EntRGi: Entropy-Aware Reward Guidance for Diffusion Language Models

Atula Tejaswi

Litu Rout

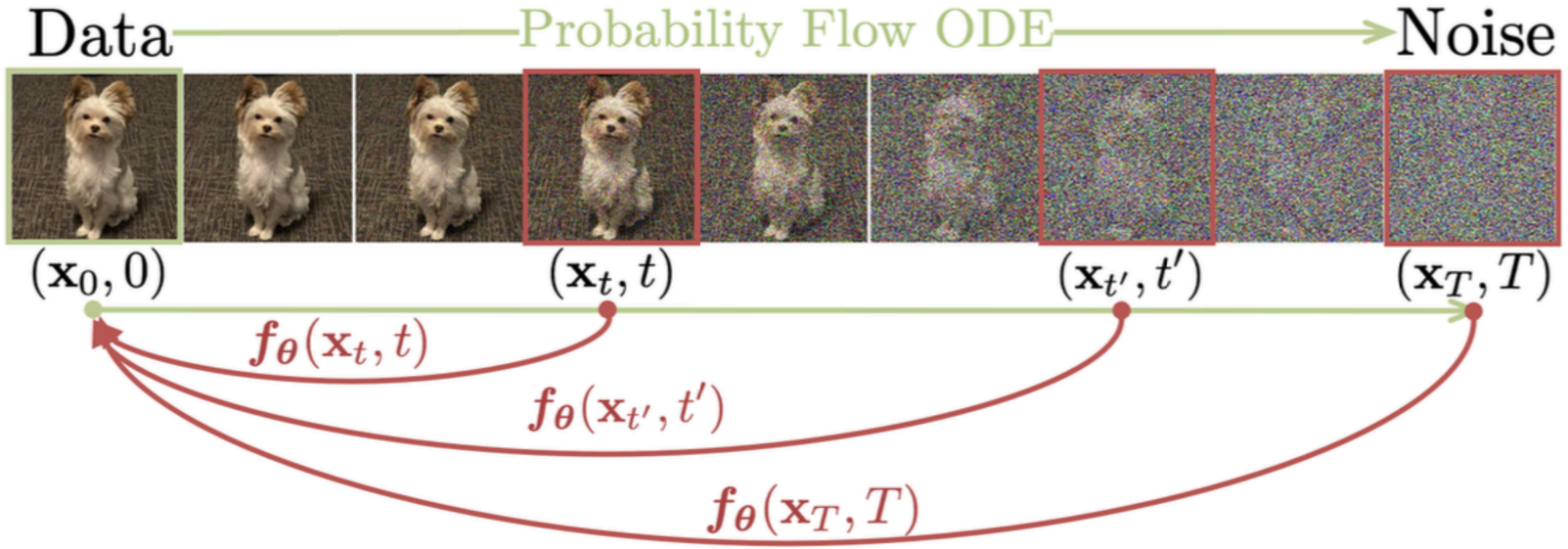
Constantine Caramanis

Sanjay Shakkottai

Sujay Sanghavi

TEXAS
The University of Texas at Austin

Diffusion Models



- Start with a clean Image
- Add noise at various ‘levels’
- Train a network to reconstruct the original image
- Inference - Start from noise and run the network with time ‘backwards’

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

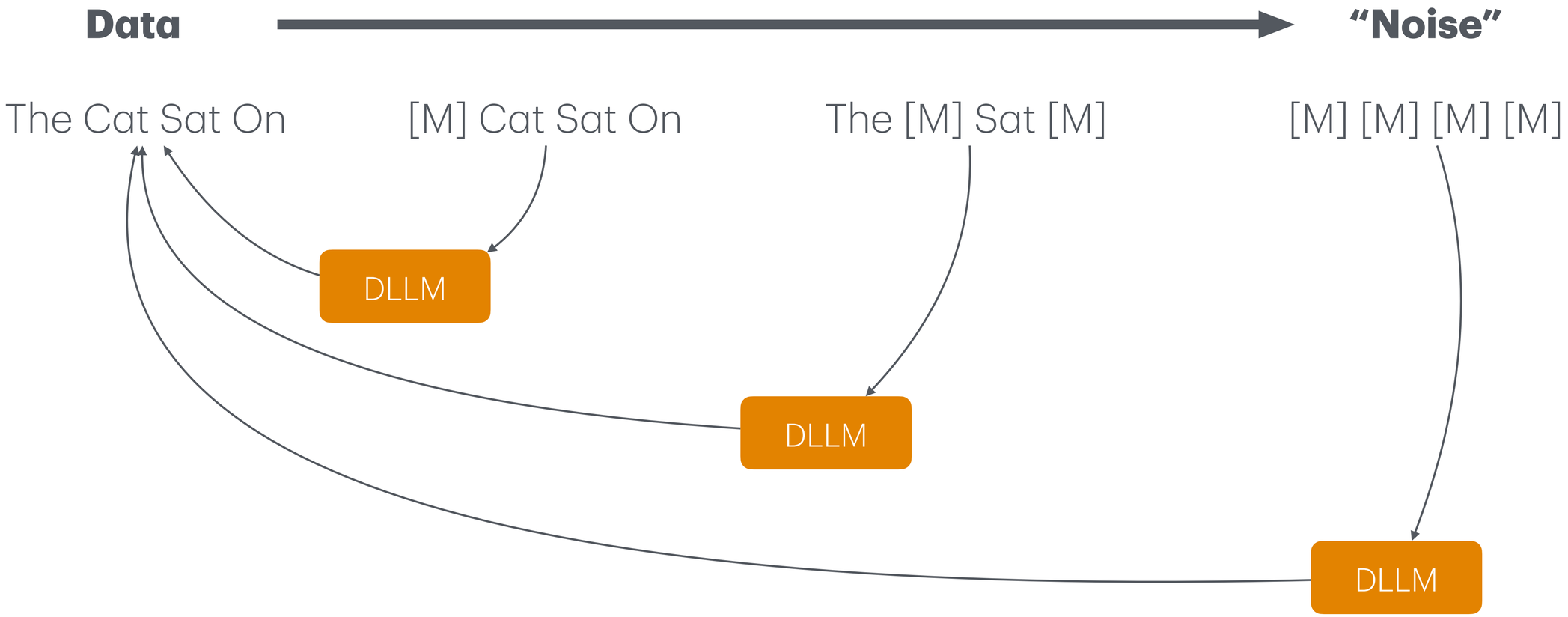
(Discrete) Diffusion Language Models

- Almost the same as continuous/image diffusion
- Follows an “Absorbing State Markov chain” — where the Noise is just a sequence of mask tokens
- Data is sequences (tokens) of text

(Masked) Diffusion Language Models

- Follows an “Absorbing State Markov chain” — where the Noise is just a sequence of mask tokens
- Data is sequences (tokens) of text

(Masked) Diffusion Language Models



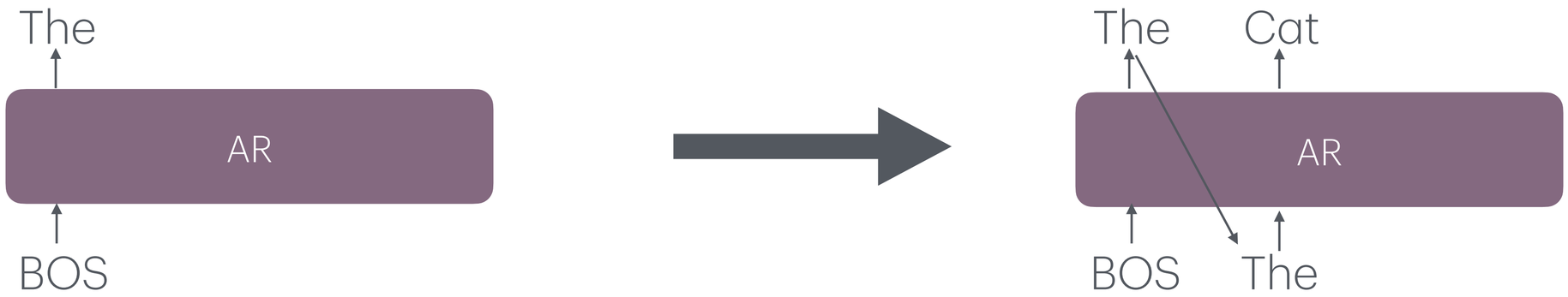
- Follows an “Absorbing State Markov chain” — where the Noise is just a sequence of mask tokens
- Data is sequences (tokens) of text

Autoregressive vs Masked Diffusion LLMs

AR Generates Left to Right

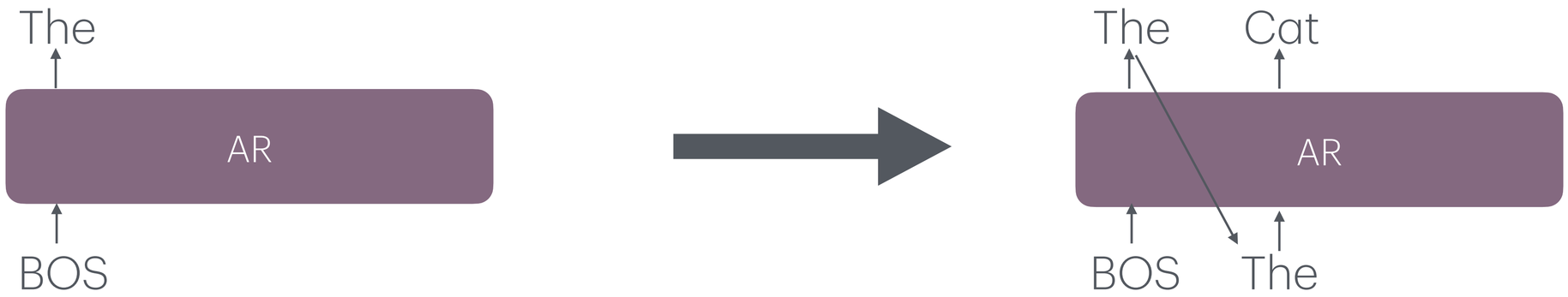
Autoregressive vs Masked Diffusion LLMs

AR Generates Left to Right

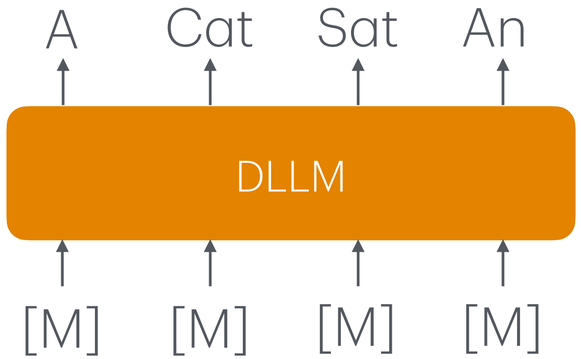


Autoregressive vs Masked Diffusion LLMs

AR Generates Left to Right

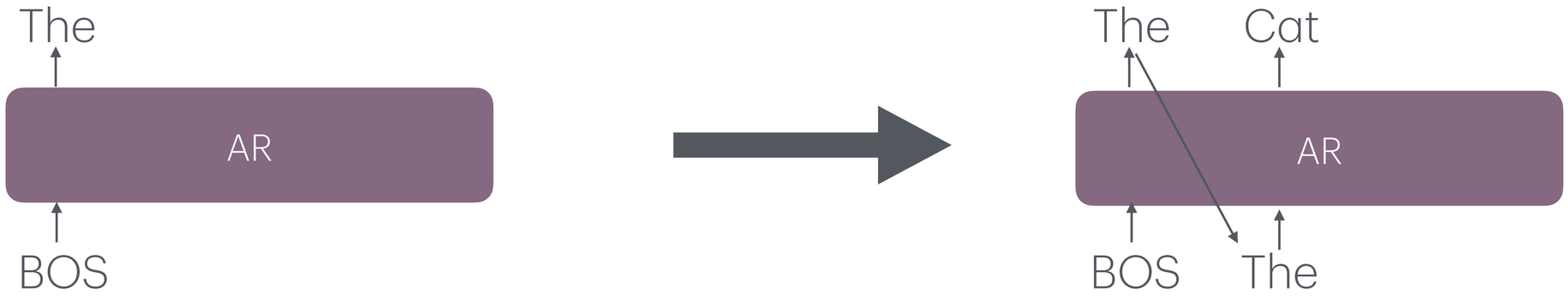


Discrete Diffusion (Sahoo et. al.) starts from mask tokens and iteratively selects tokens in an arbitrary/selected order



Autoregressive vs Masked Diffusion LLMs

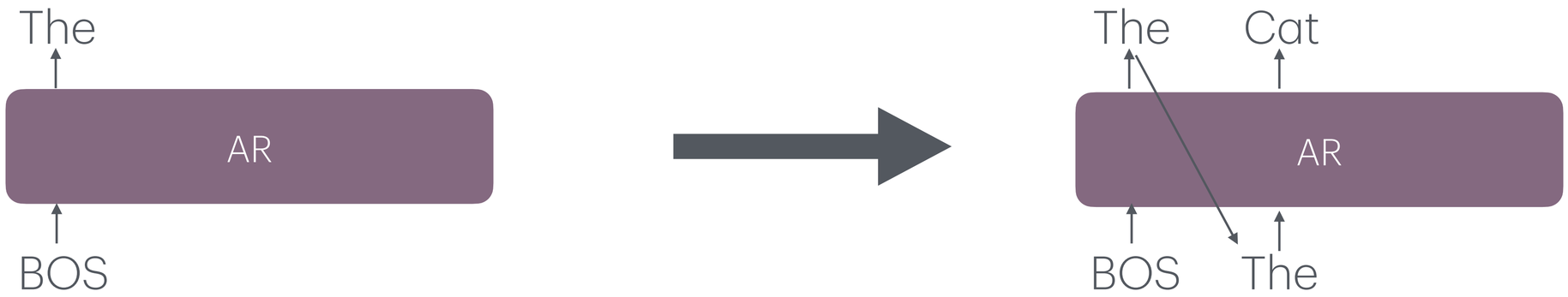
AR Generates Left to Right



Discrete Diffusion (Sahoo et. al.) starts from mask tokens and iteratively selects tokens in an arbitrary/selected order

Autoregressive vs Masked Diffusion LLMs

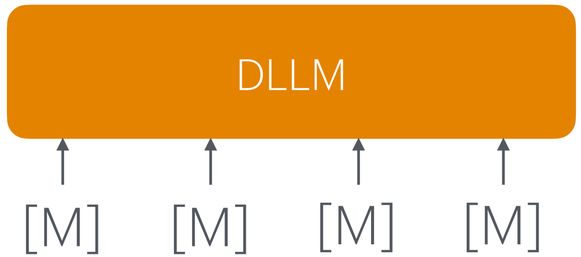
AR Generates Left to Right



Discrete Diffusion (Sahoo et. al.) starts from mask tokens and iteratively selects tokens in an arbitrary/selected order

