Diffusion Language Models

Aaron Mueller
CS599 B1: Advanced Natural Language Processing
Nov. 19, 2026

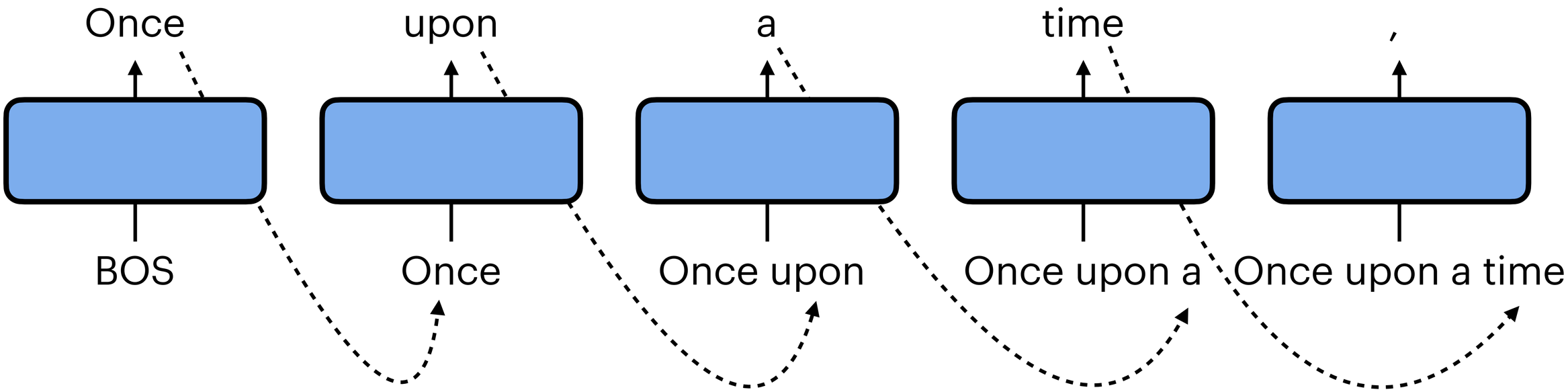
Administrivia

- Peer reviews are due **tomorrow, Nov. 20** at 11:59pm
- No classes next week; enjoy the Thanksgiving holiday! 🦃
- Grades for midway reports and peer reviews will likely be released the weekend after Thanksgiving.
- For our Thursday lecture the week after Thanksgiving, you all will pick the topic! Put your ideas here:



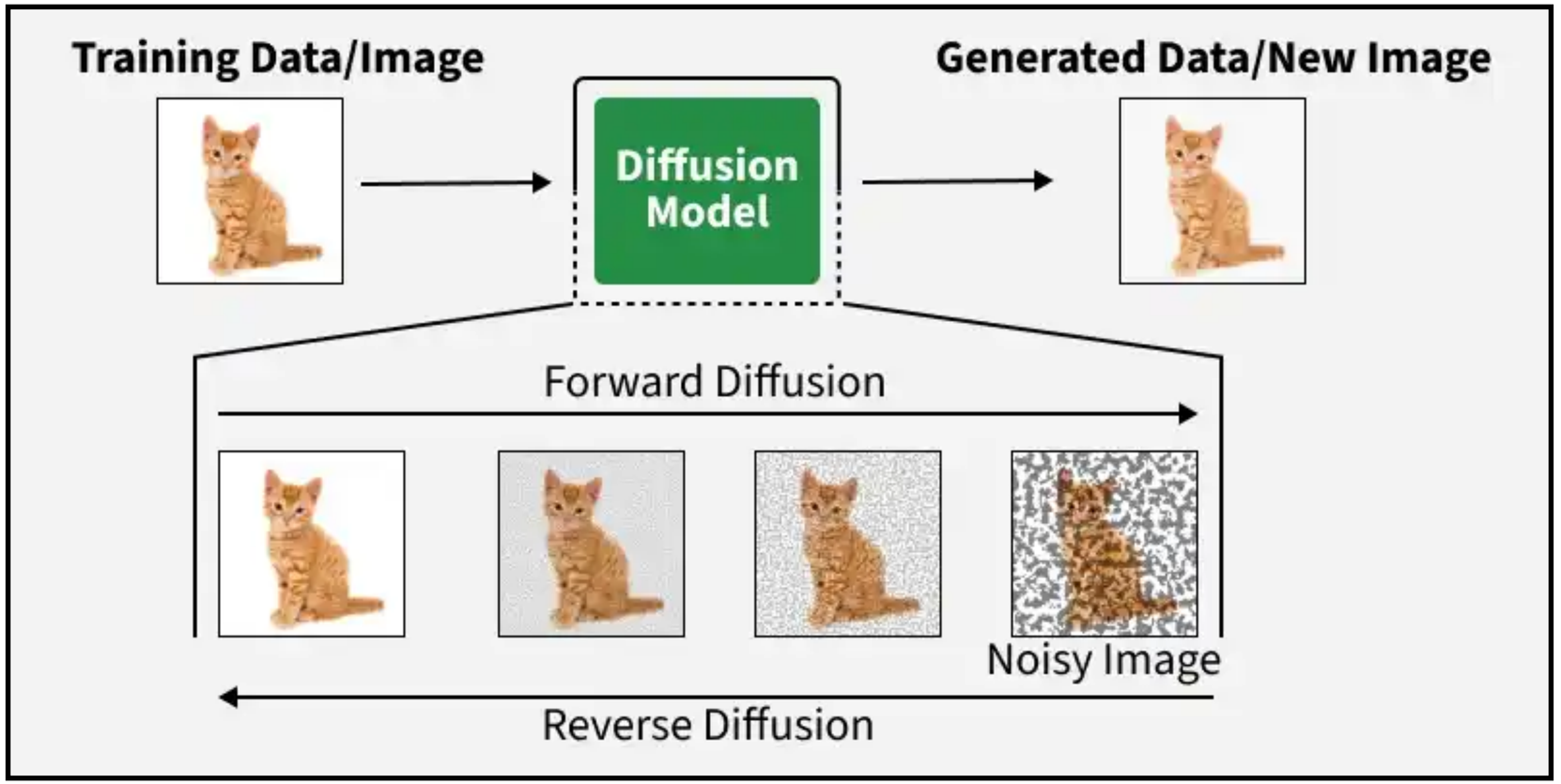
https://docs.google.com/forms/d/e/1FAIpQLSeTS2sANzgbAjpqD3IbJ1Qek-yKd7DrT-RlXpPQXfDOMY-AlA/viewform?usp=dialog

Review: Autoregressive Language Modeling



Autoregressive language models generate text by generating one token at a time, appending that token to the context, and using that new sequence as the input in the next timestep.

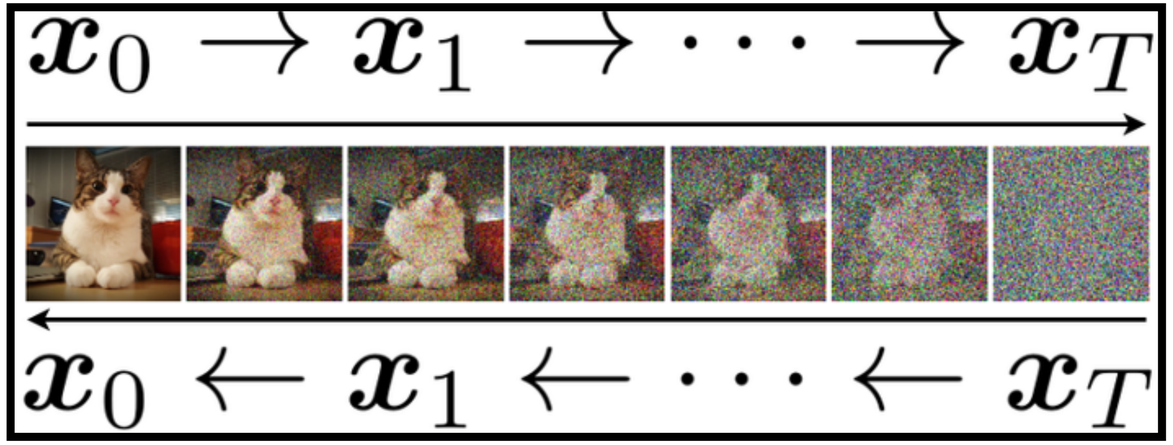
What is diffusion?



In the ML context, diffusion refers to a (usually gradual) noising or denoising process.

We train the model by generating data via forward diffusion, and train it to reverse the process such that we can generate new outputs during inference.

Diffusion Models (in Computer Vision)

