frac{1}{p_{\text{mask}}}$ weighting.

The final loss is a linear combination:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{\text{dif}} + (1 - \lambda) \cdot \mathcal{L}_{\text{seq}}, \quad (1)$$

where λ is diffusion loss proportion.

This design enables the model to simultaneously learn next-token prediction (AR) and masked-token reconstruction (MDM) on the *same* input, with explicit slot boundaries providing a structured inductive bias.

Optional flags include:

- **ordered_sampling**: inside each slot modify per-token p_{mask} so that it increases left-to-right, as $\mathcal{L}_{\text{dif}}$ is scaled with the value of $\frac{1}{p_{\text{mask}}}$, this makes the model learn that correct prediction of the slot beginning tokens is more important;

4.1.1 Slot-Size Curriculum

Slot sizes evolve during training via `slot_step_borders` (epoch or step thresholds). For example, with

$$\begin{aligned} \text{slot_size_set} &= [2, 4, 8, 16, 32, 64], \\ \text{slot_step_borders} &= [0.06, 0.2, 0.4, 0.6, 0.8, 1.0], \end{aligned}$$

the slot size gradually increases from 2 to 64 over the course of training. This curriculum eases the model from fine-grained AR learning (small slots, many predictions) to coarse-grained diffusion learning (large slots, holistic reconstruction), providing a smooth transition between paradigms. Worth noting that increase in slot size also increases calculated loss (see Figure 1).

4.2 Non-Slotted Training

When `slotted_training=False`, the model runs objective over raw sequences without slot partitioning (this regime is heavily inspired by [Sahoo et al. \(2026\)](#)):

1. For each example in batch, **sample noise level** (t) uniformly.
2. **Construct noisy input**. Noise schedule is $\alpha_t = \alpha_0(1 - t)$ and the per-token mask probability is $p_{\text{mask}} = 1 - \alpha_t$.
3. Run **two separate forward passes**:
 - *sequential phase*: clean sequence x_0 , predict only tokens at masked positions (others set to `ignore_index`).
 - *diffusion phase*: noisy sequence x_t , reconstruct masked tokens.
4. Final loss is linear combination of losses at sequential and diffusion phases, calculated the same way as in [Section 4.1](#).

Optional flags include:

- `noise_sorting`: reorders tokens by mask/unmask state before both forward passes, with logits permuted back afterward;
- `grouped_noise`: masks contiguous spans (Prefix Completion, Fill-in-the-Prefix, Fill-in-the-Middle, random spans) up to `max_span_length`;
- `ordered_sampling`: offsets t per token position so that p_{mask} increases left-to-right.

5 Inference: Dual-Mode Unified Interface

ZARYA provides two distinct decoding paradigms through a single `model.generate()` call, routed via `generation_config.slotted_generation`. Critically, the training and inference regimes are fully decoupled: a model trained with `slotted_training=False` can still be deployed with `slotted_generation=True`, and vice versa.

5.1 Mode A: MDM Sampling

When `slotted_generation=False`, inference code executes the first-hitting denoising process ([Sahoo et al. 2026](#); [Zheng et al. 2025](#)). Starting from the prompt padded with `<mdm_mask>` tokens up to `max_length`, the model iteratively reveals tokens:

1. **Mask budget planning**: `_tokens_unmasked_per_step()` determines how many masks to reveal per step. α_0 is the expected fraction of masked tokens generated with diffusion process.
 - If number of discretization steps is set to $T > 0$: exactly T diffusion steps are used. Binomial distribution is used to calculate number of masked tokens to denoise through diffusion process and tokens that's left after that to denoise sequentially (this mode is heavily inspired by [Sahoo et al. \(2026, Appendix B.5\)](#)).

- If number of discretization steps is set to $T = 0$ (ignoring noise calculations): T steps are auto-calculated as $\frac{1}{4}$ of masked tokens.
- 2. **Reordering:** Input sequence is reordered so that masked tokens are always after unmasked.
- 3. **Per-step sampling from categorical distribution:** At each step, the model receives the progressively filled sequence (with KV cache reuse) and yields logits for masked positions. Gumbel noise (Gumbel 1935; Zheng et al. 2025, Appendix F) is added for categorical sampling, and standard sampling parameters (`temperature`, `top_p`, `repetition_penalty`) are honored.
- 4. **Restoration:** After all steps, the sequence is restored to the original token order.

The KV cache can be reused because the sequence is reordered so that tokens whose values are fixed at a given denoising step precede the remaining masked positions. Under causal attention, the cached prefix states therefore remain unchanged when masked positions are progressively filled. This property would not hold for a bidirectional masked-diffusion attention pattern, where changing any previously masked token could affect the representations of other masked positions.

