

What's not an Autoregressive LLM?

Lingpeng Kong, The University of Hong Kong



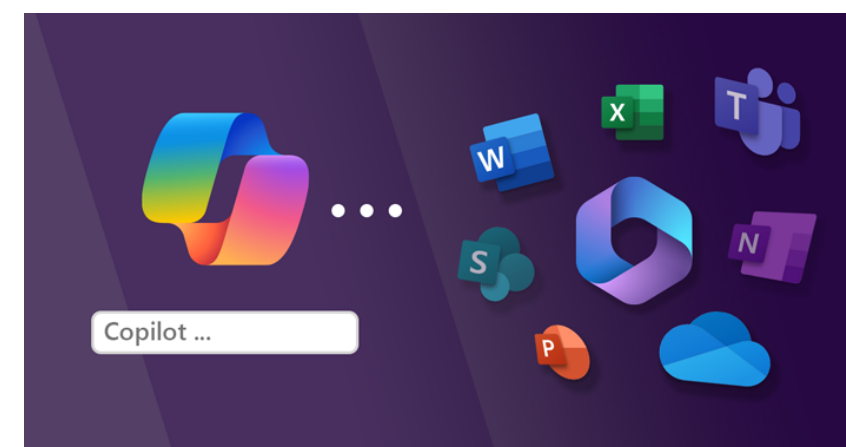
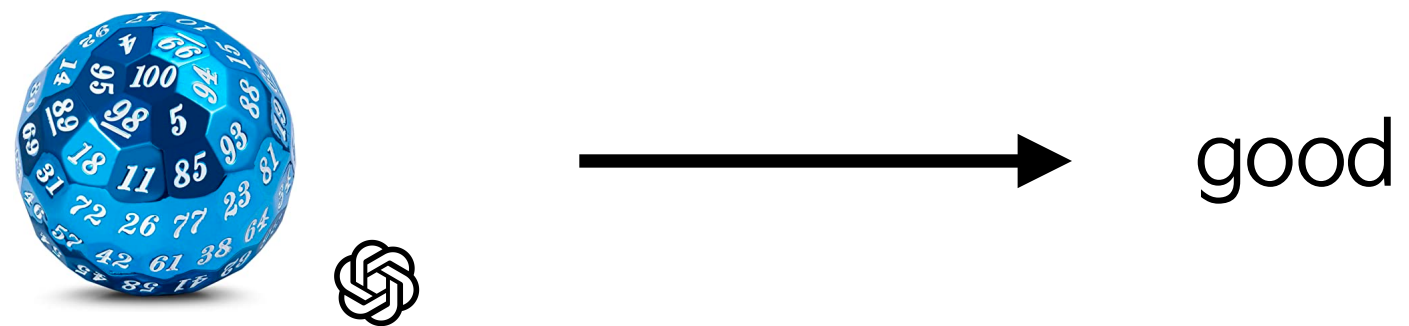
香港大學自然語言處理實驗室

Natural Language Processing Group, The University of Hong Kong

Generate a sentence

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{<i})$$

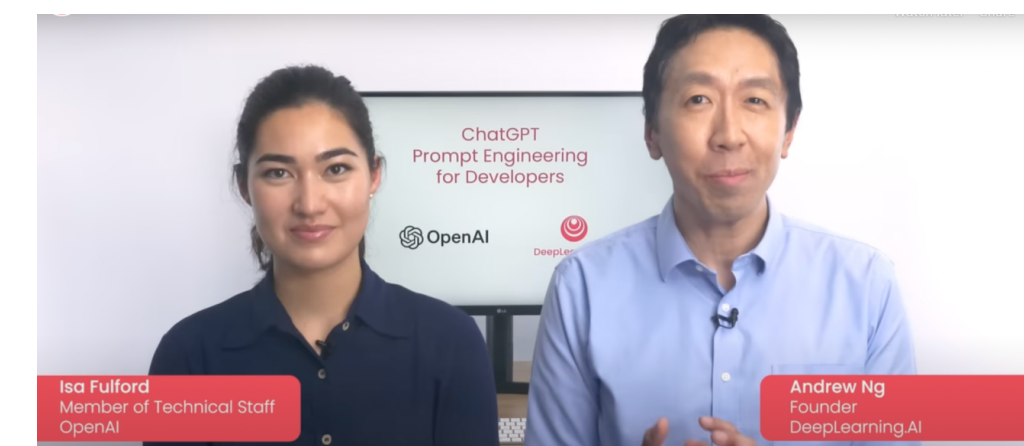
Don't just believe that it is because something is trendy that it is ...



LLMs as Agents



LLM Reasoning



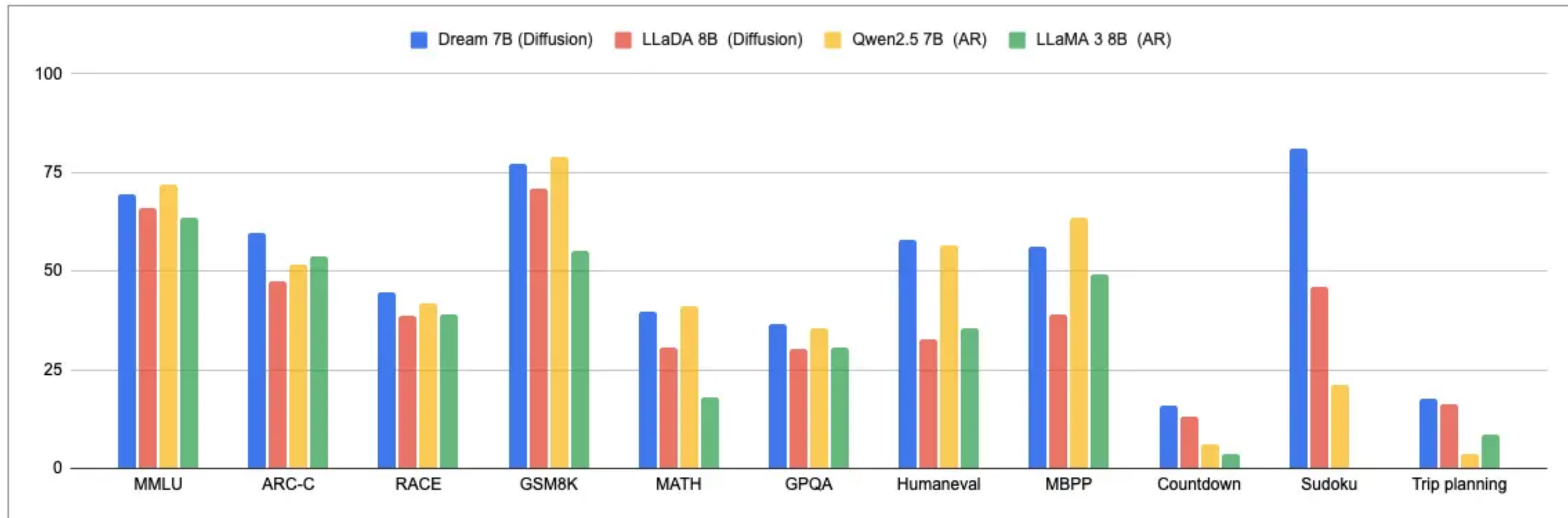
Prompt Engineering



Jiacheng Ye @JiachengYe15 · Apr 2



🚀 Excited to announce Dream 7B (Diffusion reasoning model): the most powerful open diffusion large language model to date.



Please write a Python class that implements a PyTorch trainer capable of training a model on a toy dataset.

Please write a Python class that implements a PyTorch trainer capable of training a model on a toy dataset.

Please write a Python class that implements a PyTorch trainer capable of training a model on a toy dataset.

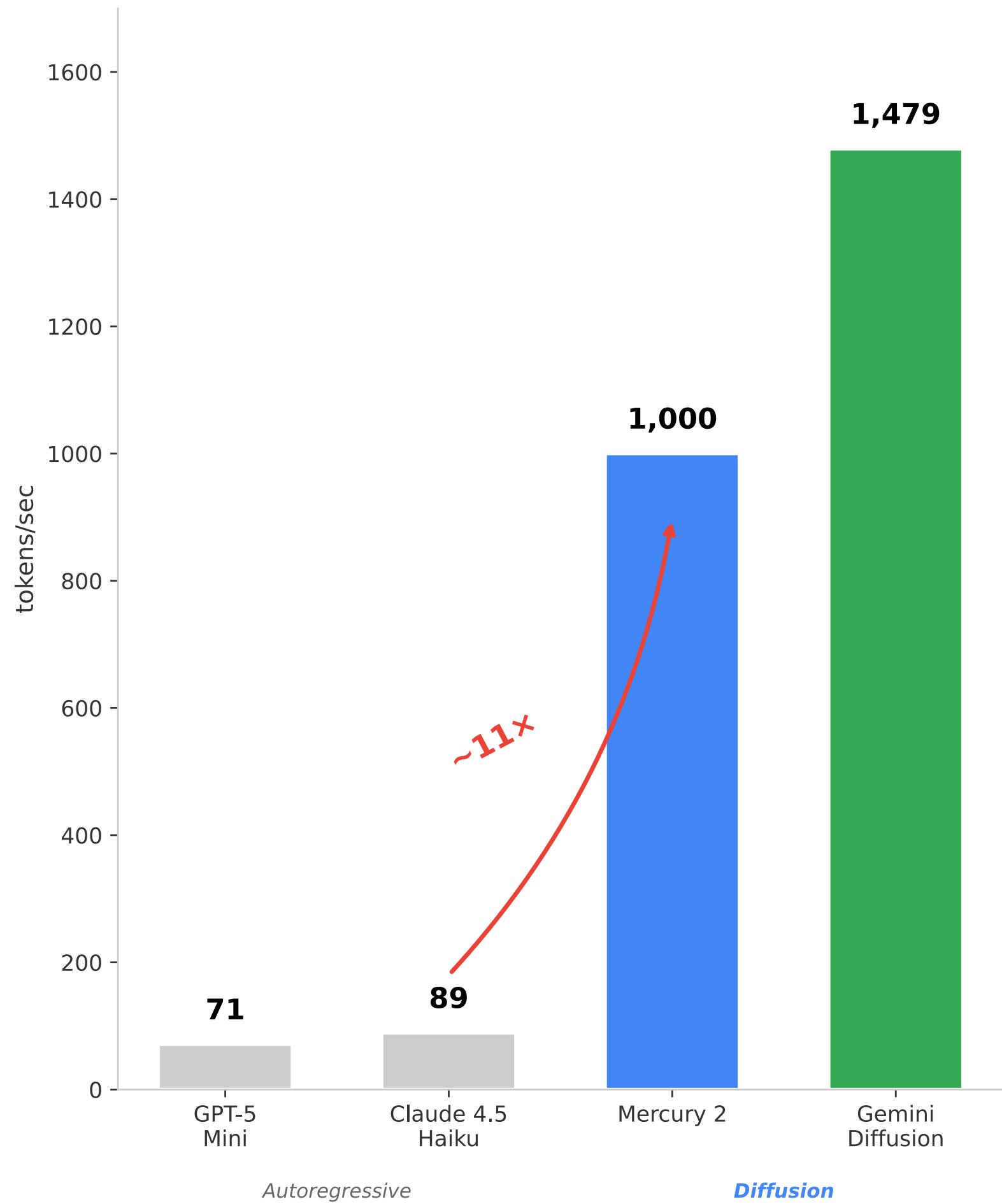
torch

the

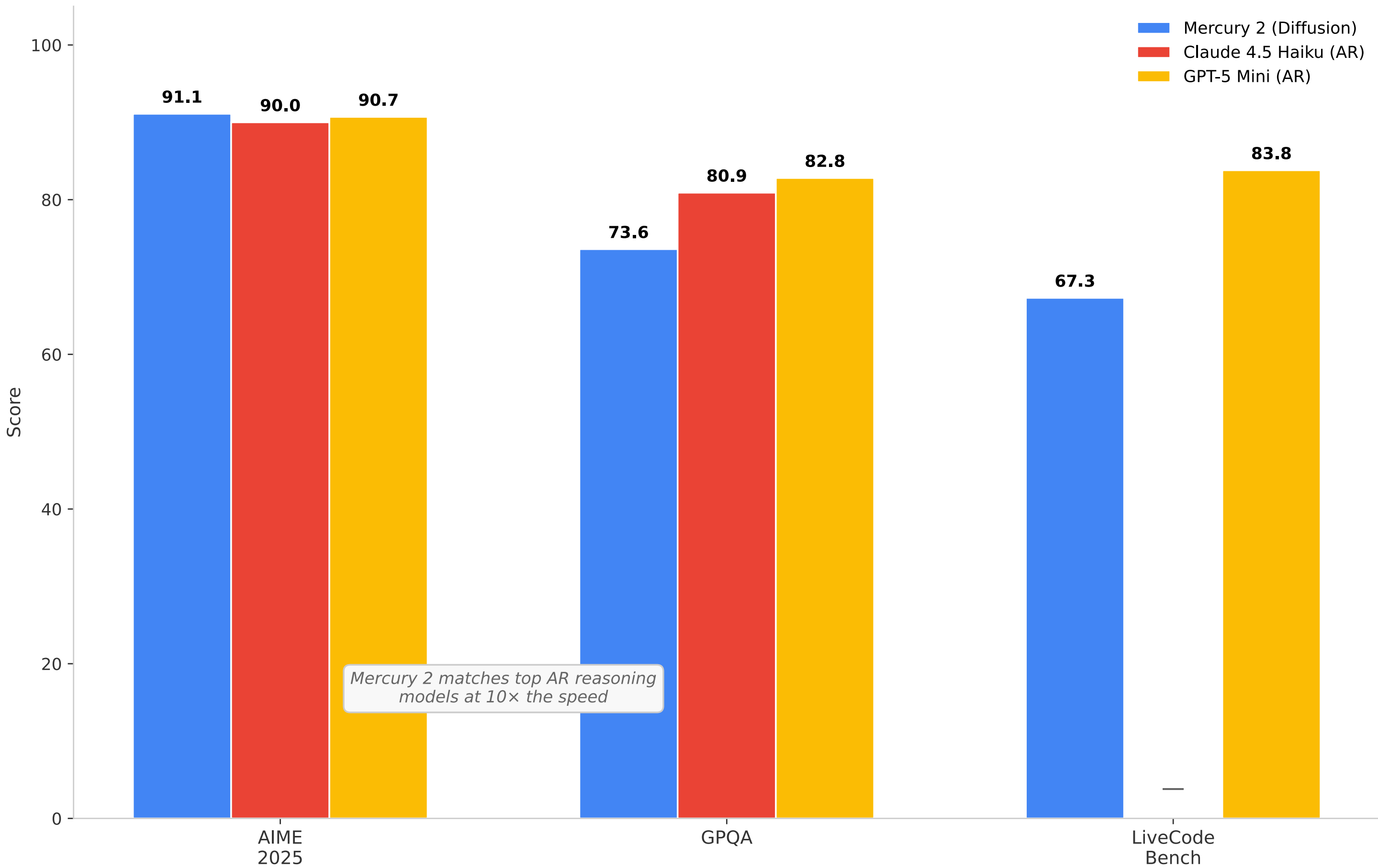
Mercury 2: The Fastest Reasoning dLLM

Inception Labs · Feb 2026 · ~1,000 tok/s · 128K context · First reasoning diffusion model

Output Speed



Reasoning Quality: dLLM vs AR Models



Iterations

0

AUTOREGRESSIVE LLM
LEFT-TO-RIGHT GENERATION

Iterations

0

INCEPTION DIFFUSION LLM
COARSE-TO-FINE GENERATION

The Bitter Lesson

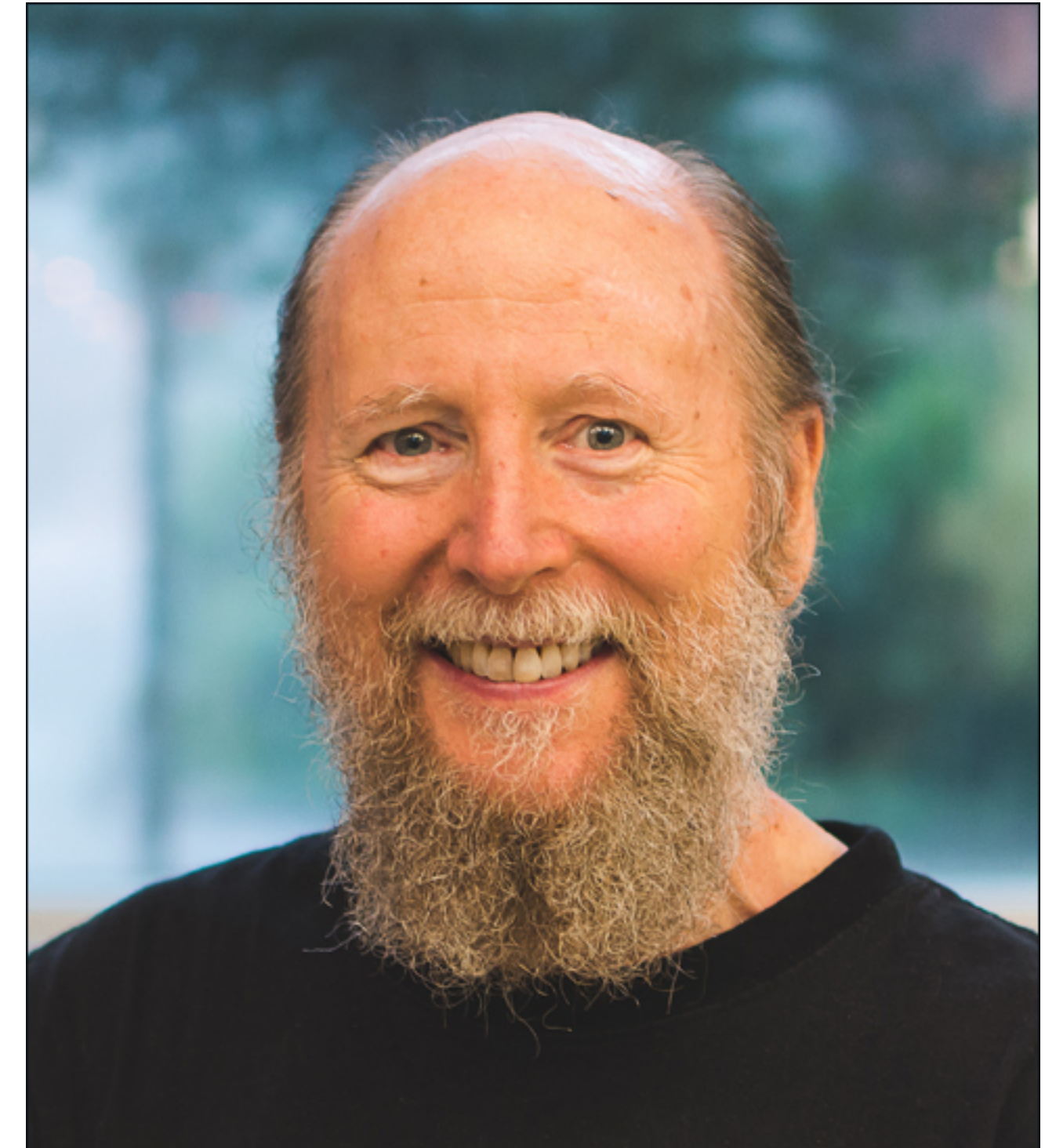
Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that "brute force" search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing search and learning.



Richard Sutton

Is Structural Bias Helpful?



Dream 7B - Diffusion Language Model Demo

[\[Model Card\]](#) [\[Blog\]](#)

🗨 Conversation

Type your message here...

Send

Word Constraints (Optional)

Format: 'pos:word, pos:word,...'. Example: '0:Once, 5:upon'

0:Hello, 10:world

Generation Settings

Clear Conversation

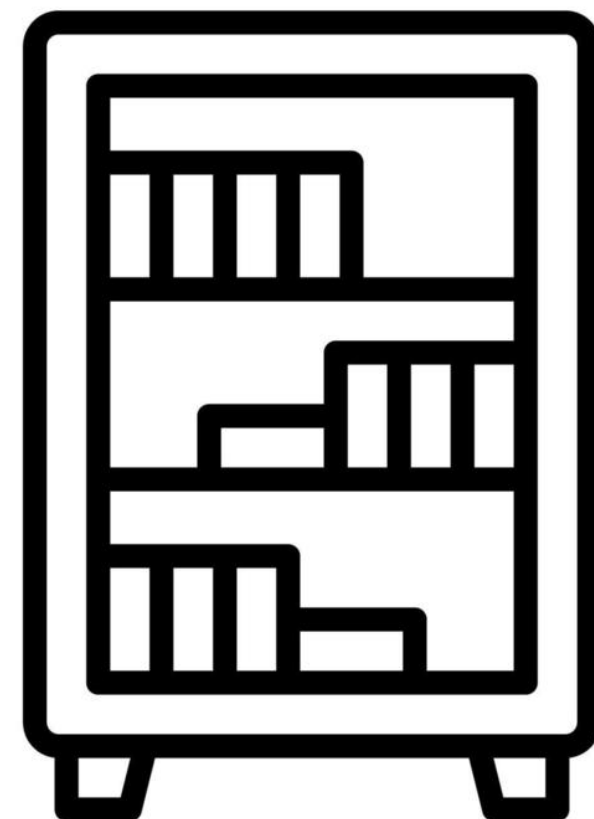
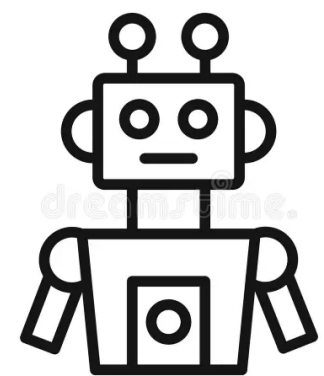
📊 Denoising Process Visualization



Autoregressive LLMs

Instruction: Put salt on steak

$p(\text{“Pick up the saltbottle.”} \mid \text{“Go to the shelf. A saltbottle is on the shelf.”})$



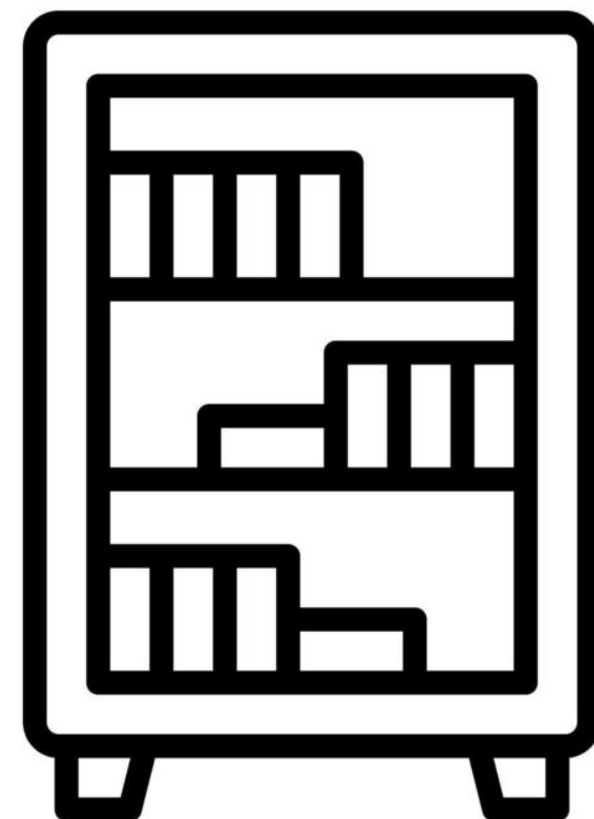
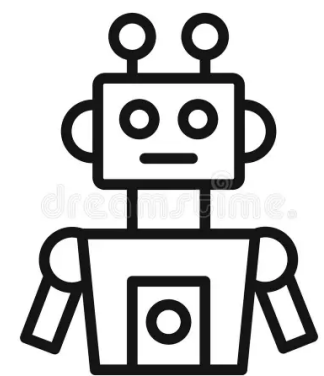
$p(\text{“Go to the shelf.”} \mid \text{“There is a shelf in front of you.”})$



Autoregressive LLMs

Instruction: Put salt on steak

$p(\text{“Pick up the } \textcolor{red}{spicebottle}\text{”} \mid \text{“Go to the shelf. A } \textcolor{red}{spicebottle}\text{ is on the shelf.”})$



$p(\text{“Go to the shelf.”} \mid \text{“There is a shelf in front of you.”})$



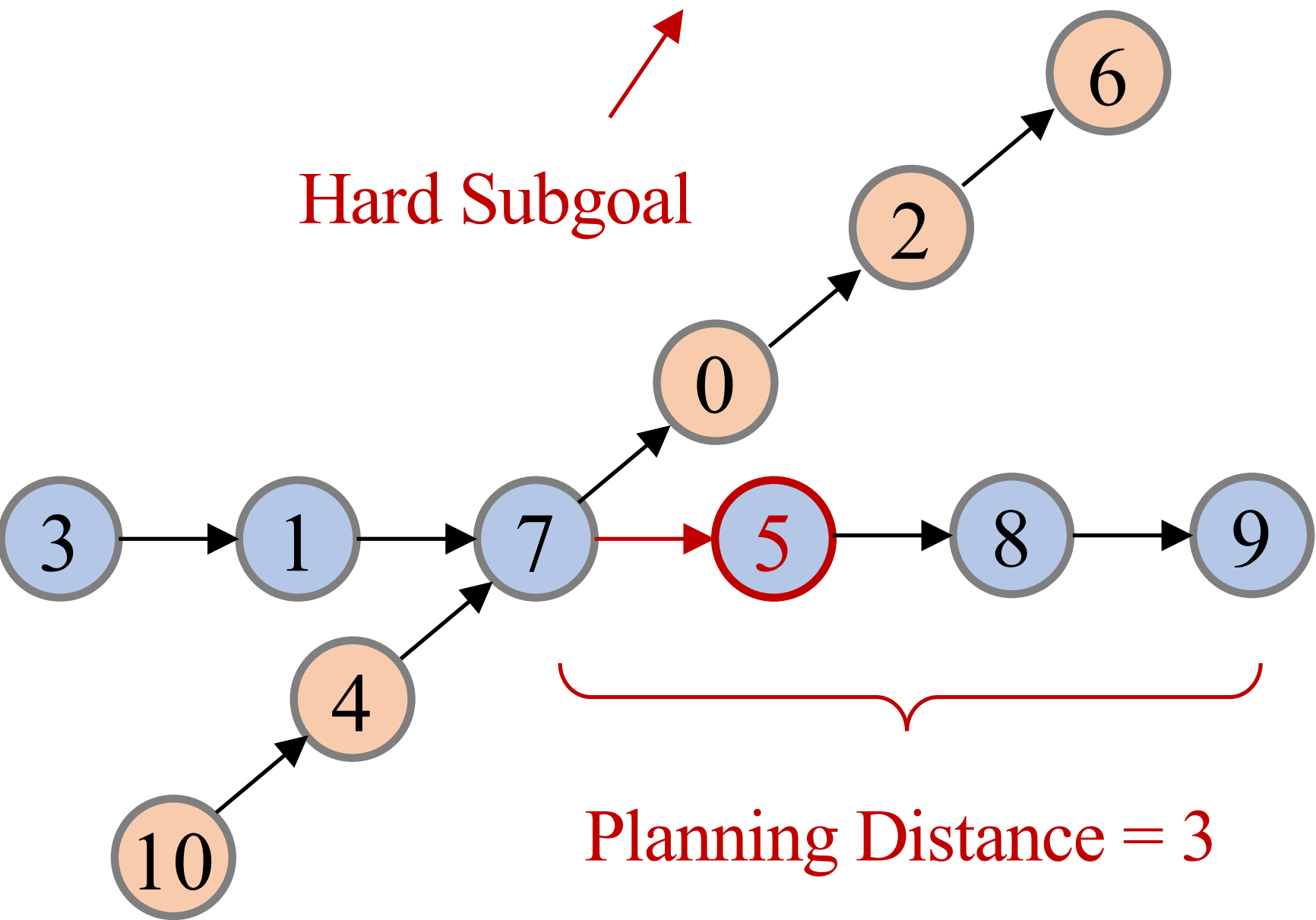
The Planning task

$$\begin{array}{l} 4+10 \quad 14 \quad 14-12 \quad 2 \quad \underline{13*2=24} \\ 3*13-39, 9+10-19, \underline{39-19-24} \\ 1+10-11 \quad 13-11=2, \underline{7*2=24} \end{array}$$

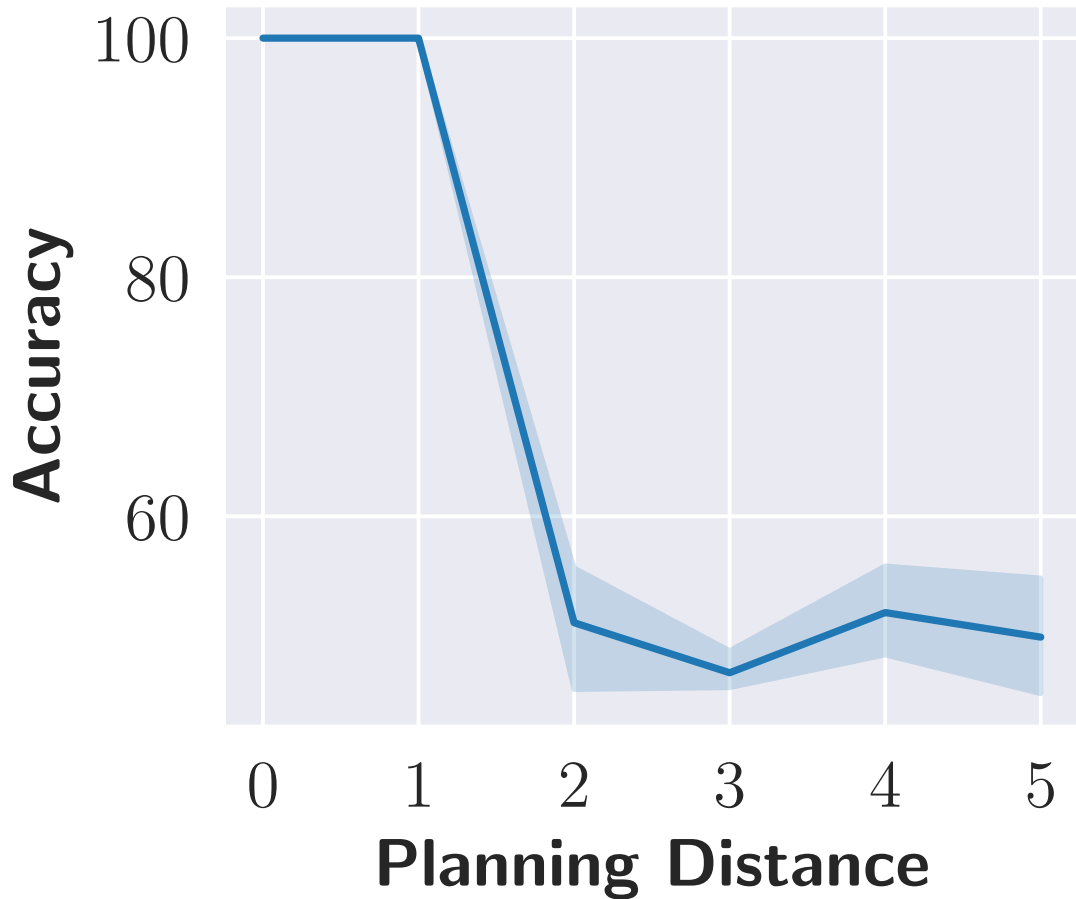
Input: 4,7 | 5,8 | 7,0 | 3,1 | ... | 0,2 / 3,9

shuffled edges start,goal

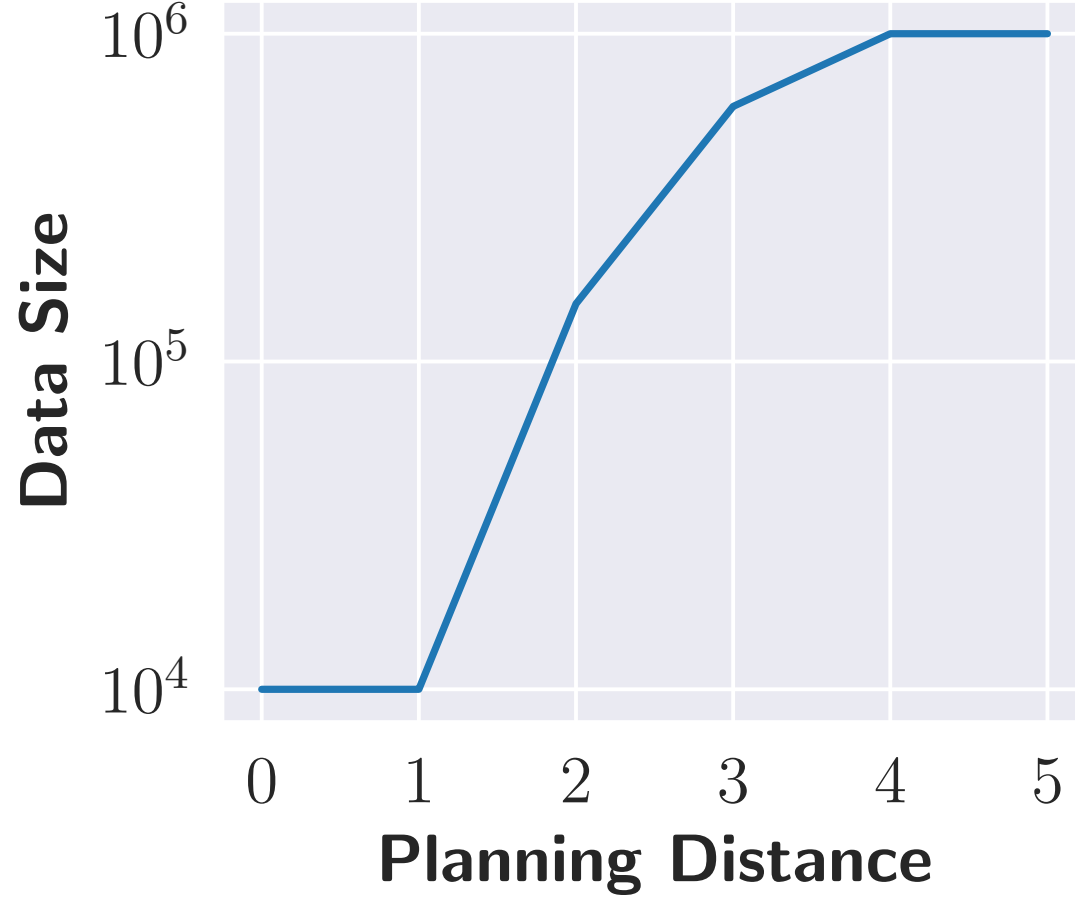
Output: 3,1 | 1,7 | 7,5 | 5,8 | 8,9



Countdown Game



(a) Accuracy of different method given 50k training data.