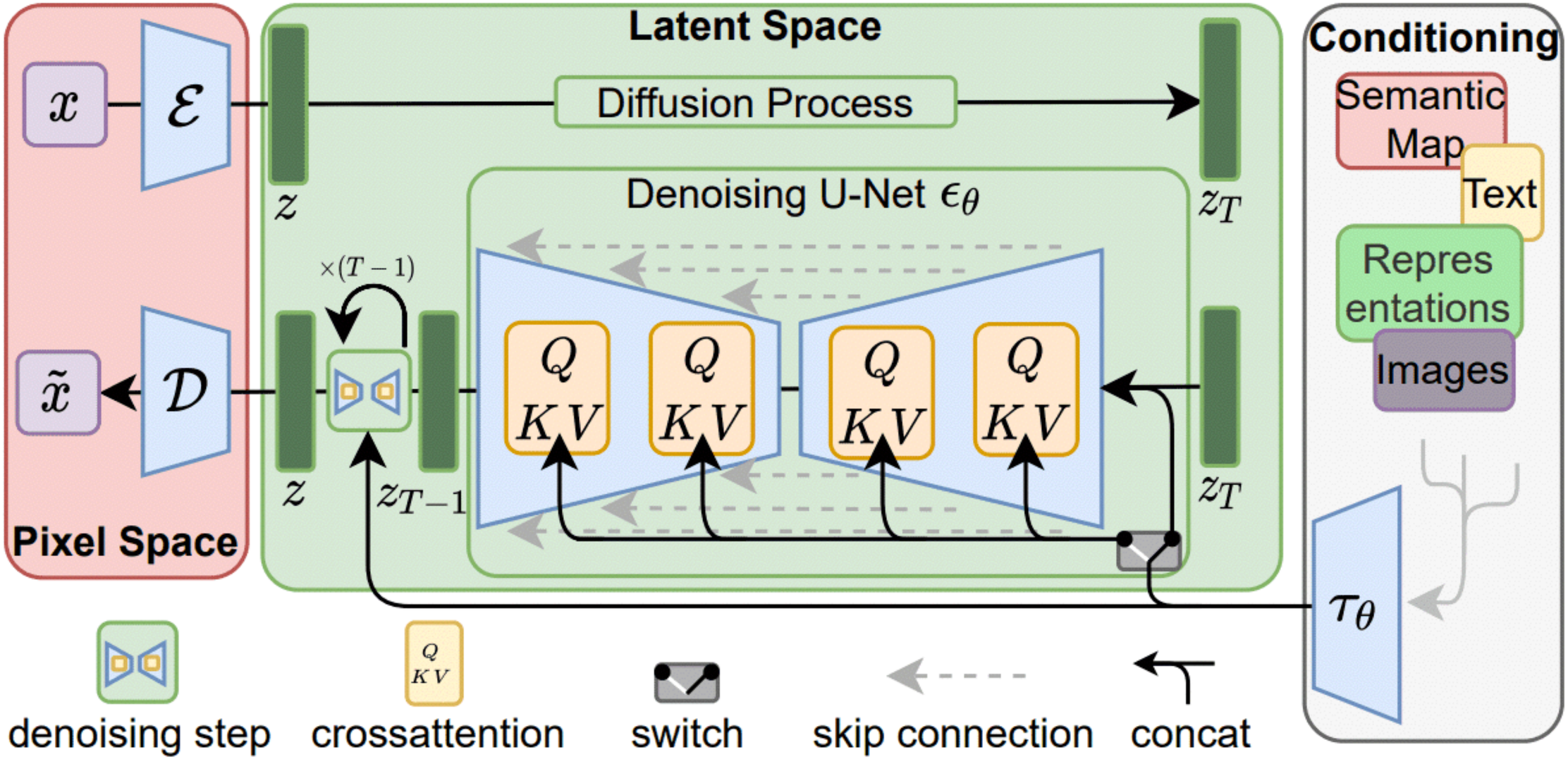


- The goal of forward diffusion is to generate training data for reverse diffusion
- Usually trained with the following loss:

$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon} = \left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|_2^2 \right]$$

- Forward diffusion involves taking the clean sample x_0 and corrupting it with Gaussian noise $\epsilon \sim N(0,I)$, cumulatively gathering noise according to the noise schedule $\bar{\alpha}_t$
- During inference, we sample a tensor of pure Gaussian noise, and use a sampler to subtract the predicted noise and move slightly closer to a cleaner image

Diffusion Models (in Computer Vision)



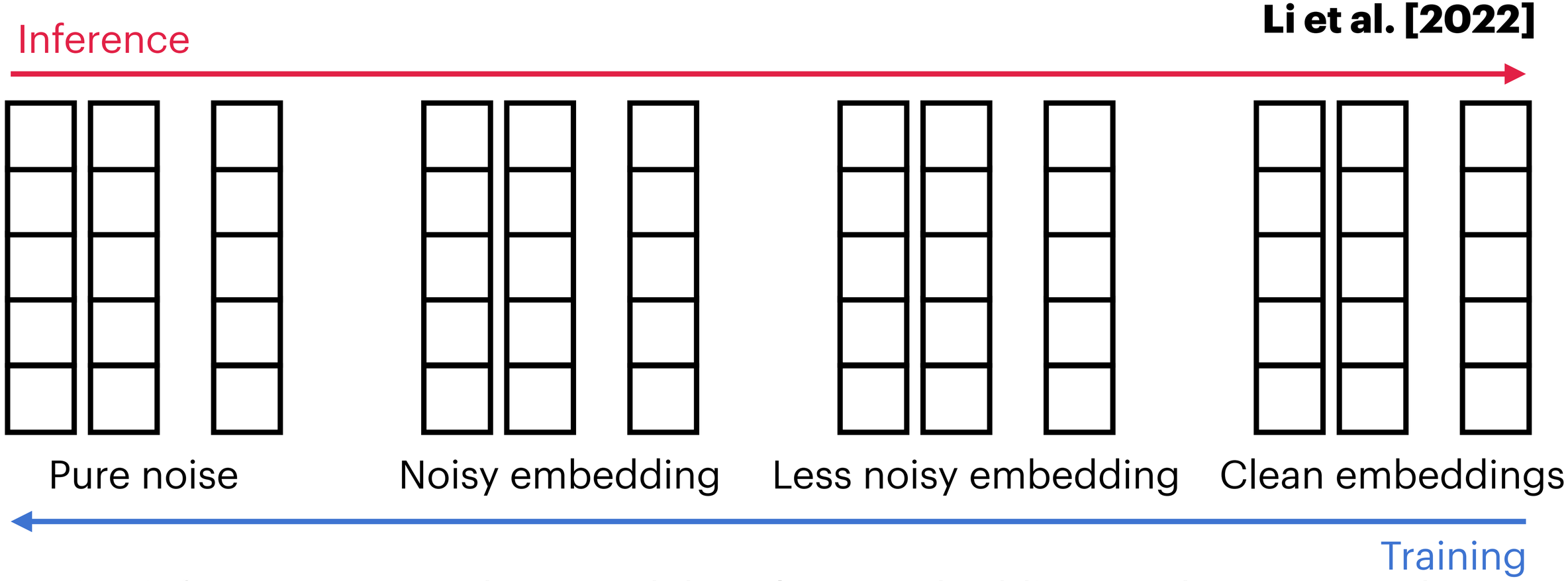
Why is applying diffusion to language models hard?

- What, exactly, are we (de)noising?
  - Tokens?
  - Full sequences?

- What does it mean to add noise to a token embedding? This may not correspond to any token at all

Diffusion-LM

Li et al. [2022]



During inference, we need to round the “clean” embeddings to the nearest token embedding representation.

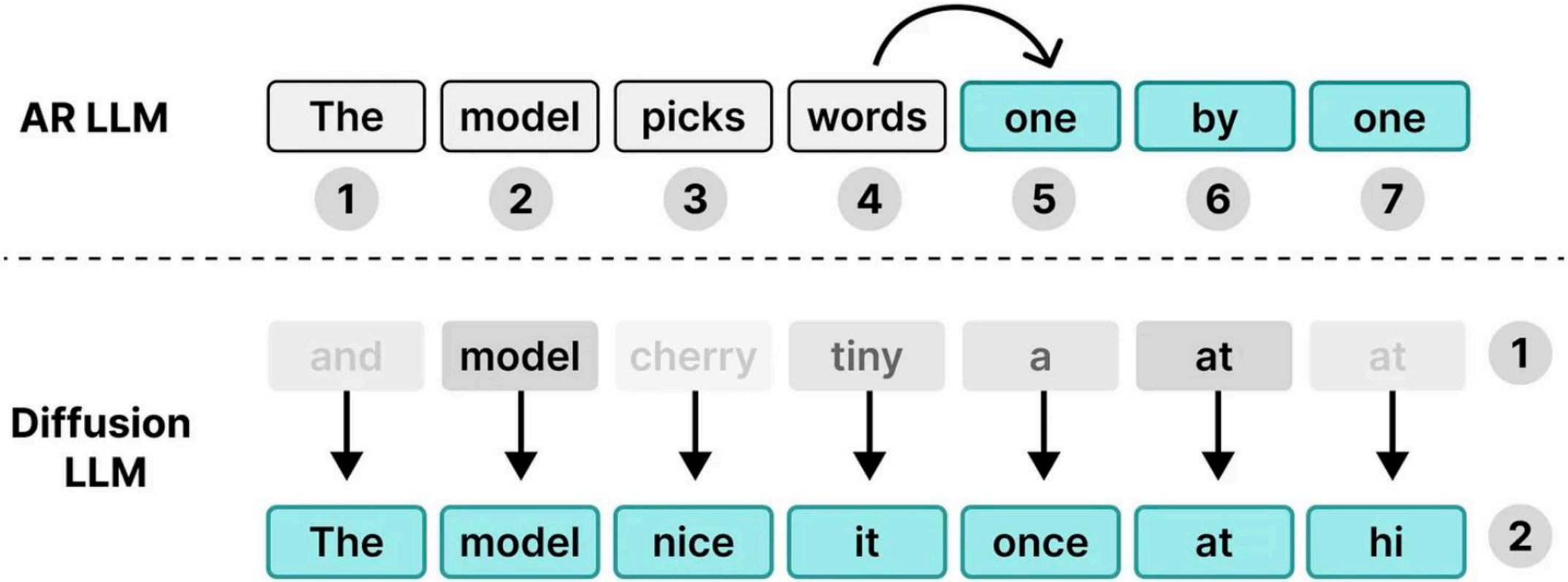
This can lead to significant rounding errors that yield nonsensical outputs.

Why is applying diffusion to language models hard?

- What, exactly, are we (de)noising?
  - Tokens?
  - Full sequences?

- What does it mean to add noise to a token embedding? This may not correspond to any token at all
  - **Uniform noise:** should we swap in random tokens?
  - **Absorbing noise:** should tokens go to some placeholder token, like a [MASK] token?

Diffusion Language Modeling



Instead of left-to-right, think of coarse-to-fine generation.

Diffusion models often generate the high-level/high-confidence tokens first, then fill in the details.

Masked Diffusion Models

- One can take clean sequences and mask a random fraction of tokens
- Here, the amount of noise corresponds to the probability that a given token is masked
- We use an ELBO training objective:

$$L_{\text{ELBO}} = \mathbb{E}_{t\sim U[0,1]}\mathbb{E}_{q(z_t|x)}\left[\frac{1}{t}\log p_\theta(x \mid z_t)\right]$$

q masks tokens with probability $\propto t$

Likelihood of clean sequence x given partially masked sequence z_t

Masked Diffusion Models

$$L_{\text{ELBO}} = \mathbb{E}_{t\sim U[0,1]}\mathbb{E}_{q(z_t|x)}\left[\frac{1}{t}\log p_\theta(x \mid z_t)\right]$$

Likelihood of clean sequence x given partially masked sequence z_t

- Recall the cross-entropy loss we used to train autoregressive models:

$$L_{\text{CL}} = -\sum_{i=1}^{t}\log p_\theta(y \mid x)$$

- Autoregressive models can be viewed as special cases of diffusion models that use iterative left-to-right unmasking
- This is a lot like masked language models, such as BERT, but with a random mask rate (BERT's mask rate is fixed as a hyperparameter)

Stand tall and be proud.