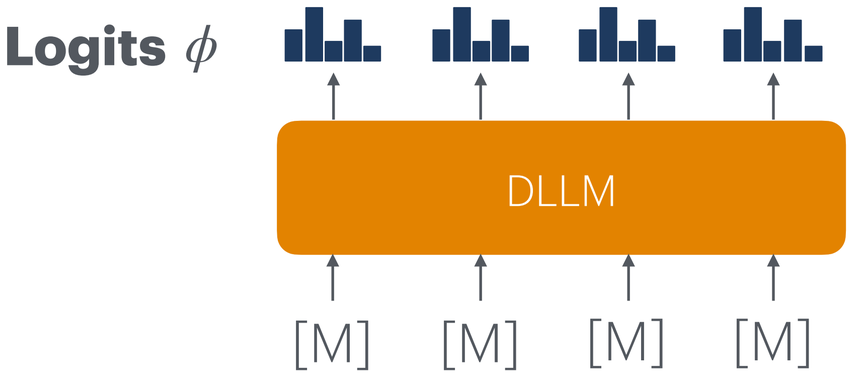
Autoregressive vs Masked Diffusion LLMs

DLLMs generate a probability distribution at ALL masked positions



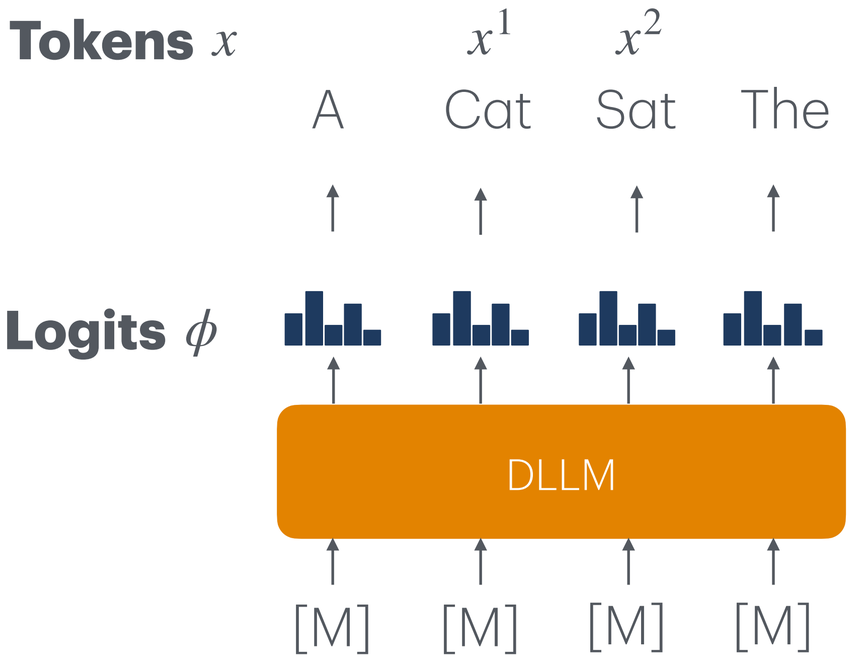
Autoregressive vs Masked Diffusion LLMs

DLLMs generate a probability distribution at ALL masked positions



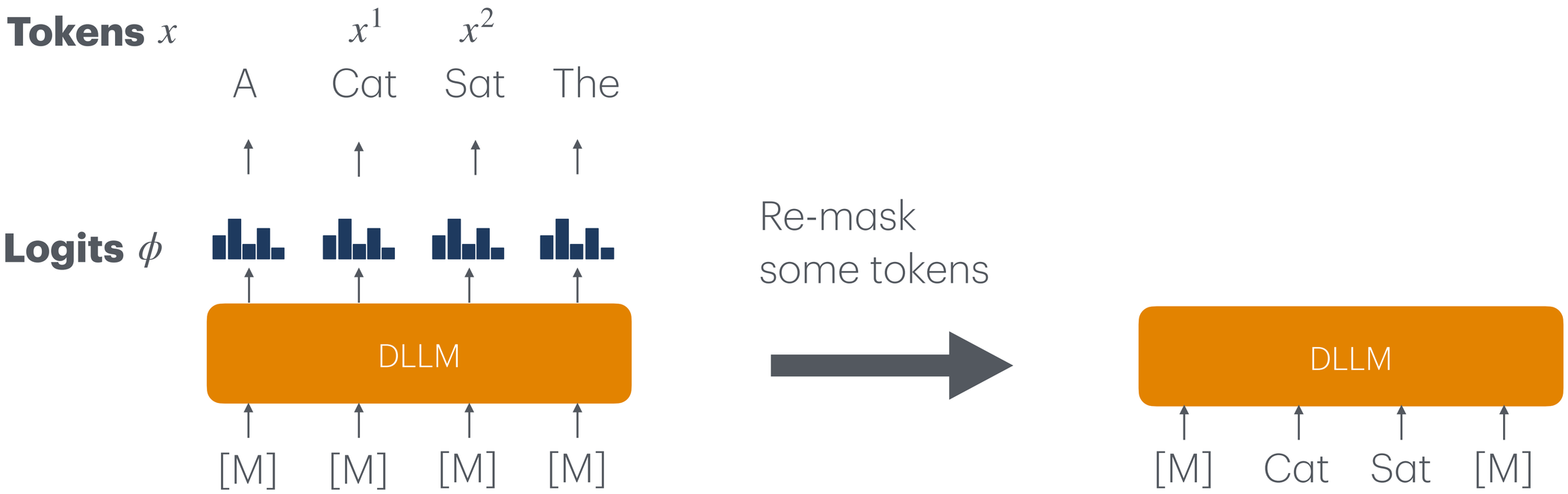
Autoregressive vs Masked Diffusion LLMs

DLLMs generate a probability distribution at ALL masked positions



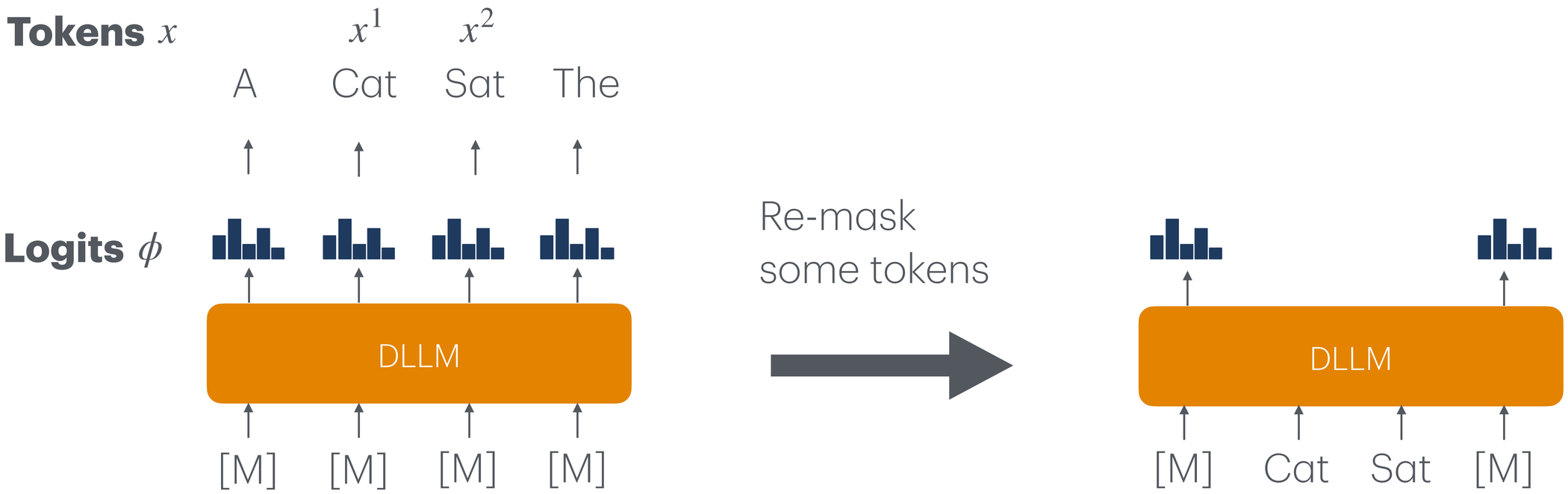
Autoregressive vs Masked Diffusion LLMs

DLLMs generate a probability distribution at ALL masked positions



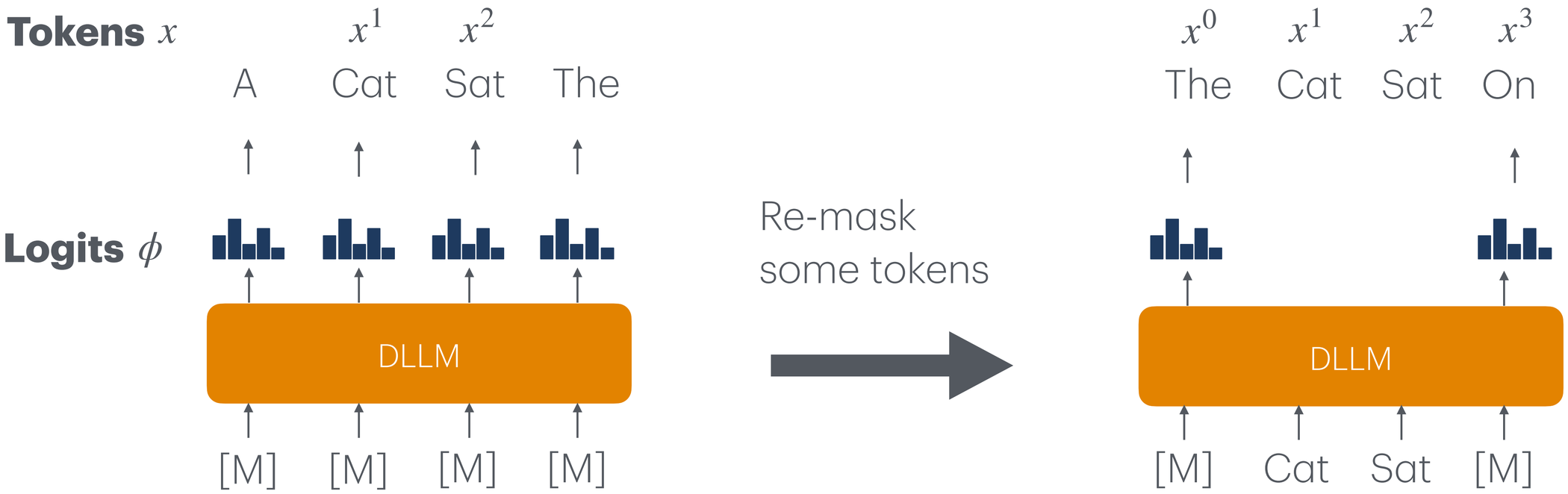
Autoregressive vs Masked Diffusion LLMs

DLLMs generate a probability distribution at ALL masked positions

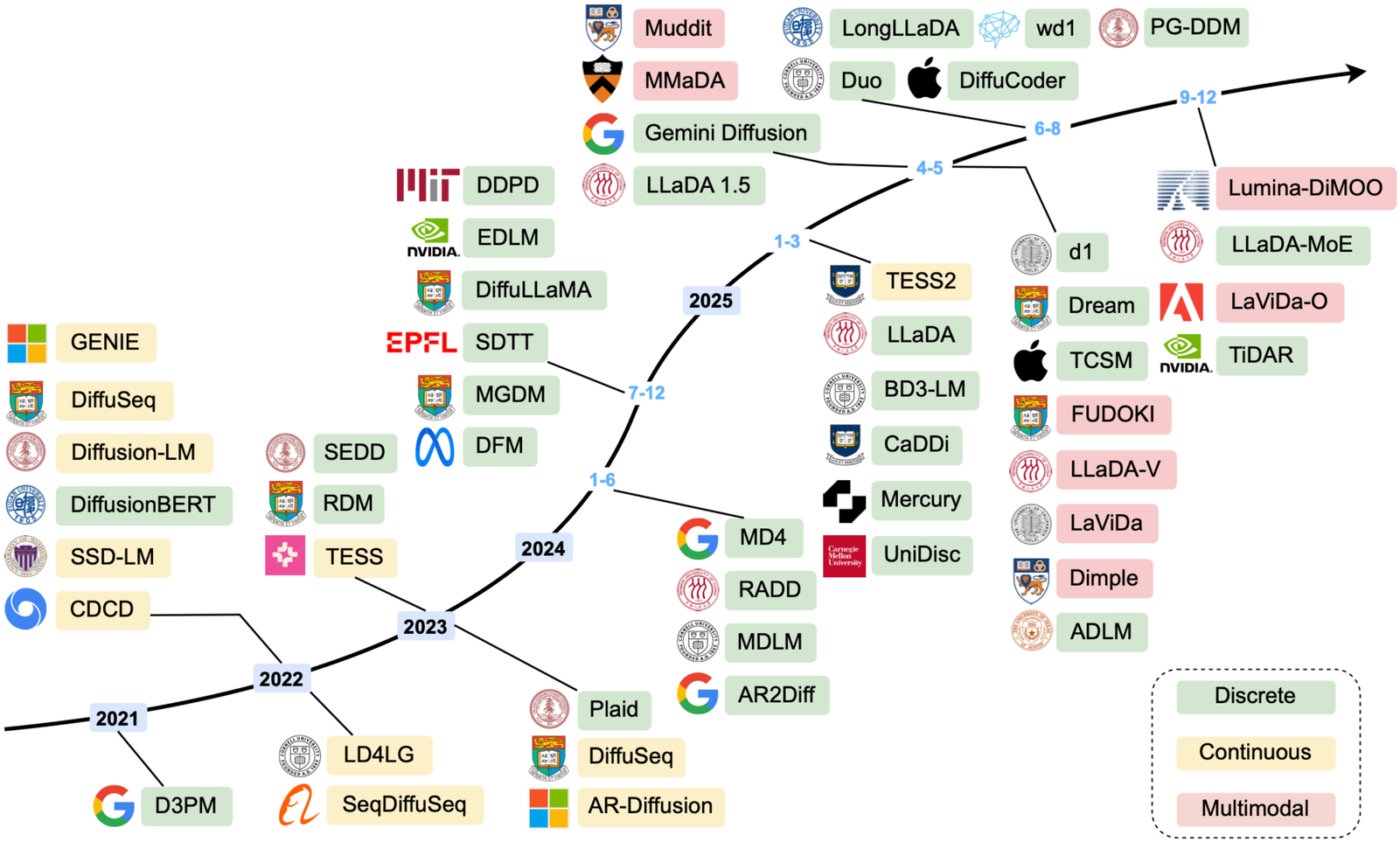


Autoregressive vs Masked Diffusion LLMs

DLLMs generate a probability distribution at ALL masked positions



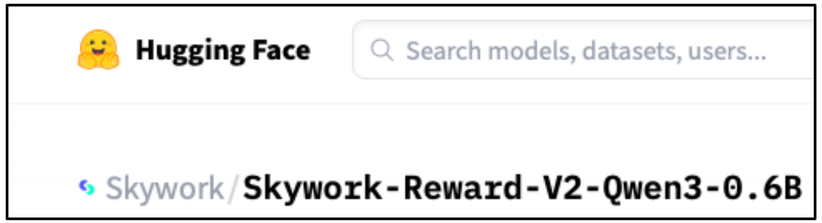
DLLMs are gaining widespread interest



Li et. al. 2025. A Survey on Diffusion Language Models

Reward models

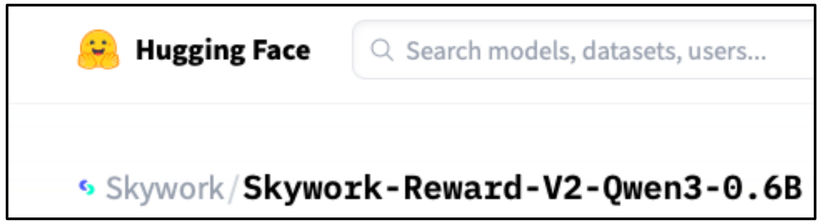
Trained and evaluated in certain domains eg: Instruction Following, Factuality