(b) Minimum data size required to solve (i.e., accuracy above 90%) subgoal at each planning distance.

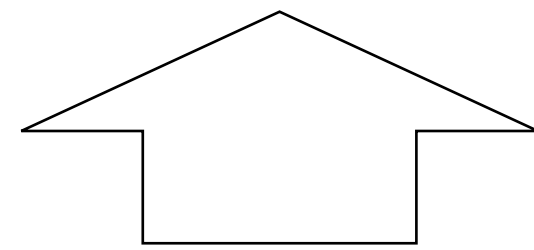
Non-autogressive Text Generation

Tokyo is the largest city in the world



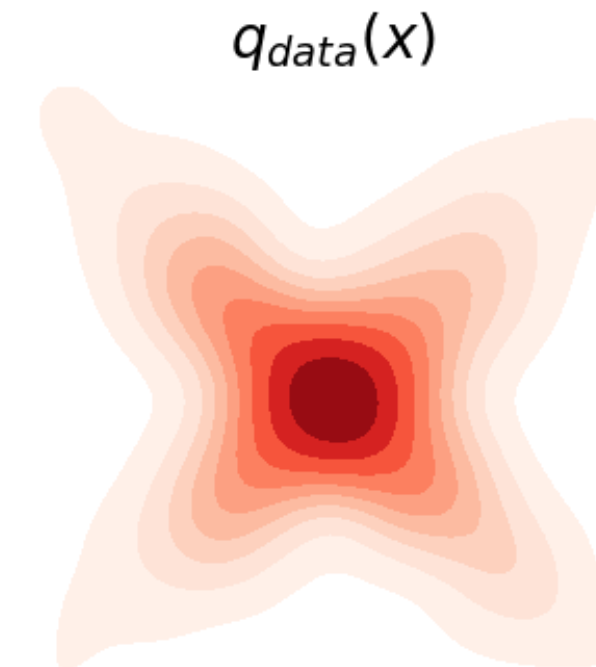
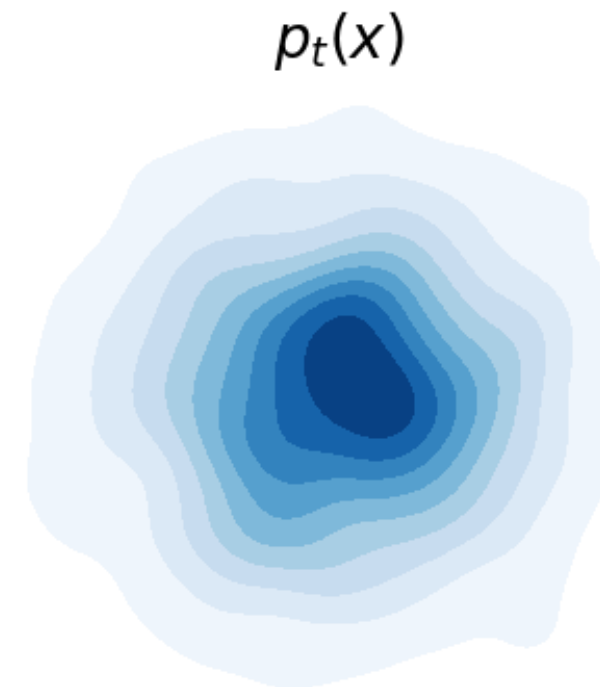
Iterative Refinement

Shanghai is the largest city in the world



Parallel Generation

[mask] [mask] [mask] [mask] [mask] [mask] [mask] [mask]



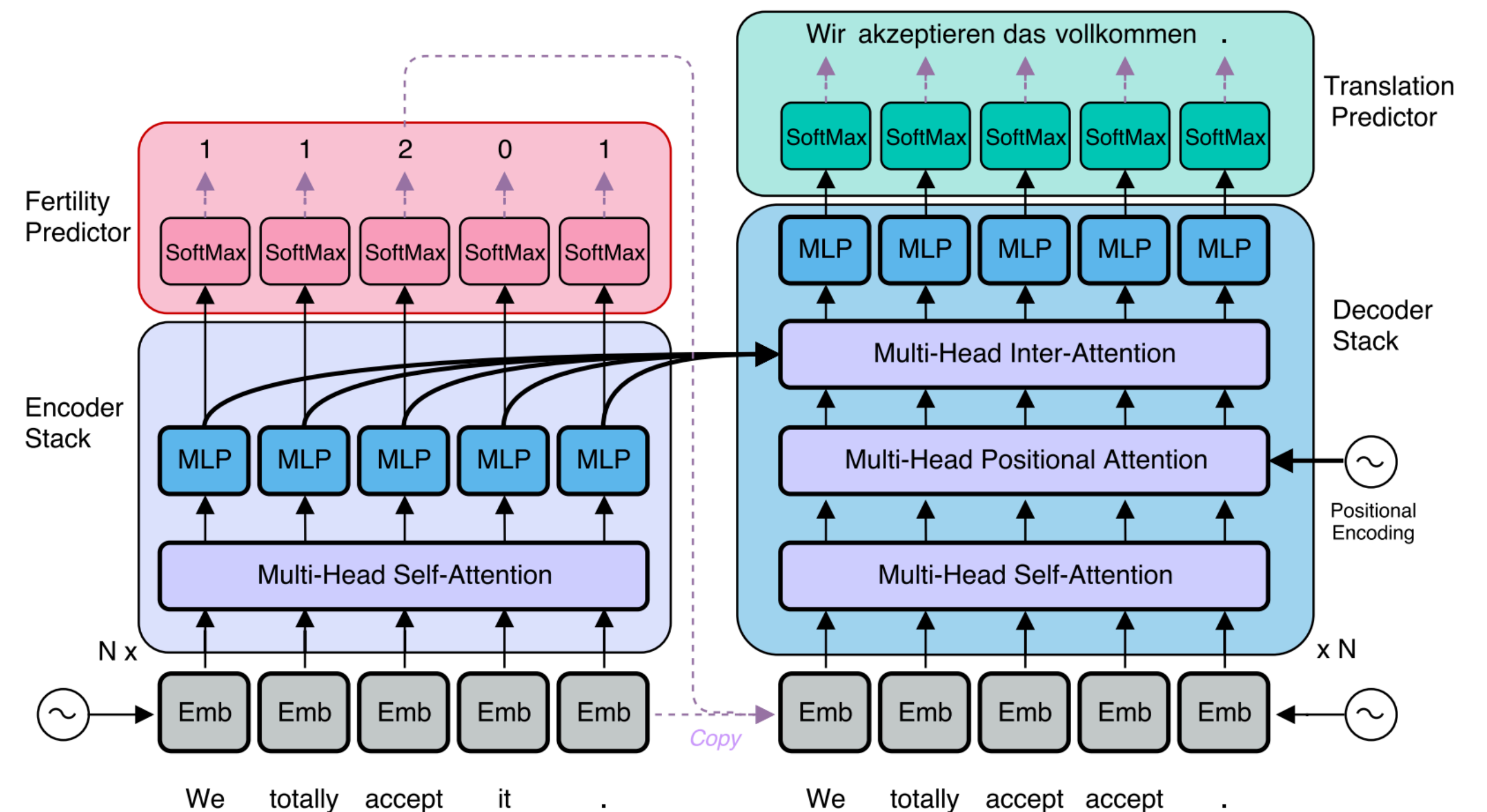
In a different “family” of $p(x)$ than the autoregressive ones

Non-autogressive Neural Machine Translation

Models	WMT14	
	En→De	De→En
NAT	17.35	20.62
NAT (+FT)	17.69	21.47
NAT (+FT + NPD $s = 10$)	18.66	22.41
NAT (+FT + NPD $s = 100$)	19.17	23.20
Autoregressive ($b = 1$)	22.71	26.39
Autoregressive ($b = 4$)	23.45	27.02

many thanks
thank you → thank thanks

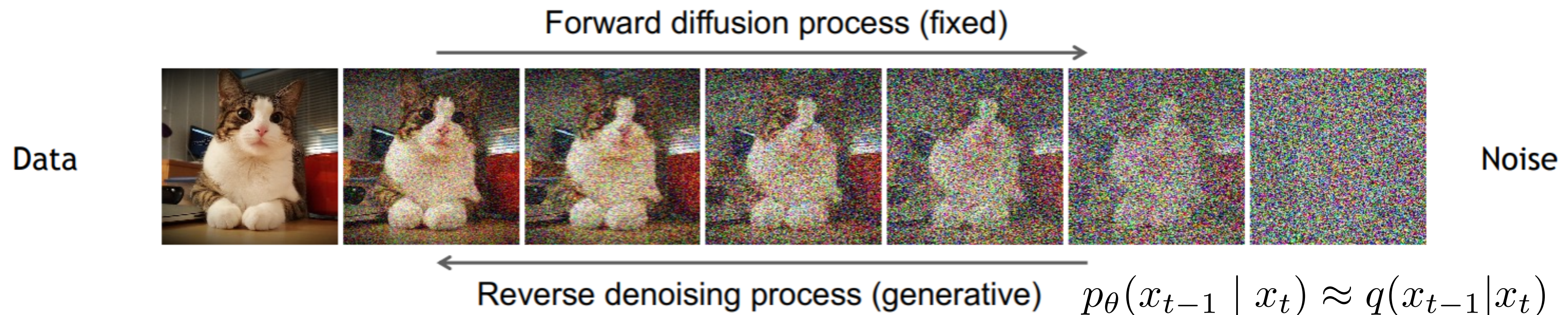
“Modality Conflicts”



Diffusion Models

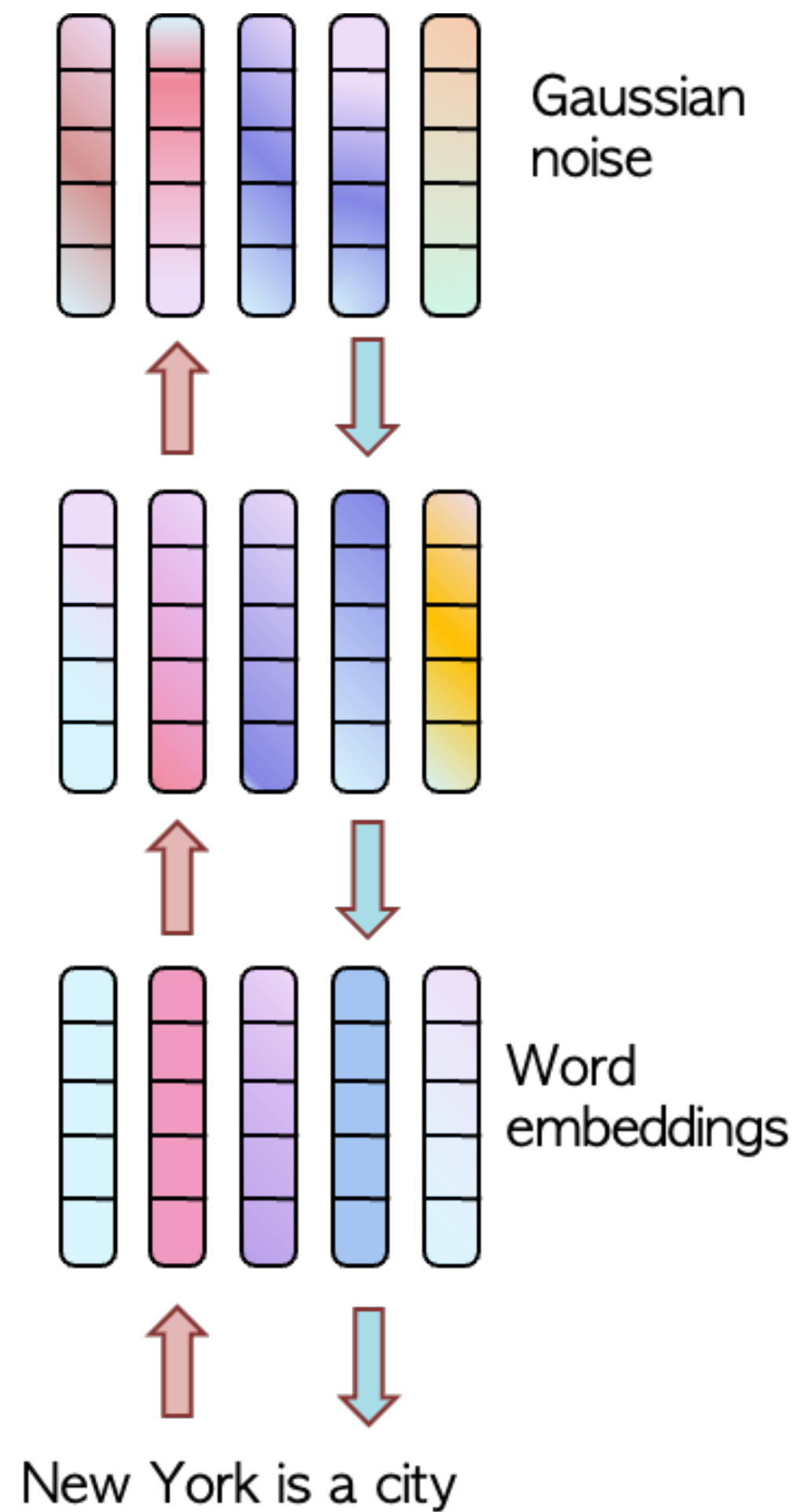
Forward-backward formulation

The forward process gradually injects noise to the input
The backward process denoises to recover the original data

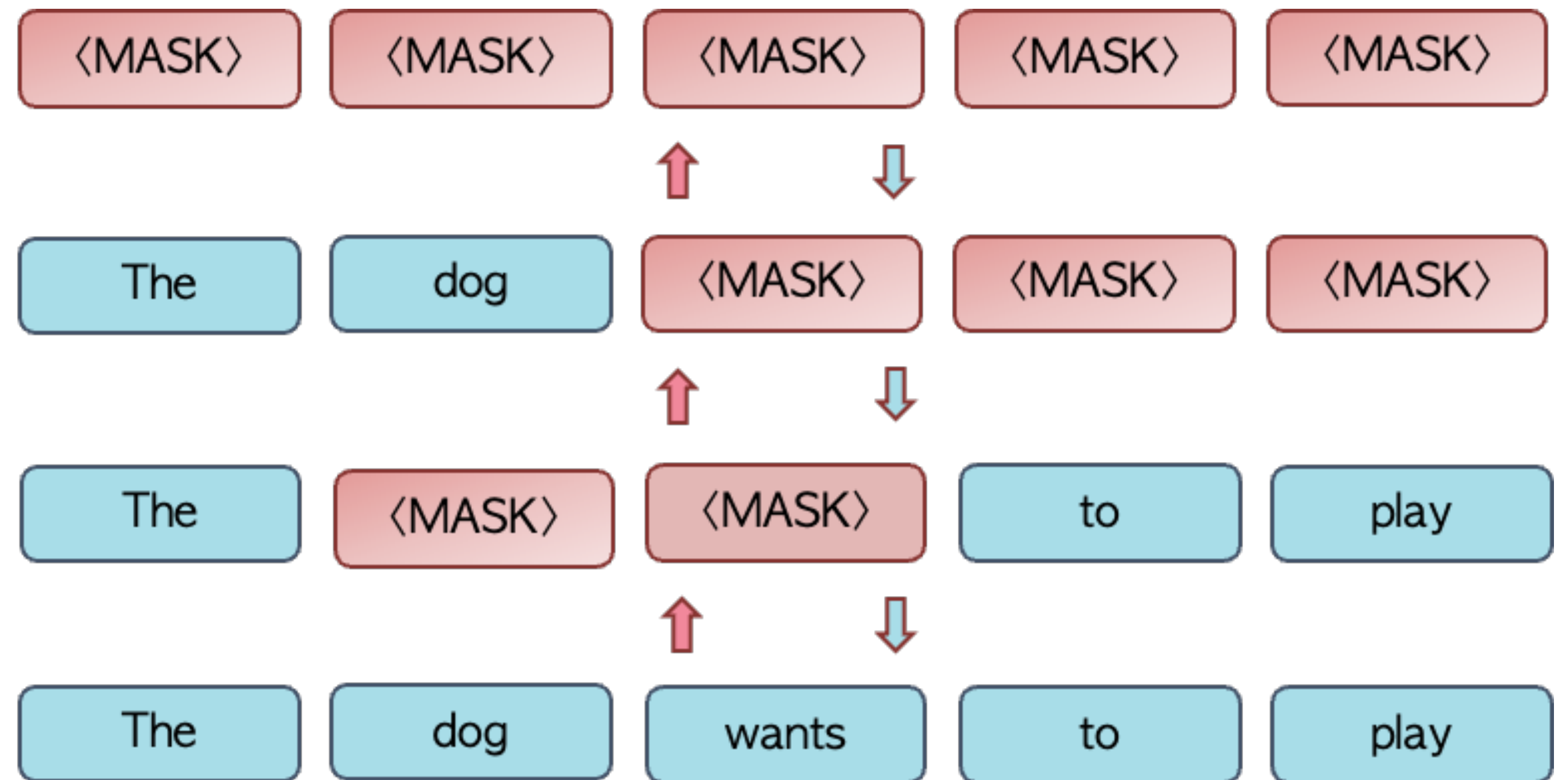


“Creating noise from data is easy; creating data from noise is generative modeling.” (Song et al., 2021)

Text Diffusion Models

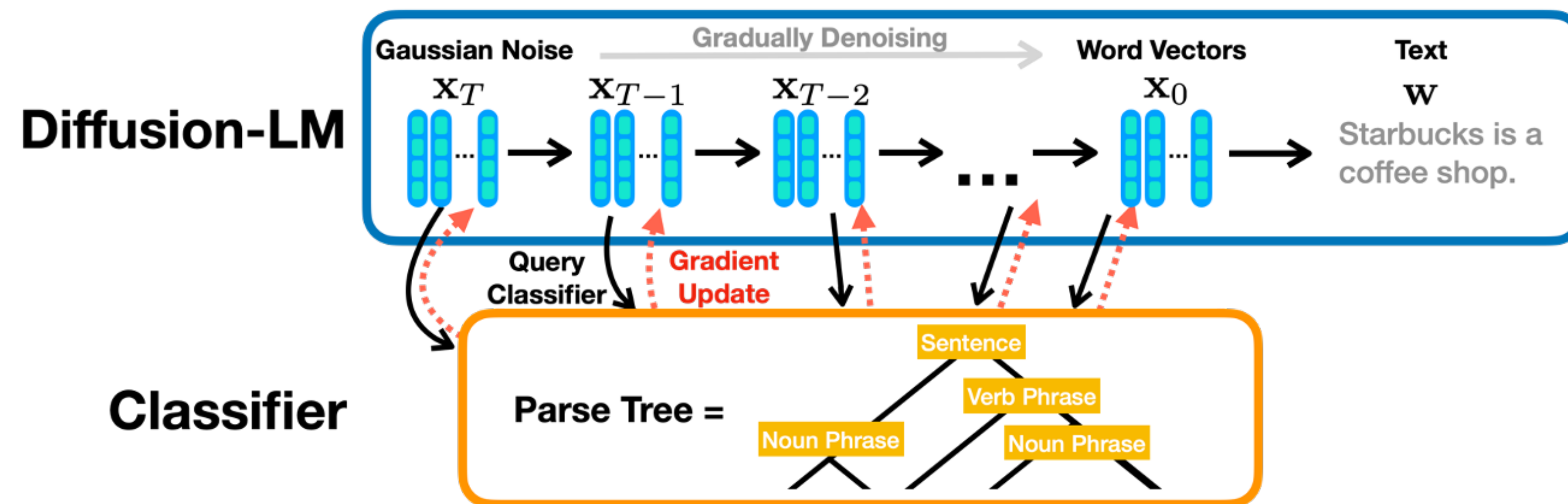
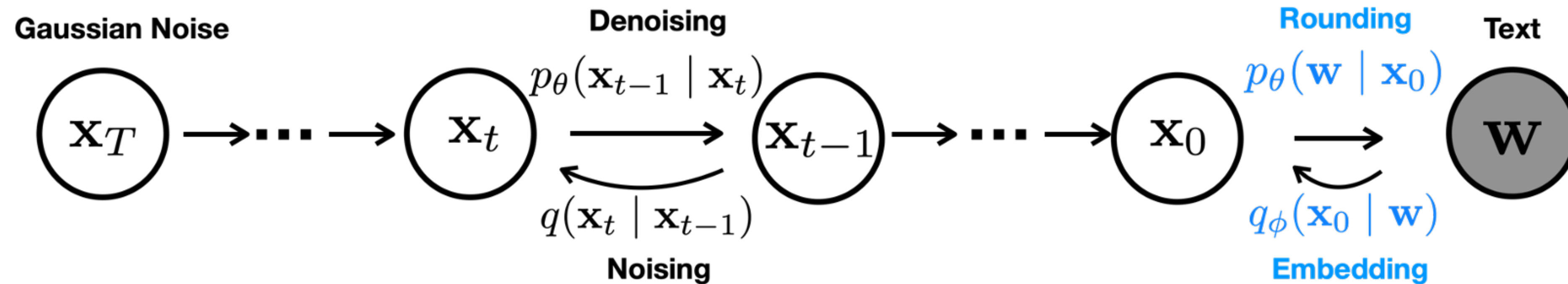


Continuous Diffusion



Discrete Diffusion

Text Diffusion Models



Discrete Diffusion - Forward Process

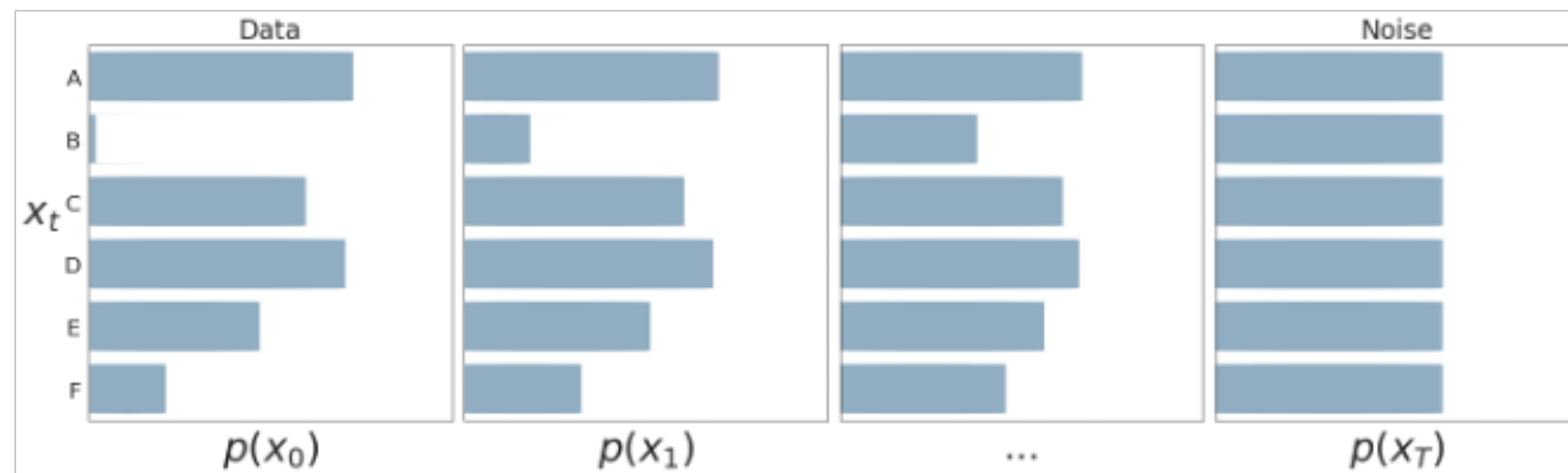
$$q(x_0) = p_{\text{data}}(x)$$

Noise: $q(x_t \mid x_{t-1}) = \beta_t x_{t-1} + (1 - \beta_t) q_{\text{noise}}$

Enjoys closed-form “marginal” distribution:

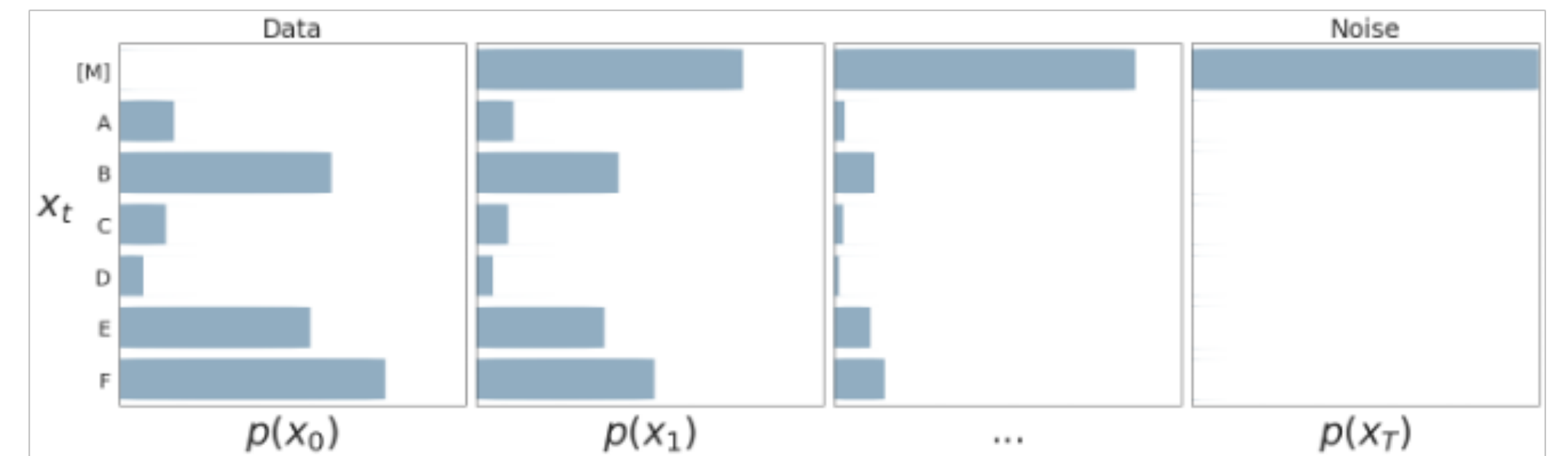
$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \alpha_t \mathbf{x}_0 + (1 - \alpha_t) q_{\text{noise}}, \quad \alpha_t := \prod_{i=1}^t \beta_i$$

“Multinomial”



$$q_{\text{noise}} = \frac{1}{K}$$

“Absorbing”



$$q_{\text{noise}} = \delta([M])$$

Discrete Diffusion - Learning

Goal: $p_{\theta}(x_{t-1} \mid x_t) \approx q(x_{t-1} \mid x_t)$

$$\begin{aligned} \log p_{\theta}(\mathbf{x}_0) &\geq \mathbb{E}_{q(\mathbf{x}_1, \dots, \mathbf{x}_T \mid \mathbf{x}_0)} \left[\log \frac{p_{\theta}(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T)}{q(\mathbf{x}_1, \dots, \mathbf{x}_T \mid \mathbf{x}_0)} \right] \\ &= \mathbb{E}_q \left[\underbrace{\log p_{\theta}(\mathbf{x}_0 \mid \mathbf{x}_1)}_{\mathcal{L}_1(\theta)} - \sum_{t=2}^T \underbrace{\text{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t))}_{\mathcal{L}_t(\theta)} \right] + \text{const.} \end{aligned}$$