No matter how weak or unworthy you feel,

keep your heart burning, grit your teeth and move forward.

If you just curl up in a ball and hide, time will pass you by.

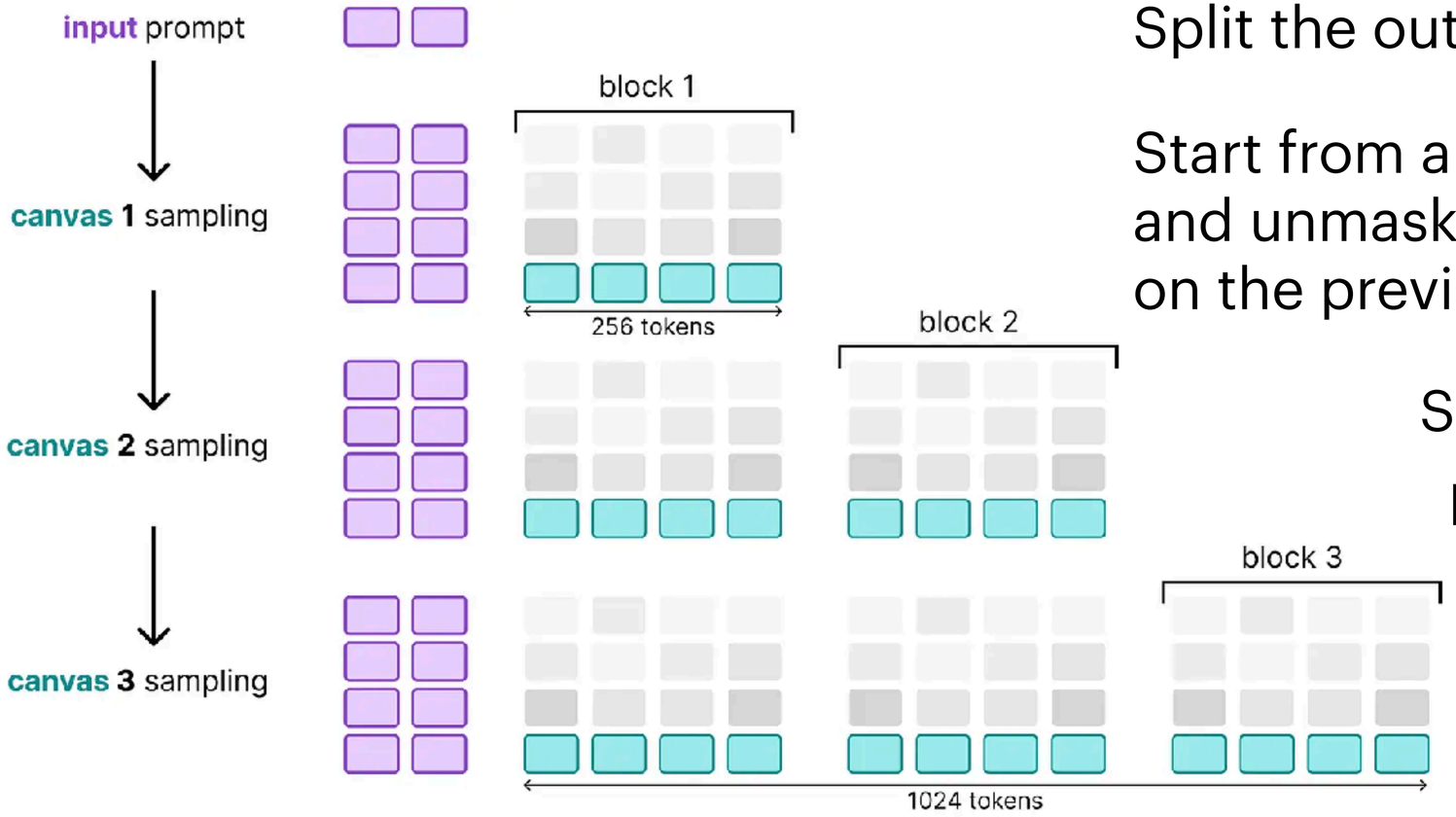
It won't stop for you while you wallow in your grief.

Issues with Masked Diffusion Models

- Masked diffusion models have various issues that make them not attractive choices compared to other architectures:
  - They can only generate fixed-length sequences
  - No error correction
  - Not particularly fast compared to a typical LM

Block Diffusion

- To fix efficiency issues and get variable-length generation, we can use a technique called **block diffusion**, where many tokens are denoised at once



Split the output into blocks of k tokens

Start from a block full of [MASK] tokens, and unmask them via denoising, conditioned on the previous blocks and the prompt

Smaller blocks -> better quality

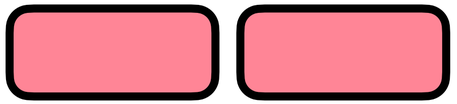
Larger blocks -> more parallelism

Arriola et al. [2025]

An Issue with Parallel Decoding

Diffusion LMs make **conditional independence** assumptions that do not accurately describe dependencies between tokens. For example:

$$p(x_i, x_j \mid z_t) \approx p(x_i \mid z_t) p(x_j \mid z_t)$$

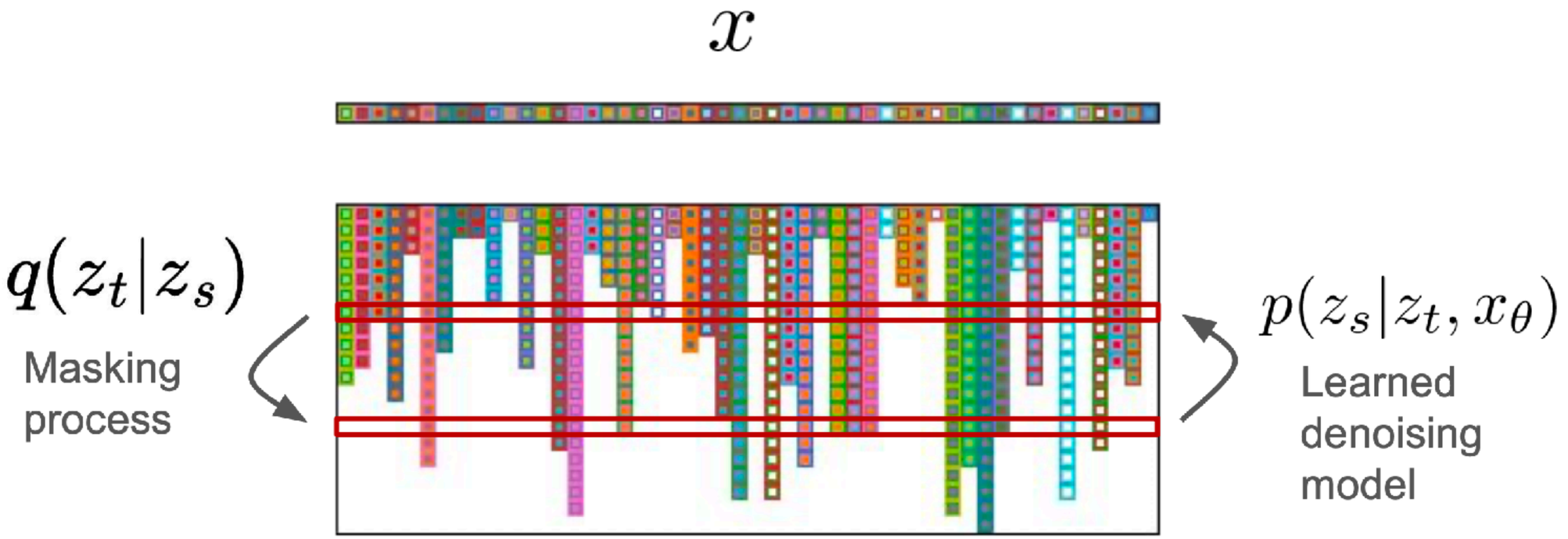
I live in 

If “New York” and “San Francisco” are both likely, we could end up with “San York”.

The original formulation of masked diffusion models does not allow the model to revise its predictions once a token has been unmasked

Iterative Refinement

- Our masked diffusion model could never remask an unmasked token, meaning it could never correct its previous mistakes or update its predictions
- Modern diffusion LMs allow the model to **remask** its tokens during inference



Remasking Diffusion

- At each step, we keep some newly remasked tokens but also remask a subset of previously unmasked tokens
- Can be tacked onto standard diffusion LMs as a predictor corrector



Wang et al. [2025]

LLaDA

- **Nie et al. [2025]** propose LLaDA, a diffusion LLM

- Pre-trained to match the size and training setup of Llama as closely as possible while varying the architecture
  - Pre-trained on 2.3T tokens
  - 8B parameters