Trained with Preference Data

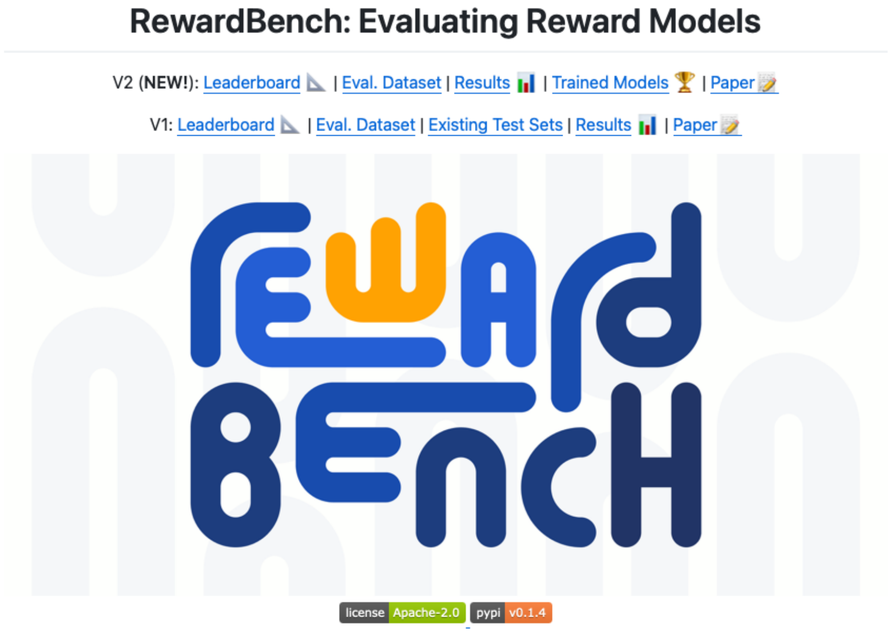
Reward models

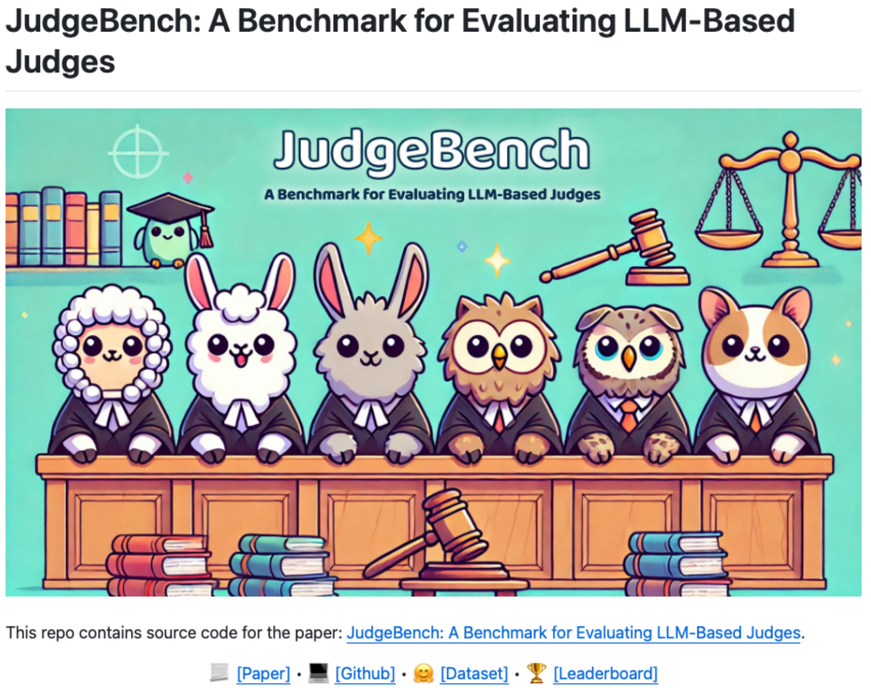
Trained and evaluated in certain domains eg: Instruction Following, Factuality



Trained with Preference Data

Evaluate the Reward Model





Goal: Generate high reward samples

According to a reward model R(x)

Goal: Generate high reward samples

According to a reward model R(x)

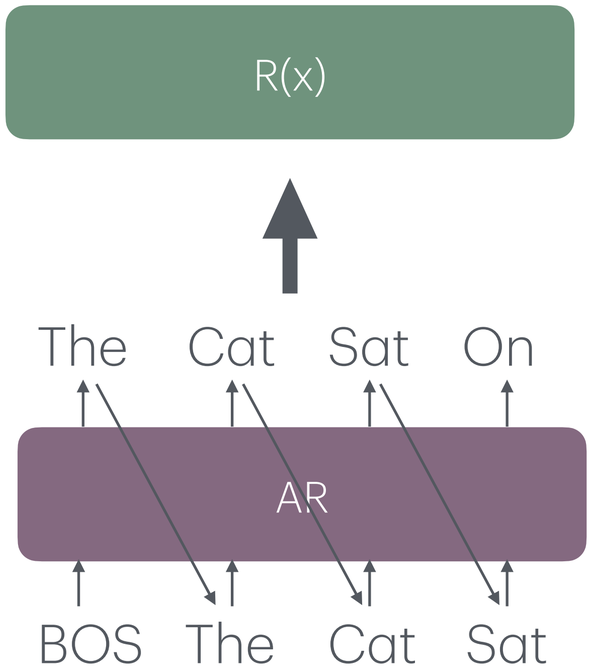
$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

Goal: Generate high reward samples

According to a reward model R(x)

$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

Typically R(x) is small-ish so you can make as many calls to R(x) as you want

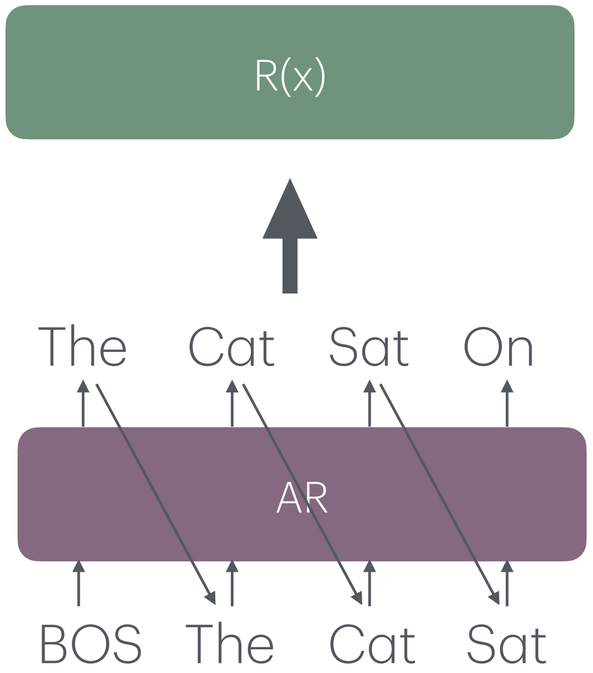


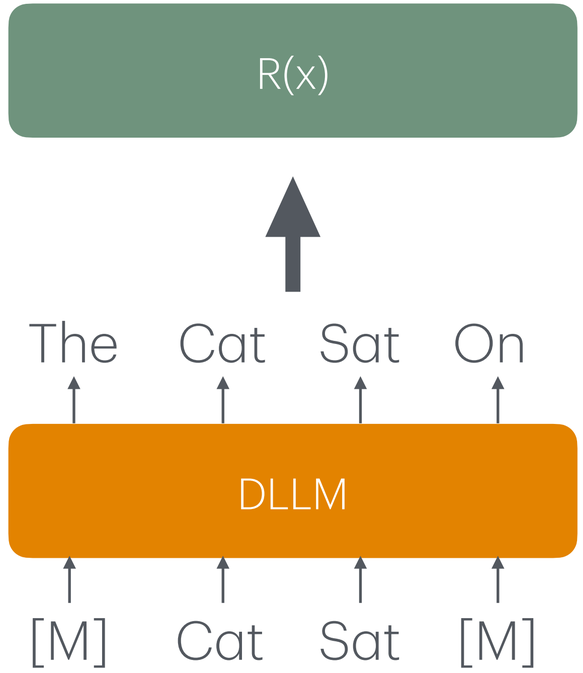
Goal: Generate high reward samples

According to a reward model R(x)

$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

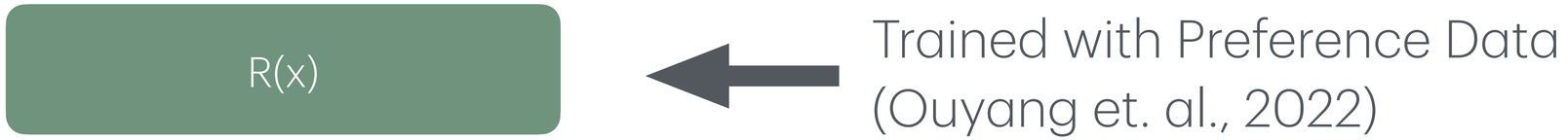
Typically R(x) is small-ish so you can make as many calls to R(x) as you want