Enjoys closed-form “marginal” distribution:

“Multinomial” $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{(\beta_t \mathbf{x}_t + (1 - \beta_t) \frac{\mathbf{1}}{K}) \odot (\alpha_{t-1} \mathbf{x}_0 + (1 - \alpha_{t-1}) \frac{\mathbf{1}}{K})}{\mathbf{x}_t^{\top} (\alpha_t \mathbf{x}_0 + (1 - \alpha_t) \frac{\mathbf{1}}{K})}$

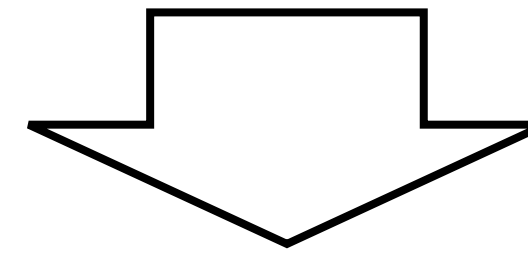
“Absorbing” $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \mathbf{x}_t, & \text{if } \mathbf{x}_t \neq \mathbf{e}_{[M]}, \\ \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \mathbf{e}_{[M]} + \frac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t} \mathbf{x}_0, & \text{if } \mathbf{x}_t = \mathbf{e}_{[M]} \end{cases}$

Reparameterized Discrete Diffusion (RDMs)

Key Observation:

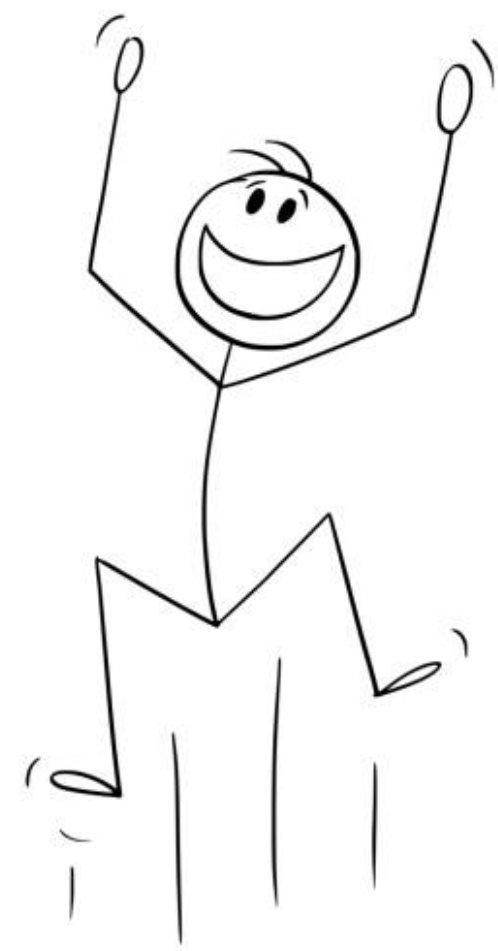
“Multinomial” $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \frac{(\beta_t \mathbf{x}_t + (1 - \beta_t) \frac{1}{K}) \odot (\alpha_{t-1} \mathbf{x}_0 + (1 - \alpha_{t-1}) \frac{1}{K})}{\mathbf{x}_t^\top (\alpha_t \mathbf{x}_0 + (1 - \alpha_t) \frac{1}{K})}$

“Absorbing” $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \mathbf{x}_t, & \text{if } \mathbf{x}_t \neq \mathbf{e}_{[M]}, \\ \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \mathbf{e}_{[M]} + \frac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t} \mathbf{x}_0, & \text{if } \mathbf{x}_t = \mathbf{e}_{[M]} \end{cases}$



$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \lambda_{t-1}^{(1)} \mathbf{x}_t + \left(1 - \lambda_{t-1}^{(1)}\right) q_{\text{noise}}, & \text{if } \mathbf{x}_t = \mathbf{x}_0 \\ \lambda_{t-1}^{(2)} \mathbf{x}_0 + \left(1 - \lambda_{t-1}^{(2)}\right) q_{\text{noise}}(\mathbf{x}_t), & \text{if } \mathbf{x}_t \neq \mathbf{x}_0 \end{cases}$$

$q_{\text{noise}}(\mathbf{x}_t) = \beta_t \mathbf{x}_t + (1 - \beta_t) q_{\text{noise}}$

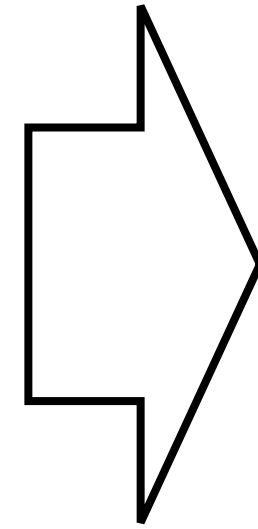


Sampling Reparameterization

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \lambda_{t-1}^{(1)}\mathbf{x}_t + \left(1 - \lambda_{t-1}^{(1)}\right) q_{\text{noise}}, & \text{if } \mathbf{x}_t = \mathbf{x}_0 \\ \lambda_{t-1}^{(2)}\mathbf{x}_0 + \left(1 - \lambda_{t-1}^{(2)}\right) q_{\text{noise}}(\mathbf{x}_t), & \text{if } \mathbf{x}_t \neq \mathbf{x}_0 \end{cases}$$

“Sampling from a mixture distribution”

$$\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$$



$$b_t = \mathbf{1}_{\mathbf{x}_t = \mathbf{x}_0}$$

$$v_{t-1}^{(1)} \sim \text{Bernoulli}\left(\lambda_{t-1}^{(1)}\right), \quad \mathbf{u}_t^{(1)} \sim q_{\text{noise}}$$

$$v_{t-1}^{(2)} \sim \text{Bernoulli}\left(\lambda_{t-1}^{(2)}\right), \quad \mathbf{u}_t^{(2)} \sim q_{\text{noise}}(\mathbf{x}_t)$$

$$\mathbf{x}_{t-1} = b_t \left[v_{t-1}^{(1)}\mathbf{x}_t + \left(1 - v_{t-1}^{(1)}\right) \mathbf{u}_t^{(1)} \right] + \\ (1 - b_t) \left[v_{t-1}^{(2)}\mathbf{x}_0 + \left(1 - v_{t-1}^{(2)}\right) \mathbf{u}_t^{(2)} \right]$$

$$q_{\text{noise}}(\mathbf{x}_t) = \beta_t\mathbf{x}_t + (1 - \beta_t)q_{\text{noise}}$$

Sampling Reparameterization

Reveals a latent routing mechanism

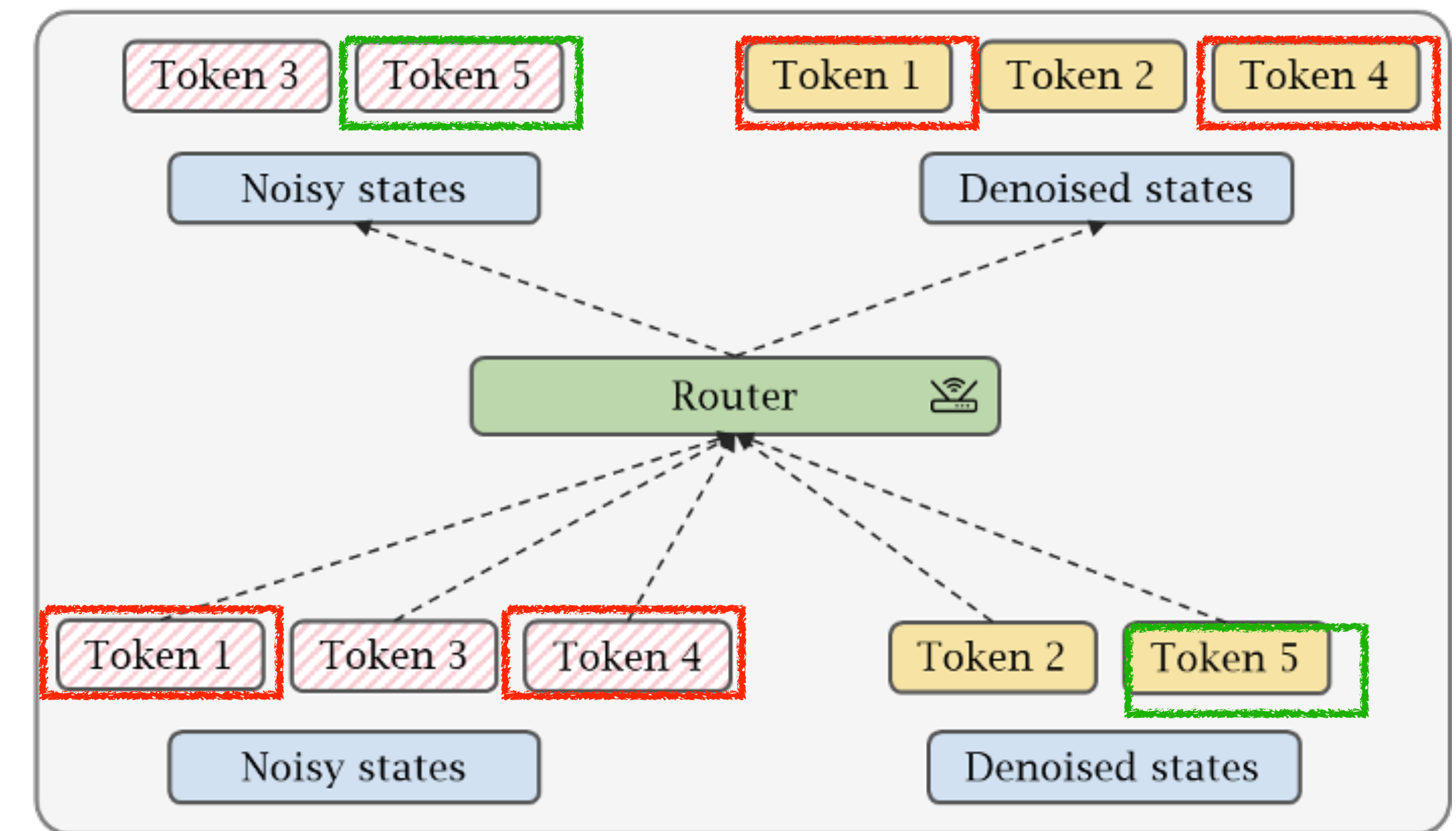
$$\begin{aligned}
 b_t &= \mathbf{1}_{\mathbf{x}_t = \mathbf{x}_0} \\
 v_{t-1}^{(1)} &\sim \text{Bernoulli}(\lambda_{t-1}^{(1)}), \quad \mathbf{u}_t^{(1)} \sim q_{\text{noise}} \\
 v_{t-1}^{(2)} &\sim \text{Bernoulli}(\lambda_{t-1}^{(2)}), \quad \mathbf{u}_t^{(2)} \sim q_{\text{noise}}(\mathbf{x}_t) \\
 \mathbf{x}_{t-1} &= b_t \left[v_{t-1}^{(1)} \mathbf{x}_t + \left(1 - v_{t-1}^{(1)}\right) \mathbf{u}_t^{(1)} \right] + \\
 &\quad (1 - b_t) \left[v_{t-1}^{(2)} \mathbf{x}_0 + \left(1 - v_{t-1}^{(2)}\right) \mathbf{u}_t^{(2)} \right]
 \end{aligned}$$

Route with $v_{t-1} := [v_{t-1}^{(1)}, v_{t-1}^{(2)}]$

Calculate $b_t = \mathbf{1}_{\mathbf{x}_t = \mathbf{x}_0}$

Sequence x_{t-1}

Token 1 Token 2 Token 3 Token 4 Token 5



Sequence x_t

Token 1 Token 2 Token 3 Token 4 Token 5

Training RDMs

Trained with ELBO:

$$\mathcal{L}_t := -\lambda_{t-1}^{(2)} \sum_{n=1}^N \mathbf{1}_{\mathbf{x}_{t,n} \neq \mathbf{x}_{0,n}} \mathbf{x}_{0,n}^\top \log f(\mathbf{x}_{t,n}; \boldsymbol{\theta})$$

Time-dependent weighting

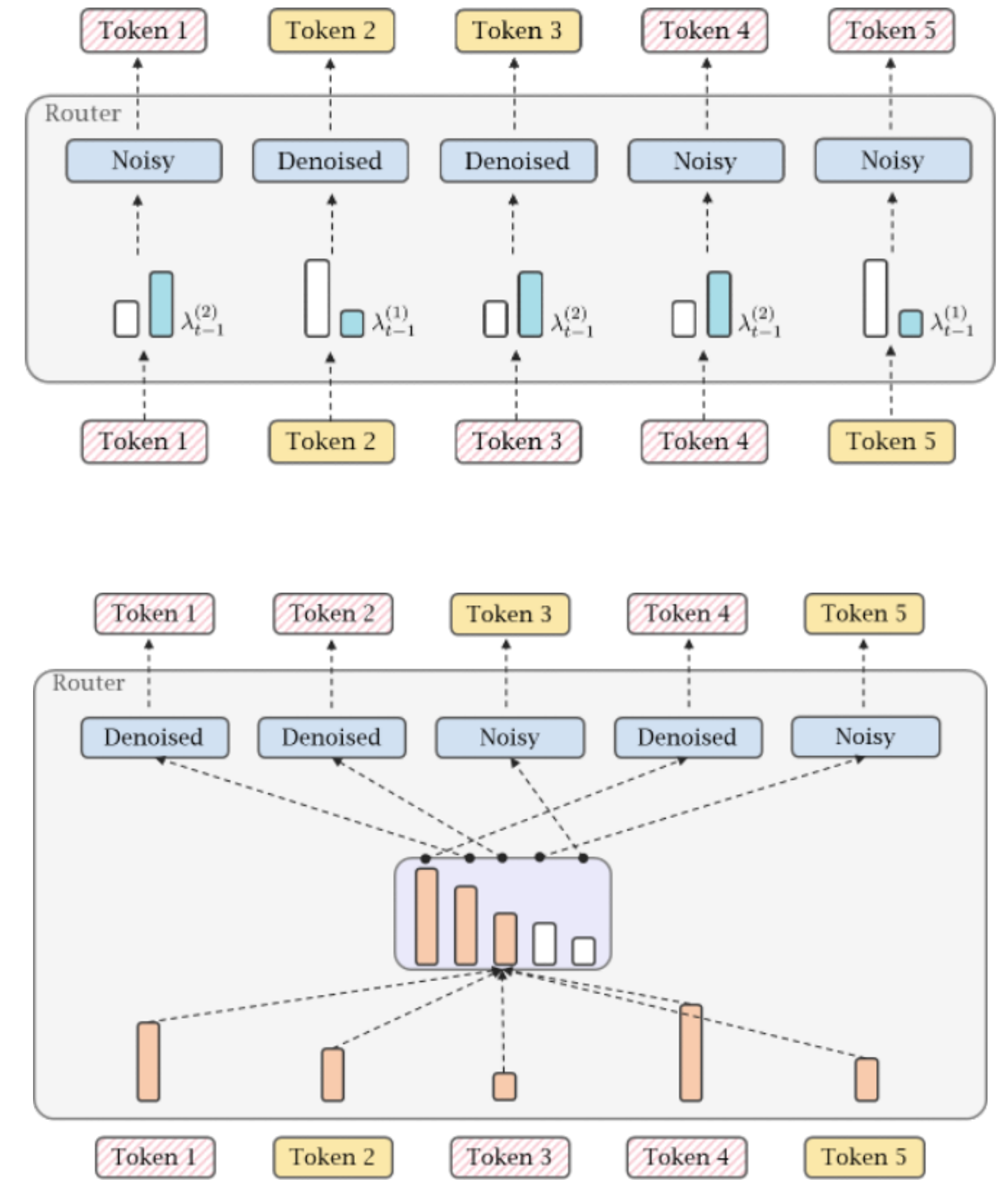
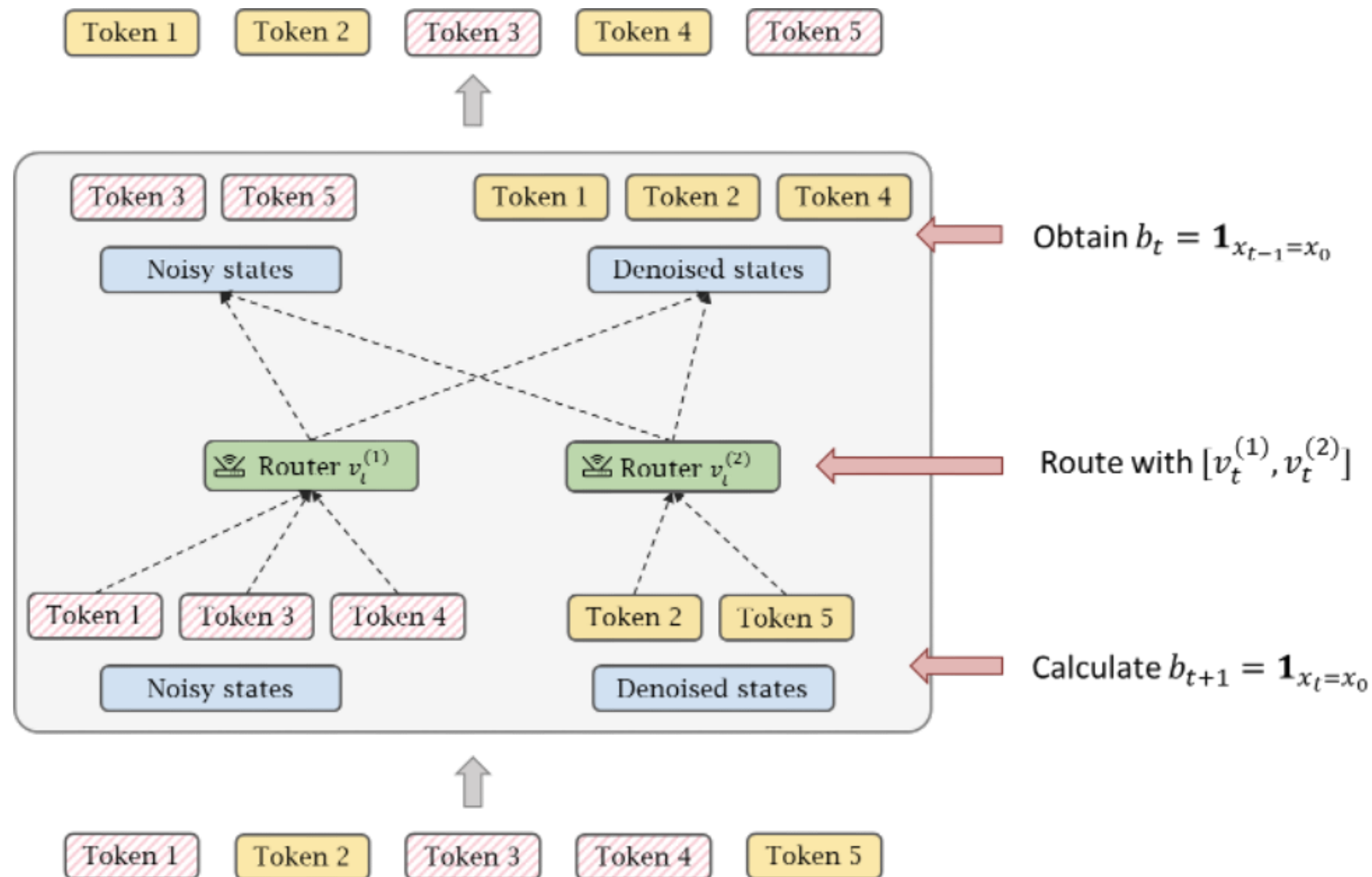
$$\lambda_{t-1}^{(2)} = 1 - \frac{t-1}{T}$$

Router-agnostic

Evaluated at noisy tokens

Simple Cross-Entropy Loss

Decoding RDMs



Confident-based Router

Experimental Results of RDMs

Close the gap with autoregressive models (Machine Translation)

	Model	# Iterations	IWSLT14 DE-EN		WMT16 EN-RO		WMT14 EN-DE	
			Vanilla	Reparam.	Vanilla	Reparam.	Vanilla	Reparam.
Continuous Diffusion	CDCD (Dieleman et al., 2022)	200	–		–		20.0*	
Discrete Diffusion	Multinomial Diffusion (Hoogeboom et al., 2021)	2	23.05	28.01	26.61	30.16	4.28	21.43
		4	24.24	30.57	27.81	31.70	4.31	24.05
		10	21.28	32.23	25.25	33.00	6.94	25.63
		16	20.59	32.58	24.36	33.11	6.07	25.64
		25	20.06	32.84	23.94	33.31	3.69	26.04
	Absorbing Diffusion (Austin et al., 2021)	2	25.24	27.60	27.24	30.72	16.46	21.00
		4	26.93	31.47	29.16	32.60	19.48	24.26
		10	28.32	33.91	30.41	33.38	21.62	26.96
		16	28.38	34.41	30.79	33.82	22.07	27.58
		25	28.93	34.49	30.56	33.99	22.52	27.59
Auto-regressive Models	Transformer-base (Vaswani et al., 2017)	n.a.	34.51		34.16		27.53	

Experimental Results of RDMs

Fix previously made errors in where vanilla discrete diffusion models get stuck

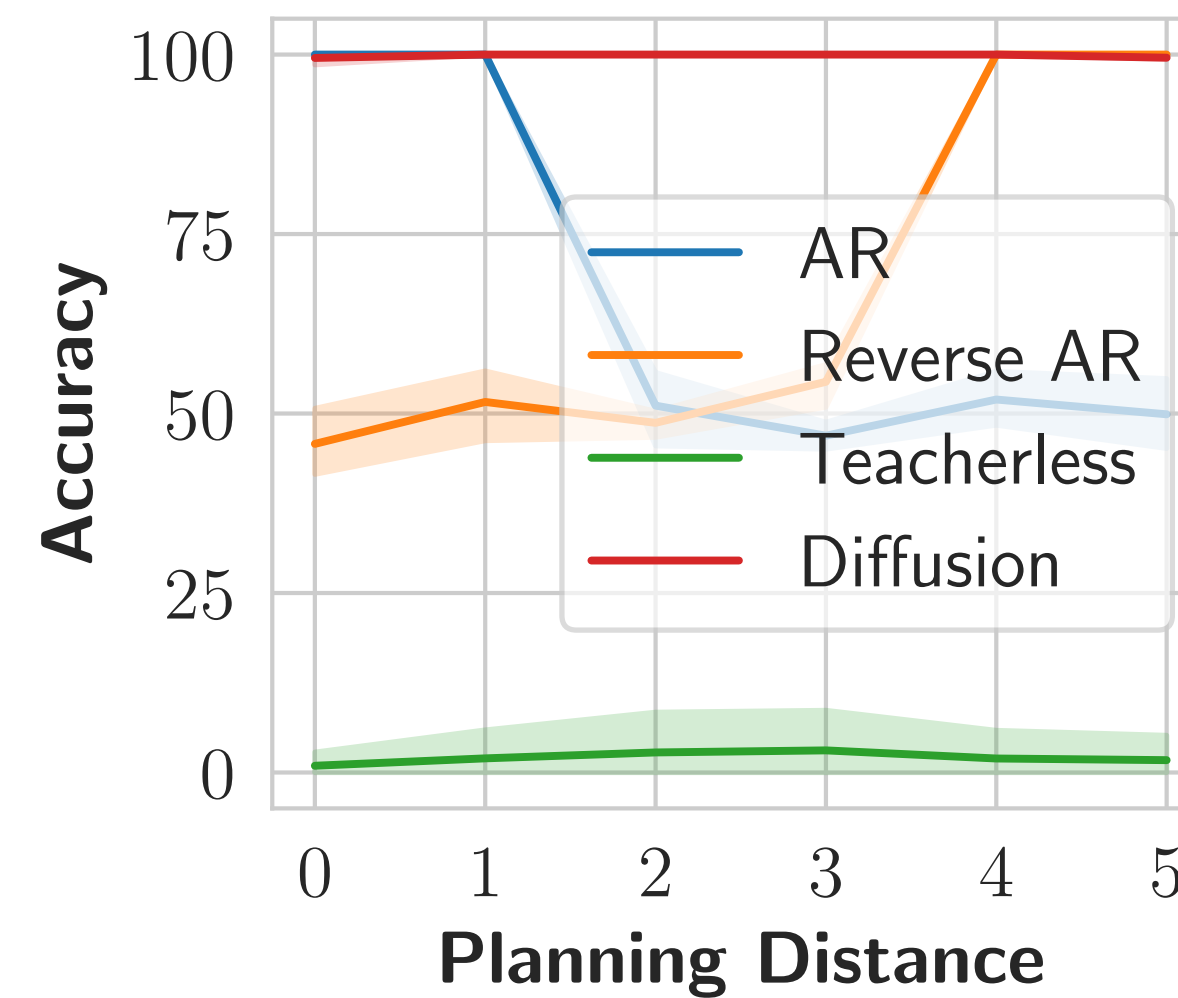
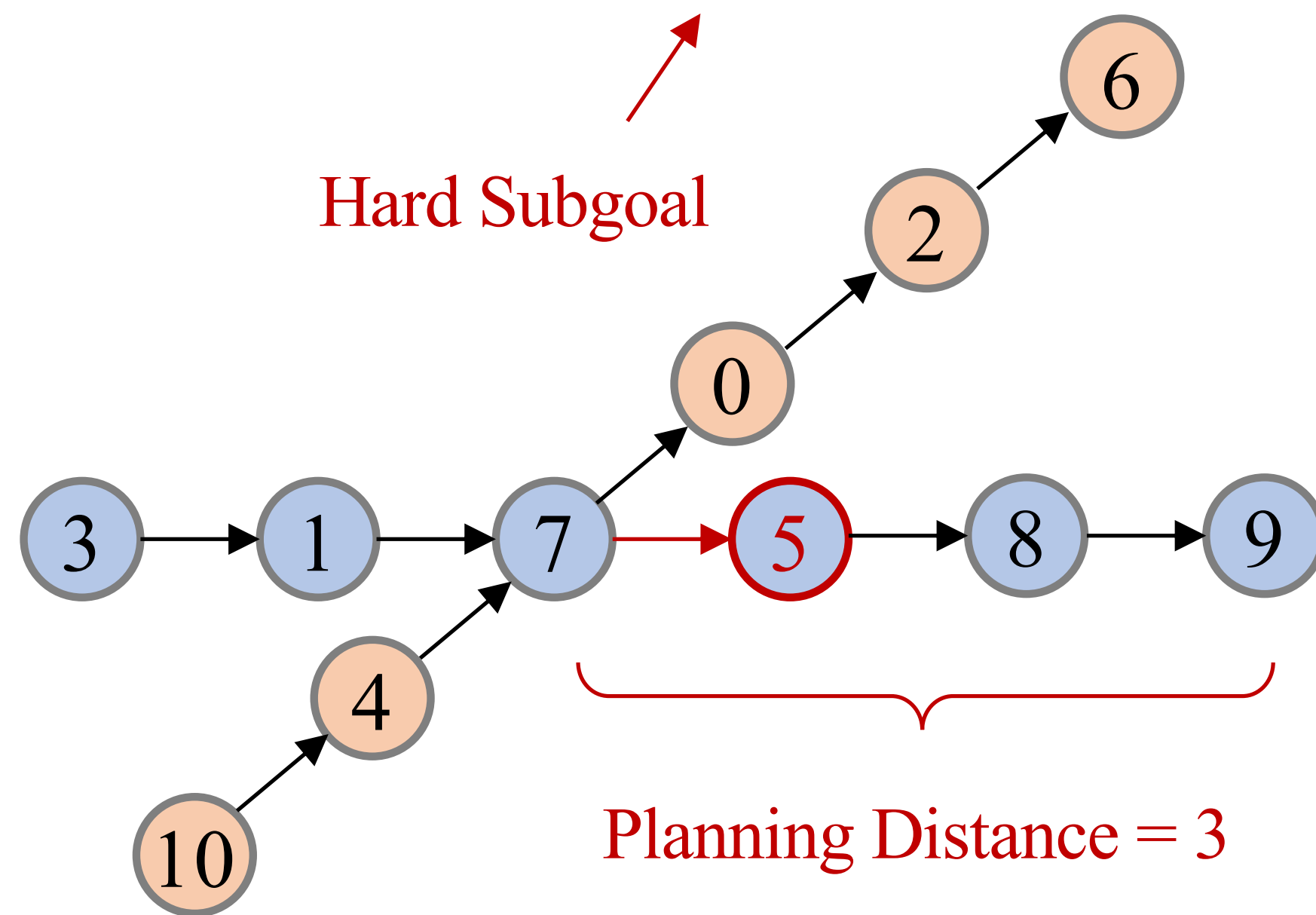
		Source: i have only 2 months for my ca cpt exams how do i prepare?
		Reference: i want to crack ca cpt in 2 months. how should i study?
		# Iter. Decodes
Absorbing	0	o <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M>
	1	o <M> <M> <M> <M> <M> ca <M> ca <M> <M> ca <M> <M> <M> <M>
	2	o <M> <M> <M> <M> <M> ca <M> ca <M> <M> ca <M> <M> <M> <M>
	3	o <M> <M> <M> <M> <M> ca - ca cp <M> ca <M> <M> exam <M>
	4	o <M> can <M> prepare <M> ca - ca cp <M> ca <M> ##t exam ?
	5	o how can i prepare for ca - ca cp ##t ca cp ##t exam ?
RDM-absorbing	0	o <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M>
	1	o how <M> i <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> <M> ?
	2	o how <M> i prepare for ca <M> ##t <M> <M> <M> <M> <M> ?
	3	o how <M> i prepare for ca cp ##t <M> <M> <M> months months ?
	4	o how <M> i prepare for ca cp ##t exam in two months <M> ?
	5	o how can i prepare for ca cp ##t exam in two months left ?