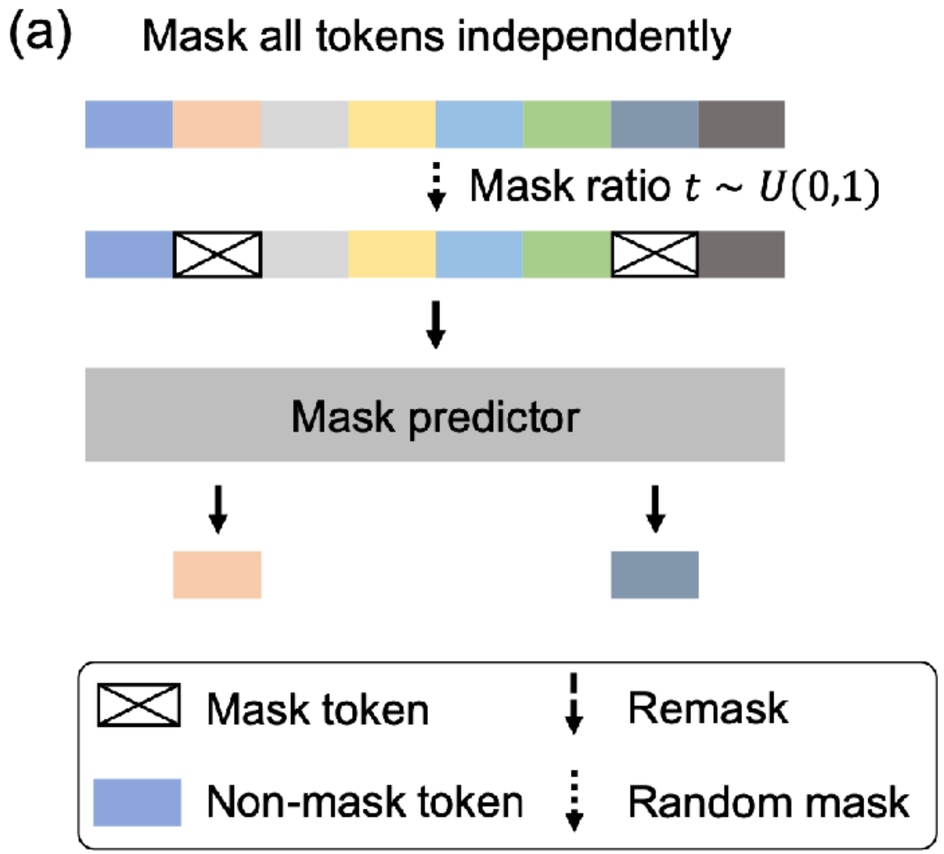
- Same Transformer architecture, too, except that we remove the causal mask
  - Diffusion LMs are also incompatible with KV caching according to Nie et al., so we use vanilla multi-head attention instead of grouped query attention

Pre-training

Nie et al. [2025]

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i=\mathbf{M}]\log p_\theta(x_0^i|x_t)\right]$$

- Trained to take masked input x_t and predict all tokens simultaneously to recover the clean input x_0
- Loss is computed only over masked tokens



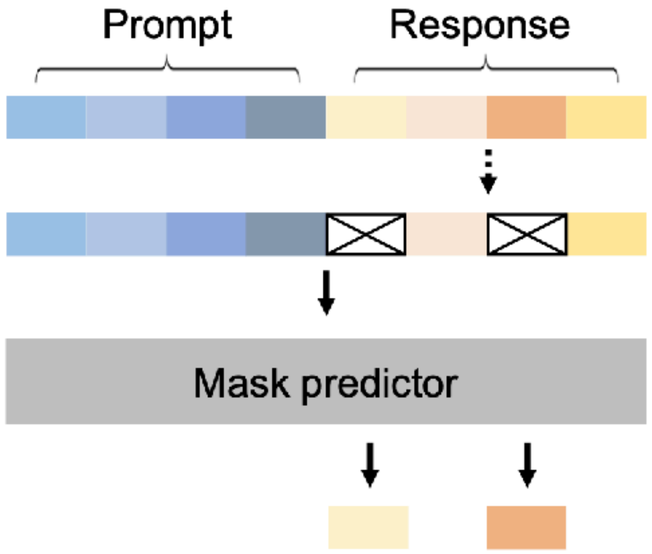
Fine-tuning

Nie et al. [2025]

- We use paired data (p_0, r_0), where p_0 is the prompt and r_0 is the response
- Implemented similarly to pre-training:

$$-\mathbb{E}_{t,p_0,r_0,r_t}\left[\frac{1}{t}\sum_{i=1}^{L'}\mathbf{1}[r_t^i=\mathrm{M}]\log p_\theta(r_0^i|p_0,r_t)\right]$$

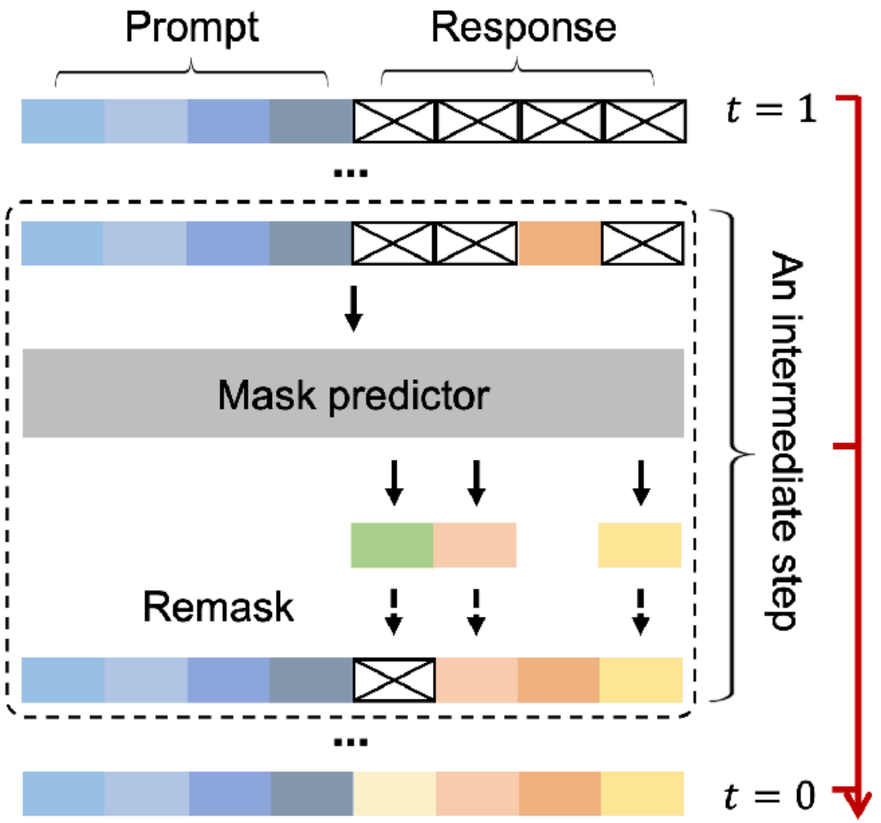
- Nie et al. perform instruction fine-tuning using 4.5M pairs



Inference

Nie et al. [2025]

- Given prompt p_0, from time $t \in (0,1]$ to $s \in [0,t)$, we feed p_0 and a fully masked response r_t into the model and predict all masked tokens simultaneously
  - Response length is a hyperparameter
  - If an intermediate predicted token is EOS, every token afterwards is truncated post hoc
- After each timestep, we remask $\frac{s}{t}$ predicted tokens to obtain r_s
  - Remask tokens the model predicted with the lowest confidence



Results

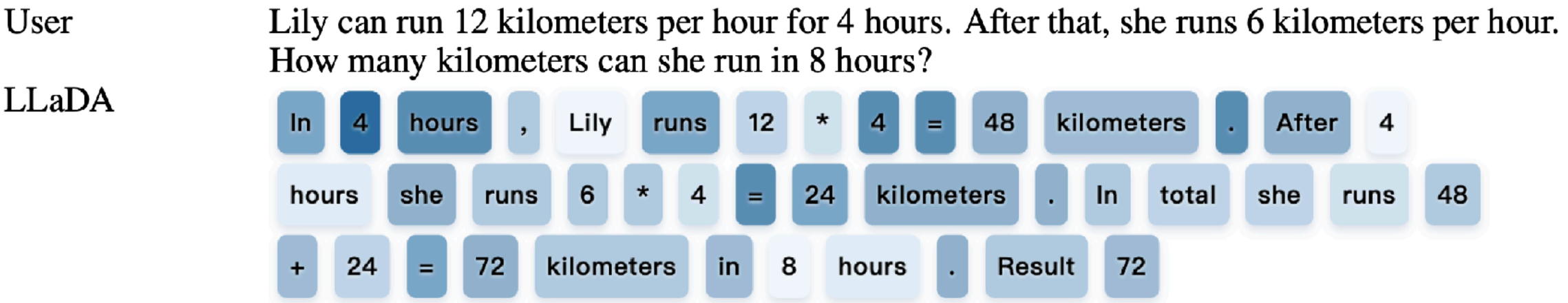
Nie et al. [2025]

- Mixed results, but LLaDA performs generally in the same vicinity of Llama on average across tasks

	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†
Model	Diffusion	AR	AR	AR
Training tokens	2.3T	15T	2T	7T
			General Tasks	
MMLU	**65.9** (5)	65.4 (5)	45.9 (5)	70.3 (5)
BBH	49.7 (3)	**62.1** (3)	39.4 (3)	62.3 (3)
ARC-C	45.9 (0)	**53.1** (0)	46.3 (0)	60.6 (25)
Hellaswag	70.5 (0)	**79.1** (0)	76.0 (0)	80.7 (10)
TruthfulQA	**46.1** (0)	44.0 (0)	39.0 (0)	54.2 (0)
WinoGrande	74.8 (5)	**77.3** (5)	72.5 (5)	77.0 (5)
PIQA	73.6 (0)	**80.6** (0)	79.1 (0)	-
			Mathematics & Science	
GSM8K	**70.3** (4)	48.7 (4)	13.1 (4)	80.2 (4)
Math	**31.4** (4)	16.0 (4)	4.3 (4)	43.5 (4)
GPQA	25.2 (5)	**25.9** (5)	25.7 (5)	30.8 (5)
			Code	
HumanEval	**35.4** (0)	34.8 (0)	12.8 (0)	51.2 (0)
HumanEval-FIM	**73.8** (2)	73.3 (2)	26.9 (2)	-
MBPP	40.0 (4)	**48.8** (4)	23.2 (4)	64.2 (0)
			Chinese	
CMMLU	**69.9** (5)	50.7 (5)	32.5 (5)	83.9 (5)
C-Eval	**70.5** (5)	51.7 (5)	34.0 (5)	83.2 (5)

Visualizing the Sampling Process

- We can visualize how many timesteps it takes for the model to generate each token in its response