5.2 Mode B: Slotted Speculative Decoding

When `slotted_generation=True`, inference code achieves parallelization by elevating decoding units from tokens to slots, fully reusing KV cache to avoid recomputation. This mode follows the REFUSION paradigm (J.-N. Li et al. 2026):

1. **Reordering:** Before the first forward pass, input sequence is reordered so that masked tokens are always after unmasked.
2. **Block construction:** `max_new_tokens` count of masked tokens is divided into `serial_num_blocks` blocks of length `block_size = floor(max_new_tokens/serial_num_blocks)`. Within each block, tokens are grouped into slots of size `slot_size`. If `max_new_tokens` is small, `serial_num_blocks` is forced to 1 to prevent zero-length blocks.
3. **Draft phase:** An MDM forward pass drafts tokens for all slots in the current block in parallel. The confidence of each slot is estimated as the probability of its first token.
4. **Sampling from categorical distribution:** If `temperature` is positive, apply Gumbel noise to logits for sampling from categorical distributions.
5. **Slot selection:** Slots with confidence exceeding `slot_threshold` are accepted immediately. If no slots are confident enough, select the most confident one, so we always have at least one slot to process further.
6. **Verification phase:** Selected slots undergo an AR verification forward pass. Tokens with probability exceeding `token_threshold` are accepted; those below are iteratively refined in a speculative loop.
7. **KV cache update:** The cache is updated incrementally with accepted tokens, avoiding recomputation for verified prefixes.
8. **Restoration:** After all steps, the sequence is restored to the original token order.

Table 2: Scope of configuration parameters across training and inference modes.

Parameter	Training	Inference
slotted_training	✓	—
slot_size_set / slot_step_borders	✓	—
diffusion_loss_proportion (λ)	✓	—
noise_sorting	✓	—
ordered_sampling	✓	—
grouped_noise / max_span_length	✓	—
add_loss_path / scale_by_batch	✓	—
slotted_generation	—	✓ (routes Mode A/B)
T (diffusion steps)	—	✓ (Mode A)
sequential_shuffle / diffusion_shuffle	✓	✓ (Mode A only)
slot_size / serial_num_blocks	—	✓ (Mode B)
slot_threshold / token_threshold	—	✓ (Mode B)
temperature / top_p / repetition_penalty	—	✓ (both modes)

5.3 Parameter Decoupling Summary

Table 2 summarizes the scope of key configuration parameters.

6 Evaluation

We evaluate on standard benchmarks including GSM8K (mathematical reasoning) (Cobbe et al. 2021), HellaSwag (commonsense reasoning) (Zellers et al. 2019), IFEval (instruction following) (Zhou et al. 2023), and MBPP (code generation) (Austin et al. 2021). Preliminary results for Zarya-0.6B are shown in Table 3. Hardware info: GPU A100; GPU driver CUDA version 13.2; GPU driver version 595.71.05; Docker info: Torch: 2.9.0+cu128; Transformers: 5.12.1; CUDNN in torch: 91002; lm-eval 0.4.12; Inference info: BF16, apply chat template, slotted_generation=true, slot_size=16, serial_num_blocks=4, slot_threshold=0.9, token_threshold=0.4

7 Conclusion

We introduced ZARYA, a hybrid AR-MDM language model that combines slotted training, dual-mode inference, and extensive configurability. By structuring data into slots and gradually increasing slot size during training, ZARYA achieves a principled integration of autoregressive and diffusion paradigms. The unified inference interface supports both MDM sampling and slotted speculative decoding mode with full KV cache reuse. ZARYA represents a step toward flexible, efficient language models that leverage the strengths of both generation paradigms. We release models at three sizes and provide a comprehensive training framework for further research.

Table 3: Preliminary evaluation results for Zarya-0.6B

Tasks	Version	Filter	n-shot	Metric	Value	Stderr
gsm8k	3	flexible-extract	5	exact_match	↑ 0.2646	± 0.0122
		strict-match	5	exact_match	↑ 0.2646	± 0.0122
hellaswag	1	none	0	acc	↑ 0.3526	± 0.0048
		none	0	acc_norm	↑ 0.4270	± 0.0049
ifeval	4	none	0	inst_level_loose_acc	↑ 0.5372	± N/A
		none	0	inst_level_strict_acc	↑ 0.5108	± N/A
		none	0	prompt_level_loose_acc	↑ 0.4177	± 0.0212
		none	0	prompt_level_strict_acc	↑ 0.3993	± 0.0211
mbpp	1	none	3	pass_at_1	↑ 0.1500	± 0.0160
mbpp_plus	1	none	3	pass_at_1	↑ 0.2249	± 0.0215

Limitations

While ZARYA demonstrates strong potential, several limitations remain:

1. **Threshold sensitivity:** The performance of slotted speculative decoding depends critically on slot selection threshold and token selection threshold. Poorly tuned thresholds can lead to low acceptance rates or quality degradation. An adaptive thresholding mechanism is needed.
2. **Mode-specific ignorance:** Researchers must remember that `ordered_sampling` and `noise_sorting` configuration options do *nothing* during inference. If these are tuned heavily during training, users may see no effect at inference time.