The Planning task (Revisit)

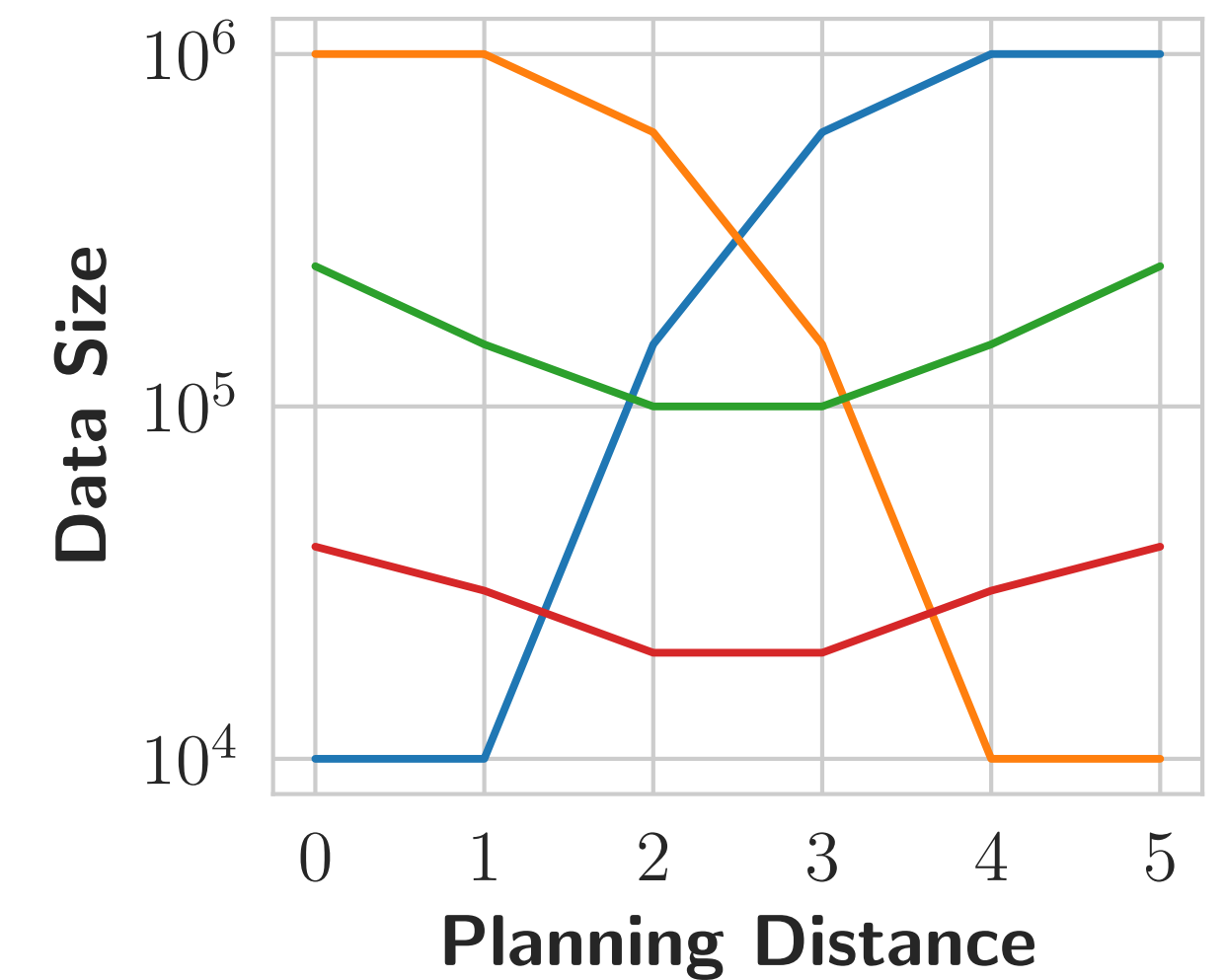
Input: $4,7 \mid 5,8 \mid 7,0 \mid 3,1 \mid \dots \mid 0,2 \mid 3,9$

shuffled edges start,goal

Output: 3,1 | 1,7 | 7,5 | 5,8 | 8,9



(a) Accuracy of different method given 50k training data.

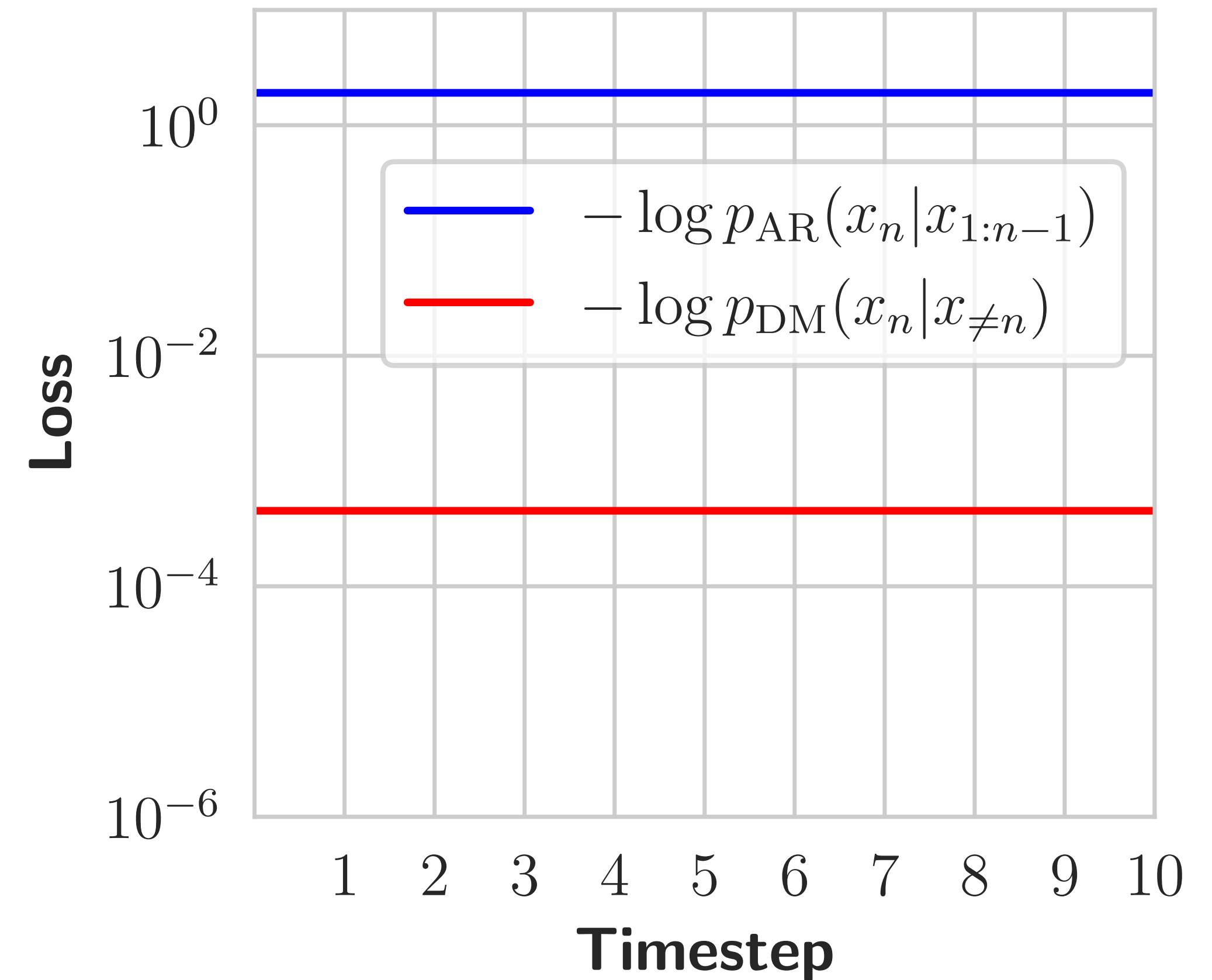


(b) Minimum data size required to solve (i.e., accuracy above 90\%) subgoal at each planning distance.

Effective Hard Subgoal Learning

$$L_{\text{AR}} = -\mathbb{E}_{q(\mathbf{x})} \sum_{n=1}^N \log p_{\theta}(\mathbf{x}_n \mid \mathbf{x}_{1:n-1})$$

$$L_{\text{DM}} = \mathbb{E}_{q(\mathbf{x}_0)} \sum_{n=1}^N \underbrace{\sum_{t=1}^T w(t) \mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0)} u(\mathbf{x}_0, \mathbf{x}_t, n; \boldsymbol{\theta})}_{-\log p_{\text{DM}}(\mathbf{x}_n \mid \mathbf{x}_{\neq n})}$$

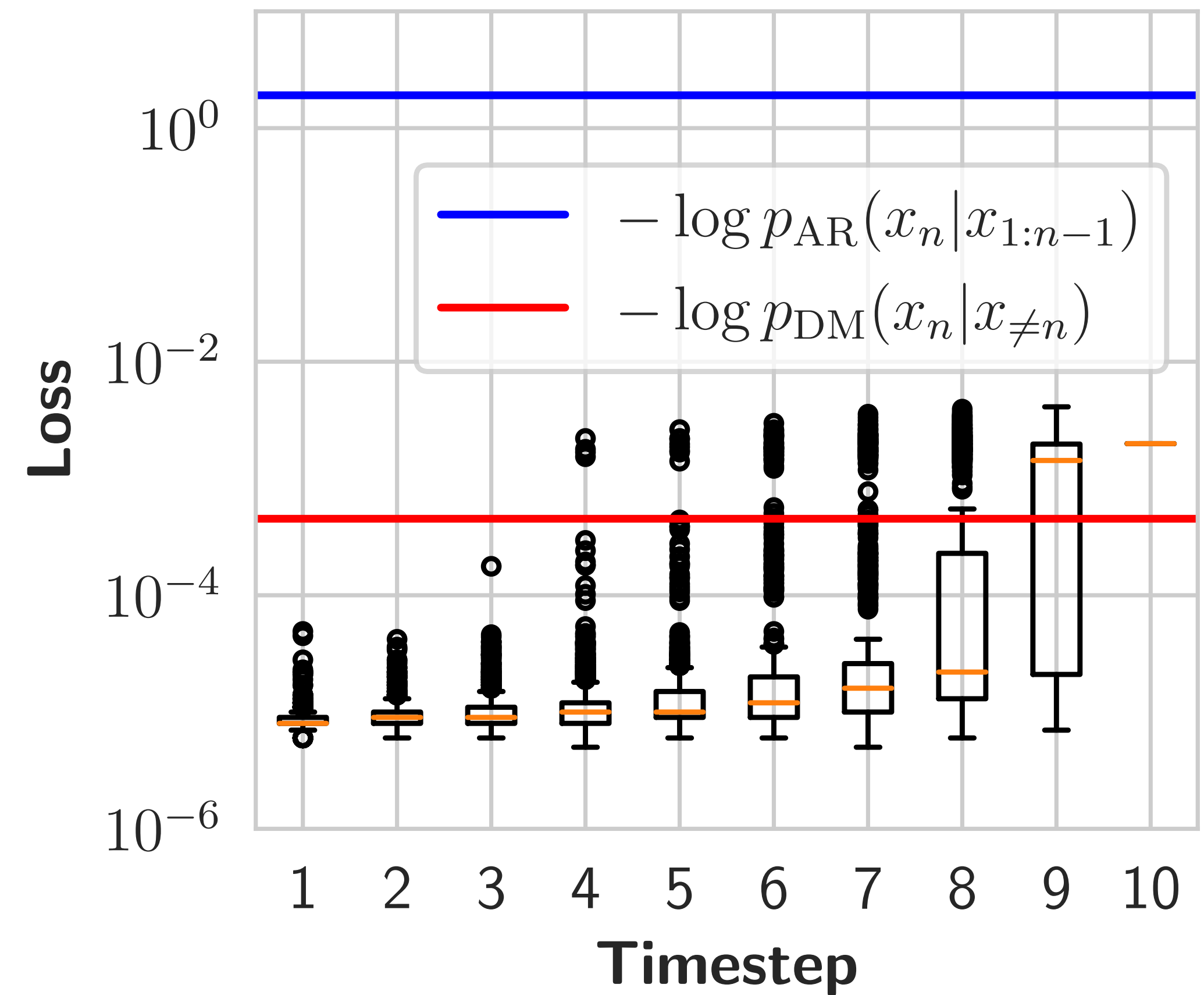


Loss for a specific hard subgoal, i.e., PD=3, in Diffusion and AR modeling.

Effective Hard Subgoal Learning

unweighted loss

$$u(\mathbf{x}_0, \mathbf{x}_t, n; \boldsymbol{\theta})$$



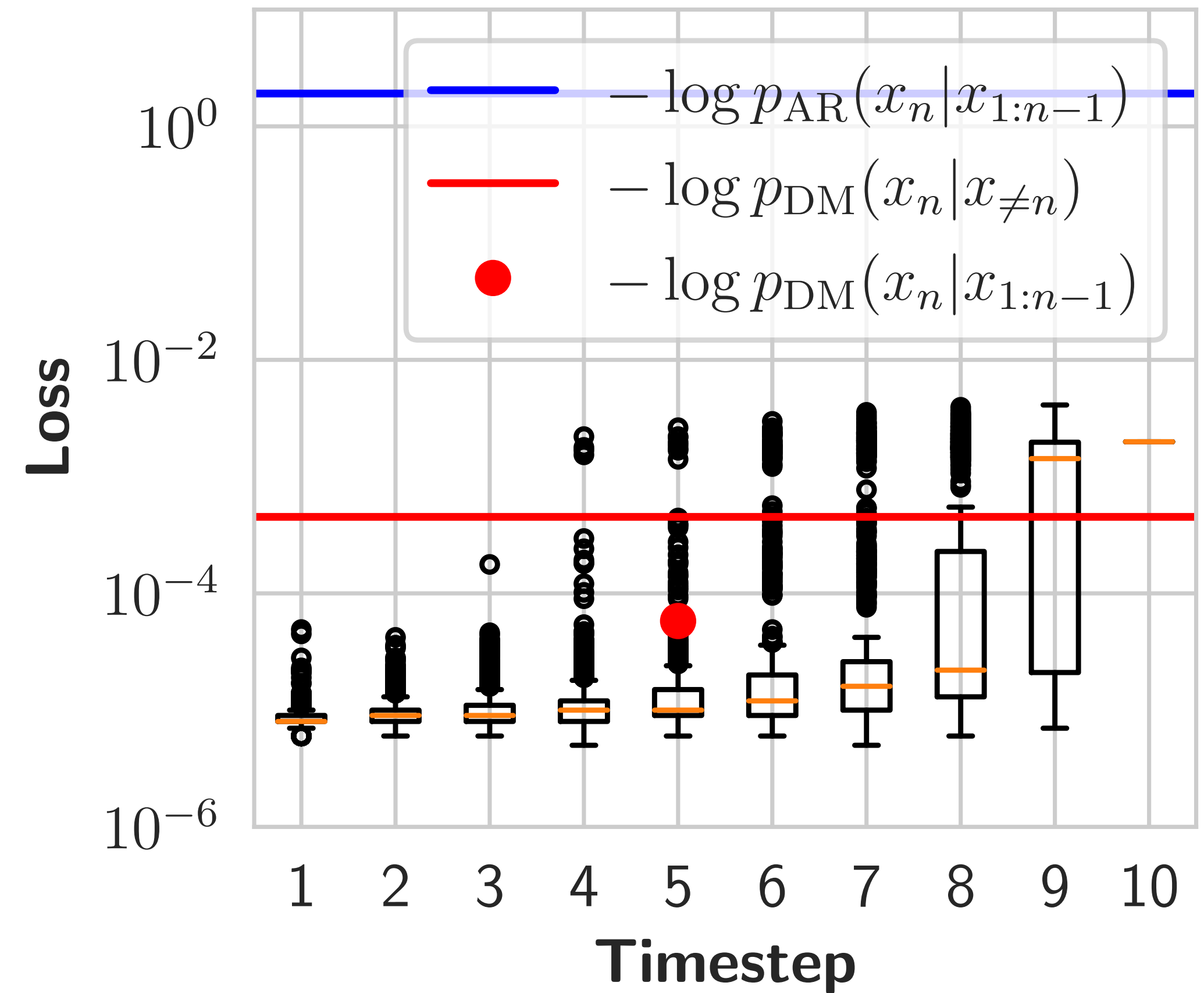
Effective Hard Subgoal Learning

unweighted loss

$$u(\mathbf{x}_0, \mathbf{x}_t, n; \boldsymbol{\theta})$$

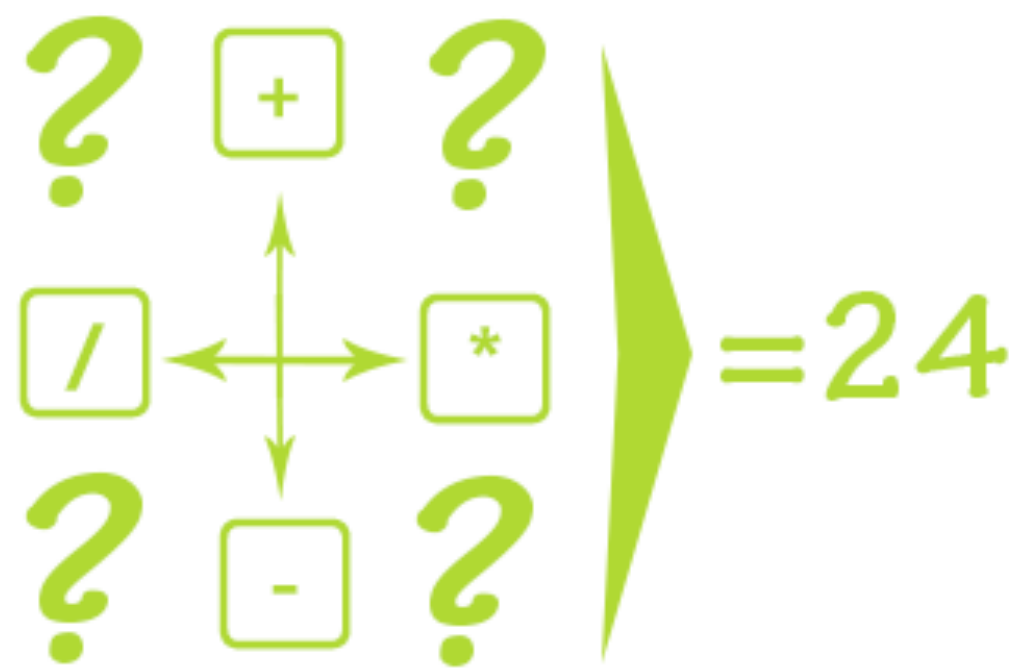
a diverse range of interrelated objectives
(Multi-view learning)

👍 more effective, promising, and exhibit
better generalization



Discrete Diffusion Models for Countdown

$$L_{\text{MDM}} = \sum_{n=1}^N \sum_{t=1}^T w(t) v(\mathbf{x}_{t,n}) u(\mathbf{x}_0, \mathbf{x}_t, n; \boldsymbol{\theta}) \quad v(\mathbf{x}_{t,n}) = \alpha(1 - \exp(-u(\cdot)))^\beta$$

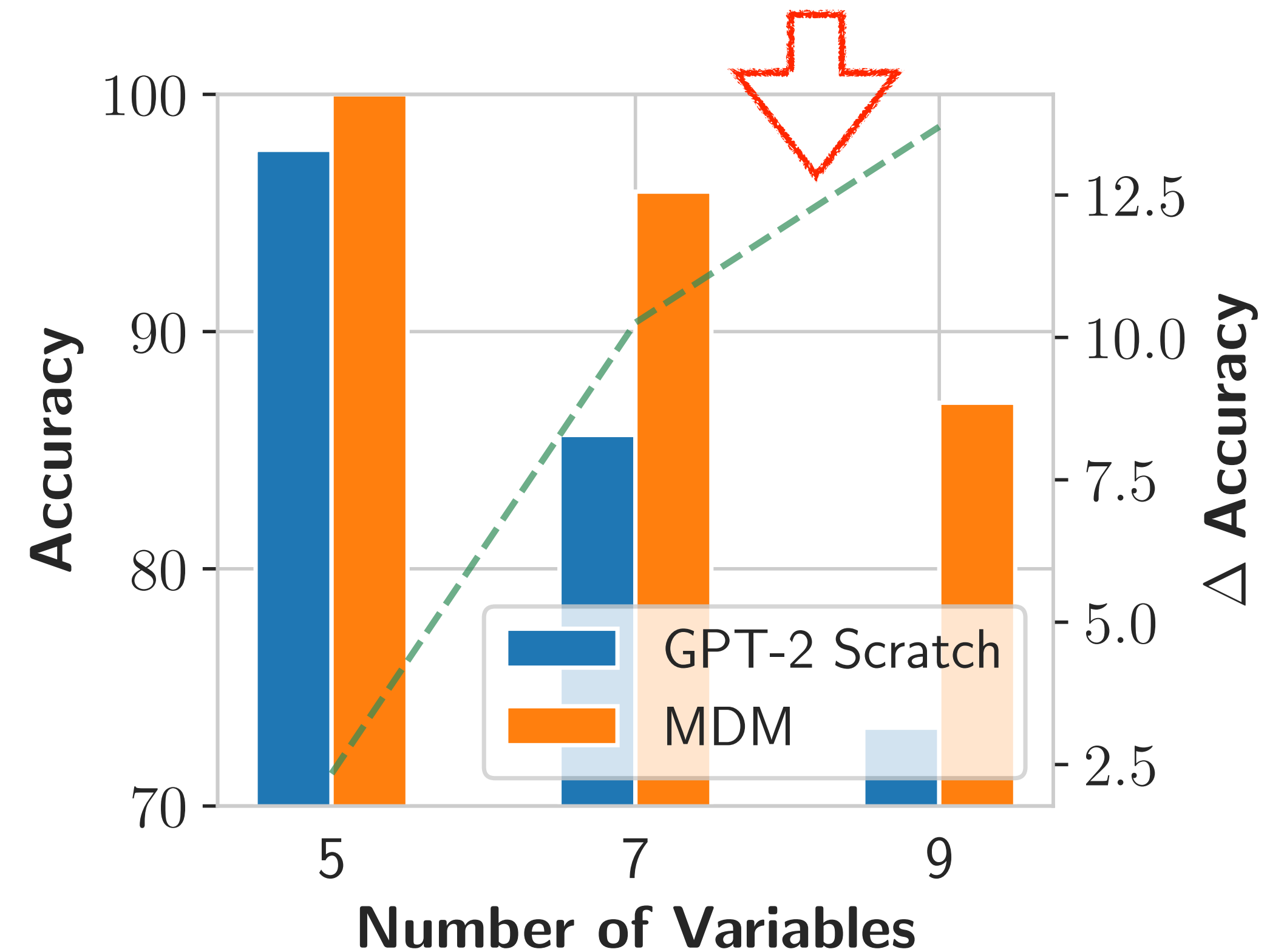
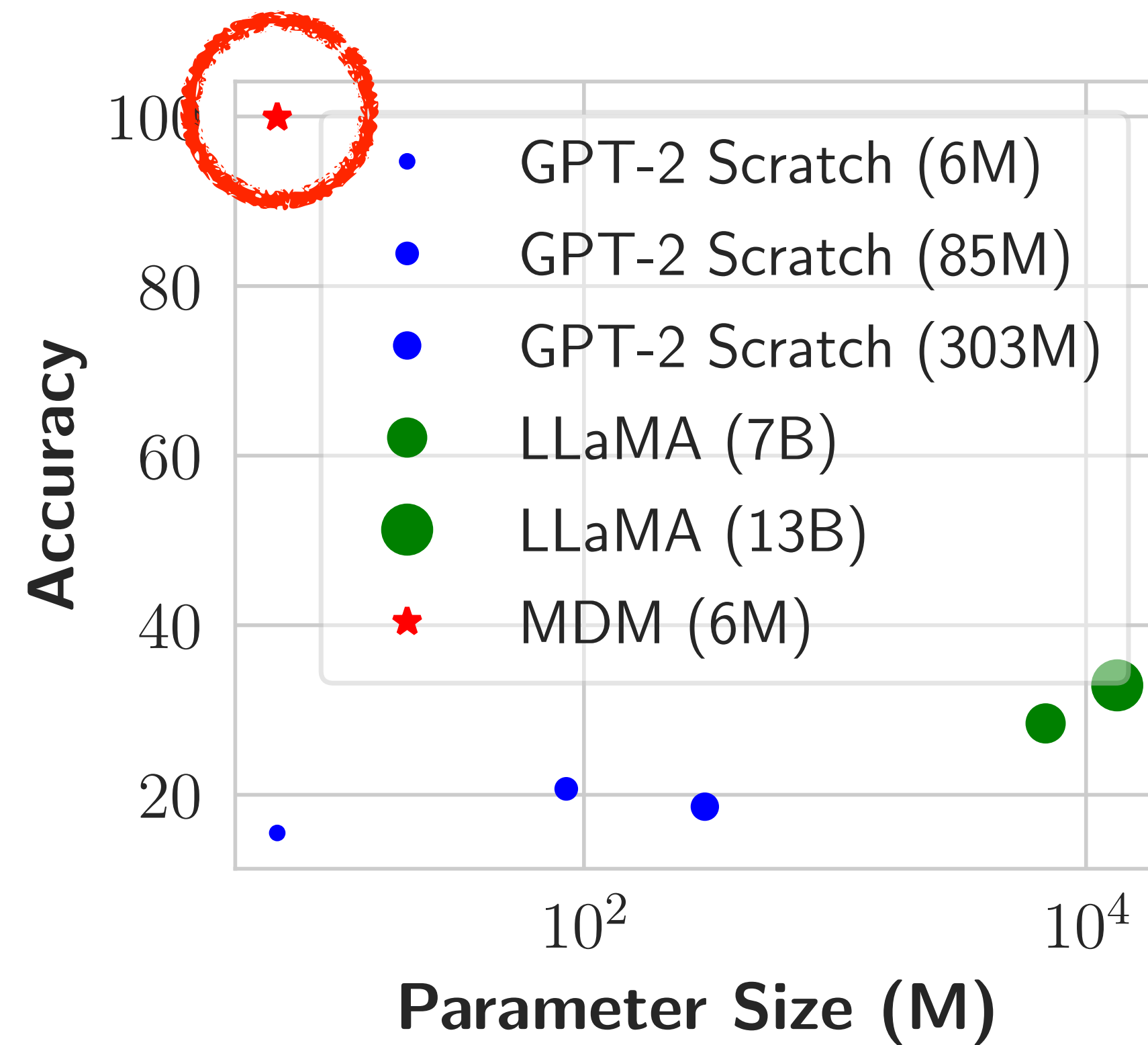


	Params	CD 3	CD 4	CD 5
GPT-2 Scratch	6M	94.1	31.9	4.3
	85M	95.9	45.8	5.1
	303M	96.4	41.3	4.5
LLaMA	7B	95.7	41.1	6.7
	13B	96.5	51.1	7.4
Diffusion Reasoning Model	85M	99.5	87.0	45.8

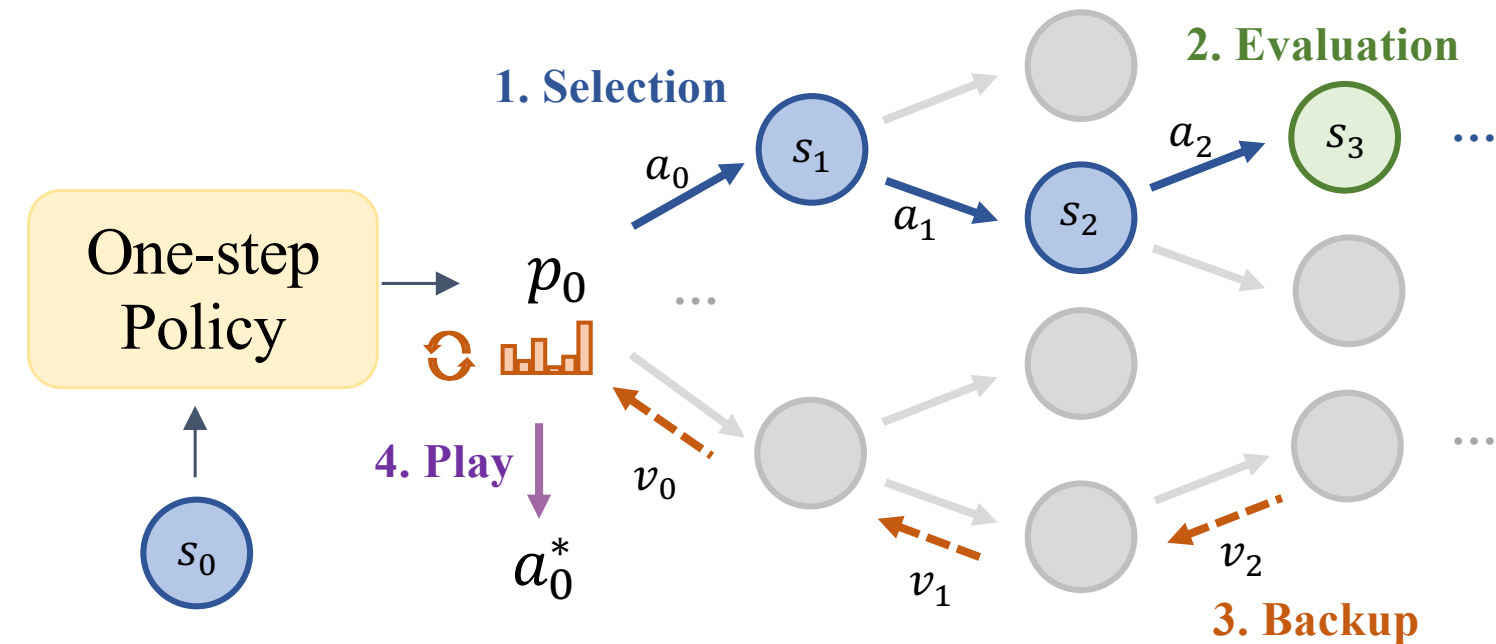
Discrete Diffusion Models for Sudoku

$$L_{\text{MDM}} = \sum_{n=1}^N \sum_{t=1}^T w(t) v(\mathbf{x}_{t,n}) u(\mathbf{x}_0, \mathbf{x}_t, n; \boldsymbol{\theta}) \quad v(\mathbf{x}_{t,n}) = \alpha(1 - \exp(-u(\cdot)))^\beta$$

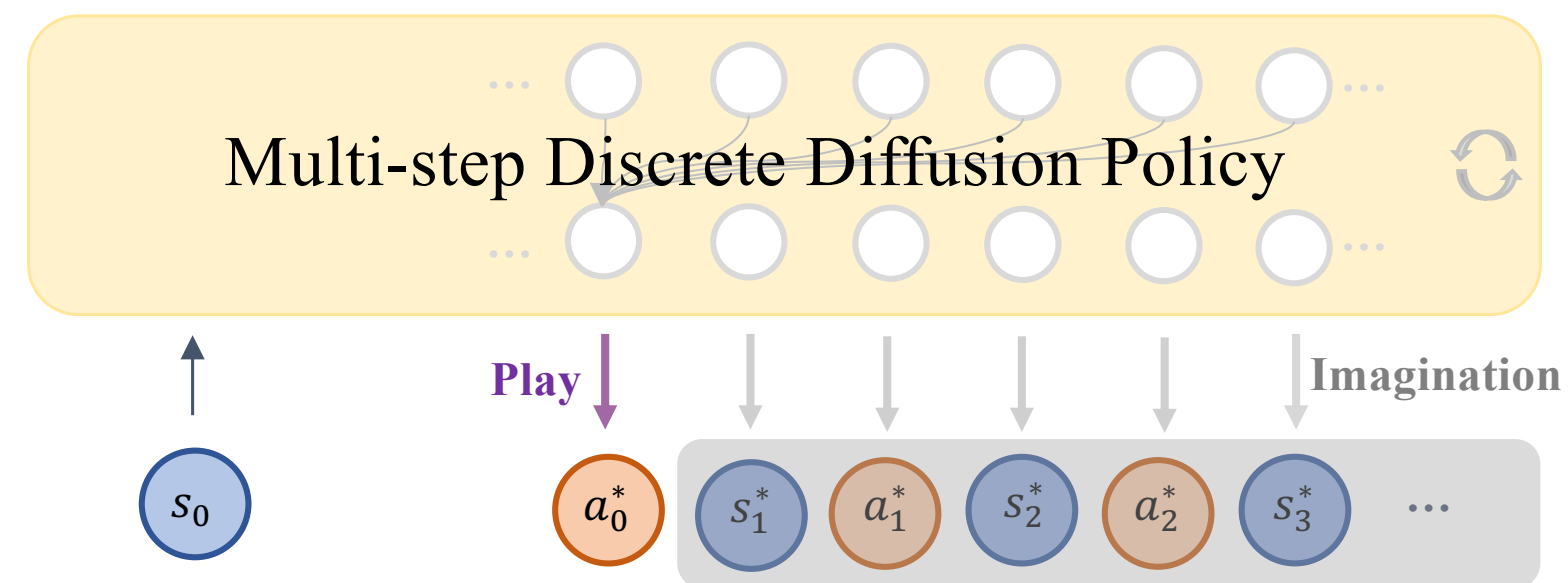
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