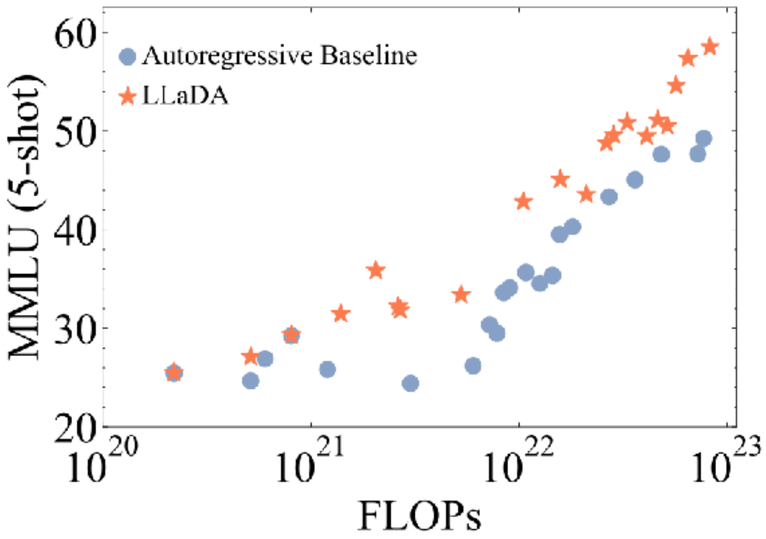


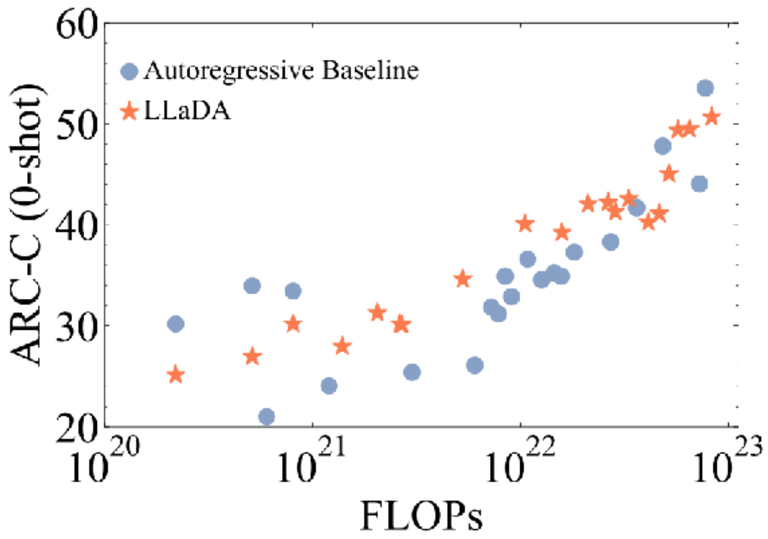
- Lighter colors are earlier predictions

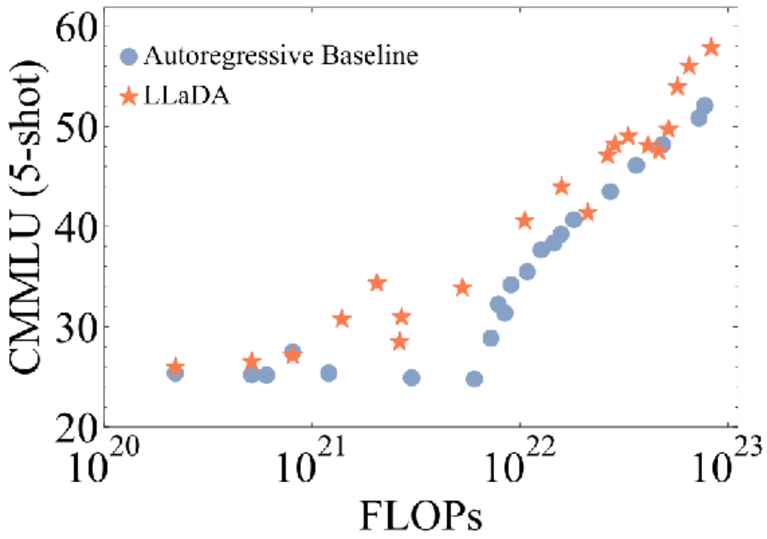
Nie et al. [2025]

Scaling

Nie et al. [2025]







- Scaling relative to Transformers looks decent! Not uniformly better across all tasks, though
- FLOPs is still a large determinant of performance

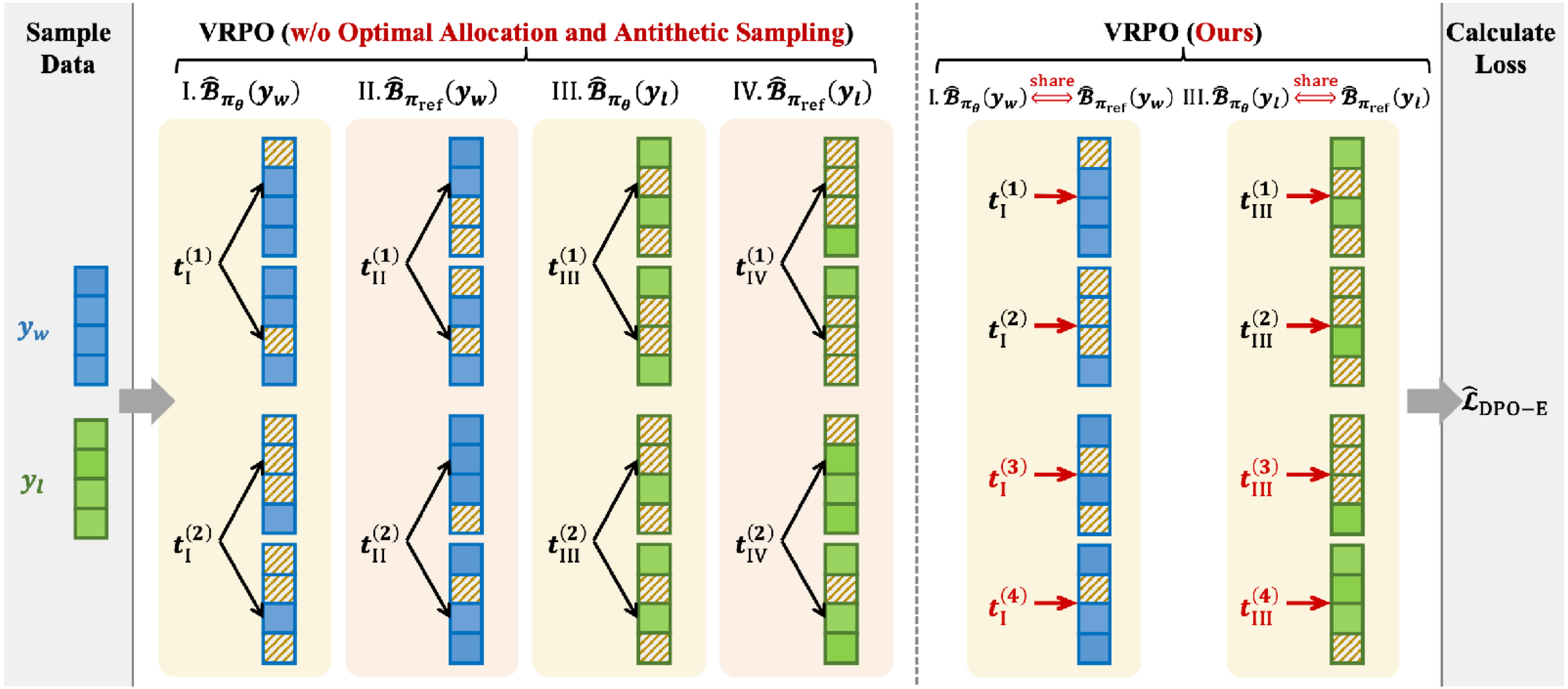
Post-training Diffusion LMs

- **Zhu et al. [2025]** propose Variance-reduced Preference Optimization (VRPO), which is similar to DPO
- DPO requires probabilities from a policy and frozen reference model. In a diffusion model, there is no tractable exact likelihood
- Thus, we can swap probabilities for ELBOs estimated via Monte Carlo sampling:

$$\widehat{\ell}_{\mathrm{DPO-E}}(y_w, y_l; \theta) \triangleq -\log \sigma \left(\beta \left(\widehat{\mathcal{B}}_{\pi_\theta}(y_w) - \widehat{\mathcal{B}}_{\pi_{\mathrm{ref}}}(y_w) \right) - \beta \left(\widehat{\mathcal{B}}_{\pi_\theta}(y_l) - \widehat{\mathcal{B}}_{\pi_{\mathrm{ref}}}(y_l) \right) \right)$$

- We can also use a couple extra tricks to reduce the variance of the ELBOs:
  - Optimal allocation: allocate sampling budget across timesteps to sample only one masked item per timestep
  - Antithetic sampling: Share Monte Carlo samples between policy and reference for same response