Goal: Generate high reward samples

One way - we can update the parameters of the model

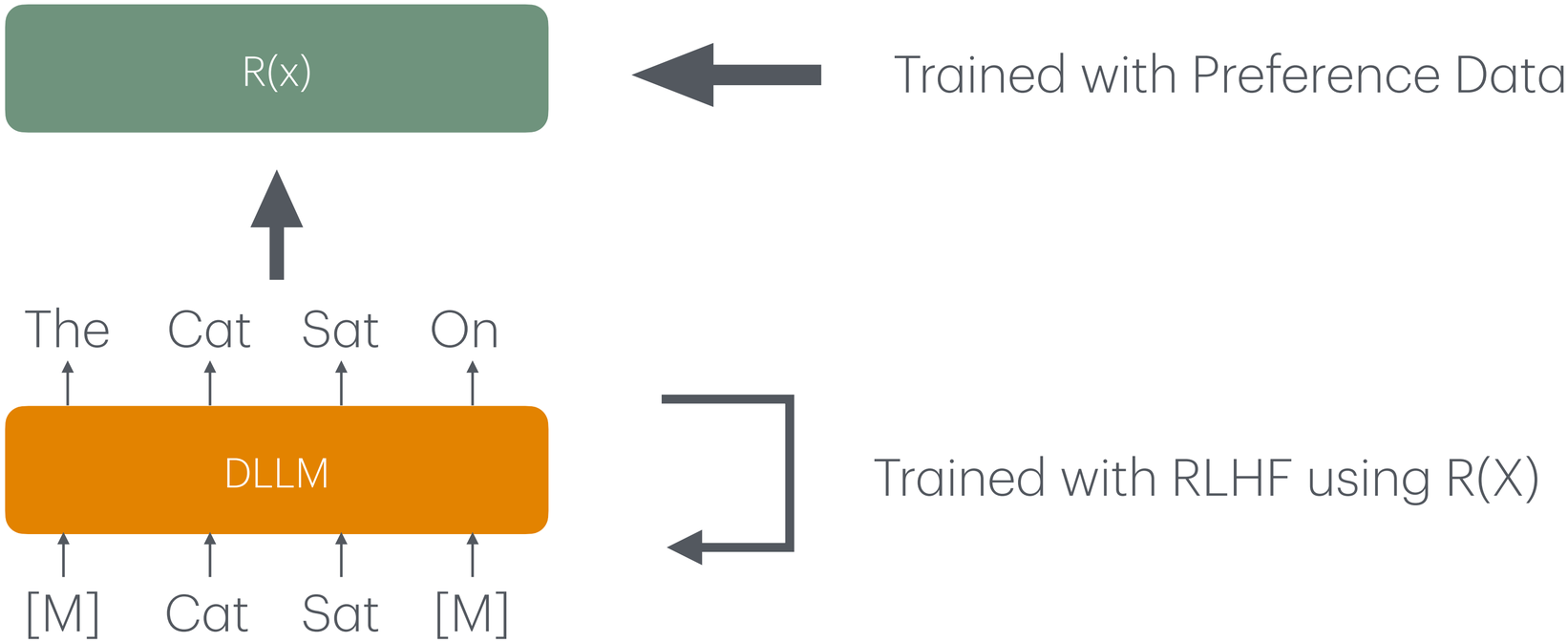


Goal: Generate high reward samples

One way - we can update the parameters of the model

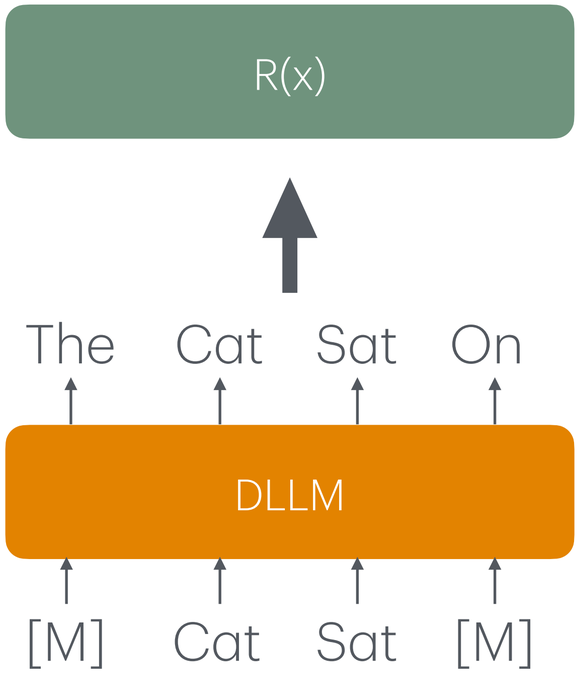
Goal: Generate high reward samples

One way - we can update the parameters of the model



Goal: Generate high reward samples

One way - we can update the parameters of the model



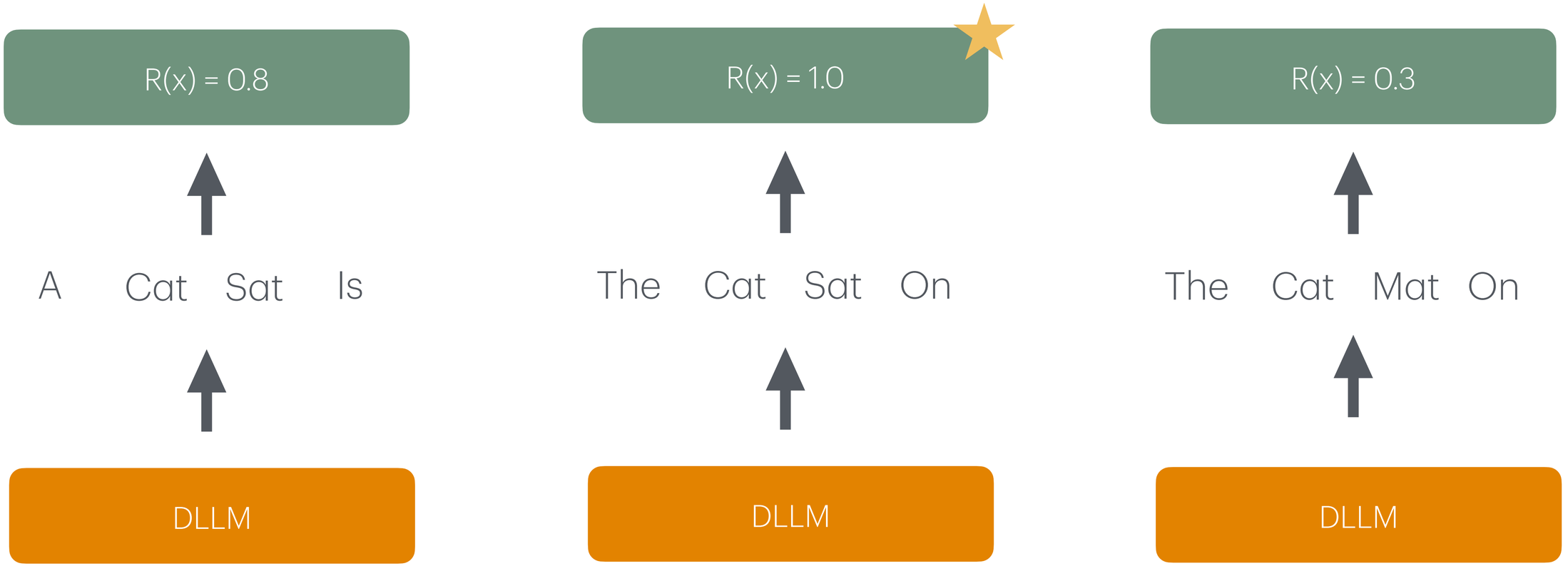
Trained with Preference Data

We don't want to modify weights!

Trained with RLHF using R(X)

Goal: Generate high reward samples

Best-of-N Sampling - Generate ’N’ and pick the best one



Goal: Generate high reward samples

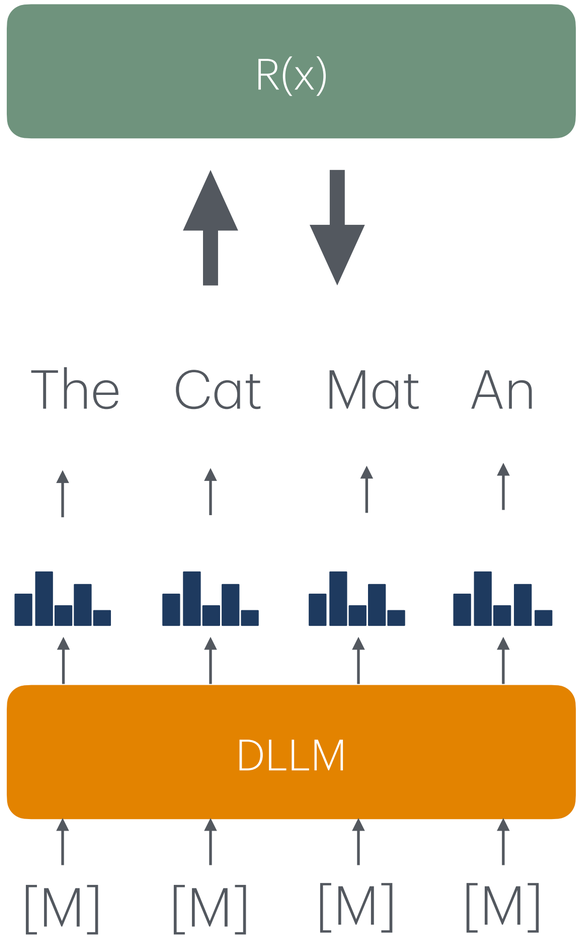
Recall - DLLMs produce a distribution at ALL positions

- With AR -> need to generate full string ‘x’ before assessing quality
- Or use Process Reward Models to assess intermediate completions

Goal: Generate high reward samples

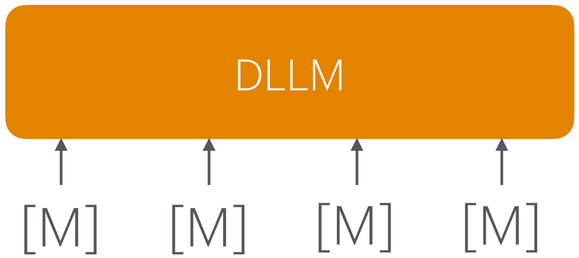
Recall - DLLMs produce a distribution at ALL positions

- With AR -> need to generate full string ‘x’ before assessing quality
- Or use Process Reward Models to assess intermediate completions
- With DLLMs -> you can sample (imperfect strings) at ALL stages
- Can we use this observation to update logits at each step?



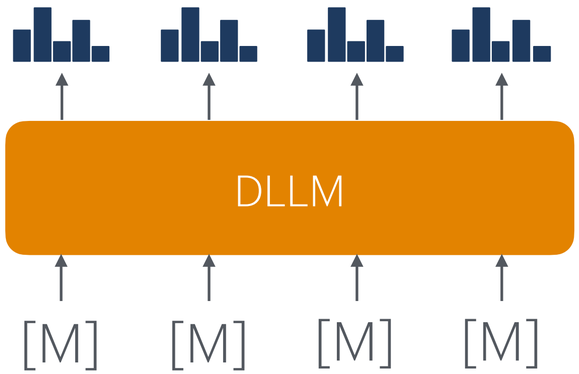
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



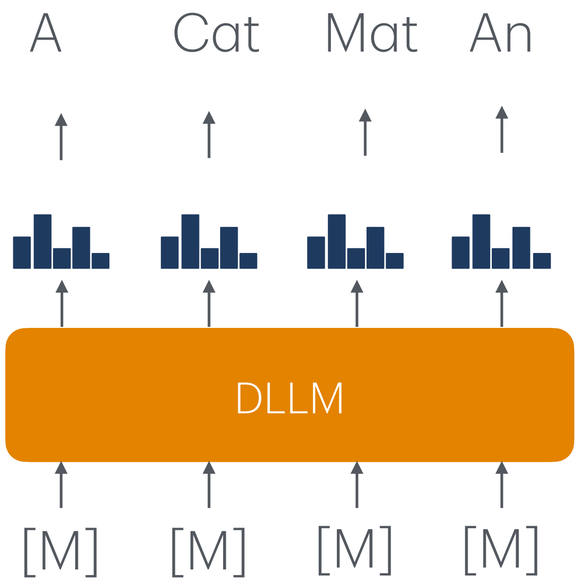
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



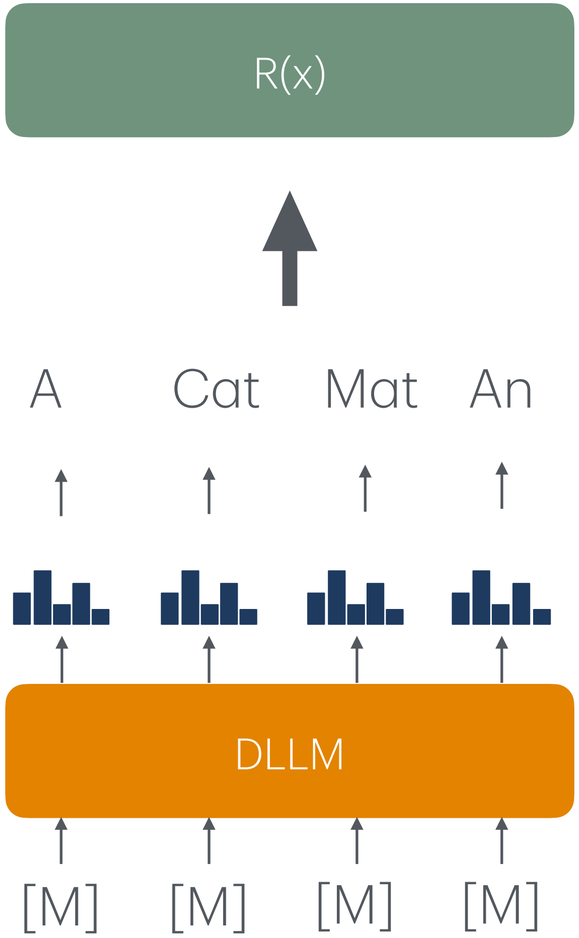
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



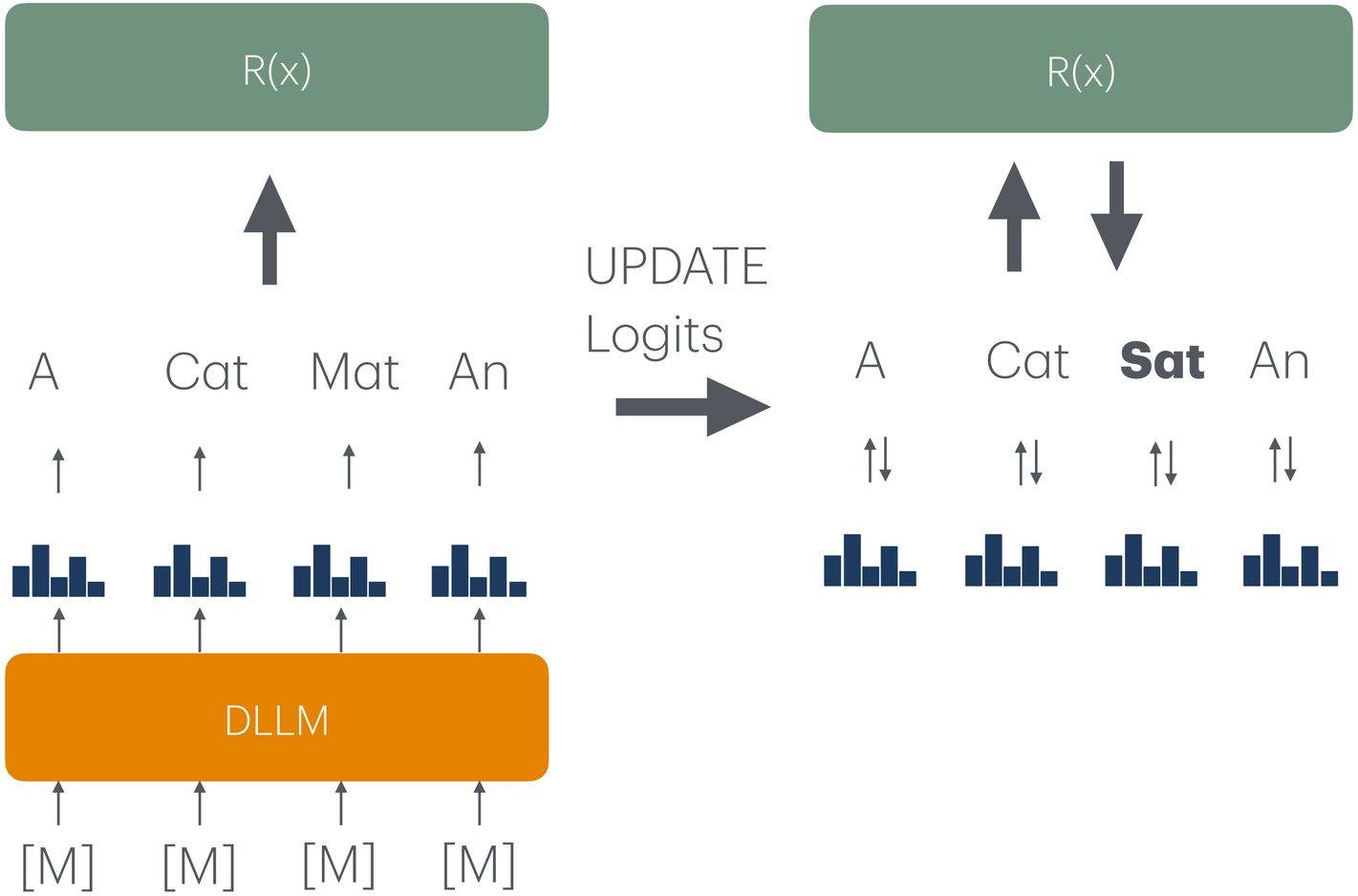
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