Acknowledgments

We thank the open-source community for providing the foundation upon which this work is built.

References

- M. Arriola, S. S. Sahoo, A. Gokaslan, Z. Yang, Z. Qi, J. Han, J. T. Chiu, and V. Kuleshov. 2025. “Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models.” In: *The Thirteenth International Conference on Learning Representations*. <https://openreview.net/forum?id=tyEyYT267x>.
- J. Austin et al.. 2021. *Program Synthesis with Large Language Models*. (2021). <https://arxiv.org/abs/2108.07732> arXiv: 2108.07732 (cs.PL).
- T. Brown et al.. 2020. “Language Models are Few-Shot Learners.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf.

- T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao. 2024. “Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads.” In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. <https://openreview.net/forum?id=PEpbUobfJv>.
- C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. 2014. *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*. (2014). <https://arxiv.org/abs/1312.3005> arXiv: 1312.3005 (cs.CL).
- C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper. 2023. *Accelerating Large Language Model Decoding with Speculative Sampling*. (2023). <https://arxiv.org/abs/2302.01318> arXiv: 2302.01318 (cs.CL).
- K. Cobbe et al.. 2021. *Training Verifiers to Solve Math Word Problems*. (2021). <https://arxiv.org/abs/2110.14168> arXiv: 2110.14168 (cs.LG).
- A. Gokaslan, V. Cohen, E. Pavlick, and S. Tellex. 2019. *OpenWebText Corpus*. (2019). <http://Skylio.n007.github.io/OpenWebTextCorpus>.
- E. J. Gumbel. 1935. “Les valeurs extrêmes des distributions statistiques.” 5, 2, 115–158. https://www.numdam.org/item/AIHP_1935__5_2_115_0/.
- J.-N. Li, J. Guan, W. Wu, and C. Li. 2026. “ReFusion: A Diffusion Large Language Model with Parallel Autoregressive Decoding.” In: *International Conference on Learning Representations*. Ed. by C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust. Vol. 2026, 53846–53869. https://proceedings.iclr.cc/paper_files/paper/2026/file/585979c057a1b30796cf317063559638-Paper-Conference.pdf.
- T. Li, M. Chen, B. Guo, and Z. Shen. 2026. *A Survey on Diffusion Language Models*. (2026). <https://arxiv.org/abs/2508.10875> arXiv: 2508.10875 (cs.CL).
- OpenAI. 2023. *GPT-4 Technical Report*. (2023). <https://arxiv.org/abs/2303.08774> arXiv: 2303.08774 (cs.CL).
- S. S. Sahoo et al.. 2026. “Esoteric Language Models: Bridging Autoregressive and Masked Diffusion LLMs.” In: *ICLR 2026 Workshop on Multimodal Intelligence*. <https://openreview.net/forum?id=CKrPJveQIr>.
- H. Touvron et al.. 2023. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. (2023). <https://arxiv.org/abs/2307.09288> arXiv: 2307.09288 (cs.CL).
- A. Yang et al.. 2025. *Qwen3 Technical Report*. (2025). <https://arxiv.org/abs/2505.09388> arXiv: 2505.09388 (cs.CL).
- R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. July 2019. “HellaSwag: Can a Machine Really Finish Your Sentence?” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by A. Korhonen, D. Traum, and L. Màrquez. Association for Computational Linguistics, Florence, Italy, (July 2019), 4791–4800. doi:10.18653/v1/P19-1472.
- K. Zheng, Y. Chen, H. Mao, M.-Y. Liu, J. Zhu, and Q. Zhang. 2025. “Masked Diffusion Models are Secretly Time-Agnostic Masked Models and Exploit Inaccurate Categorical Sampling.” In: *International Conference on Learning Representations*. Ed. by Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu. Vol. 2025, 63186–63227. https://proceedings.iclr.cc/paper_files/paper/2025/file/9e3b203e72c4e058de26d02a92a81844-Paper-Conference.pdf.
- J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. 2023. *Instruction-Following Evaluation for Large Language Models*. (2023). <https://arxiv.org/abs/2311.07911> arXiv: 2311.07911 (cs.CL).