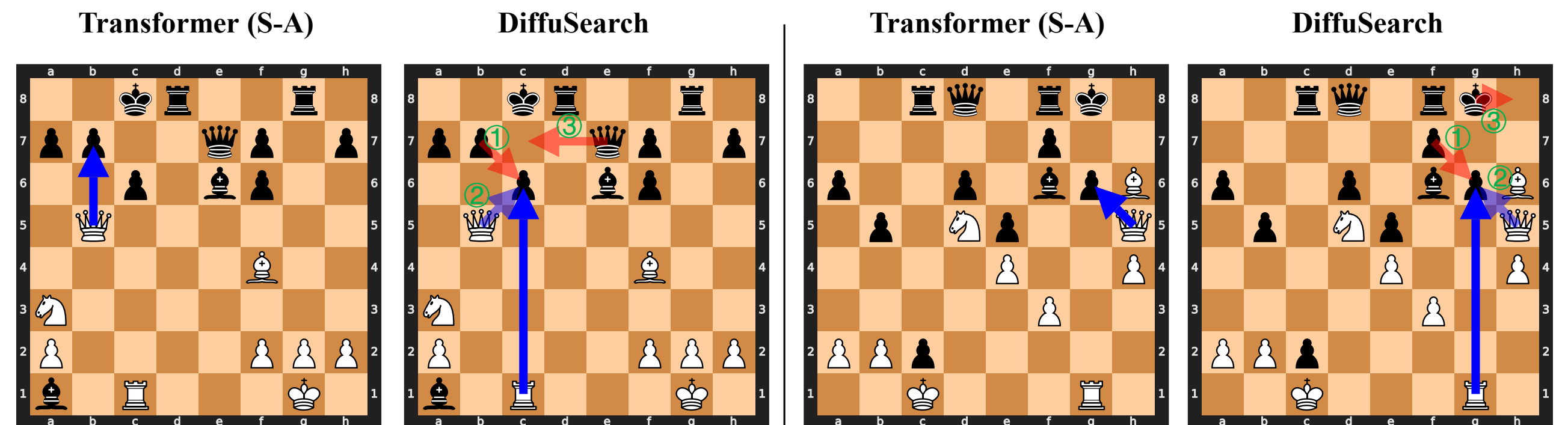
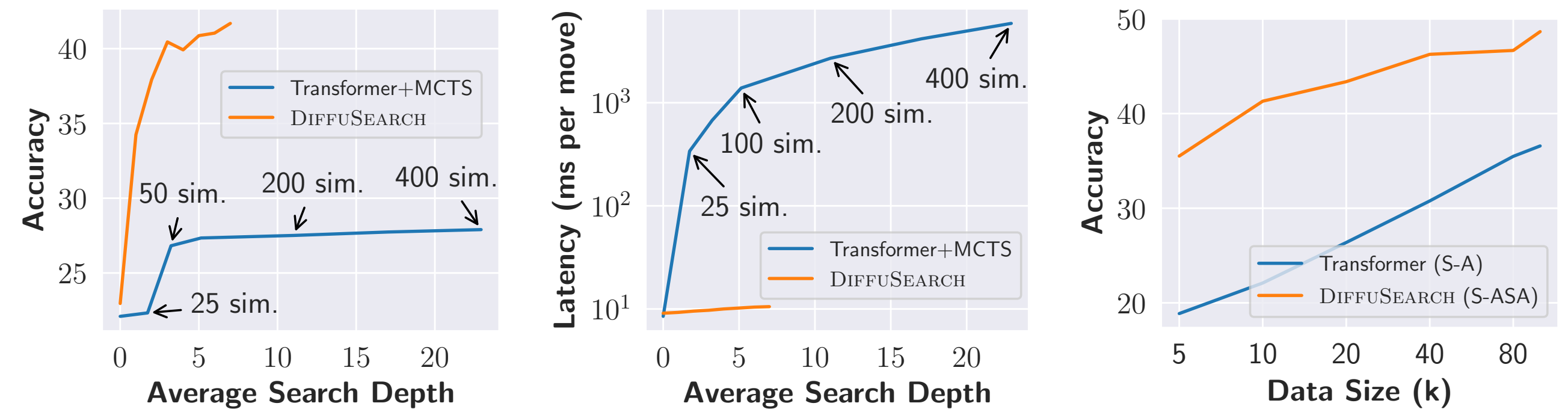
Implicit Search via Discrete Diffusion Models



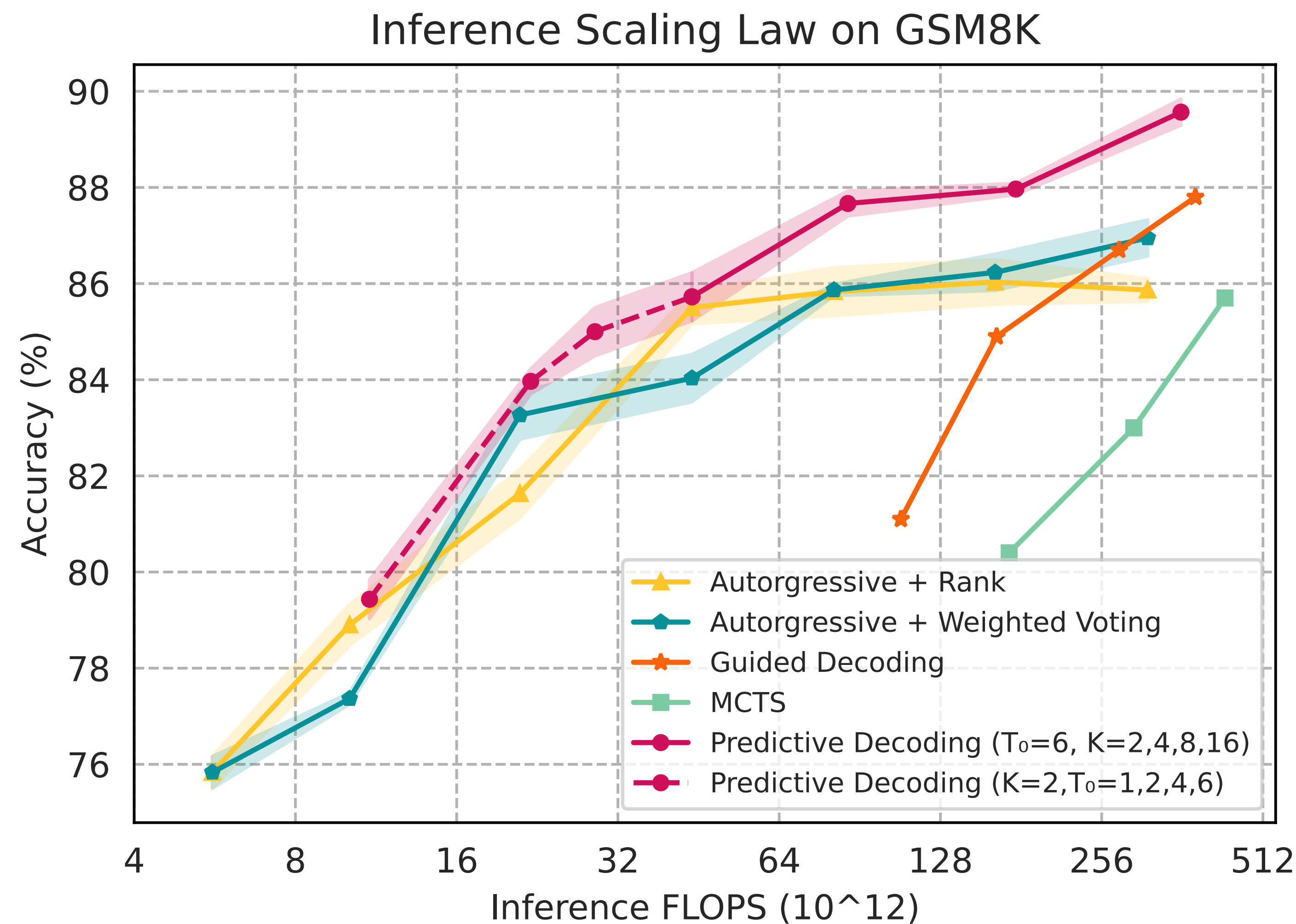
(a) Explicit Search via MCTS



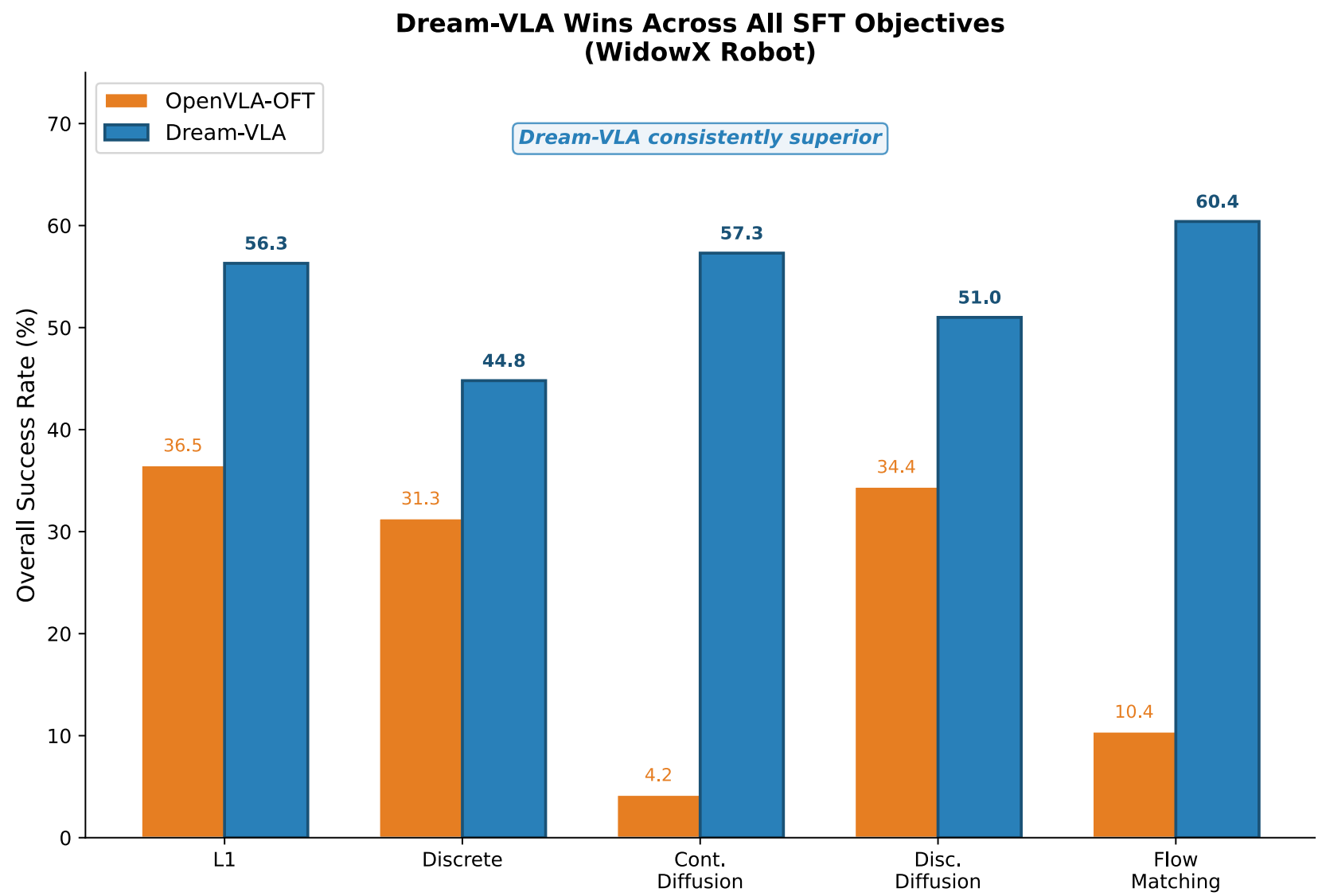
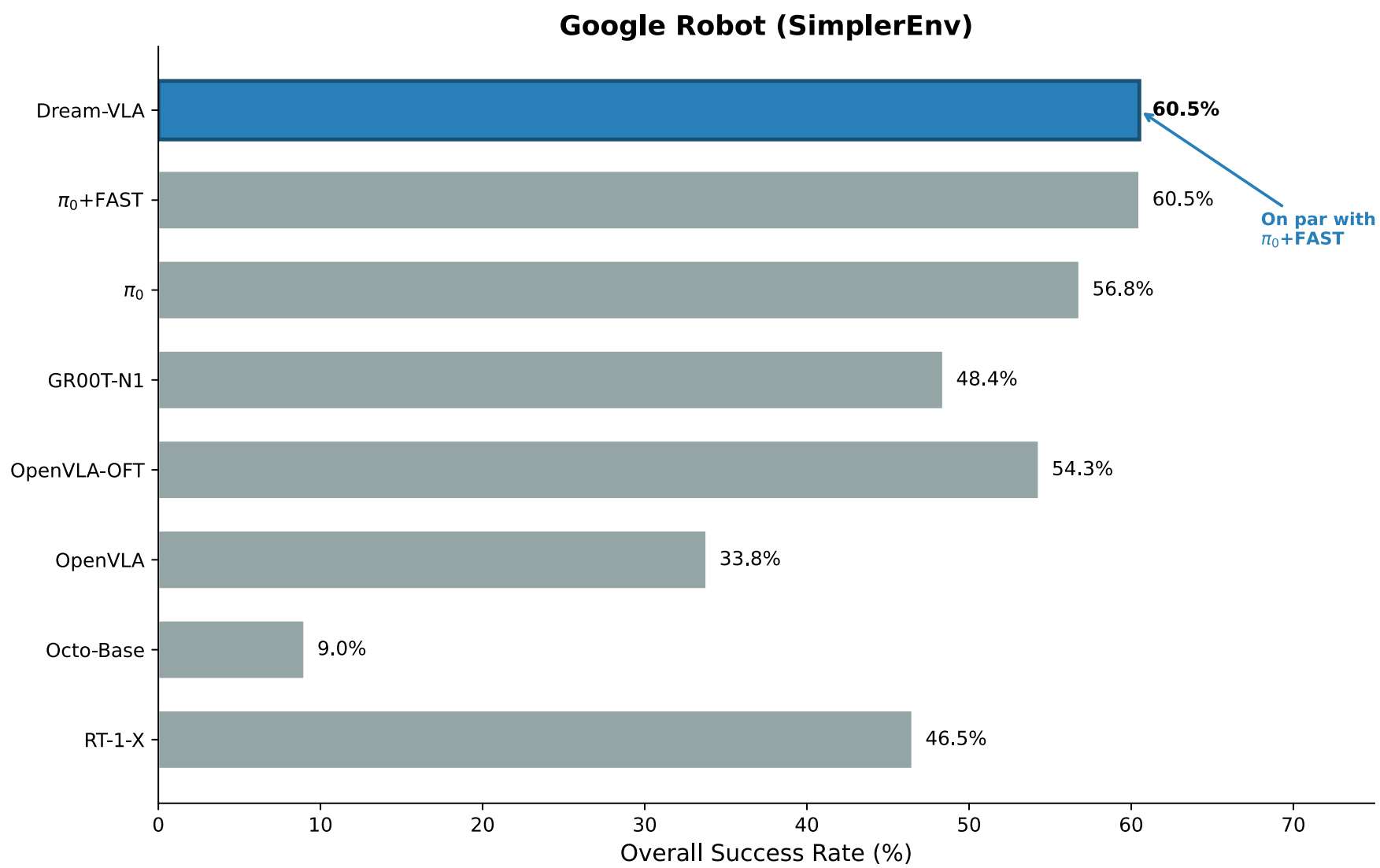
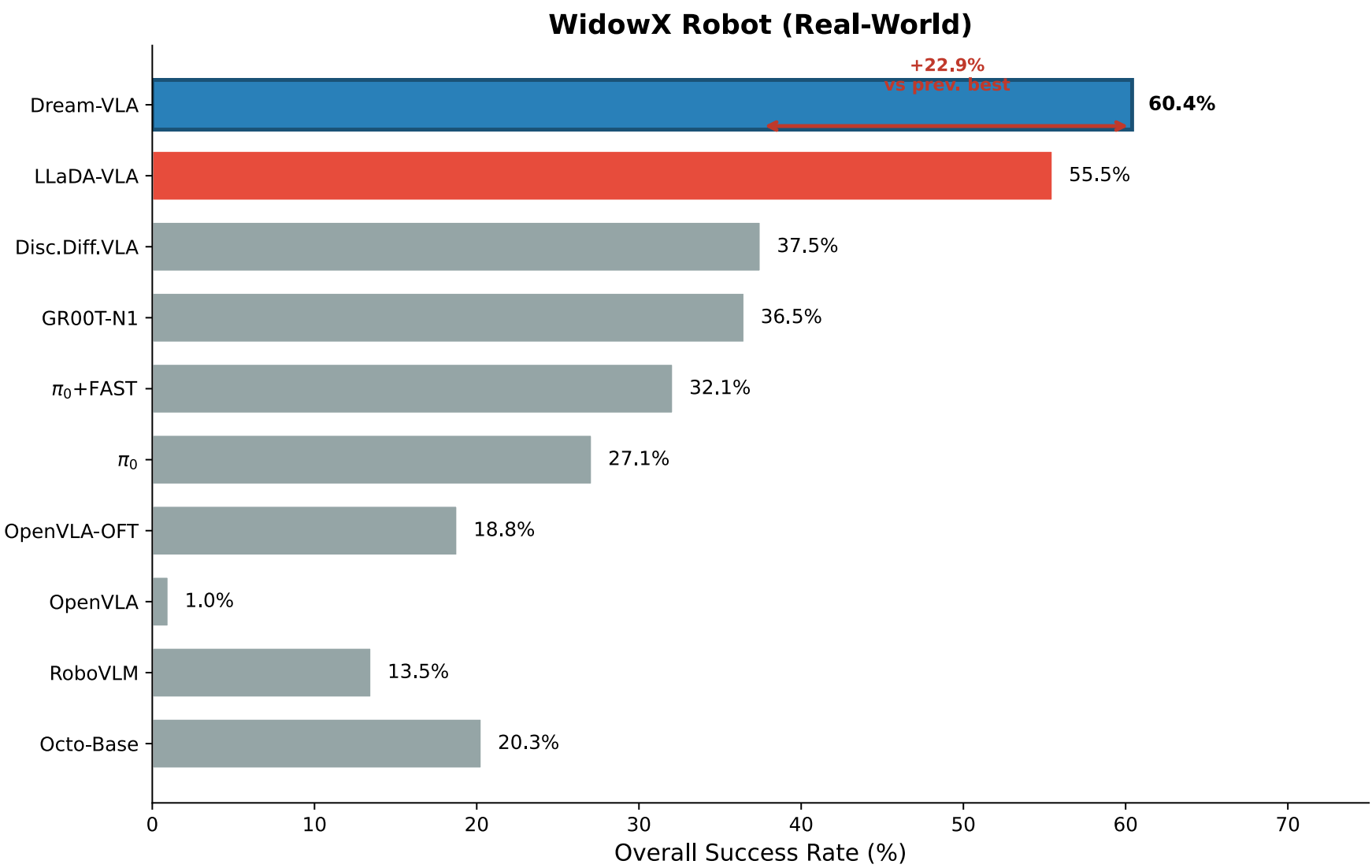
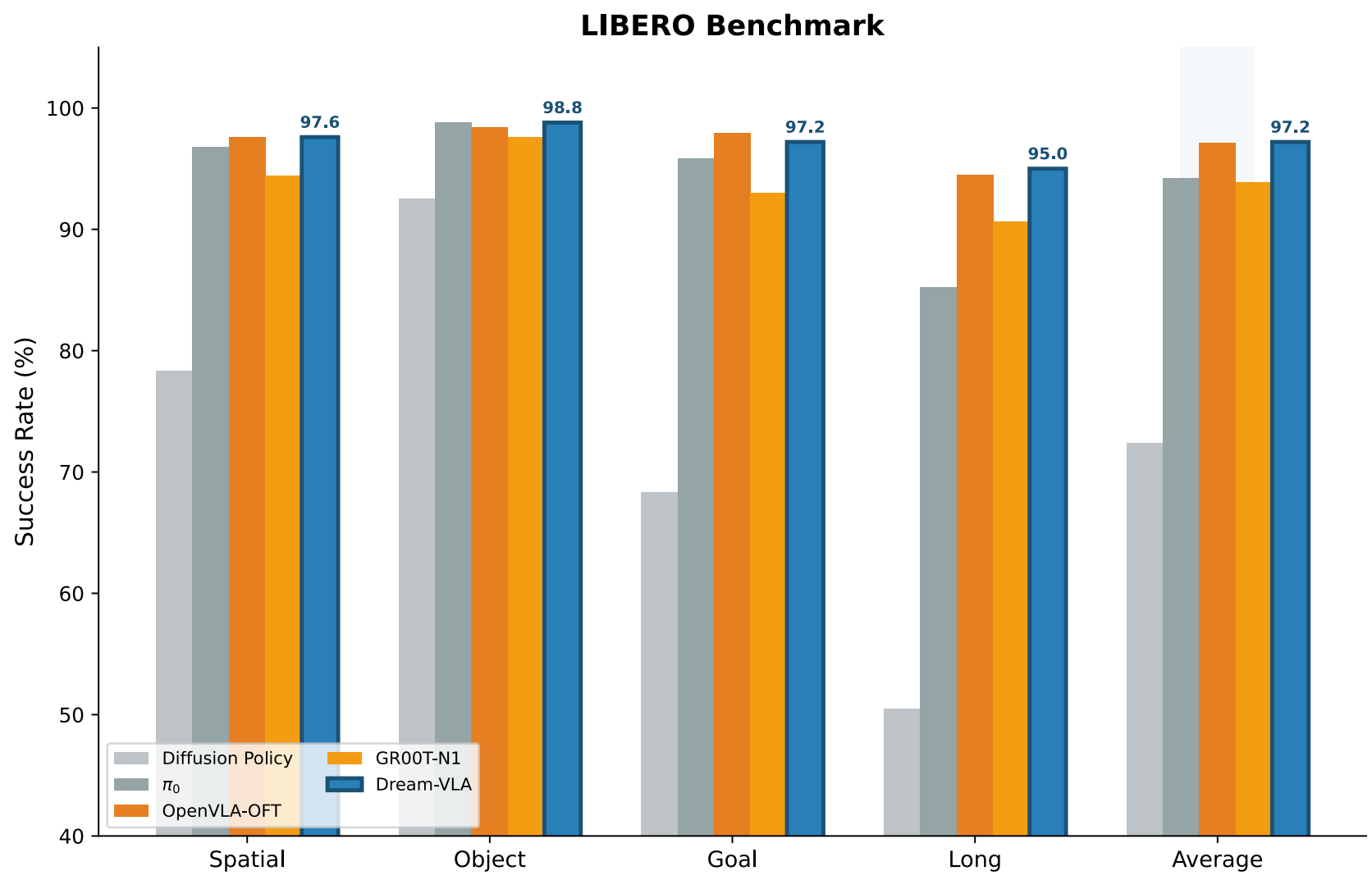
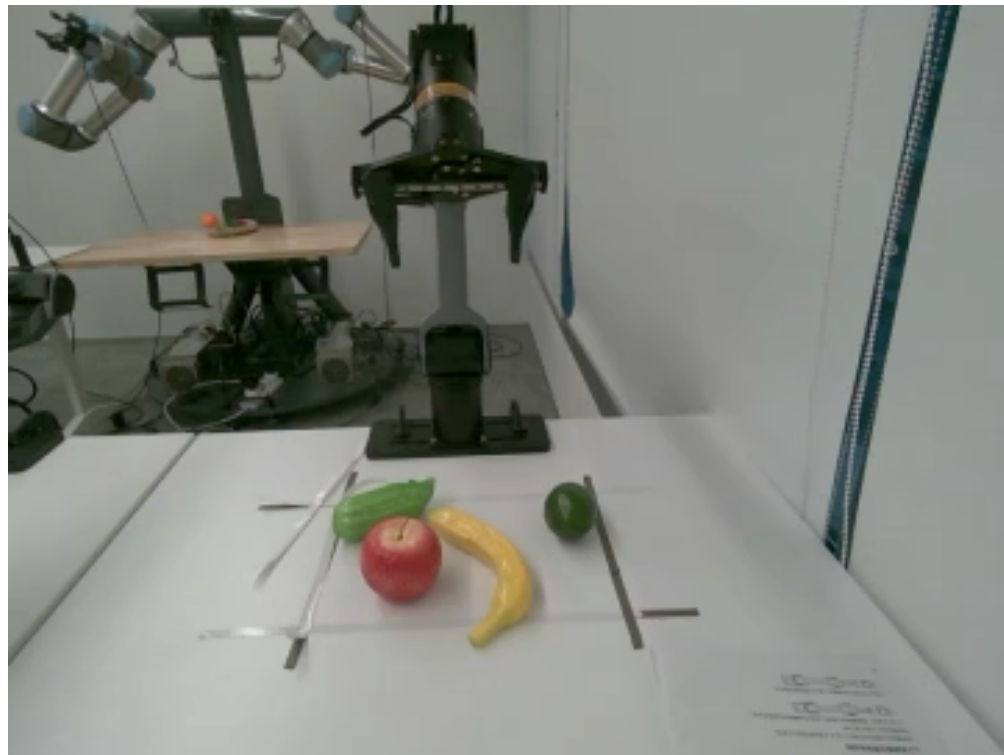
(b) Implicit Search via Discrete Diffusion