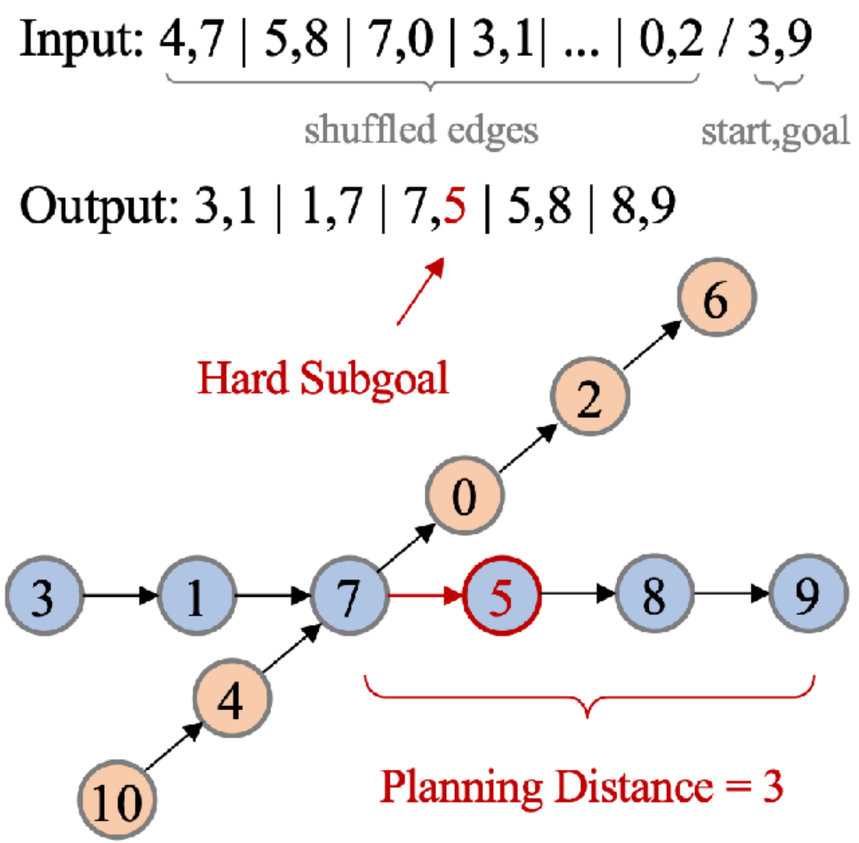
Post-training Diffusion LMs



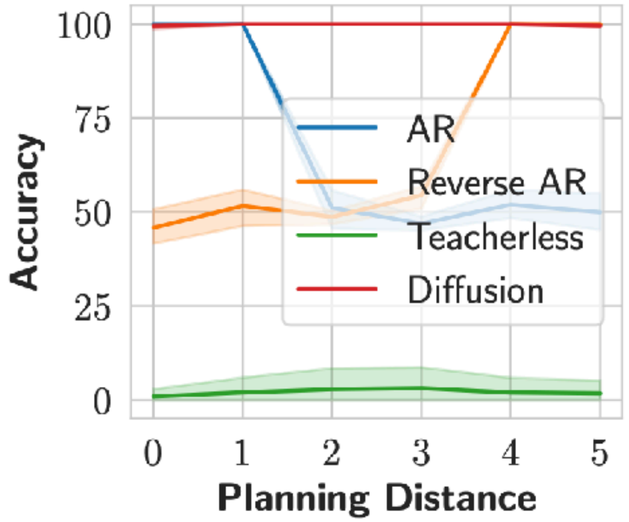
Zhu et al. [2025]

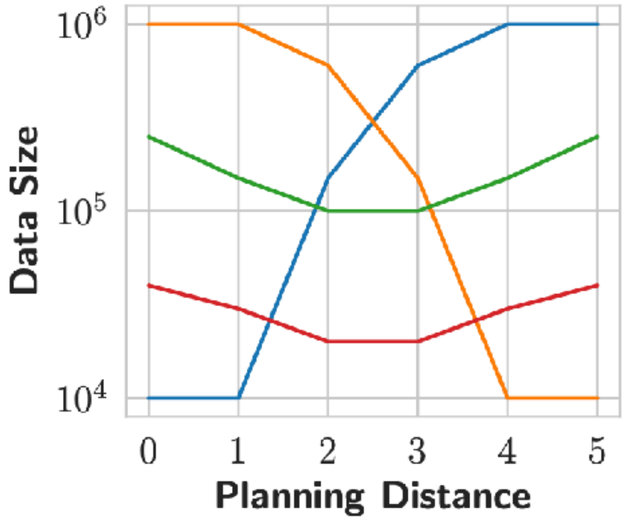
Strengths of Diffusion Models

- If we have **subgoal imbalance**, where some subgoals are significantly more difficult than others, autoregressive models can suffer



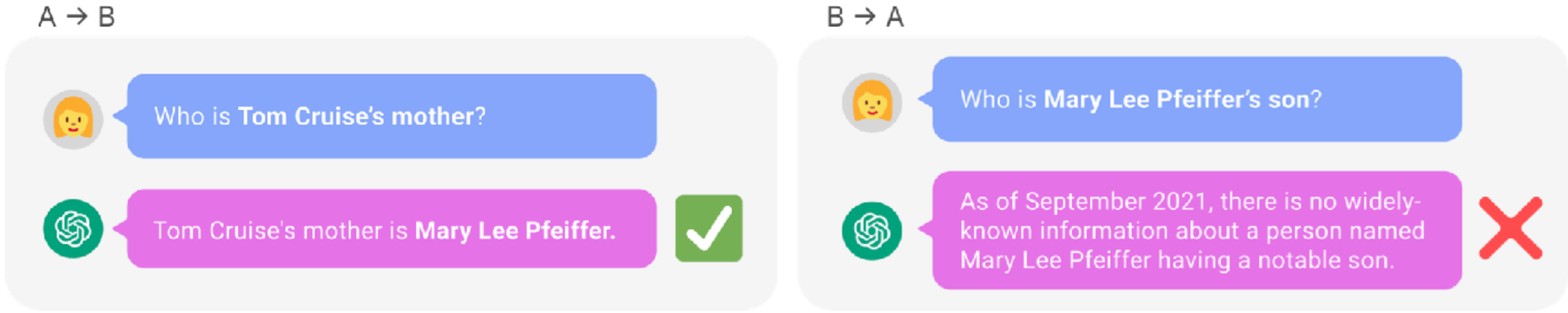
- Diffusion models are far better at tasks like these:





The Reversal Curse

LLMs trained on “A is B” fail to learn “B is A”



Jane Austen Novel Task

Inverse search: “In <Pride and Prejudice>, what’s the sentence ***before***: <sentence2>?”
Forward search: “In <Pride and Prejudice>, what’s the sentence ***after***: <sentence1>?”

	Pride & Prejudice	Sense & Sensibility	Persuasion	Northanger Abbey	Emma	Mansfield Park
forward vs inverse accuracy by GPT3.5	0.5% vs 14.4%	0.3% vs 5.4%	0.07% vs 4.3%	0.6% vs 5.5%	0.8% vs 7.2%	0.7% vs 5.5%
forward vs inverse accuracy by GPT4	0.8% vs 65.9%	0.9% vs 40.2%	0.5% vs 33.9%	0.9% vs 41.0%	0.6% vs 42.7%	0.3% vs 31.7%

Wiki Bio Task

Inverse search: “what’s the full name of the celebrity born on <date> in <city> who is a <occupation>?” GPT3.5 acc = **23.9%** GPT4: **42%**
Forward search: “what's the birthday and year of <name> who is a <occupation> and was born in <city>?” GPT3.5 acc = **89.5%** GPT4: **99%**

Chinese Idiom Task

Given a common 4-letter Chinese idiom such as *指鹿为马*, mask out its i-th letter (for i=1,2,3, or 4) and let GPT fill out the missing letter.

Prompt 1: 成语“X鹿为马”的X是什么字? GPT3.5 accuracy **9.4%**, GPT4 accuracy **17.6%**
Prompt 2: 成语“指X为马"的X是什么字? GPT3.5 accuracy **29.5%**, GPT4 accuracy **36.1%**
Prompt 3: 成语"指鹿X马"的X是什么字? GPT3.5 accuracy **32.0%**, GPT4 accuracy **76.7%**
Prompt 4: 成语"指鹿为X"的X是什么字? GPT3.5 accuracy **56.7%**, GPT4 accuracy **90.6%**

Chinese Poem Task

Given a famous two-sentence Chinese poem such as *劝君更尽一杯酒, 西出阳关无故人*, let GPT answer what’s the sentence ***before***/***after*** <sentence2/1>

Inverse search: “西出阳关无故人”的上一句是什么? GPT3.5 accuracy **2.1%**, GPT4 accuracy **7.3%**
Forward search: “劝君更尽一杯酒”的下一句是什么? GPT3.5 accuracy **33.0%**, GPT4 accuracy **66.5%**

Diffusion Models Better Mitigate The Reversal Curse?

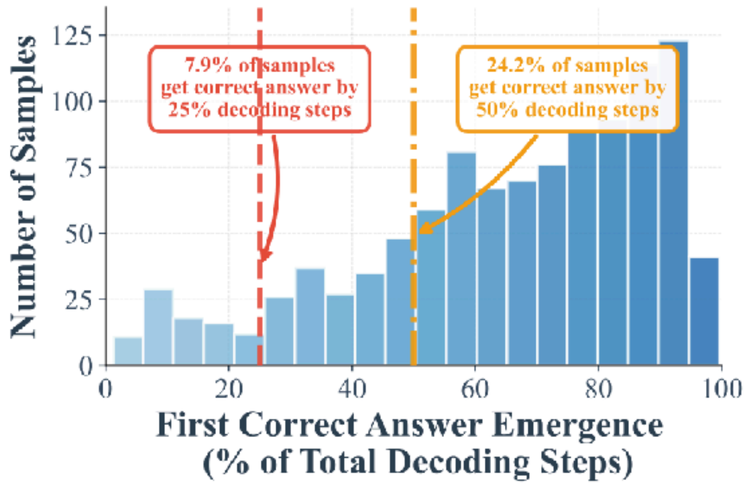
	Forward	Reversal
GPT-4o (2024-08-06)	**82.7**	34.3
Qwen2.5-7B Instruct	75.9	38.0
LLaDA-8B Instruct	51.8	**45.6**

Nie et al. [2025]

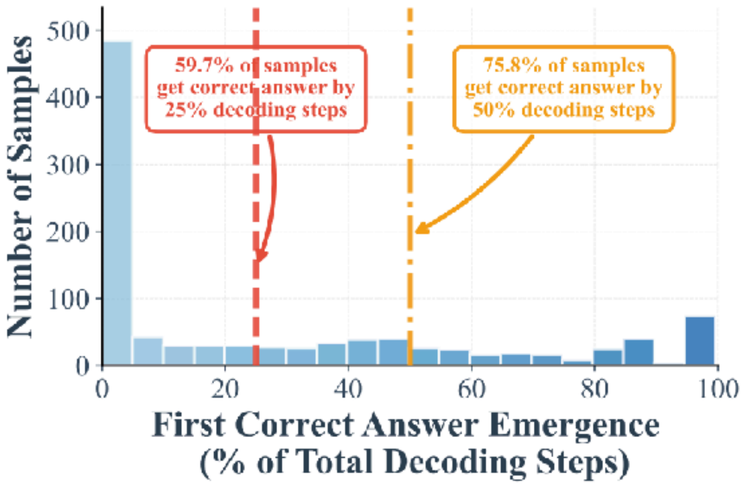
- Diffusion models tend to perform more similarly in forward and reverse directions
- Although you will notice that overall performance is not as good as autoregressive models in this cases