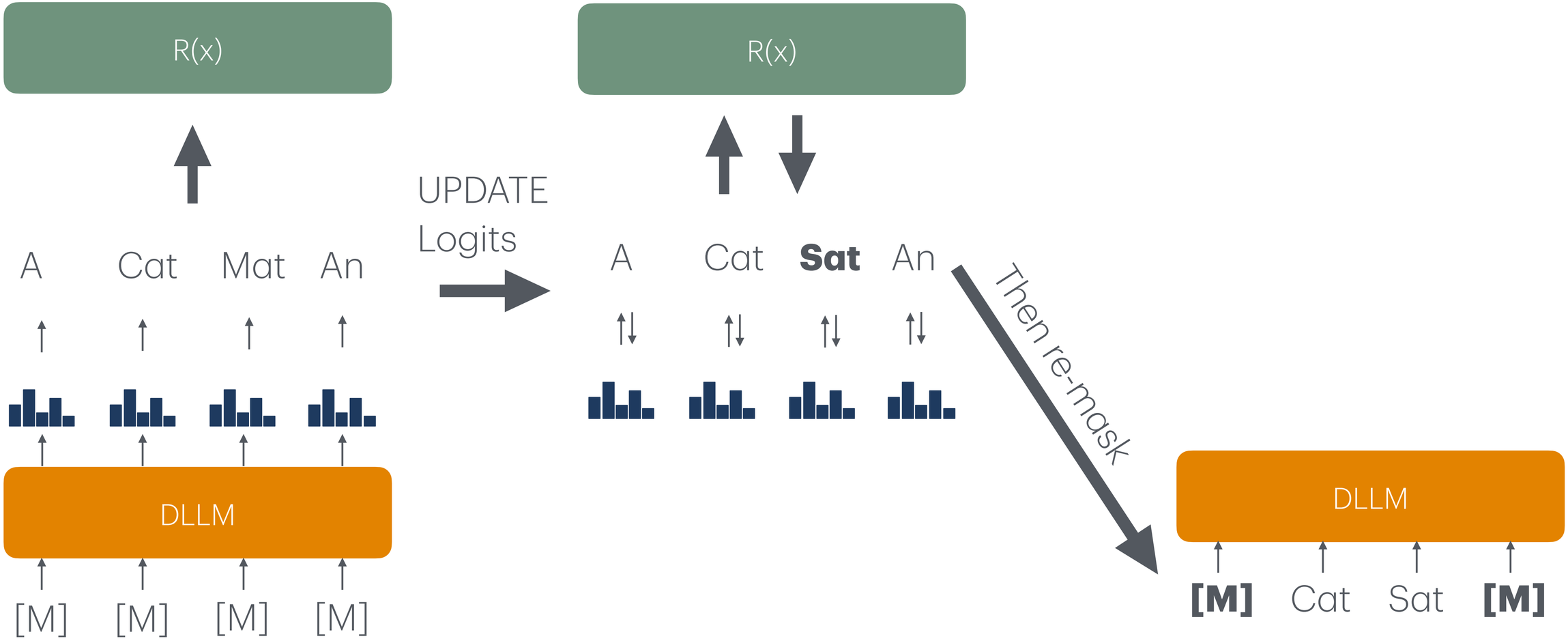
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



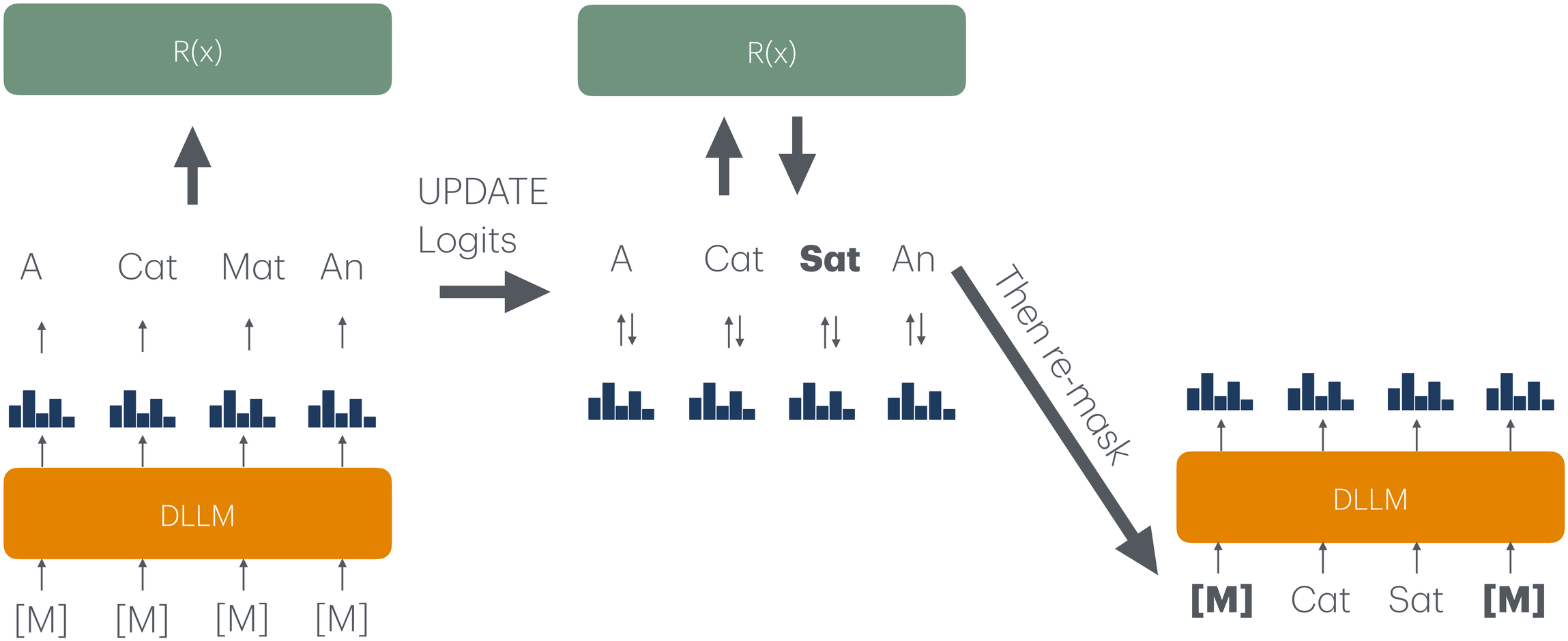
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



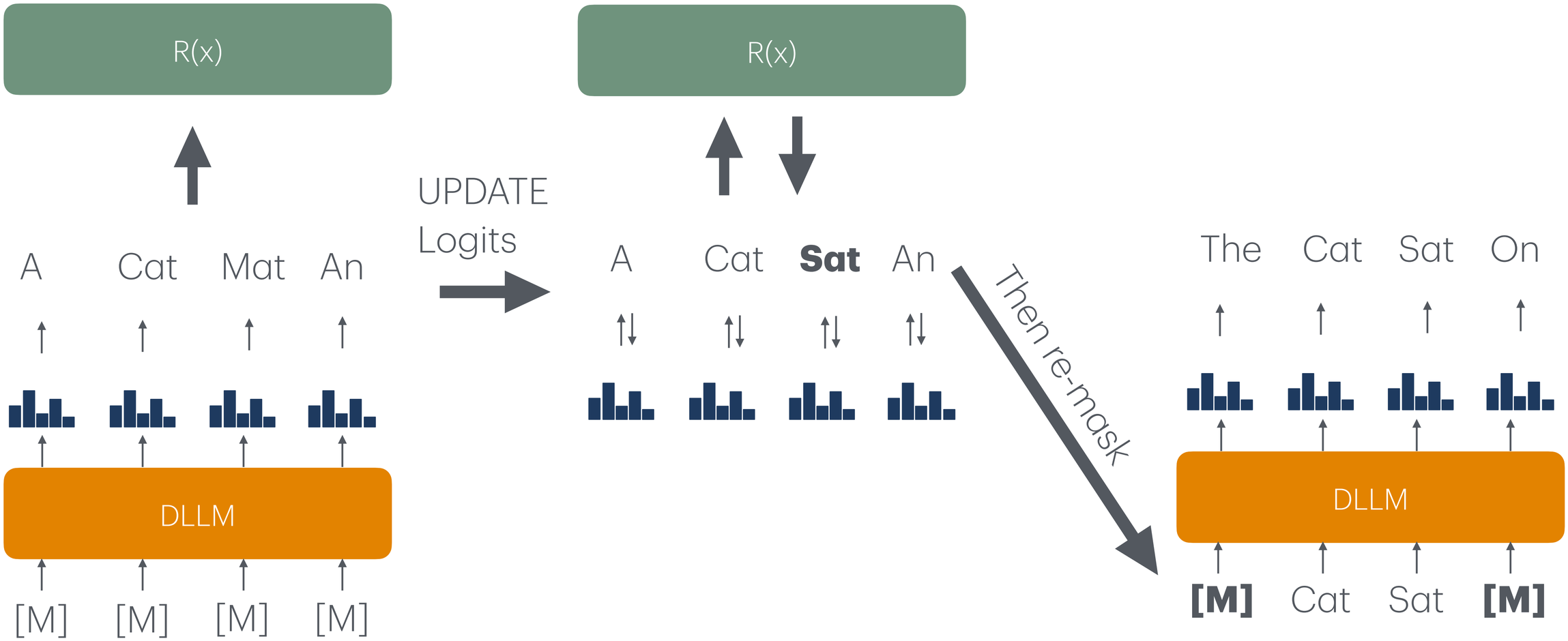
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



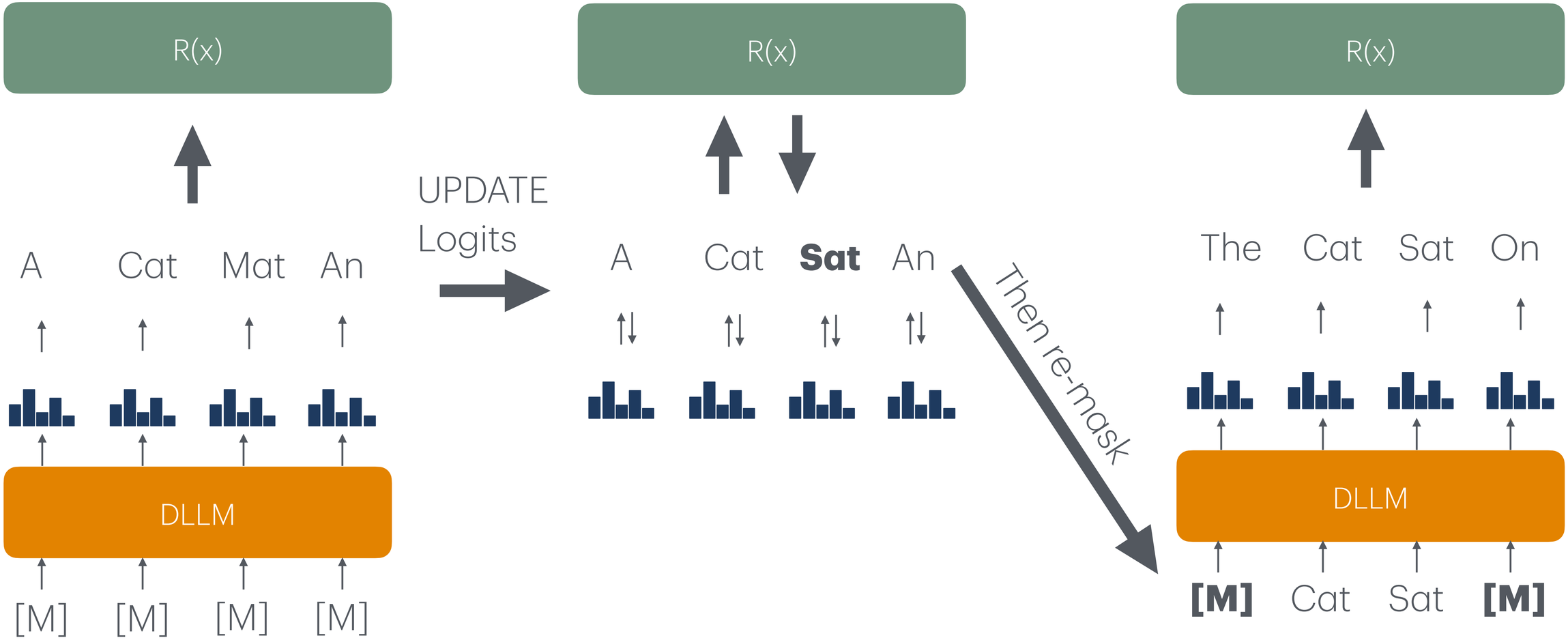
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



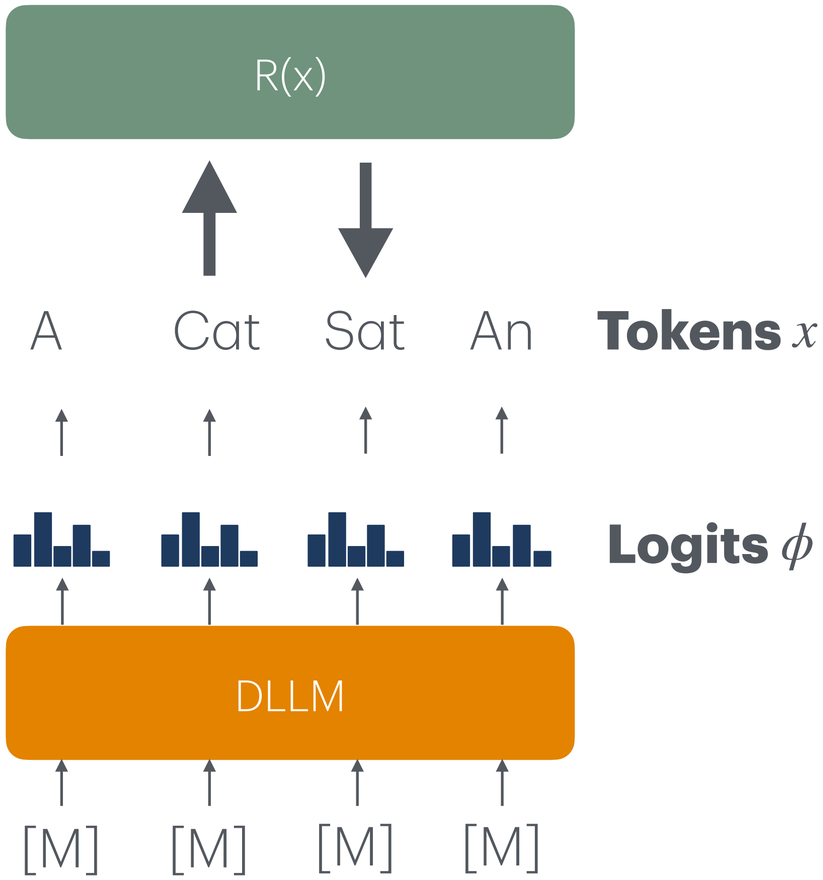
Goal: Generate high reward samples

Recall - DLLMs produce a distribution at ALL positions



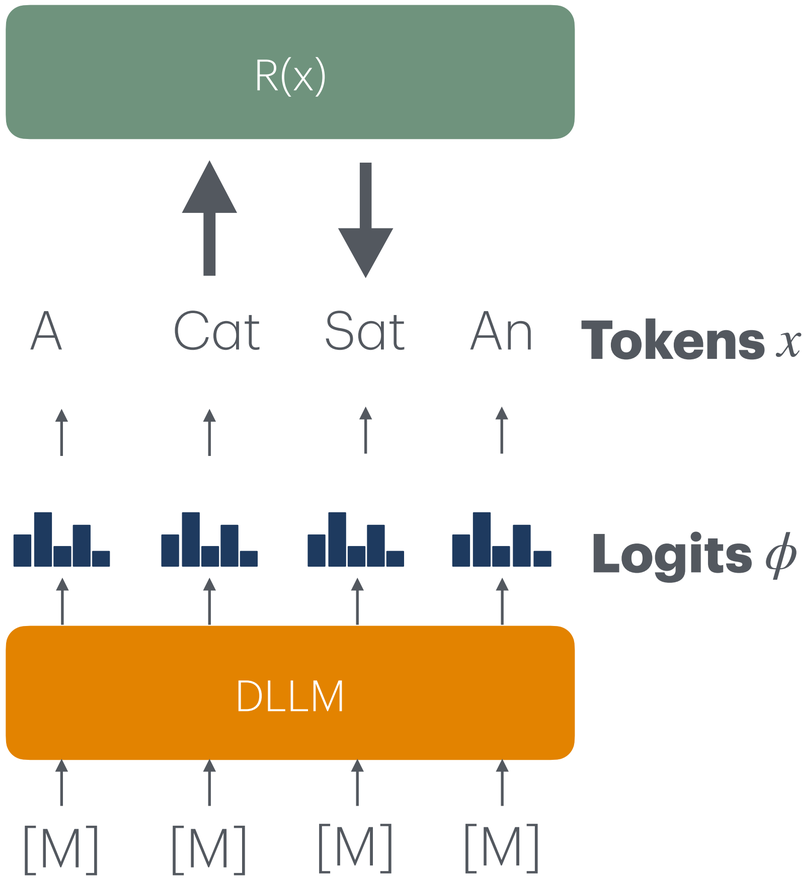
Gradient/Classifier Guidance (Dhariwal & Nichol et. al.)