Autoregressive LLMs

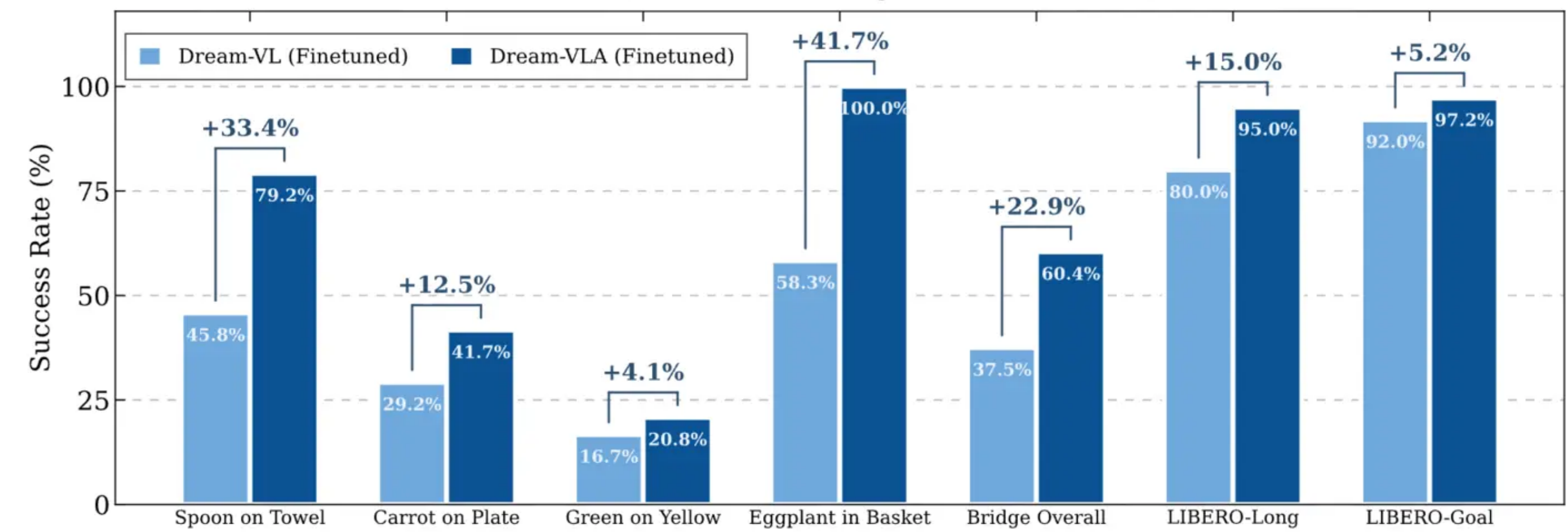


DreamVLA – Vision-Language-Action Model

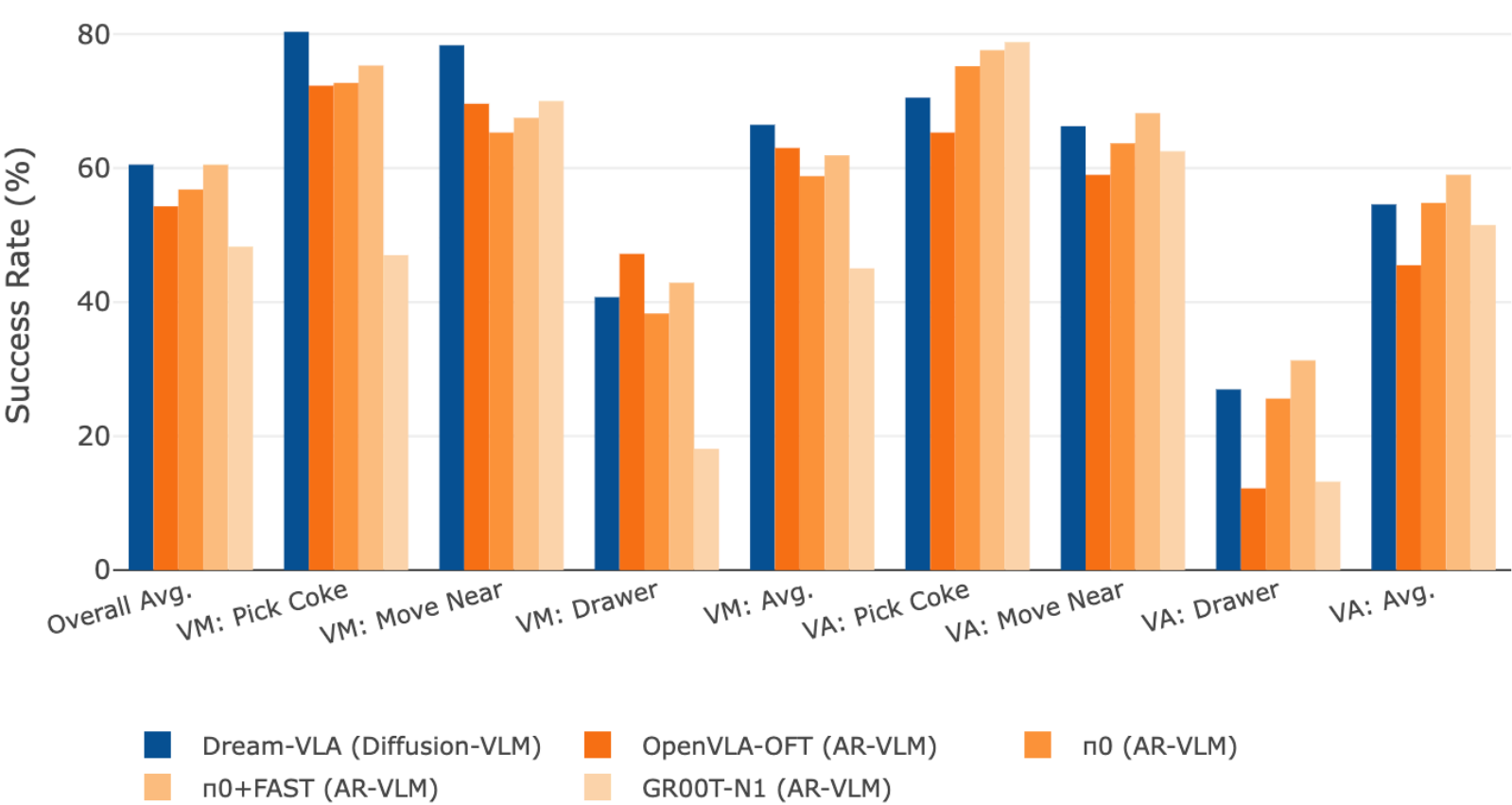


DreamVLA

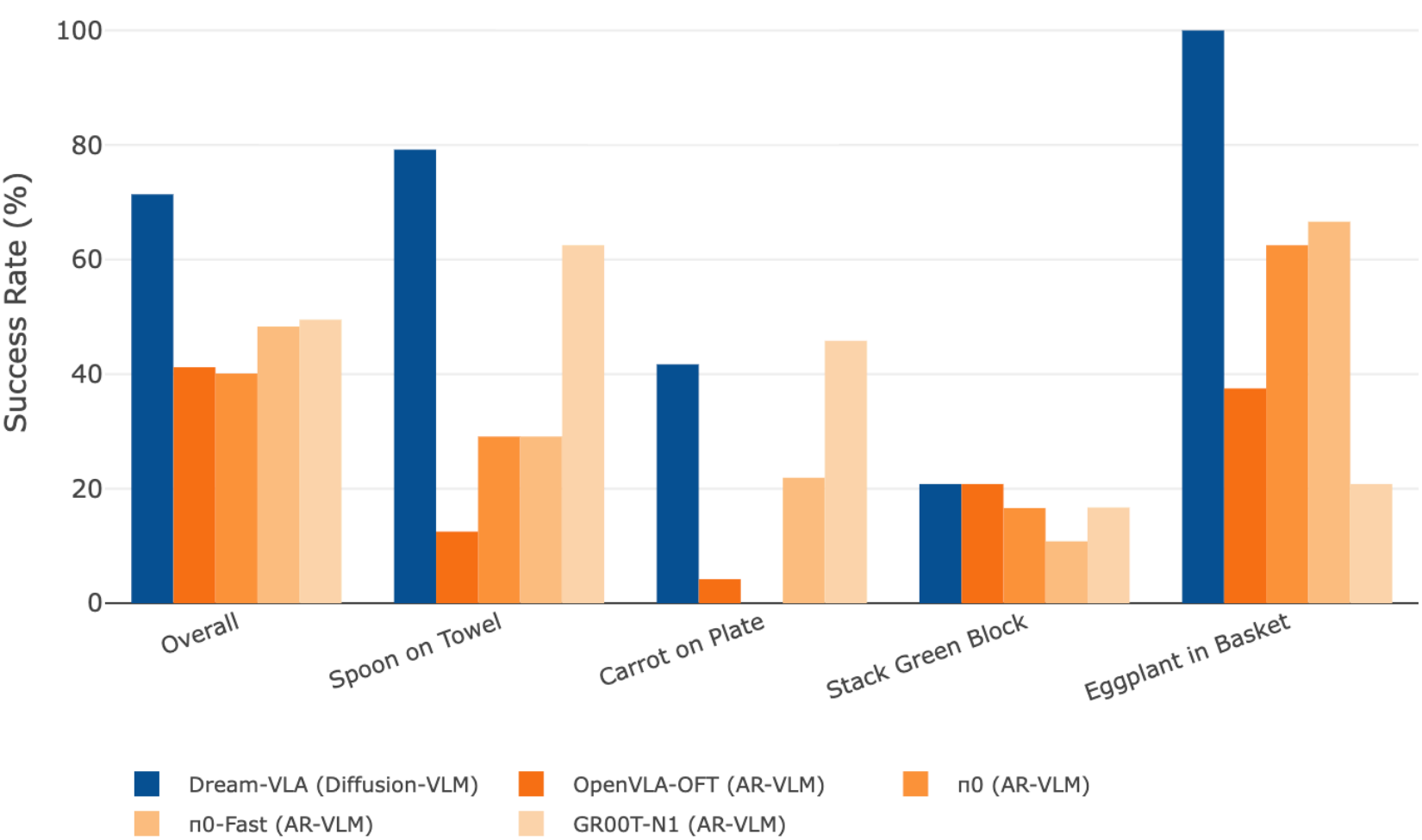
Effect of Robotic Pretraining on Success Rate (%)



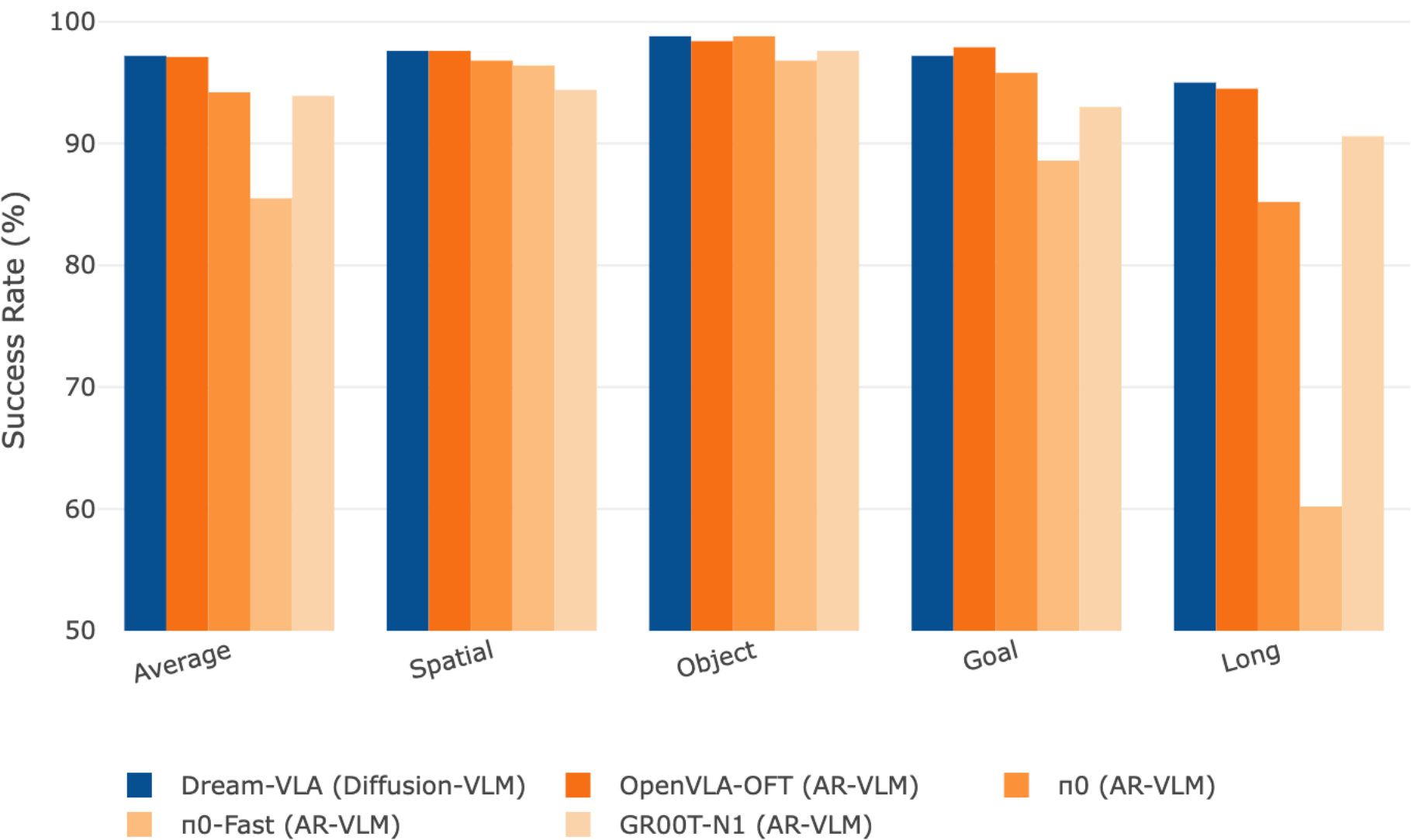
SimplerEnv Google Robot



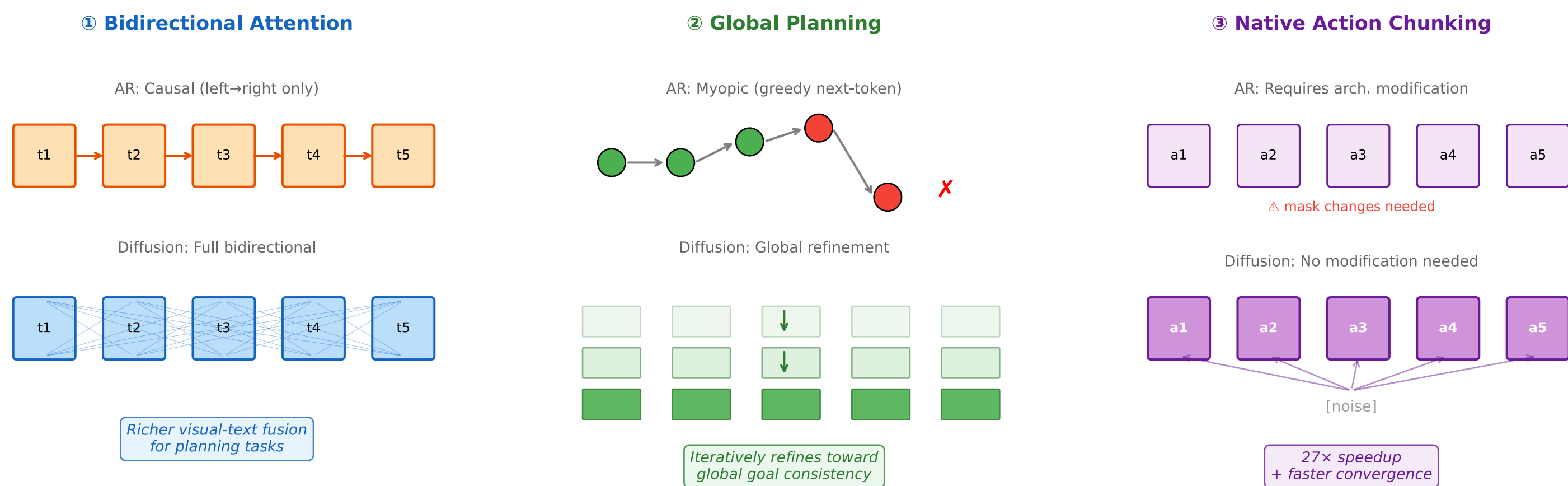
SimplerEnv WidowX Robot



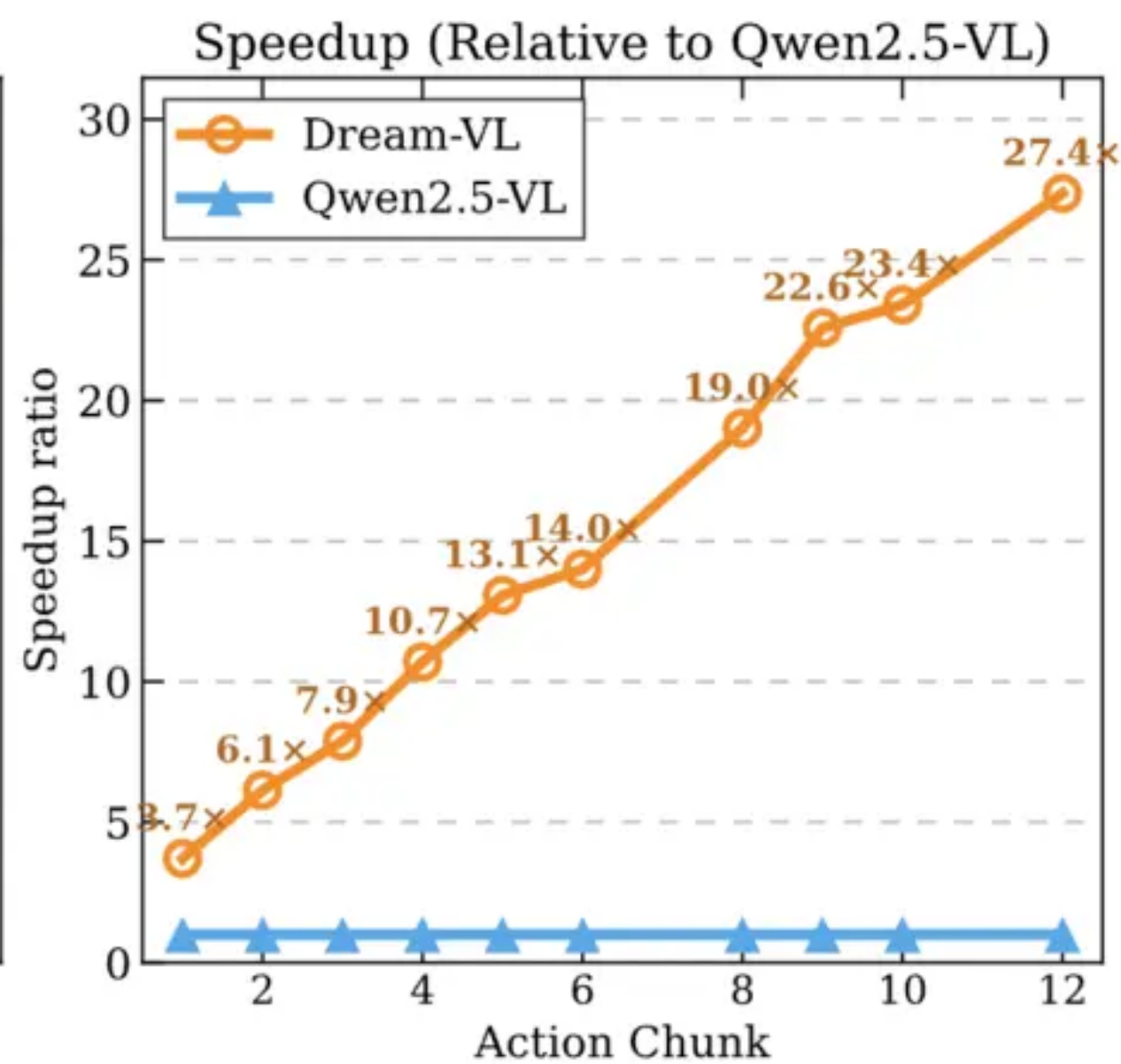
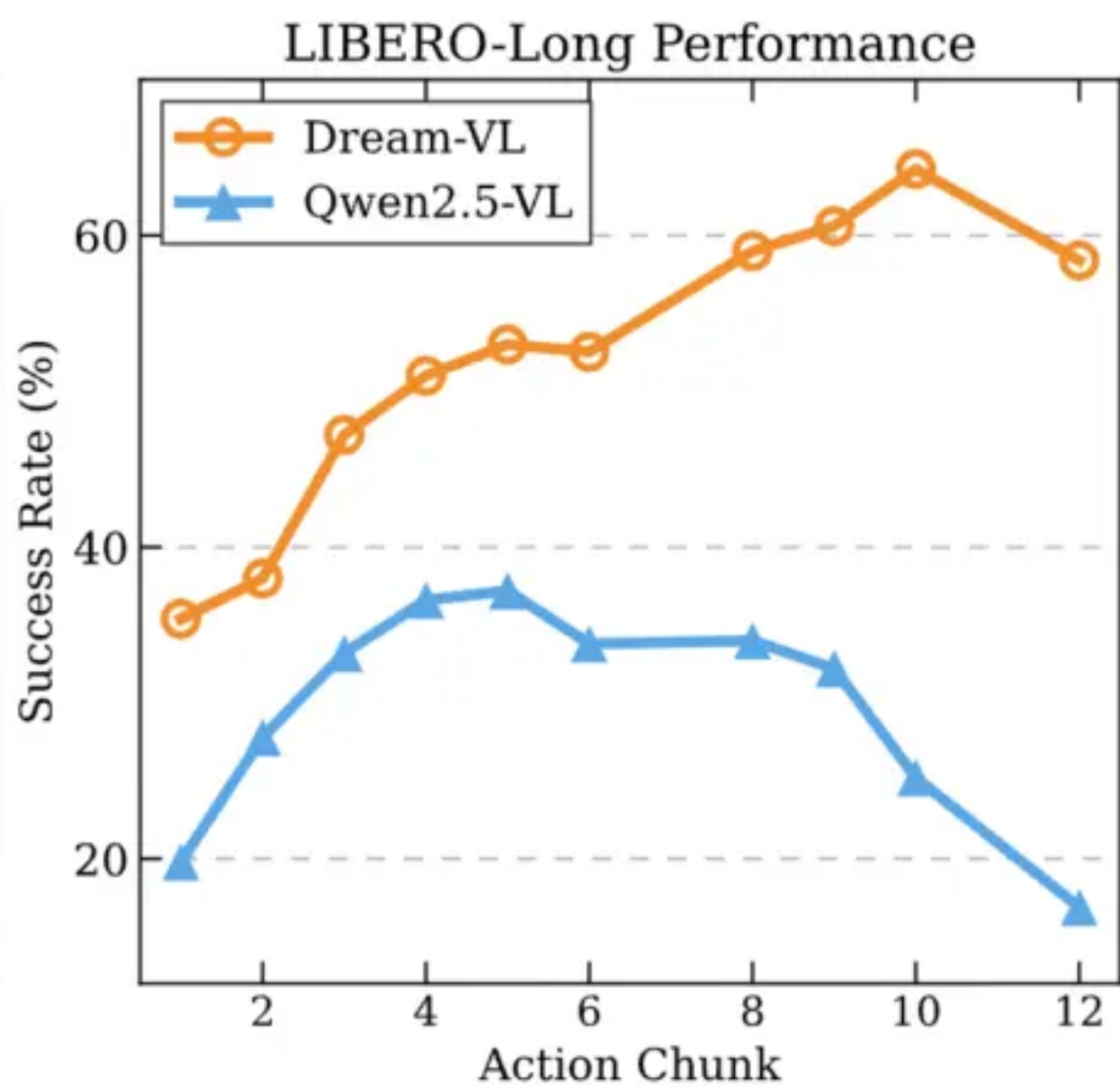
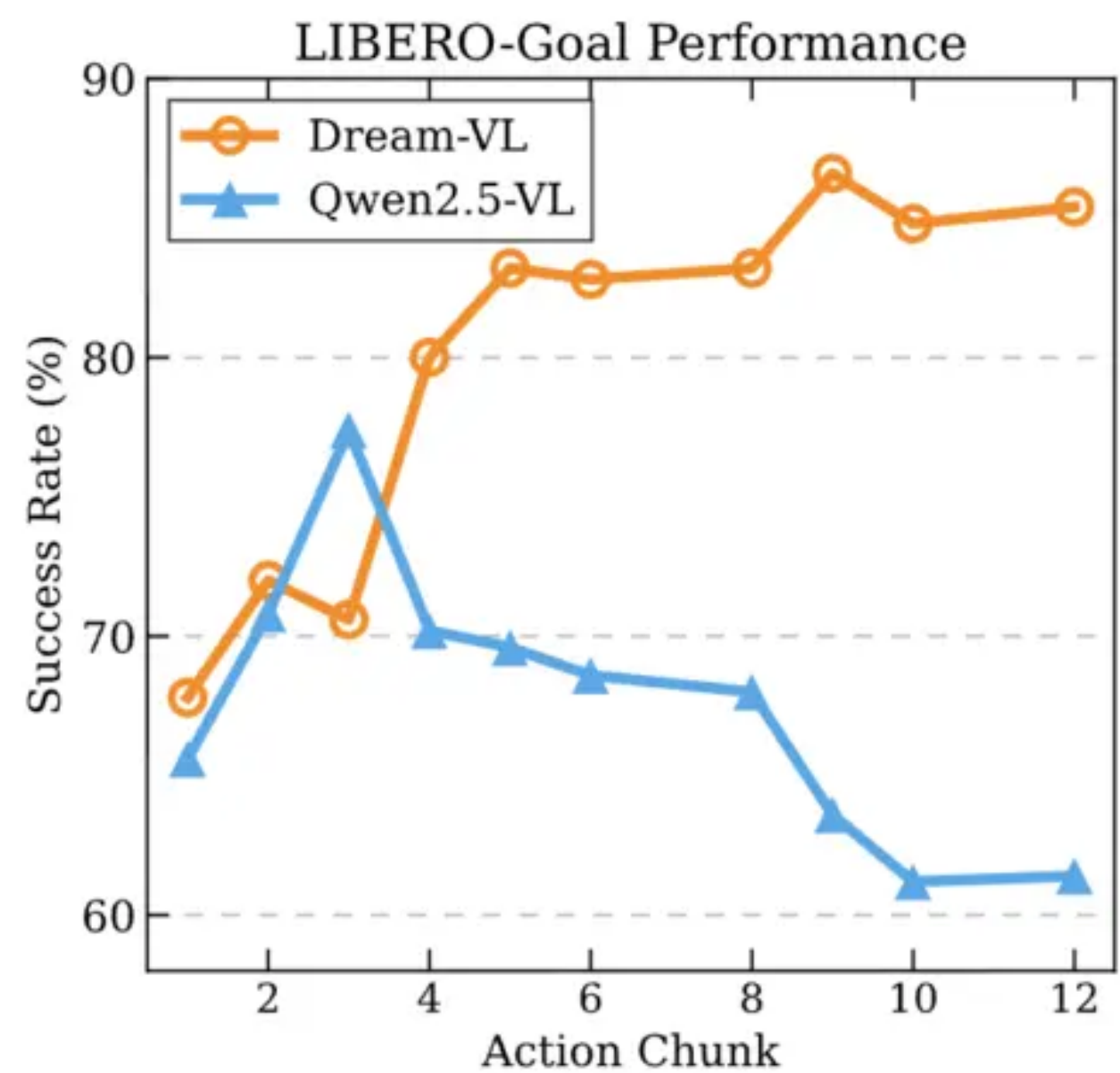
LIBERO Franka Robot



DreamVLA



With Bidirectional Attention, Global planning, and Native Action Chunking, as chunk size grows, AR collapses. Dream-VLA improves — 97.2% LIBERO, 71.4% WidowX.

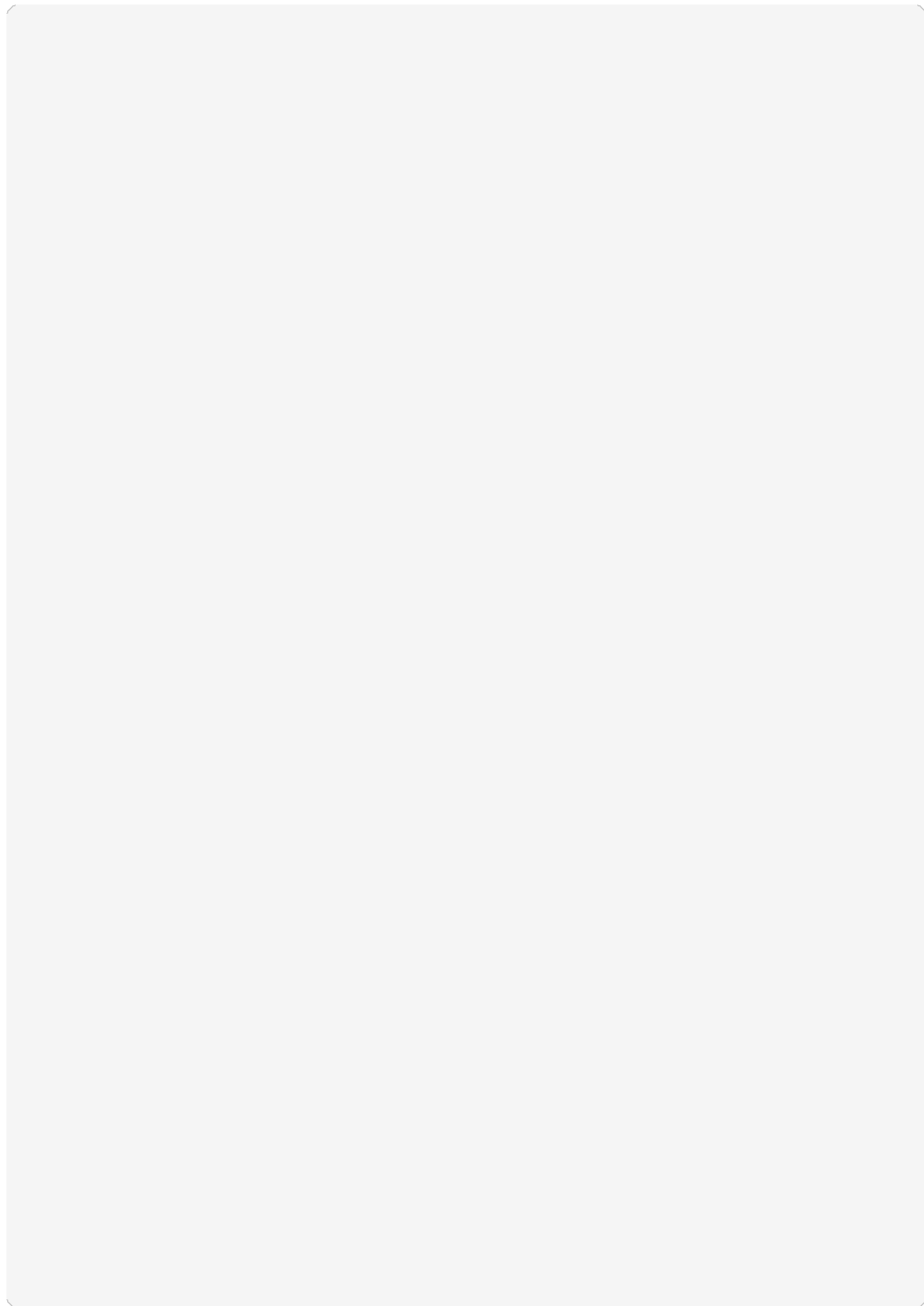


Dream-Coder 7B

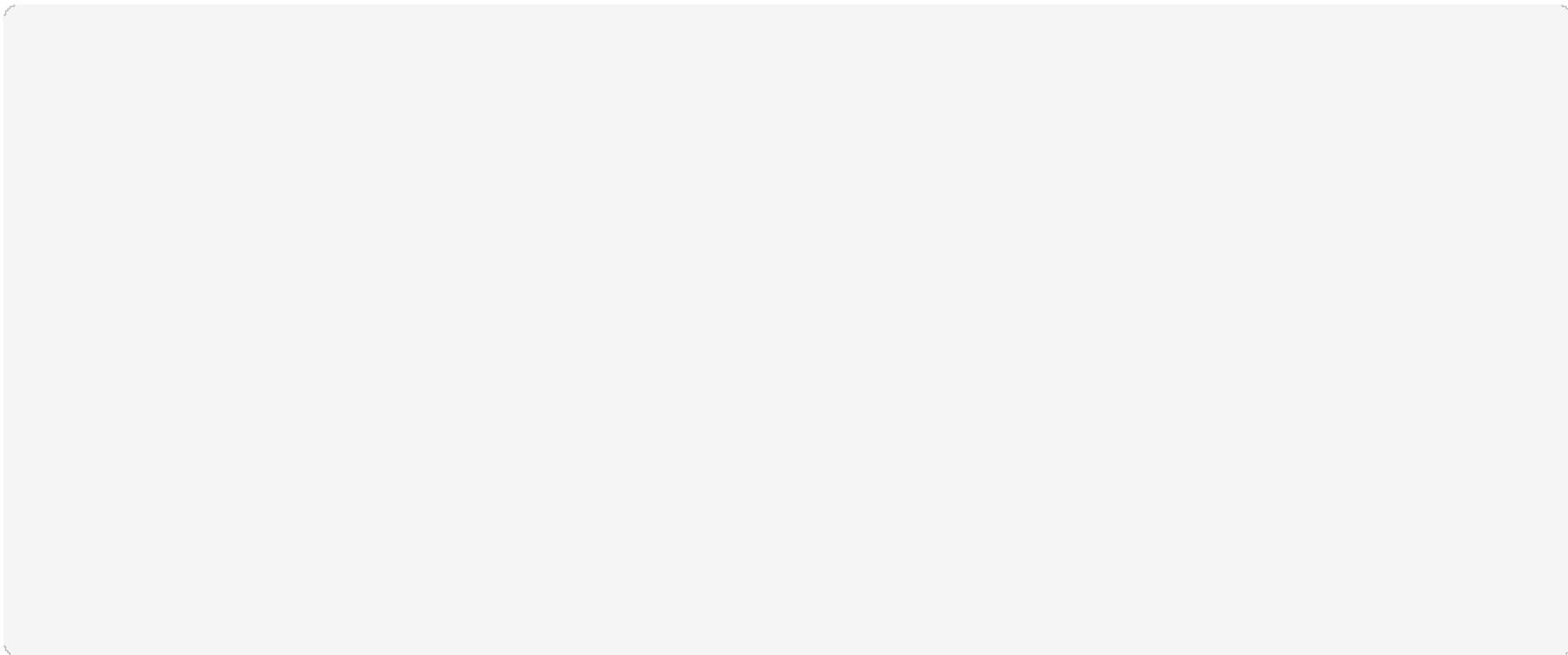
Dream-Coder 7B, the first fully open-source **diffusion LLM for code** that provides complete transparency throughout its development pipeline, with all components publicly available -- data processing scripts, implementation code, and model weights.