Diffusion Models Often Know the Answer Early

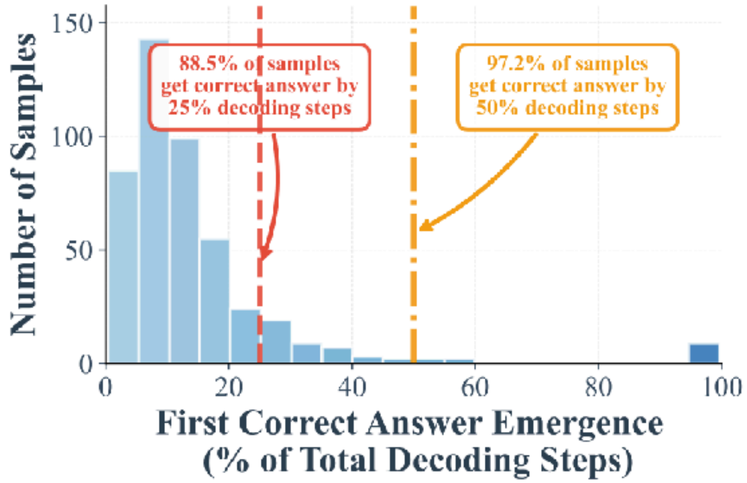
- The naive generation regime makes it seem like computing the answer requires many timesteps
- However, using low-confidence remasking OR appending “Answer: “ as a suffix to the prompt leads to far earlier answers
- We can often see the correct answer at less than halfway to the final decoding step



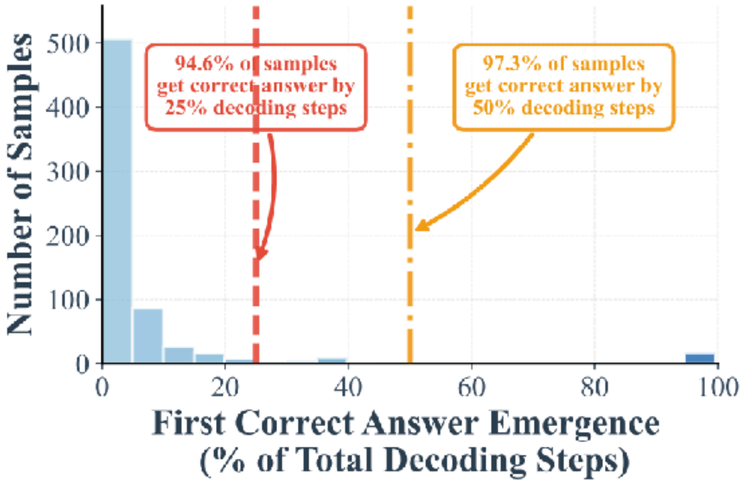
(a) w/o suffix prompt (low-confidence remasking)



(b) w/ suffix prompt (low-confidence remasking)



(c) w/o suffix prompt (random remasking)



(d) w/ suffix prompt (random remasking)

Diffusion LMs Enable Controllable Generation

- Because they refine all tokens globally and simultaneously, diffusion models are especially good at producing outputs with some desired property
- For example, we can use a classifier $p(y \mid x)$ to guide generation and yield a sample from the distribution $p(x \mid y) \propto p(y \mid x)p(x)$
  - For each step, we maximize $p(x_{t-1} \mid x_t)$ *and* $p(y \mid x_{t-1})$

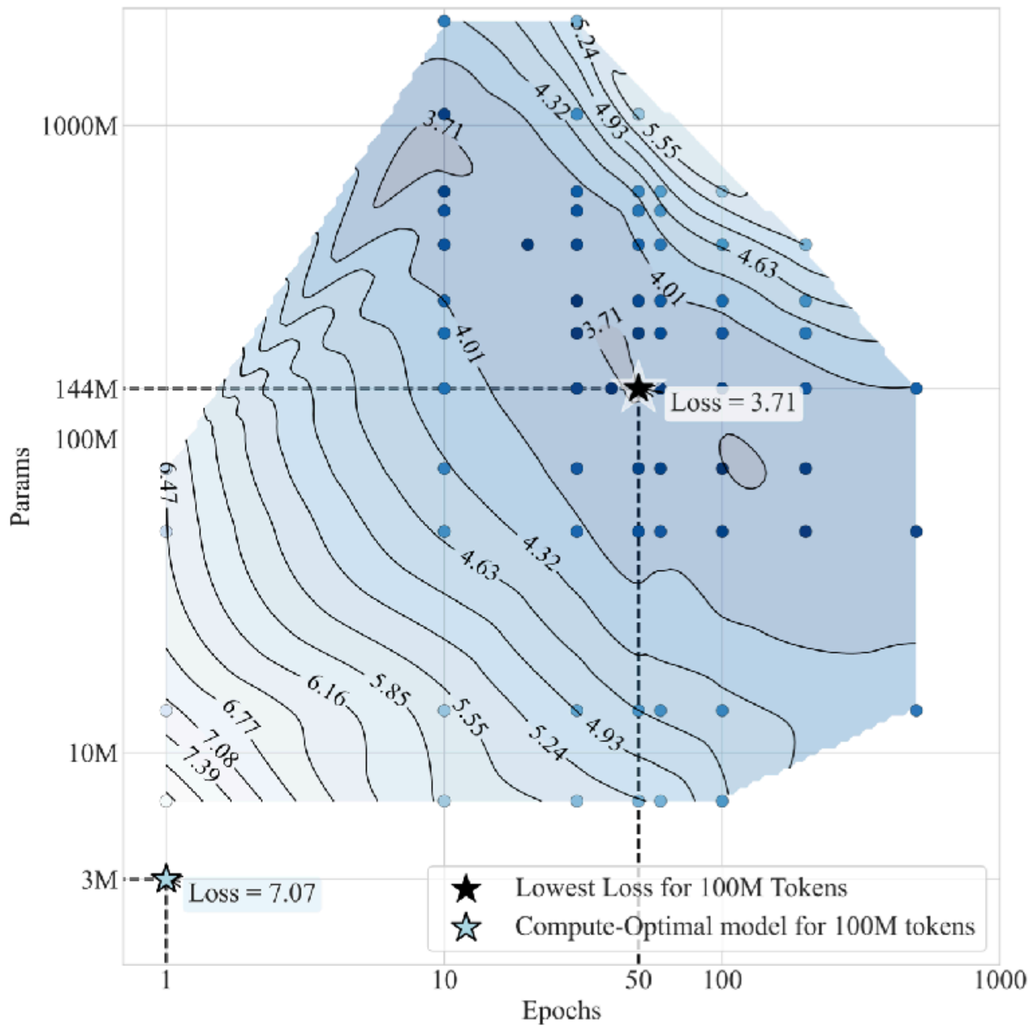
Diffusion step Classifier score

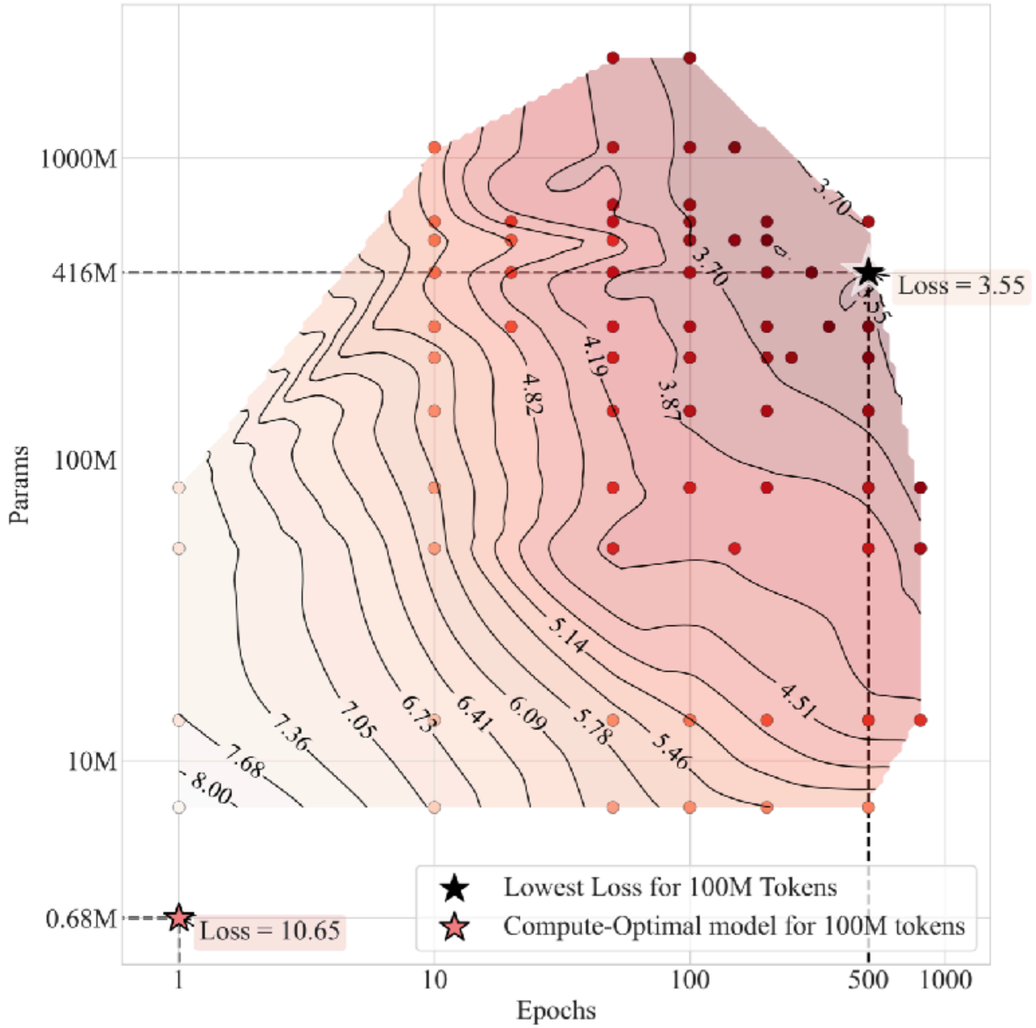
Li et al. [2022]

input (Semantic Content) output text	food : Japanese Browns Cambridge is good for Japanese food and also children friendly near The Sorrento .
input (Parts-of-speech) output text	PROPN AUX DET ADJ NOUN NOUN VERB ADP DET NOUN ADP DET NOUN PUNCT Zizzi is a local coffee shop located on the outskirts of the city .
input (Syntax Tree) output text	(TOP (S (NP (*) (*) (*)) (VP (*) (NP (NP (*) (*))))) The Twenty Two has great food
input (Syntax Spans) output text	(7, 10, VP) Wildwood pub serves multicultural dishes and is ranked 3 stars
input (Length) output text	14 Browns Cambridge offers Japanese food located near The Sorrento in the city centre .