If R(x) is differentiable, we can (potentially) update the logits. Successful in Continuous (Image) Diffusion



Gradient/Classifier Guidance (Dhariwal & Nichol et. al.)

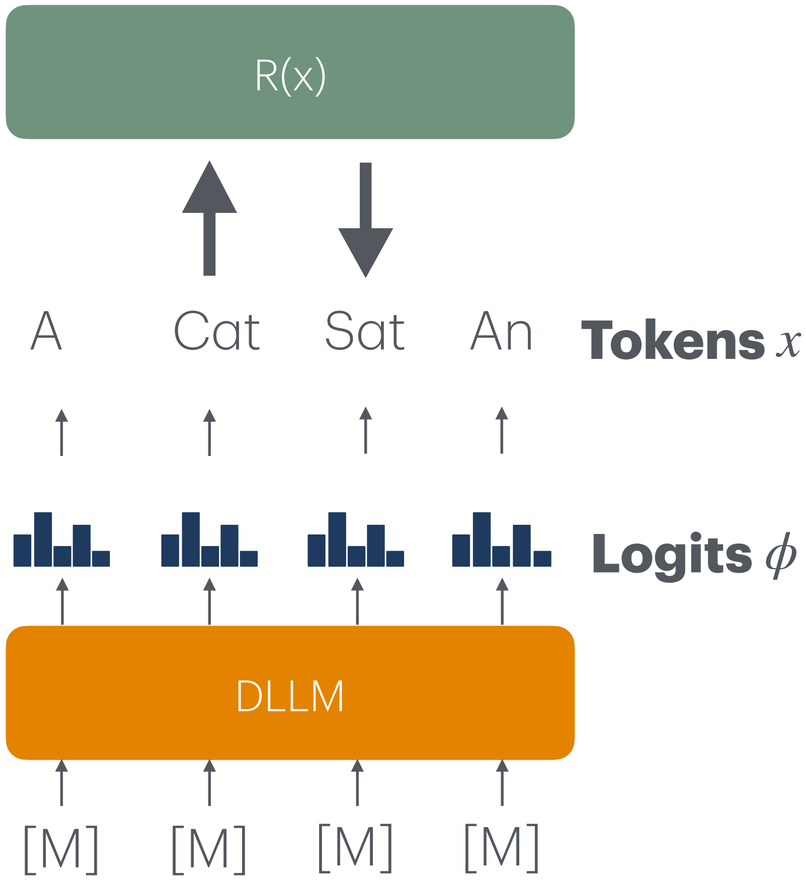
If R(x) is differentiable, we can (potentially) update the logits. Successful in Continuous (Image) Diffusion



$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

Gradient/Classifier Guidance (Dhariwal & Nichol et. al.)

If R(x) is differentiable, we can (potentially) update the logits. Successful in Continuous (Image) Diffusion

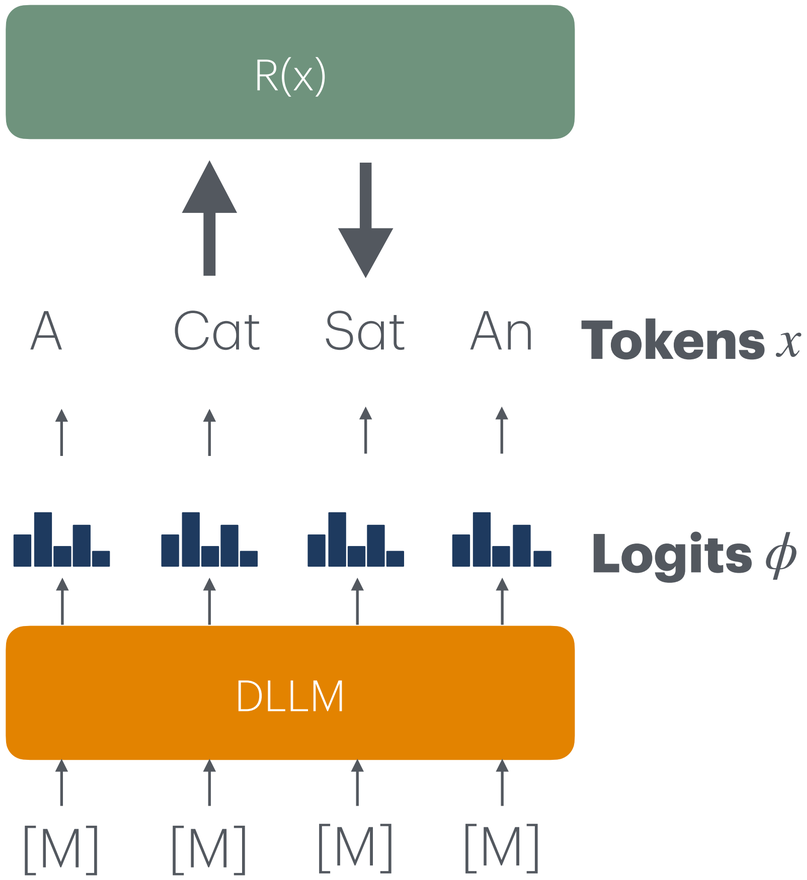


$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

$$\log p^*(x) \propto \log p_0(x) + \frac{R(x)}{\beta}$$

Gradient/Classifier Guidance (Dhariwal & Nichol et. al.)

If R(x) is differentiable, we can (potentially) update the logits. Successful in Continuous (Image) Diffusion



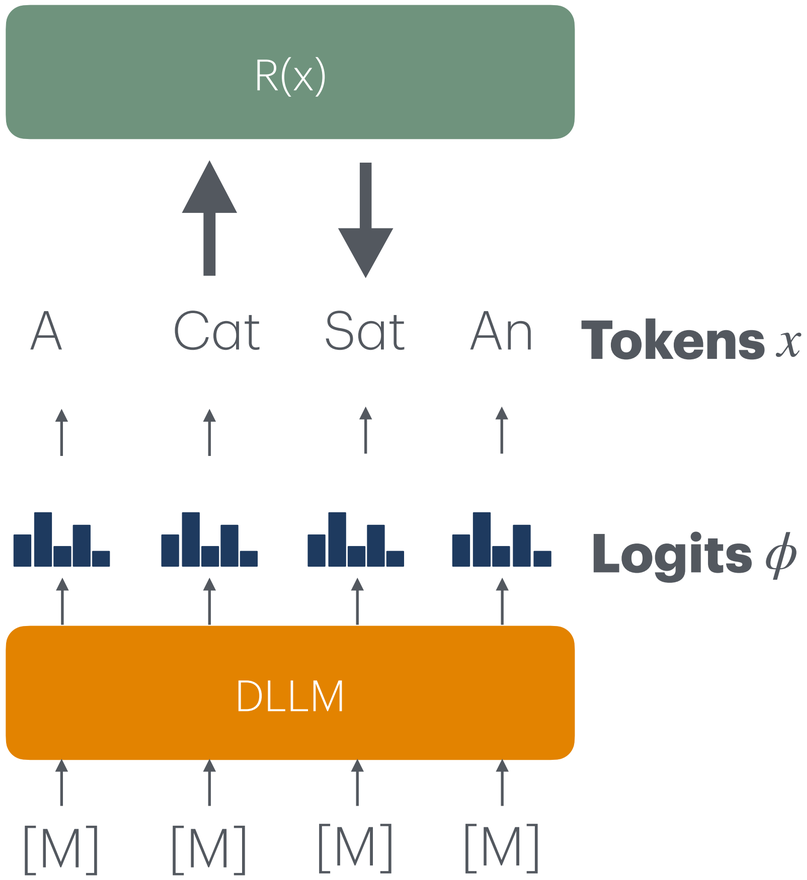
$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

$$\log p^*(x) \propto \log p_0(x) + \frac{R(x)}{\beta}$$

$$\nabla_x \log p^*(x) = \nabla_x \log p_0(x) + \frac{1}{\beta} \nabla_x R(x)$$

Gradient/Classifier Guidance (Dhariwal & Nichol et. al.)

If R(x) is differentiable, we can (potentially) update the logits. Successful in Continuous (Image) Diffusion



$$p^*(x) \propto p_\theta(x) \cdot \exp(R(x)/\beta)$$

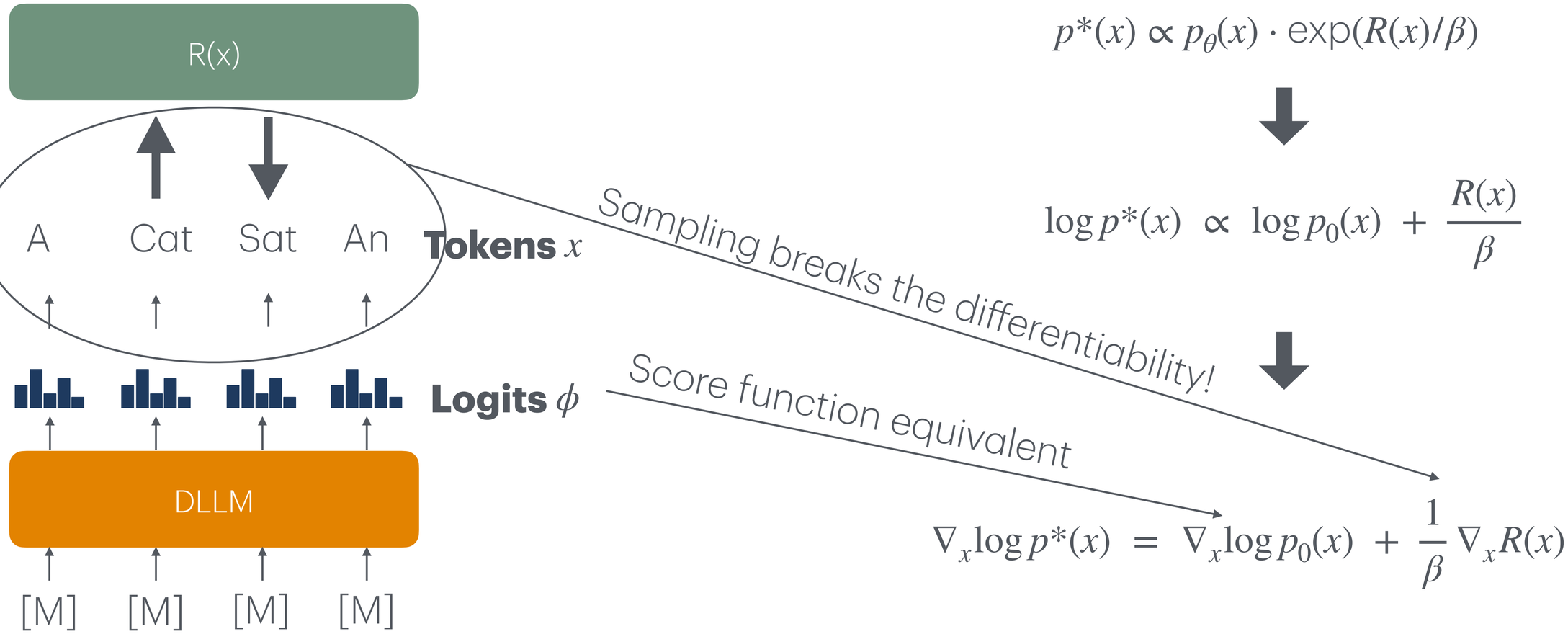
$$\log p^*(x) \propto \log p_0(x) + \frac{R(x)}{\beta}$$

Score function equivalent

$$\nabla_x \log p^*(x) = \nabla_x \log p_0(x) + \frac{1}{\beta} \nabla_x R(x)$$

Gradient/Classifier Guidance (Dhariwal & Nichol et. al.)

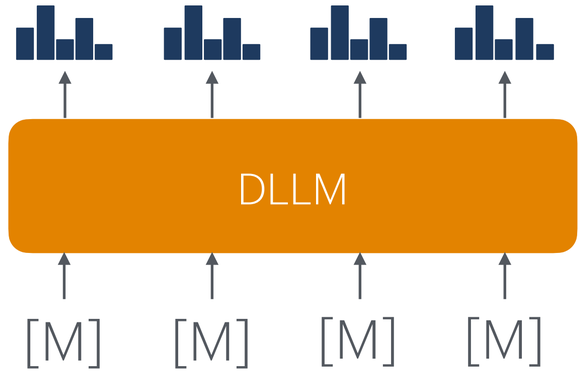
If R(x) is differentiable, we can (potentially) update the logits. Successful in Continuous (Image) Diffusion