	Type	Data	HumanEval	MBPP	EvalPlus	BigCodeBench full	hard	LiveCodeBench 2410-2505	CRUXEval Input-CoT Output-CoT		Average
Proprietary Models											
Claude 3 Haiku	AR	✗	76.8	80.2	68.9	39.4	16.9	21.2	-	-	-
Mercury Coder Small	Diffusion	✗	90.0	80.4	76.4	45.3	22.3	22.9	66.4	70.8	59.3
Gemini Diffusion	Diffusion	✗	89.6	76.0	-	45.4	-	-	-	-	-
Open-Weight AR Models											
OpenCoder-8B-Instruct	AR	✓	81.7	82.0	74.4	40.3	16.9	10.6	41.6	41.9	48.7
Qwen2.5-Coder-7B-Instruct	AR	✗	88.4	83.5	77.9	41.0	18.2	19.3	65.8	65.9	57.5
Seed-Coder-8B-Instruct	AR	✗	84.8	85.2	75.0	44.5	26.4	20.9	63.3	67.1	58.4
Open-Weight Diffusion Models											
LLaDA-1.5-8B	Diffusion	✗	52.4	42.8	-	-	-	-	-	-	-
Dream-7B-Instruct	Diffusion	✓	63.4	68.3	57.4	23.0	2.7	9.7	53.1	42.8	40.0
DiffuCoder-7B-cpGRPO	Diffusion	✓	73.2	78.6	67.9	37.5	10.8	10.3	-	-	-
Dream-Coder-7B-Instruct	Diffusion	✓	82.9	79.6	73.1	37.1	17.6	21.4	64.6	63.1	54.9

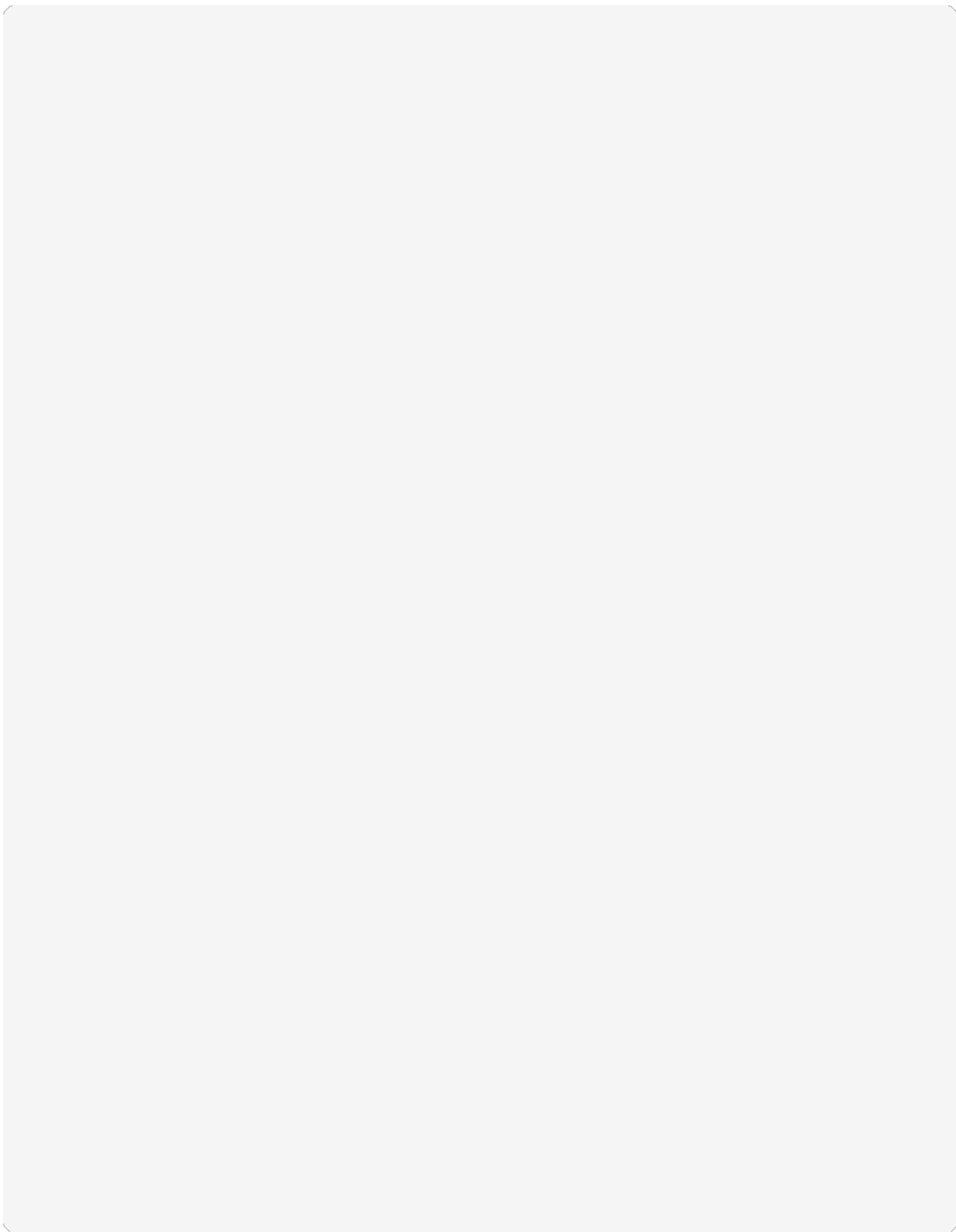
Dream-Coder 7B



Sketch-First Generation (from LiveCodeBench)

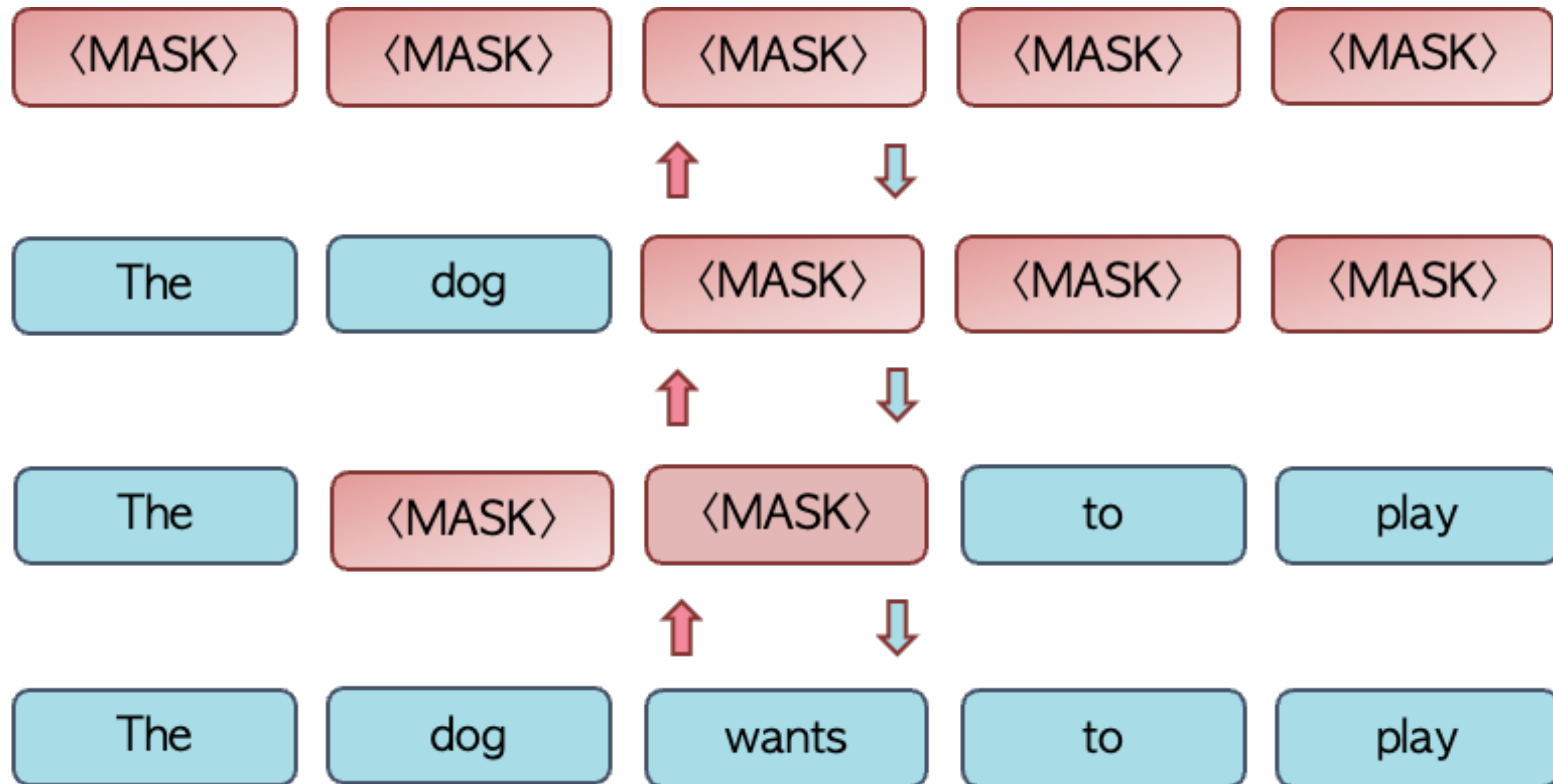


Left-to-Right Generation (from BigCodeBench)



Interleaved Reasoning Generation (from CRUXEval)

Discrete Diffusion: Fixed-Size Canvas



Discrete Diffusion: Fixed-Size Canvas

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers
    are any two numbers closer to each other than given threshold.
    """
    for idx, elem in enumerate(numbers):
        <|mask|><|mask|><|mask|><|mask|>
        if idx != idx2:
            distance = abs(elem - elem2)
            if distance < threshold:
                return True

    return False
```

Fixed-Length

With too few masked tokens, diffusion models lack sufficient room for meaningful code infilling.

```
def correct_bracketing(brackets: str):
    """ brackets is a string of "<" and ">".
    return True if every opening bracket has a corresponding closing bracket.
    """
    depth = 0
    for b in brackets:
        if b == "<":
            depth += 1
        <|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|>
        <|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|>
        if b == ">":
            depth -= 1
            if depth < 0:
                return False
    return depth == 0
```

Fixed-Length

Too many masked tokens cause overgeneration of unnecessary code snippet `depth > 0` that is incorrect.



Code infilling becomes meaningless given the oracle length.

Discrete Diffusion: Fixed-Size Canvas

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers
    are any two numbers closer to each other than given threshold.
    """
    for idx, elem in enumerate(numbers):
        <|mask|><|mask|><|mask|><|mask|>
        if idx != idx2:
            distance = abs(elem - elem2)
            if distance < threshold:
                return True

    return False
```

Fixed-Length

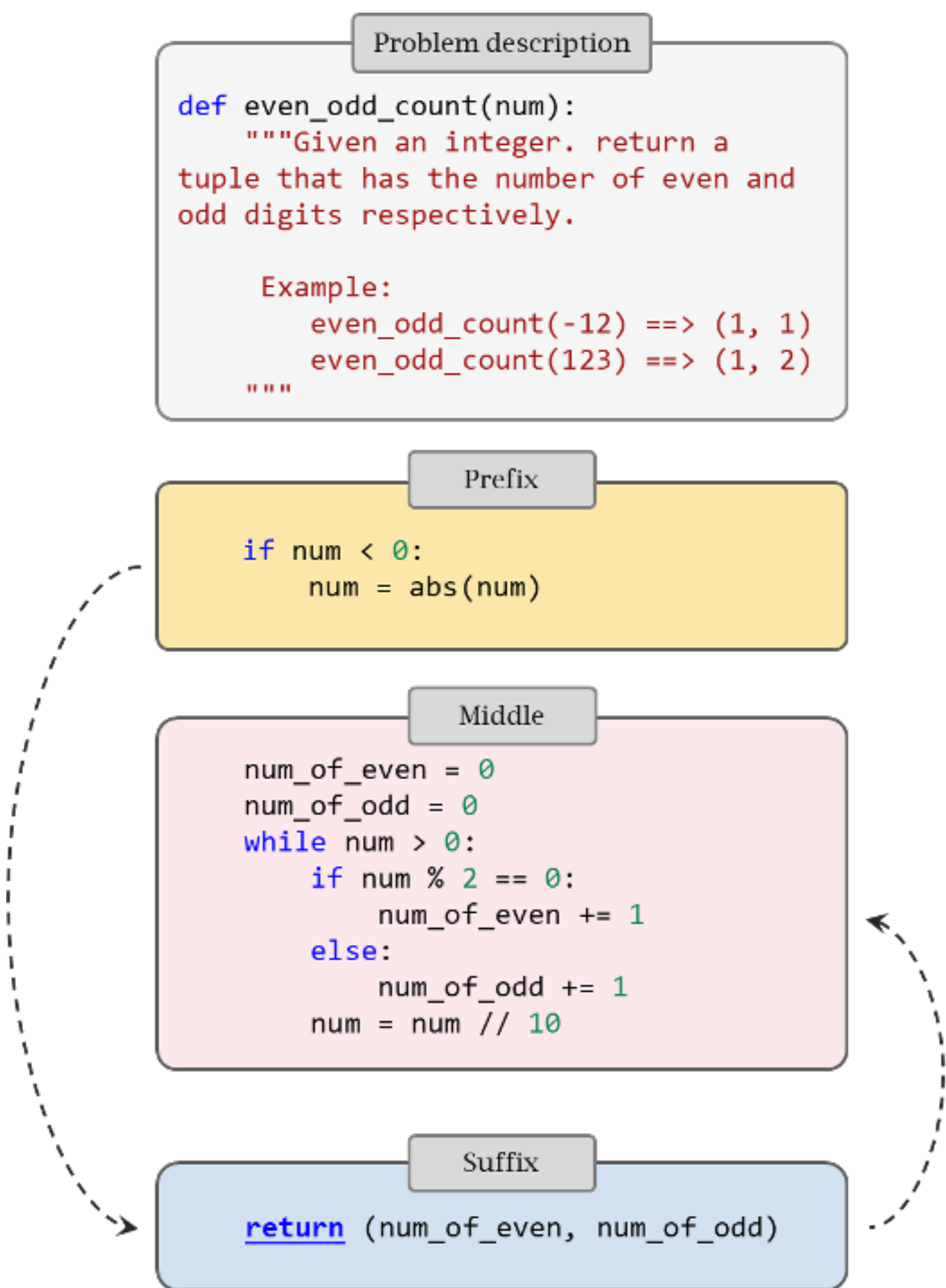
```
def correct_bracketing(brackets: str):
    """ brackets is a string of "<" and ">".
    return True if every opening bracket has a corresponding closing bracket.
    """
    depth = 0
    for b in brackets:
        if b == "<":
            depth += 1
        <|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|>
        <|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|>
        depth -= 1
        if depth < 0:
            return False
    return depth == 0
```

Fixed-Length

Models	Initial Mask Length					Avg.	Oracle
	4	8	16	32	64		
Dream-Coder-7B	24.9	61.2	72.6	62.4	55.5	55.3	93.3

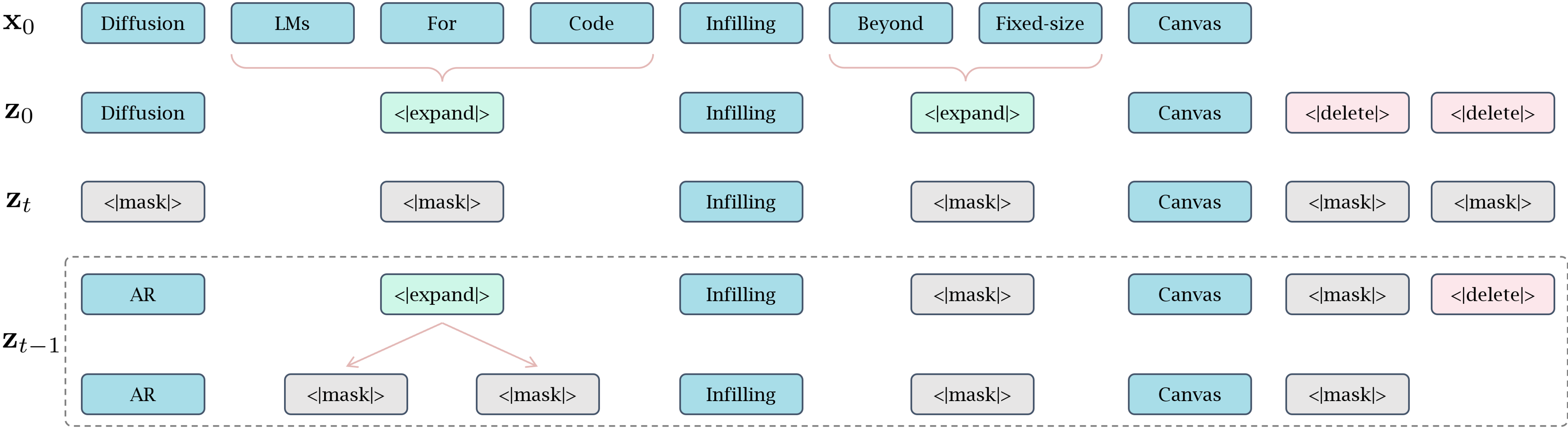
Self-infilling Code Generation (AR solution)

<prefix> ... </prefix> <suffix> ... </suffix> <middle> ... </middle>



Model	Size	Method	HUMANEVAL			MBPP		
			pass@1	pass@10	pass@100	pass@1	pass@10	pass@100
STARCORDERBASE [†]	15.5B	Left-to-right	30.4	-	-	49.0	-	-
STARCORDER [†]	15.5B		33.6	-	-	52.7	-	-
CODE LLAMA [†]	7B		33.5	59.6	85.9	41.4	66.7	82.5
CODE LLAMA [†]	13B		36.0	69.4	89.8	47.0	71.7	87.1
CODE LLAMA [†]	34B		48.8	76.8	93.0	55.0	76.2	86.6
Lemur [†]	70B		35.4	-	-	53.2	-	-
STARCORDERBASE	15.5B	Left-to-right	31.7	56.3	80.3	43.8	68.7	85.3
		Self-infill ($N=0$)	27.4	52.0	80.5	42.2	68.7	85.7
		Self-infill ($N=1$)	33.5	56.6	82.3	44.6	69.4	86.4
		Self-infill ($N=2$)	36.0	59.4	84.6	46.6	69.1	84.9
STARCORDER	15.5B	Left-to-right	35.4	62.1	85.1	48.6	71.5	86.6
		Self-infill ($N=0$)	29.2	58.0	85.9	46.4	70.7	85.6
		Self-infill ($N=1$)	37.8	63.2	86.5	47.8	71.3	86.1
		Self-infill ($N=2$)	38.4	64.7	87.3	50.0	71.1	87.2
CODE LLAMA	7B	Left-to-right	34.1	59.6	86.5	42.8	66.8	82.3
		Self-infill ($N=0$)	29.9	56.9	86.0	41.0	67.0	84.2
		Self-infill ($N=1$)	34.1	61.3	87.0	43.8	67.6	83.9
		Self-infill ($N=2$)	39.0	62.5	88.5	44.8	67.9	84.5
CODE LLAMA	13B	Left-to-right	35.4	69.7	90.0	47.2	71.9	87.3
		Self-infill ($N=0$)	32.3	69.2	90.5	44.0	71.2	87.8
		Self-infill ($N=1$)	38.4	70.7	92.5	47.2	72.6	88.3
		Self-infill ($N=2$)	40.8	72.1	91.1	49.0	73.0	89.2

DreamOn: Go Beyond Fixed-Size Canvas



DreamOn: Go Beyond Fixed-Size Canvas

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers
    are any two numbers closer to each other than given threshold.
    """
    for idx, elem in enumerate(numbers):
        <|mask|><|mask|><|mask|><|mask|>
        if idx != idx2:
            distance = abs(elem - elem2)
            if distance < threshold:
                return True