Data Efficiency

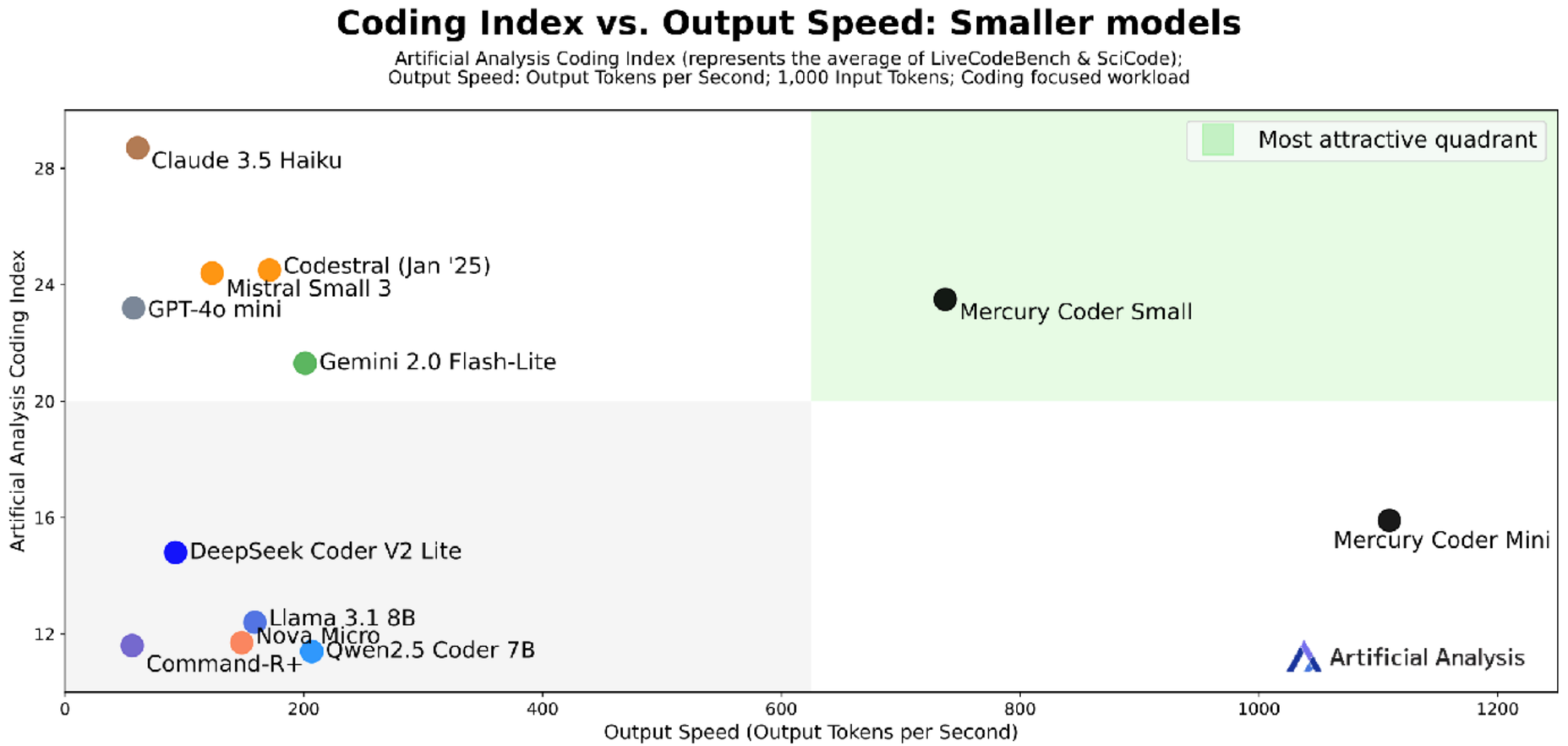
Prabhudesai et al. [2025]





Autoregression is best on the first pass through the data, but in data-limited settings, diffusion often beats autoregression given enough compute.

Token Throughput



- Note that parallel token generation doesn’t automatically make diffusion LMs faster than autoregressive! This is partially a function of Mercury’s specific design and API.

Drawbacks of Diffusion LMs

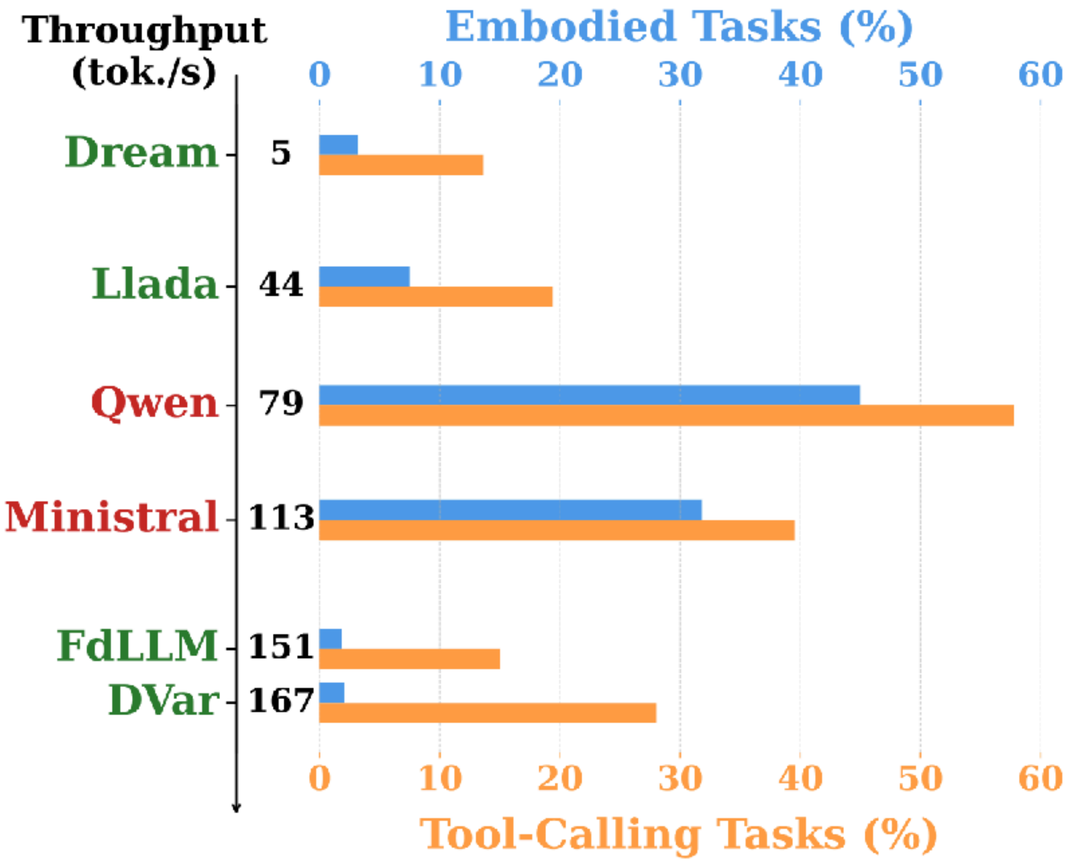
- Diffusion LMs are not necessarily more efficient than autoregressive models!
  - Can require more sampling steps during decoding to get comparable performance to autoregressive models
  - More sampling steps are generally needed for multi-step reasoning tasks
- Keys and values can drift across denoising steps, which makes standard KV caching tricky
- Performance depends heavily on sequence length and block structure

Diffusion Models Are Bad Foundations for Agentic Systems

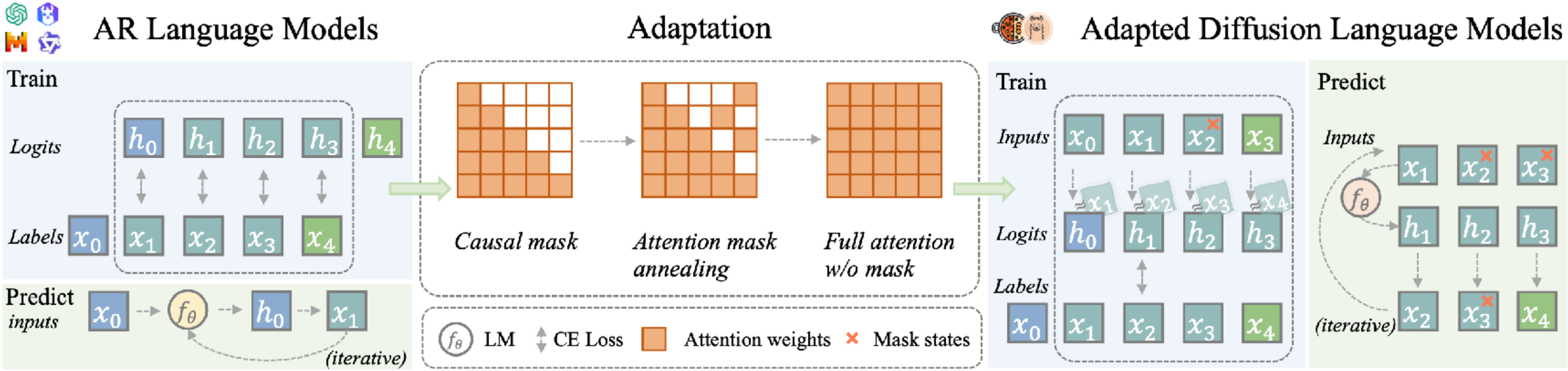
- Diffusion models often obtain far worse performance on agentic tasks than autoregressive models

- They also do not necessarily have higher throughput!

Lu et al. [2026]



Converting Autoregressive Models into Diffusion Models



- We can gradually remove the causal mask during training, which makes the model bidirectional
- Then, we can shift the logits to compute the loss with the next token; the loss for h_i is w.r.t. x_{i+1}

Summary

- Diffusion language models: an interesting alternative to autoregressive models
- Converts noisy sequences into clean ones, often by denoising many tokens at once
- Not strictly better or worse than previous architectures