We can pretend the problem doesn't exist

Feed the `expected token` to R

- Let E_R be the embedding table of the reward model R

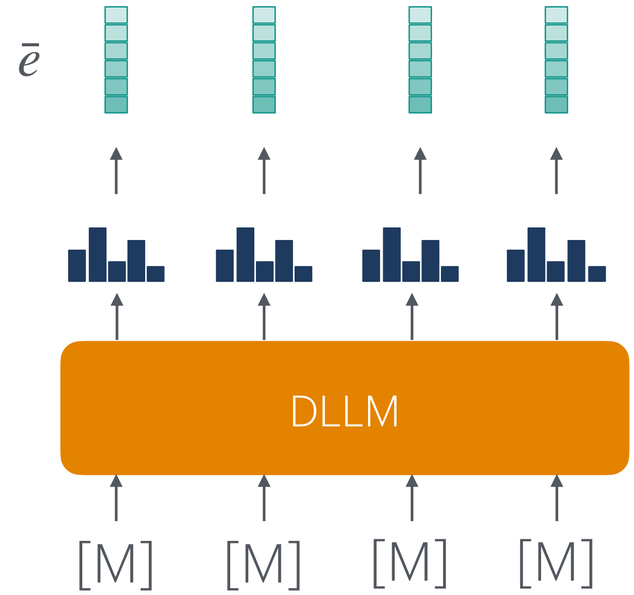


We can pretend the problem doesn't exist

Feed the `expected token` to R

- Let E_R be the embedding table of the reward model R

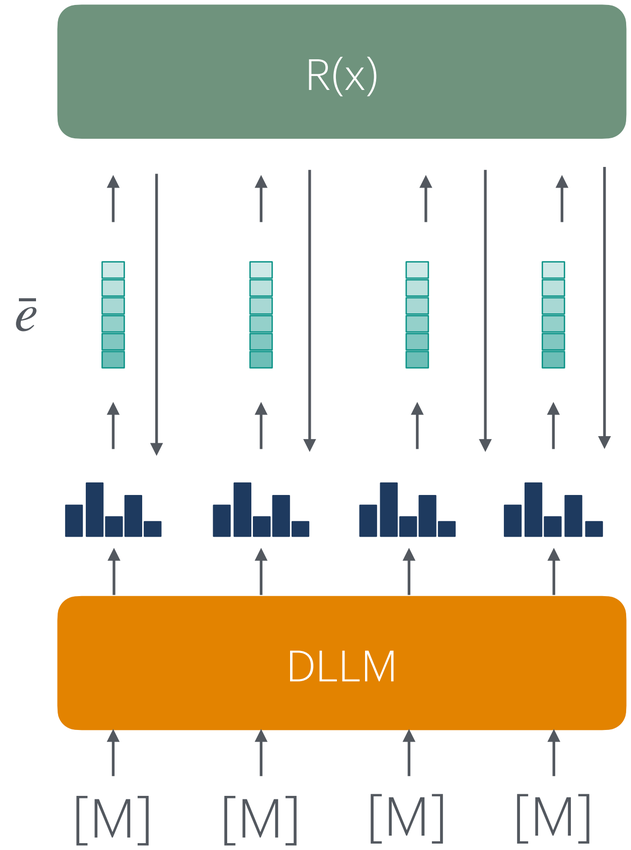
- Calculate the Expected Token Embedding as:
  $\bar{e} = \sum_{i \in \mathcal{V}} \text{softmax}(\phi) \cdot E_R^i$



We can pretend the problem doesn't exist

Feed the `expected token` to R

- Let E_R be the embedding table of the reward model R
- Calculate the Expected Token Embedding as:
  $\bar{e} = \sum_{i \in \mathcal{V}} \text{softmax}(\phi) \cdot E_R^i$
- Update logits as : $\phi \leftarrow \phi + \frac{1}{\beta} \nabla_\phi R(\bar{e})$

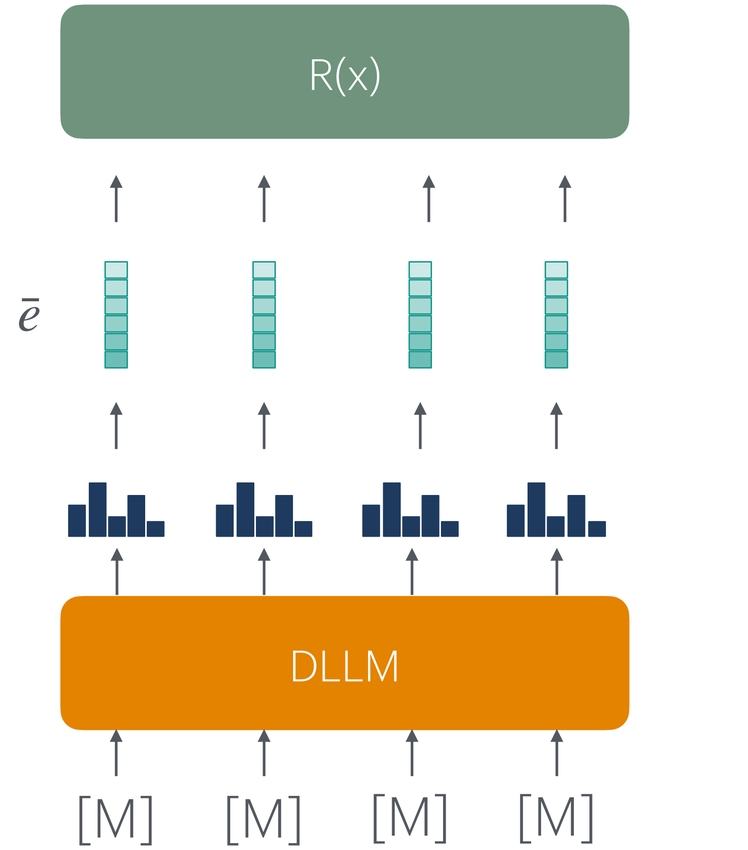


We can pretend the problem doesn't exist

Feed the `expected token` to R

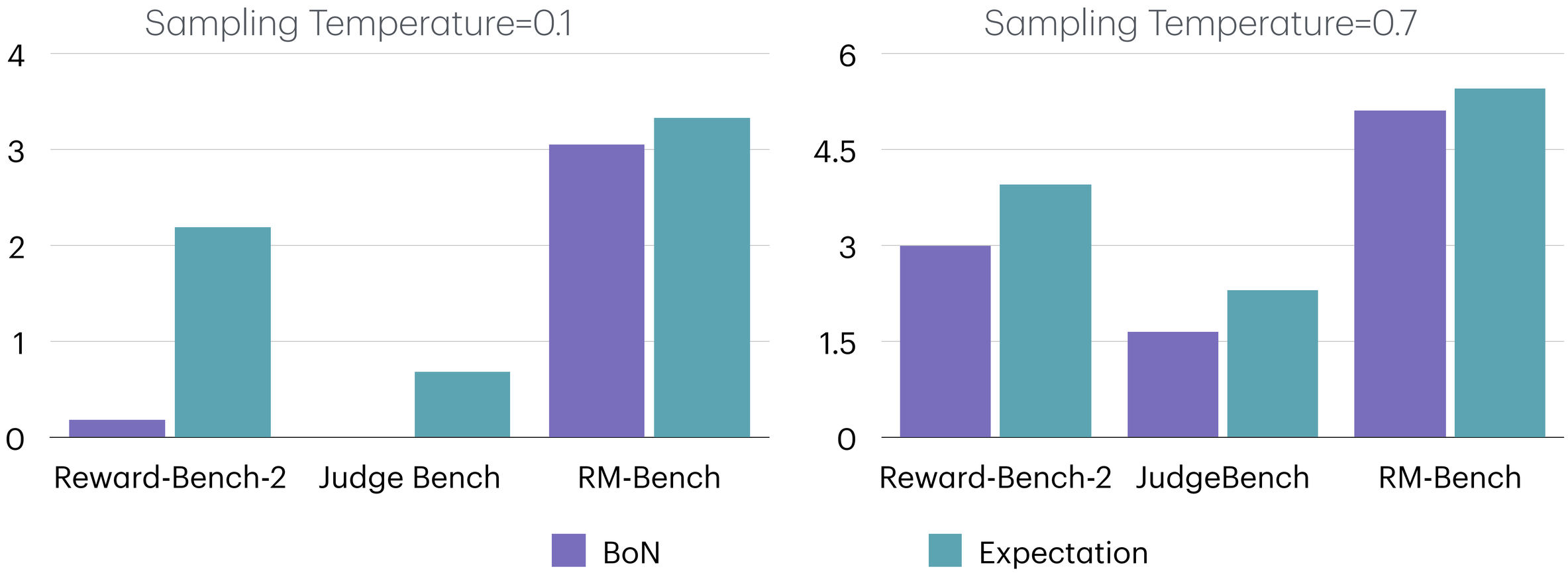
May not always know what $\bar{e}$ means!

- Let E_R be the embedding table of the reward model R
- Calculate the Expected Token Embedding as:
  $\bar{e} = \sum_{i \in \mathcal{V}} \text{softmax}(\phi) \cdot E_R^i$
- Update logits as : $\phi \leftarrow \phi + \frac{1}{\beta} \nabla_\phi R(\bar{e})$
- $\bar{e}$ may not be close to any real token embedding E_R^i



We can pretend the problem doesn't exist

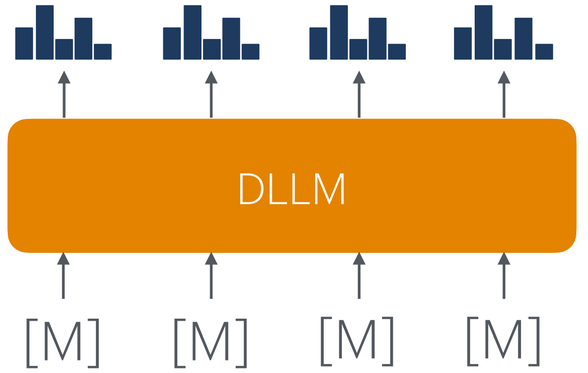
Feed the `expected token` to R



APS (Rout et al.) for Image Discrete Diffusion

Feed a discrete token, but compute gradient at the soft token (Bengio et. al., Jang et. al.)

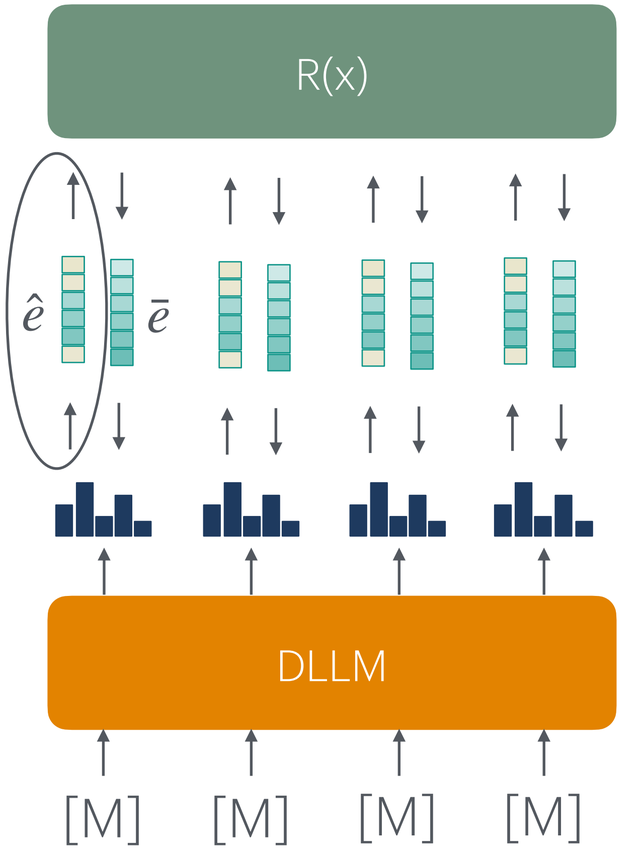
- Sample a token x
- Get its embedding $\tilde{e} = E_R[x]$



APS (Rout et al.) for Image Discrete Diffusion

Feed a discrete token, but compute gradient at the soft token (Bengio et. al., Jang et. al.)

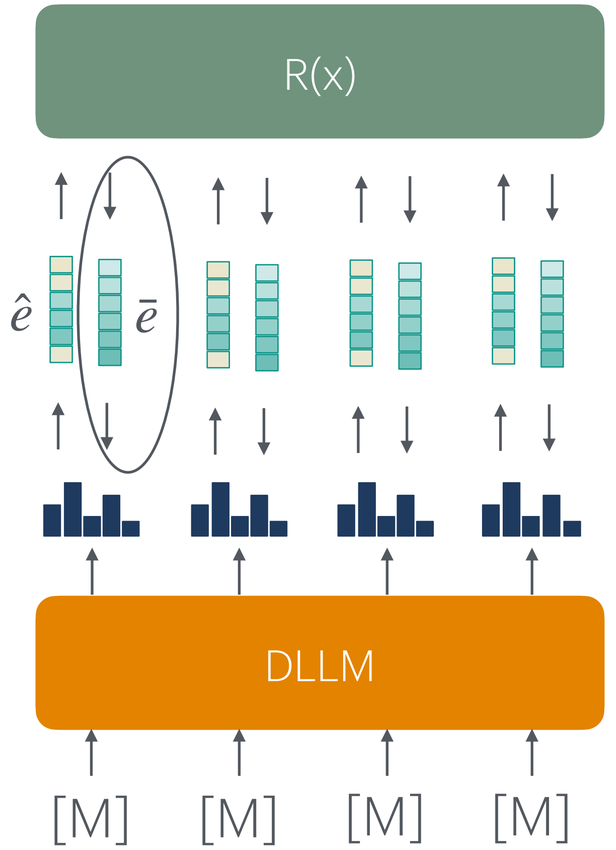
- Sample a token x
- Get its embedding $\tilde{e} = E_R[x]$
- Construct input embedding to R as $\hat{e} = \bar{e} + \text{stop-grad}(\tilde{e} - \bar{e})$



APS (Rout et al.) for Image Discrete Diffusion

Feed a discrete token, but compute gradient at the soft token (Bengio et. al., Jang et. al.)