    return False
```

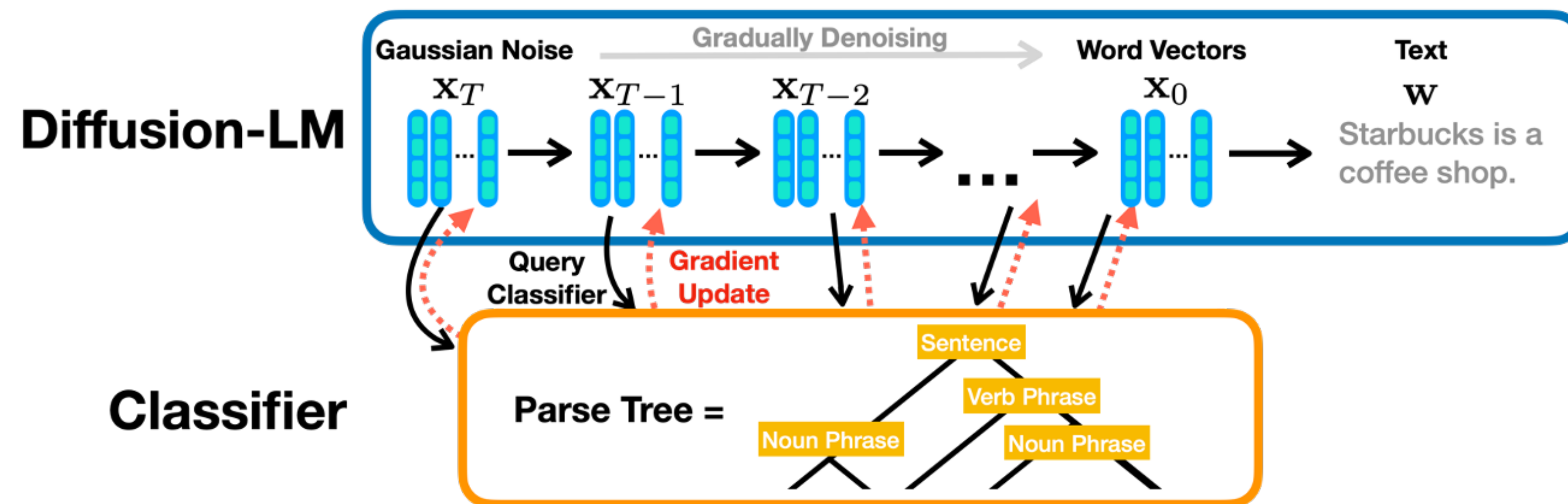
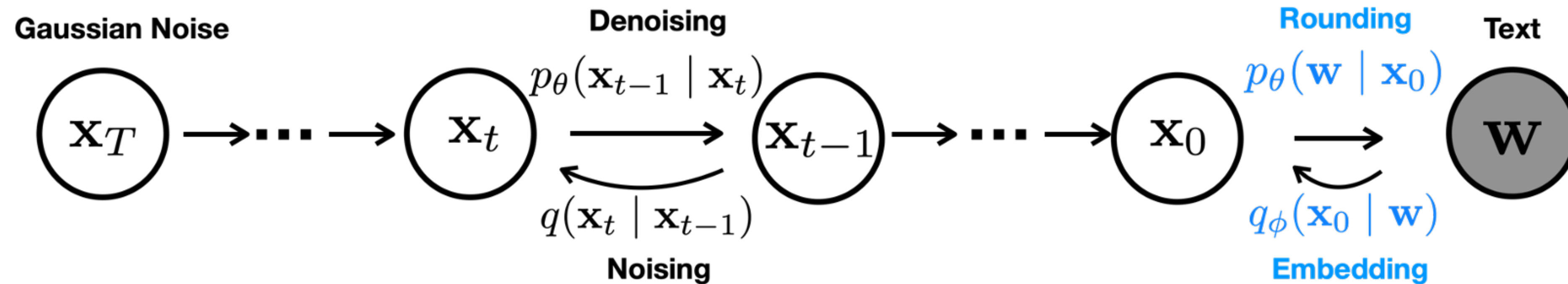
```
def correct_bracketing(brackets: str):
    """ brackets is a string of "<" and ">".
    return True if every opening bracket has a corresponding closing bracket.
    """
    depth = 0
    for b in brackets:
        if b == "<":
            depth += 1
            <|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|>
            <|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|><|mask|>
            depth -= 1
        if depth < 0:
            return False
    return depth == 0
```

DreamOn

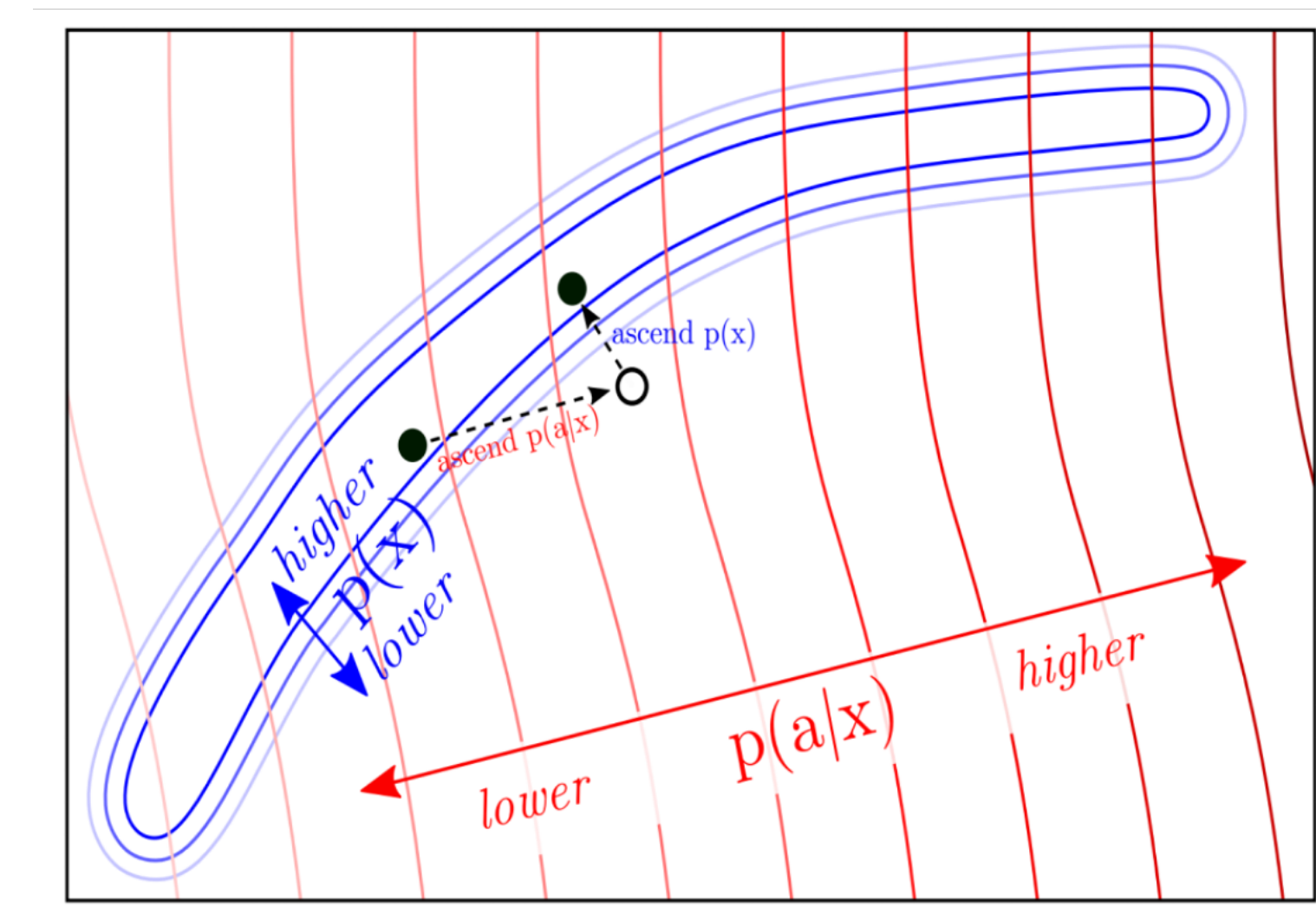
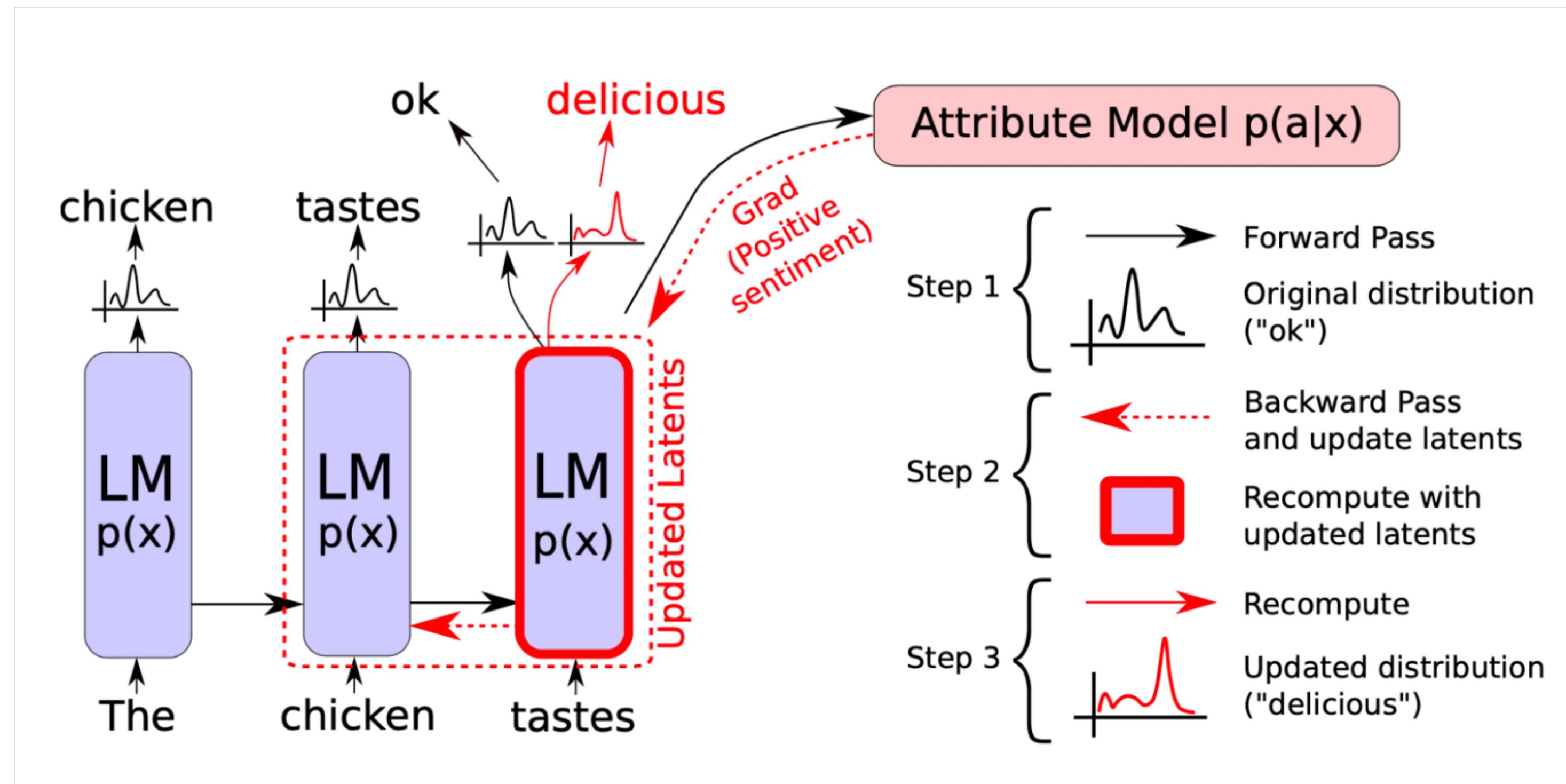
DreamOn

Models	Initial Mask Length					Avg.
	4	8	16	32	64	
Dream-Coder-7B	24.9	61.2	72.6	62.4	55.5	55.3
DREAMON-7B	88.7	90.6	91.0	91.6	92.1	90.8

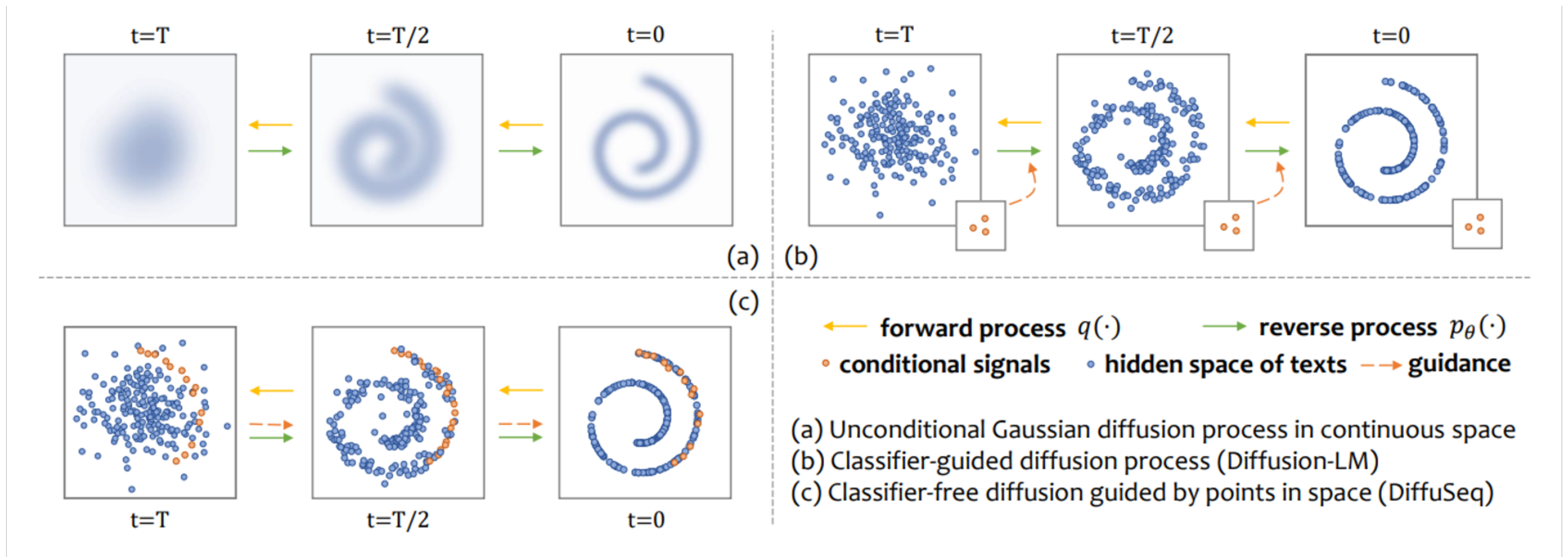
Text Diffusion Models



Classifier Guidance

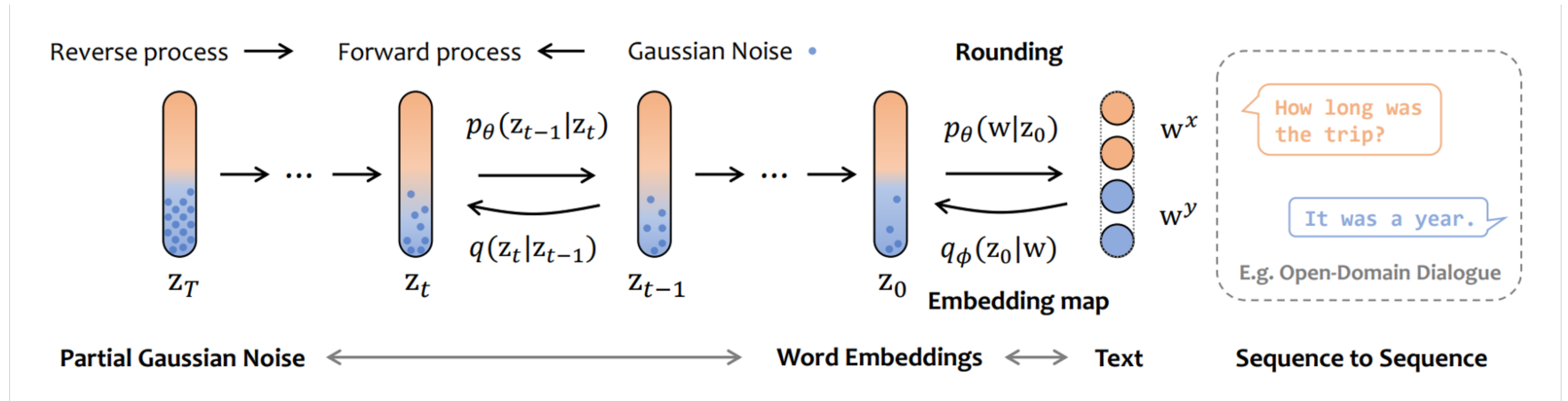


Sequence to Sequence Text Diffusion



“Seq2Seq” tasks: $\mathbf{x} \rightarrow \mathbf{y}$

Sequence to Sequence Text Diffusion



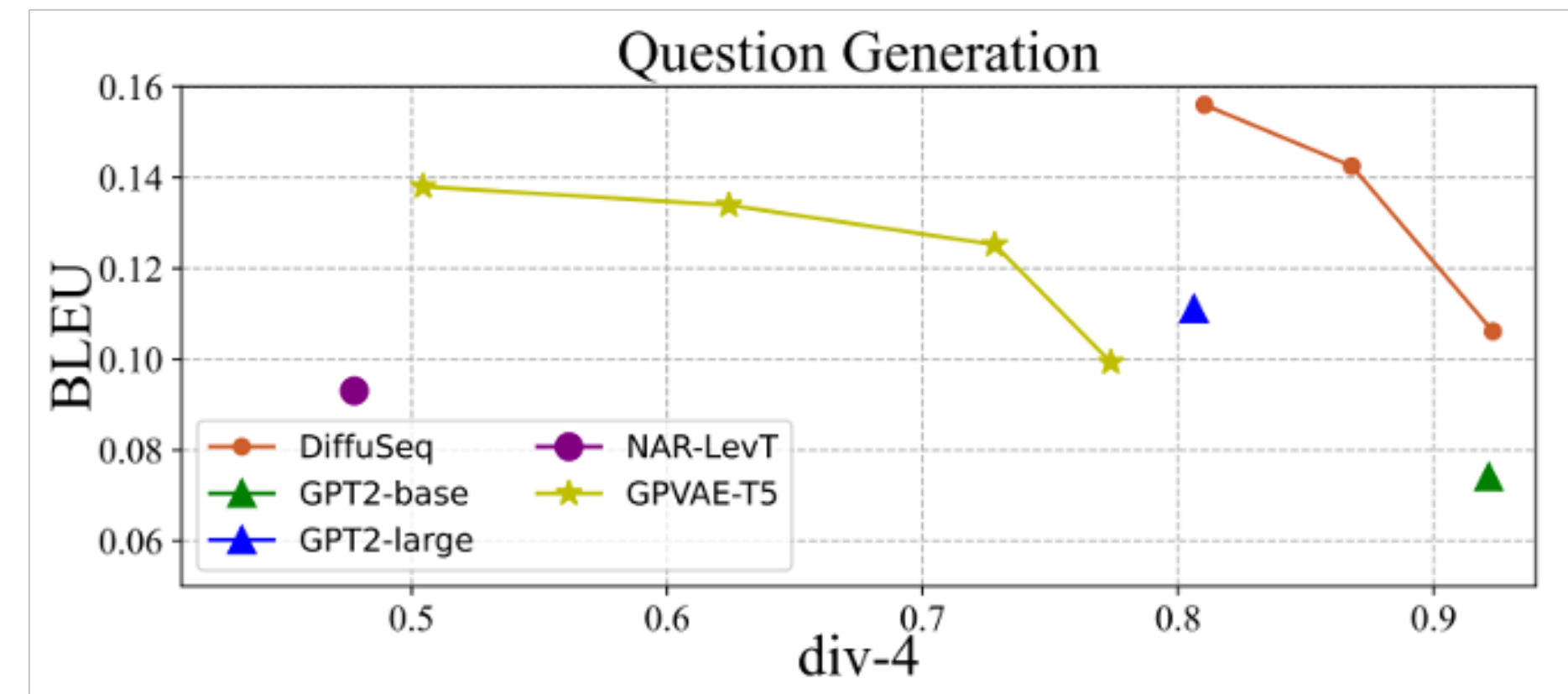
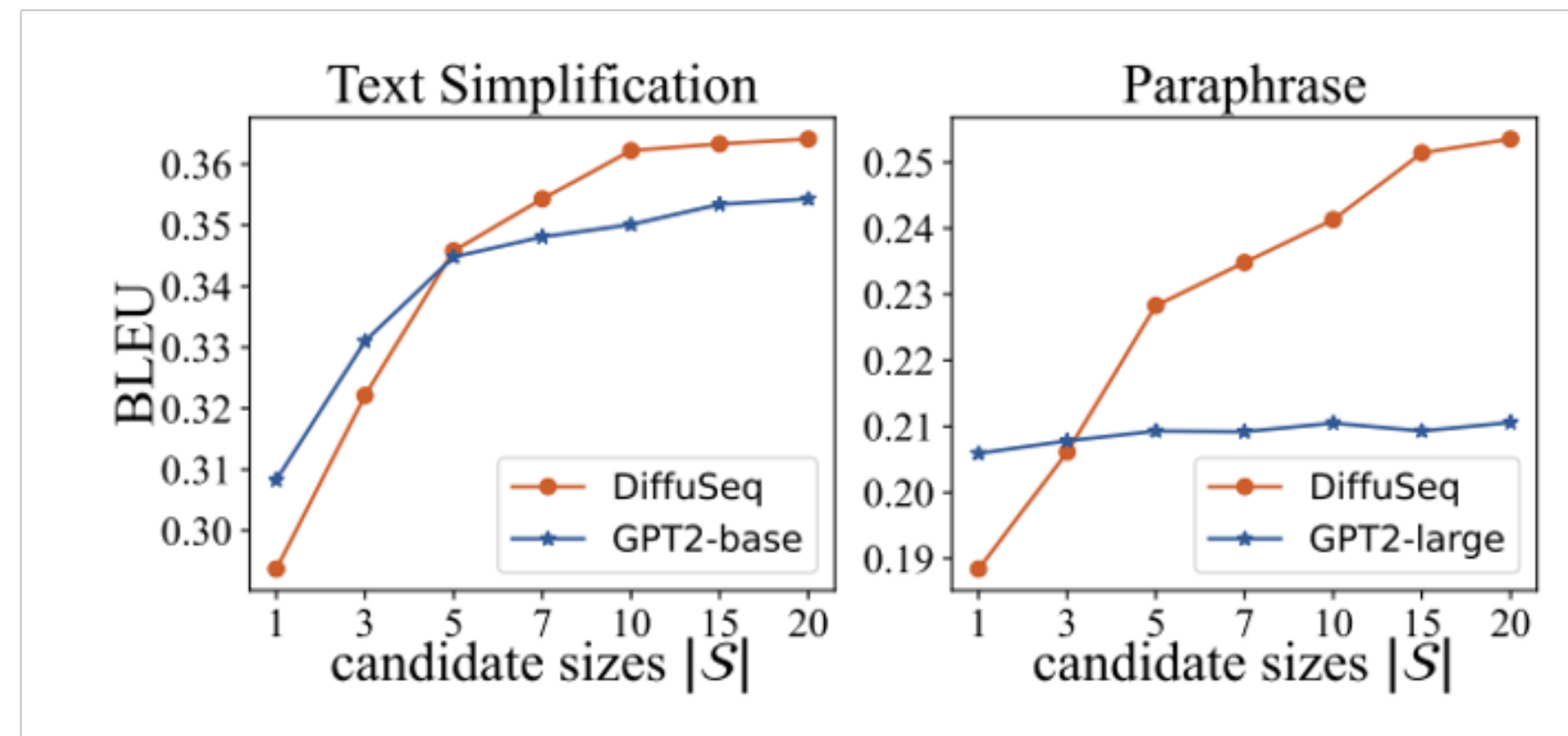
$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} \left[\underbrace{\log \frac{q(\mathbf{z}_T|\mathbf{z}_0)}{p_\theta(\mathbf{z}_T)}}_{\mathcal{L}_T} + \sum_{t=2}^T \underbrace{\log \frac{q(\mathbf{z}_{t-1}|\mathbf{z}_0, \mathbf{z}_t)}{p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)}}_{\mathcal{L}_{t-1}} + \underbrace{\log \frac{q_\phi(\mathbf{z}_0|\mathbf{w}^{x \oplus y})}{p_\theta(\mathbf{z}_0|\mathbf{z}_1)}}_{\mathcal{L}_0} - \underbrace{\log p_\theta(\mathbf{w}^{x \oplus y}|\mathbf{z}_0)}_{\mathcal{L}_{\text{round}}} \right].$$

x

$$\min_{\theta} \mathcal{L}_{\text{VLB}} = \min_{\theta} \left[\sum_{t=2}^T \|\mathbf{y}_0 - \tilde{f}_\theta(\mathbf{z}_t, t)\|^2 + \|\text{EMB}(\mathbf{w}^y) - \tilde{f}_\theta(\mathbf{z}_1, 1)\|^2 + \mathcal{R}(\|\mathbf{z}_0\|^2) \right]$$

Sequence to Sequence Text Diffusion

Diversity Ensures Quality



Statement: *The Japanese yen is the official and only currency recognized in Japan.*

Question: *What is the Japanese currency?*

GPVAE-T5

- * What is the japanese currency
- * What is the japanese currency
- * What is the japanese currency

NAR-LevT

- * What is the basic unit of currency for Japan ?
- * What is the basic unit of currency for Japan ?
- * What is the basic unit of currency for Japan ?

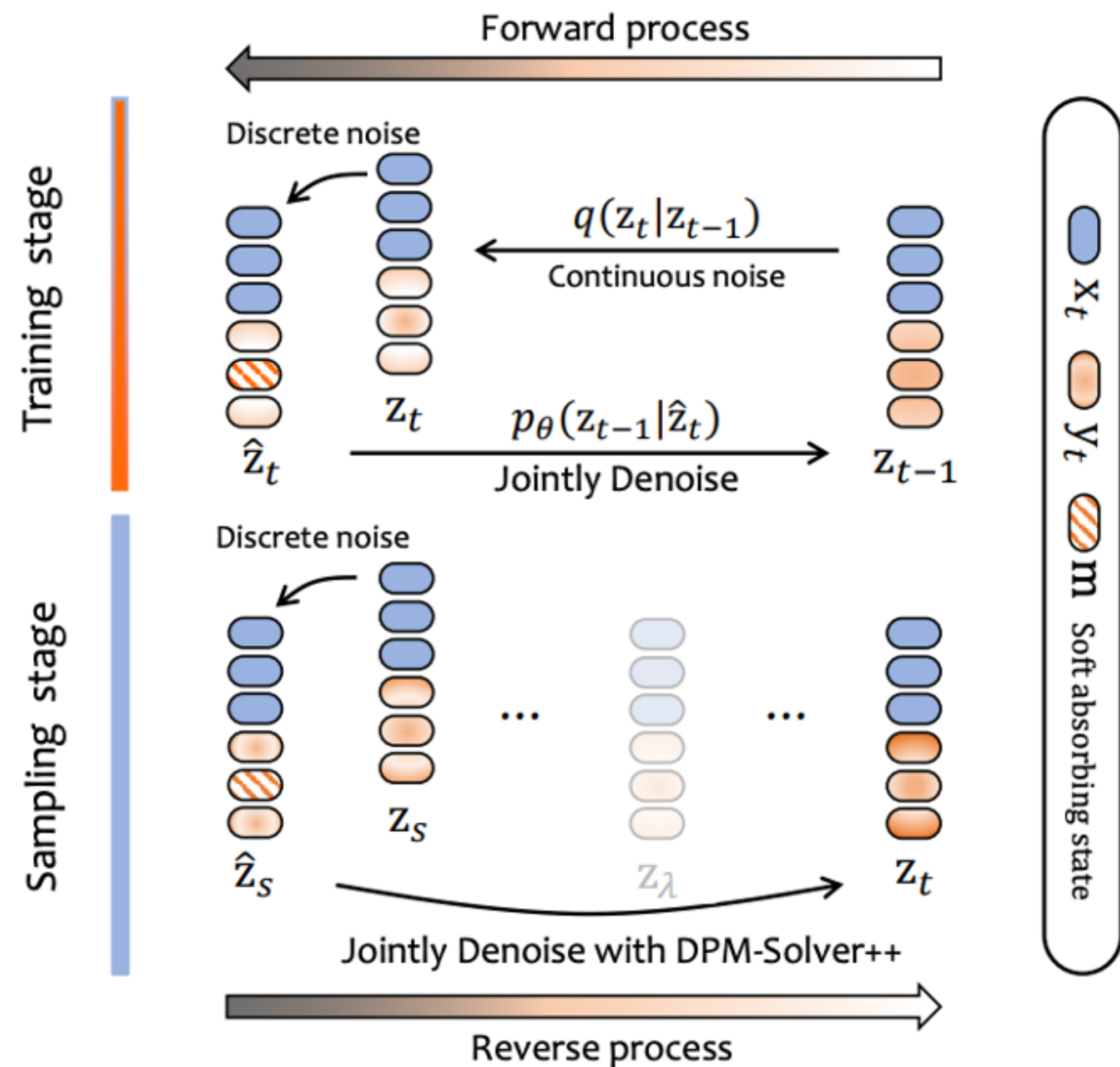
GPT2-large finetune

- * What is the basic unit of currency for Japan?
- * What is the Japanese currency
- * What is the basic unit of currency for Japan?

DiffuSeq

- * What is the Japanese currency
- * Which country uses the “yen yen” in currency
- * What is the basic unit of currency?

Accelerated Seq2Seq Diffusion Models



Training

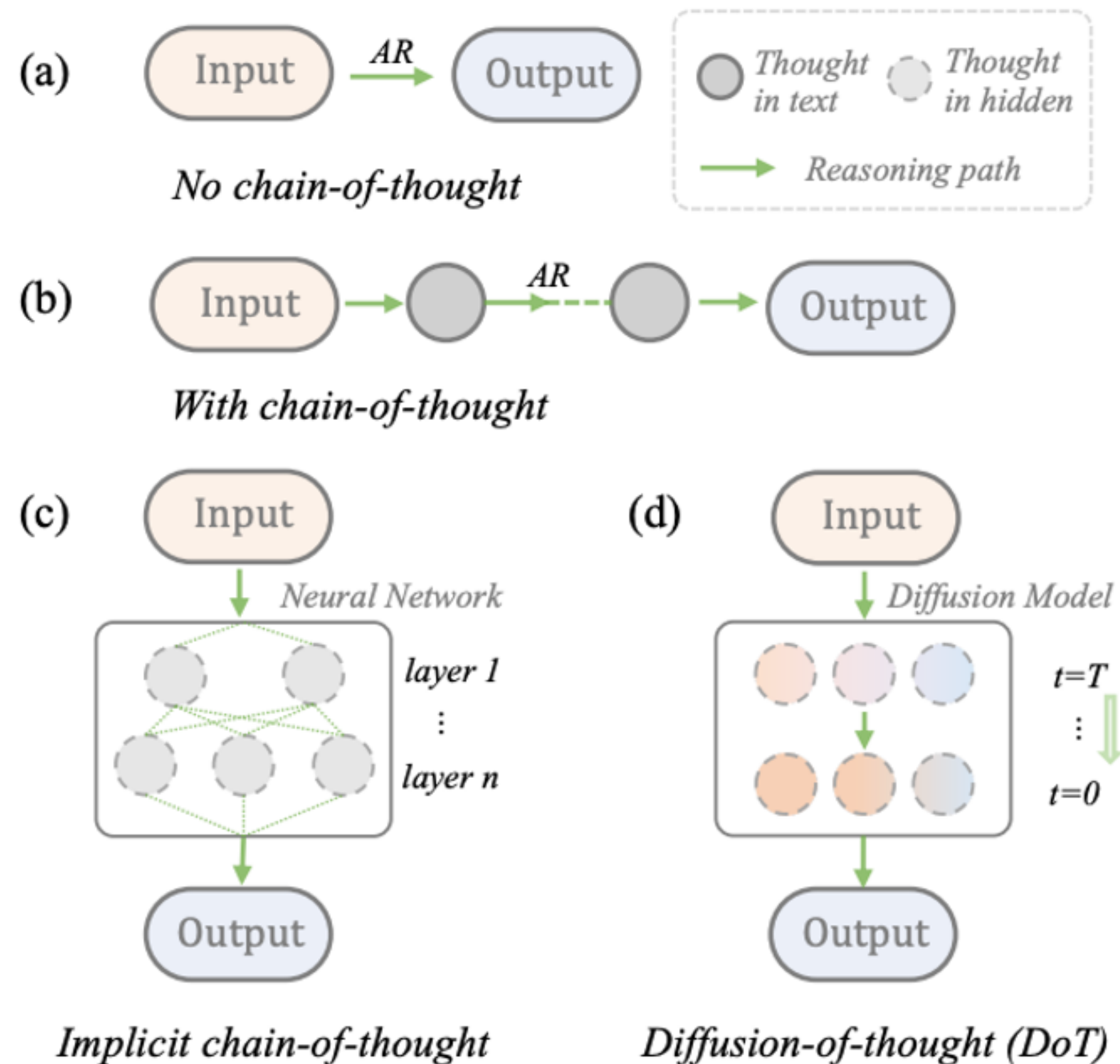
Discrete noise: replace the embedding of one token with [mask] token's embedding. 4x  (fp16)

Sampling

Use DPM-solver to speedup generation. 800x  (fp16)

Diffusion of Thoughts

Diffusion of Thoughts (DoTs)



Diffusion of Thoughts

Questions:

When Freda cooks canned tomatoes into sauce, they lose half their volume. Each 16 ounce can of tomatoes that she uses contains three tomatoes. Freda's last batch of tomato sauce made 32 ounces of sauce. How many tomatoes did Freda use?

CoT Decoding:

<

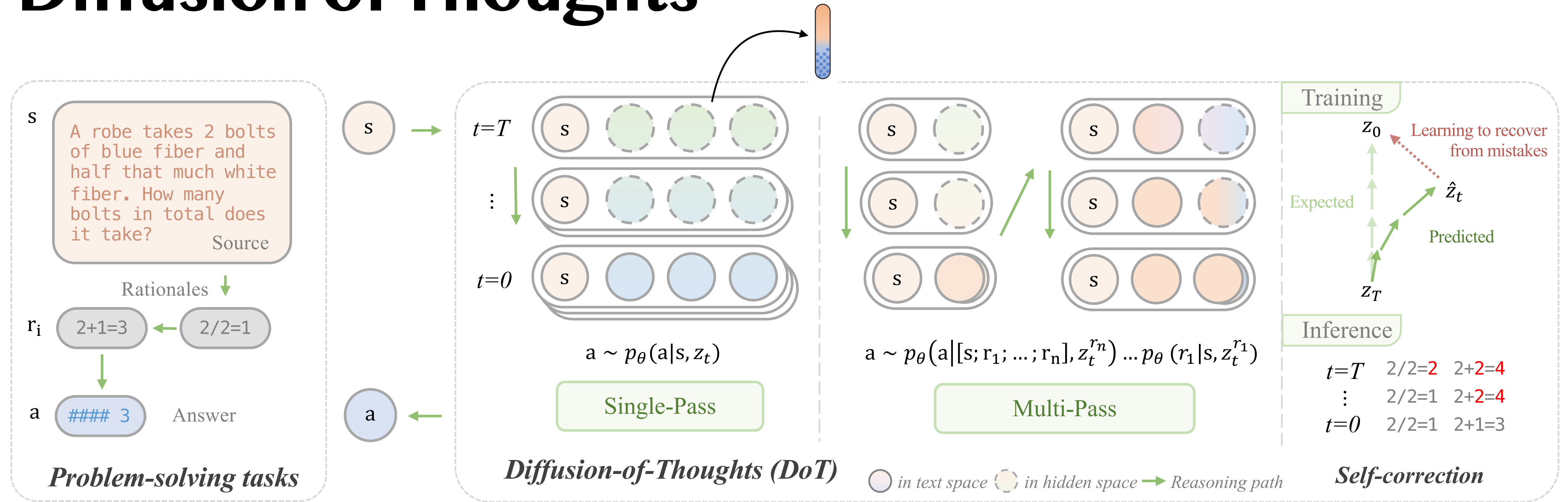
DoT Decoding:

<<32/16=2>> <<12/12=2*3 ## *2=2= #####

MP-DoT Decoding:

<<32==2>>

Diffusion of Thoughts



Models	4 × 4 Mult.		5 × 5 Mult.		Boolean logic		GSM8K-Aug	
	Acc	Throughput	Acc	Throughput	Acc	Throughput	Acc	Throughput
Chain-of-Thoughts (CoT)								
GPT2-medium	100	1.2	100	0.8	100	0.5	43.9	1.1
GPT2-large	100	0.8	99.3	0.6	100	0.3	44.8	0.7
Diffusion-of-Thoughts (DoT)								
SEDD-small	100	59.2	100	55.5	100	33.3	45.3 (51.8)	1.0
SEDD-medium	100	31.8	100	28.5	100	17.2	53.5 (59.4)	0.5



Ludwig Wittgenstein

“Imagine people who could only think aloud.
(As there are people who can only read aloud.)”

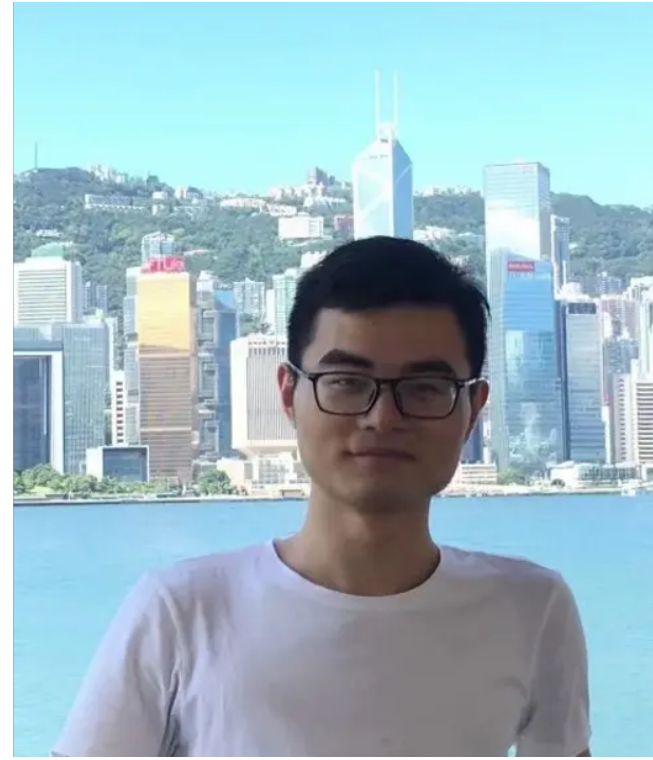
— Philosophical Investigations, §331, 1953



Shansan Gong



Lin Zheng



Jiacheng Ye



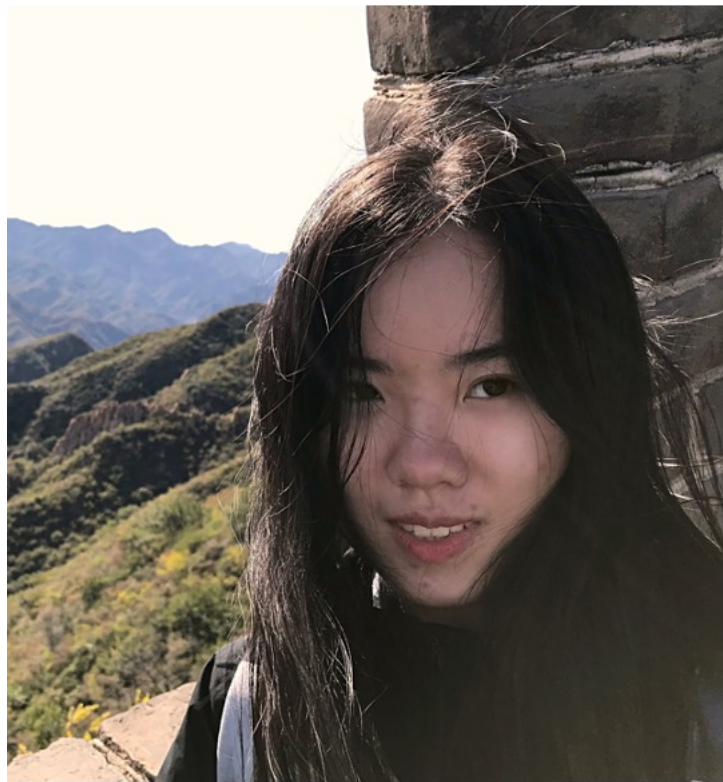
Zirui Wu



Zhihui Xie



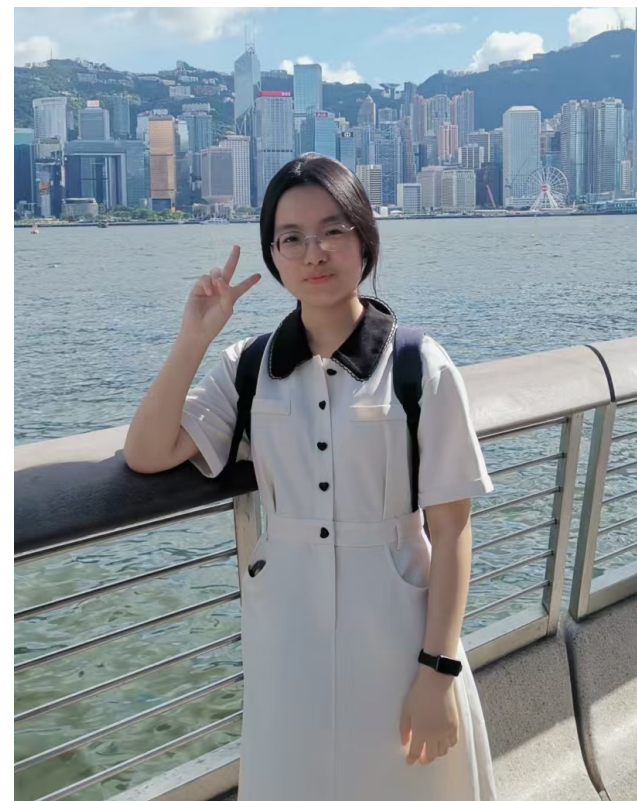
Xueliang Zhao



Chang Ma



Jiahui Gao



Jingwei Dong



All the
blogs
are here.

Thank you!