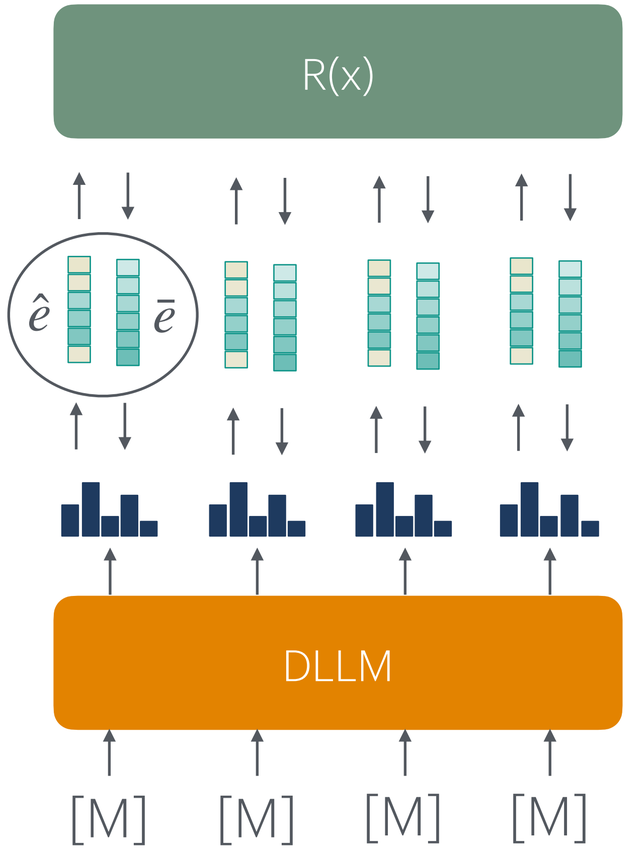
- Sample a token x
- Get its embedding $\tilde{e} = E_R[x]$
- Construct input embedding to R as
  $\hat{e} = \bar{e} +$ stop-grad$(\tilde{e} - \bar{e})$
- Intuition: R effectively 'sees' $\tilde{e}$ but computes gradient at $\bar{e}$



APS (Rout et al.) for Image Discrete Diffusion

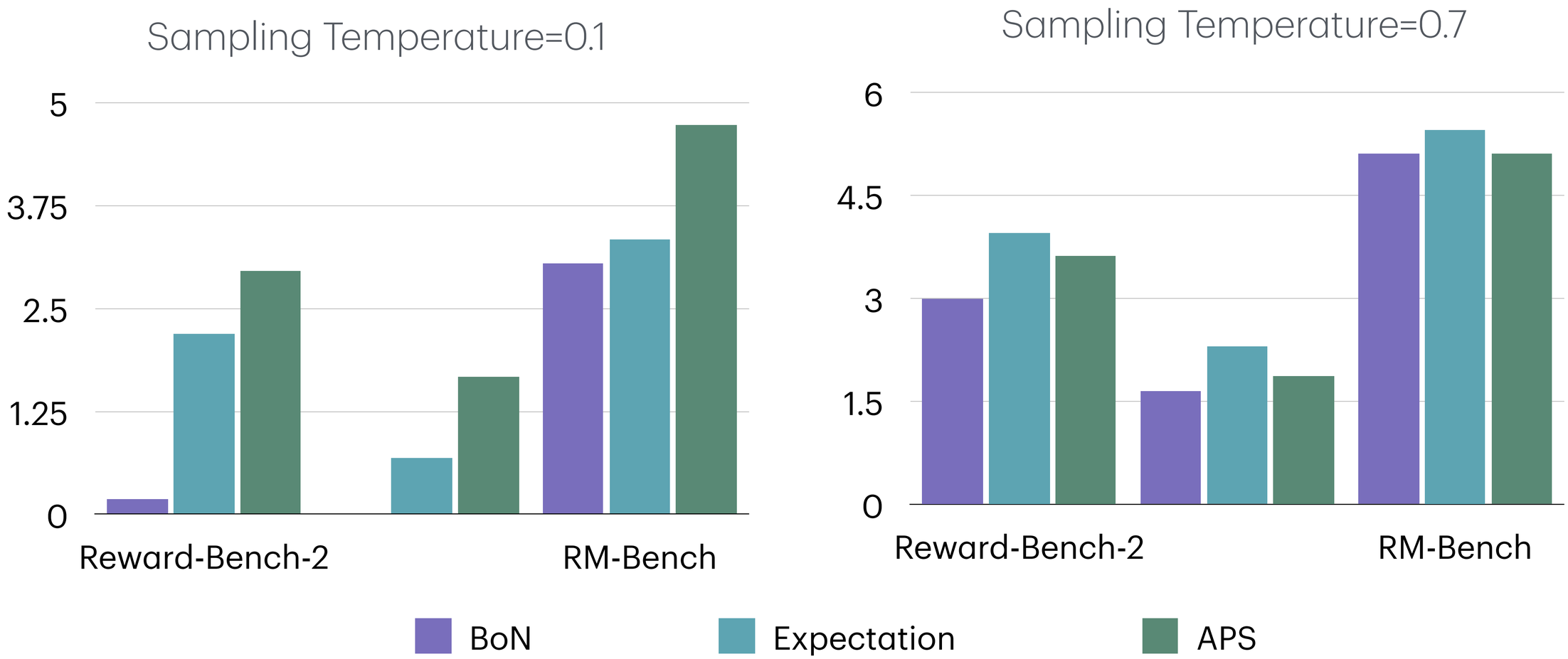
Feed a discrete token, but compute gradient at the soft token (Bengio et. al., Jang et. al.)

- Sample a token x
- Get its embedding $\tilde{e} = E_R[x]$
- Construct input embedding to R as
  $\hat{e} = \bar{e} + \text{stop-grad}(\tilde{e} - \bar{e})$
- Intuition: R effectively 'sees' $\tilde{e}$ but computes gradient at $\bar{e}$
- Issue: If $\bar{e}$ is far away from $\tilde{e}$, the gradients are wrong
- VQ-VAE in images, so this issue is not as catastrophic



APS (Rout et al.) for Image Discrete Diffusion

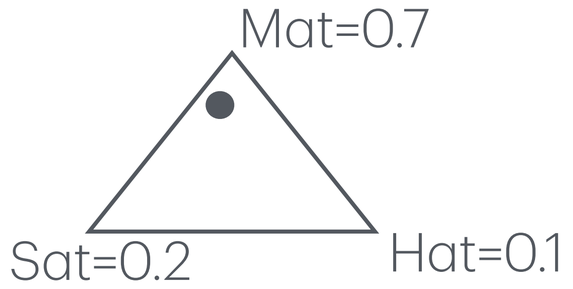
If gradients are wrong, performance drops!



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

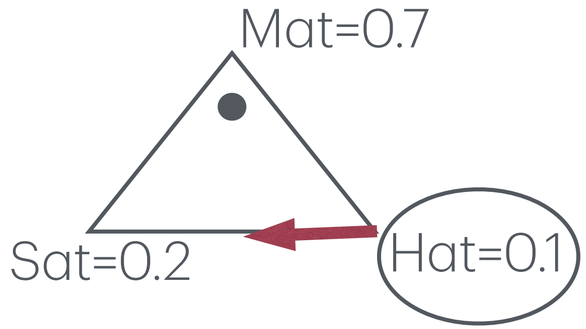
- Low Entropy -> Model is close to a real token -> don't sample, use the expected (soft) embedding



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

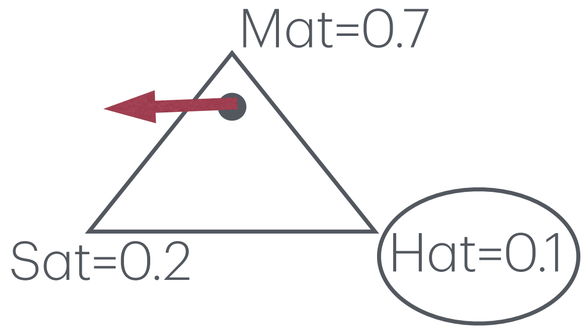
- Low Entropy -> Model is close to a real token -> don't sample, use the expected (soft) embedding



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

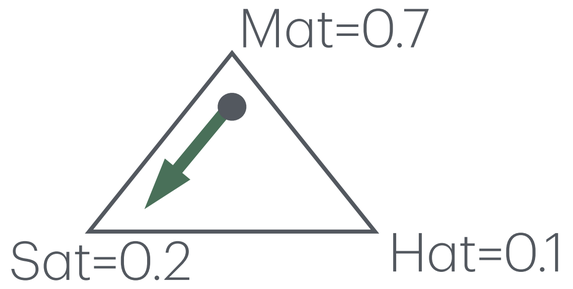
- Low Entropy -> Model is close to a real token -> don't sample, use the expected (soft) embedding



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

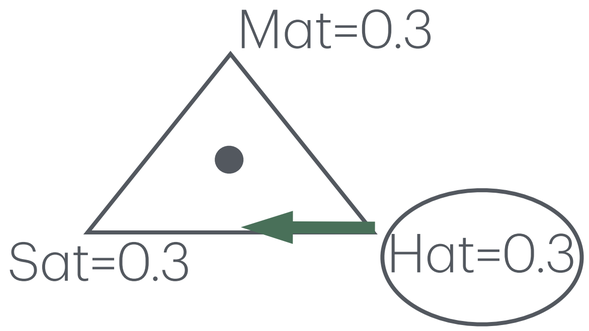
- Low Entropy -> Model is close to a real token -> don't sample, use the expected (soft) embedding



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

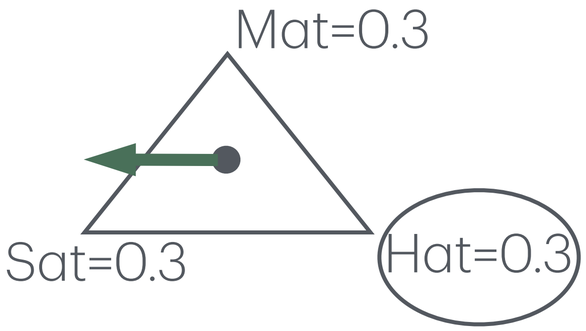
- Low Entropy -> Model is close to a real token -> don't sample, use the expected (soft) embedding
- High Entropy -> R(x) becomes unreliable, fall back to STE



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

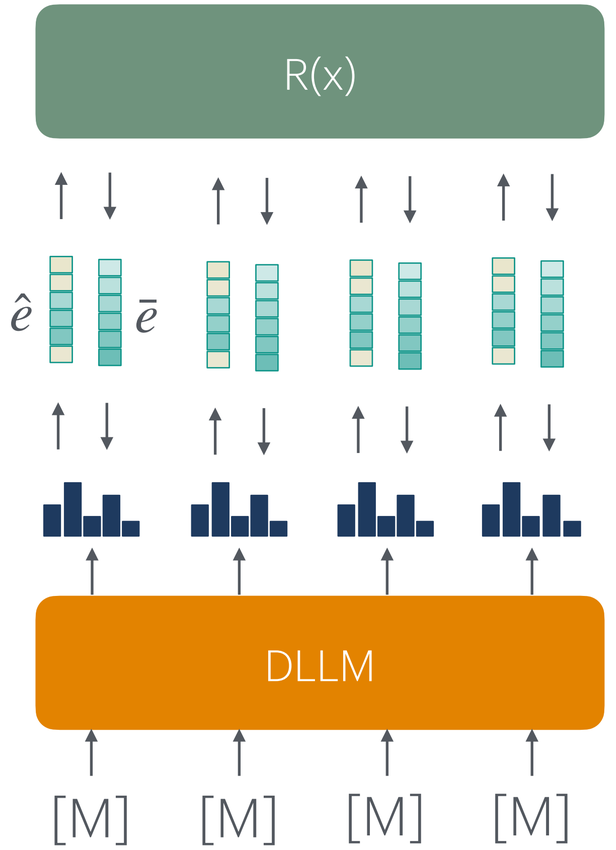
- Low Entropy -> Model is close to a real token -> don't sample, use the expected (soft) embedding
- High Entropy -> R(x) becomes unreliable, fall back to STE



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

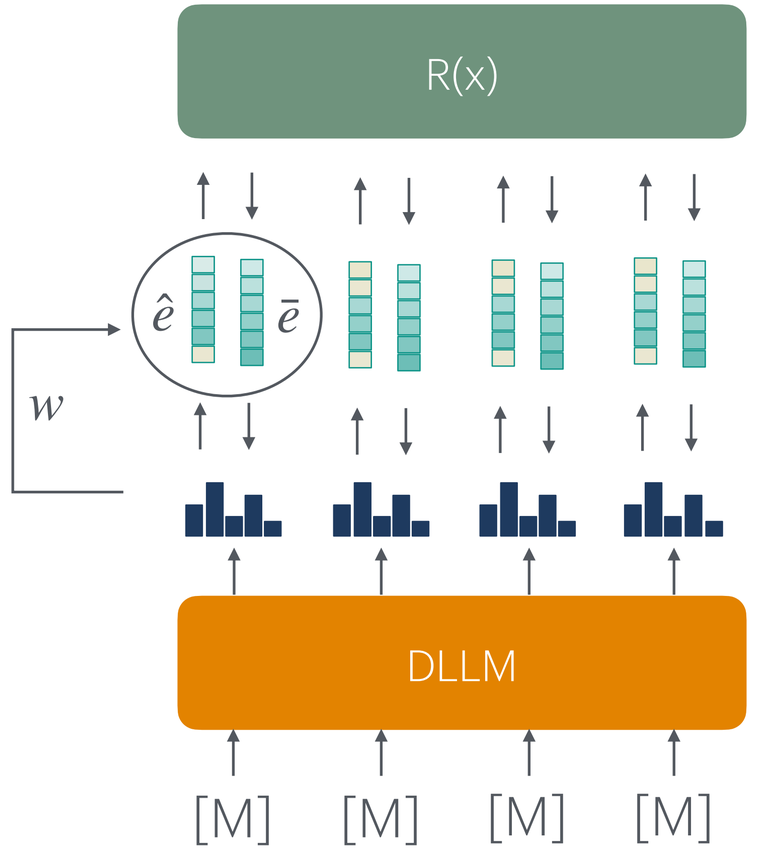
- Low Entropy -> Model is close to a real token -> don't sample, use the expected embedding
- High Entropy -> R(x) becomes unreliable, fall back to STE
- Construct input embedding to R as
  $\hat{e} = \bar{e} + \text{stop-grad}(w(\tilde{e} - \bar{e}))$
- Where $w = \dfrac{H(\text{softmax}(\phi))}{\log |\mathcal{V}|} \in [0,1]$



EntRGi: Let the Model's Entropy Decide

The right choice depends on the confidence of the model (see paper for gradient analysis)

- Low Entropy -> Model is close to a real token -> don't sample, use the expected embedding
- High Entropy -> RM becomes unreliable, fall back to STE
- Construct input embedding to R as
  $\hat{e} = \bar{e} + \text{stop-grad}(w(\tilde{e} - \bar{e}))$
- Where $w = \dfrac{H(\text{softmax}(\phi))}{\log |\mathcal{V}|} \in [0,1]$



Results

Dream-7B-Instruct with Skywork-Reward-V2-0.6B

External model to measure reward hacking

Method	Reward-Bench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
Temperature ($\tau = 0.1$)									
BoN	0.18±0.22	0.05±0.23	3.74±0.04	0.00±0.15	-0.07±0.16	3.75±0.03	3.05±0.05	3.02±0.05	3.93±0.01
Expectation	2.19±0.19	**1.62**±0.17	4.12±0.03	0.68±0.19	**-0.06**±0.21	3.81±0.02	3.33±0.20	2.59±0.12	3.89±0.04
APS	2.95±0.21	1.47±0.20	4.19±0.01	**1.67**±0.11	-0.17±0.14	3.89±0.03	4.72±0.13	2.46±0.17	4.01±0.03
EntRGi	**3.07**±0.22	**1.62**±0.18	**4.22**±0.02	**1.73**±0.14	-0.11±0.18	**3.94**±0.01	**4.90**±0.13	**2.75**±0.14	**4.06**±0.01
Temperature ($\tau = 0.7$)									
BoN	2.99±0.23	1.38±0.29	4.15±0.02	1.65±0.18	-0.84±0.16	3.91±0.02	5.11±0.20	2.98±0.15	4.02±0.03
Expectation	**3.95**±0.28	**2.23**±0.24	4.22±0.02	2.30±0.08	**0.13**±0.07	**3.97**±0.01	5.45±0.16	3.29±0.13	4.02±0.03
APS	3.62±0.27	1.80±0.24	4.22±0.02	1.87±0.14	-0.63±0.10	3.93±0.02	5.11±0.14	2.66±0.15	4.00±0.02
EntRGi	**3.91**±0.30	**2.20**±0.26	**4.25**±0.02	**2.44**±0.06	0.02±0.10	**3.98**±0.02	**5.70**±0.12	**3.41**±0.14	**4.04**±0.01

Try it Out!

TEXAS
The University of Texas at Austin

Paper

Code