



Accelerating Masked Image Generation by Learning Controlled Latent Dynamics

ACM
Multimedia 2026
Rio de Janeiro, Brazil

Kaiwen Zhu, Quansheng Zeng, Yuandong Pu, Shuo Cao, Xiaohui Li,
Yi Xin, Qi Qin, Jiayang Li, Juncheng Yan, Yu Qiao, Jinjin Gu, Yihao Liu



Paper



Project Page

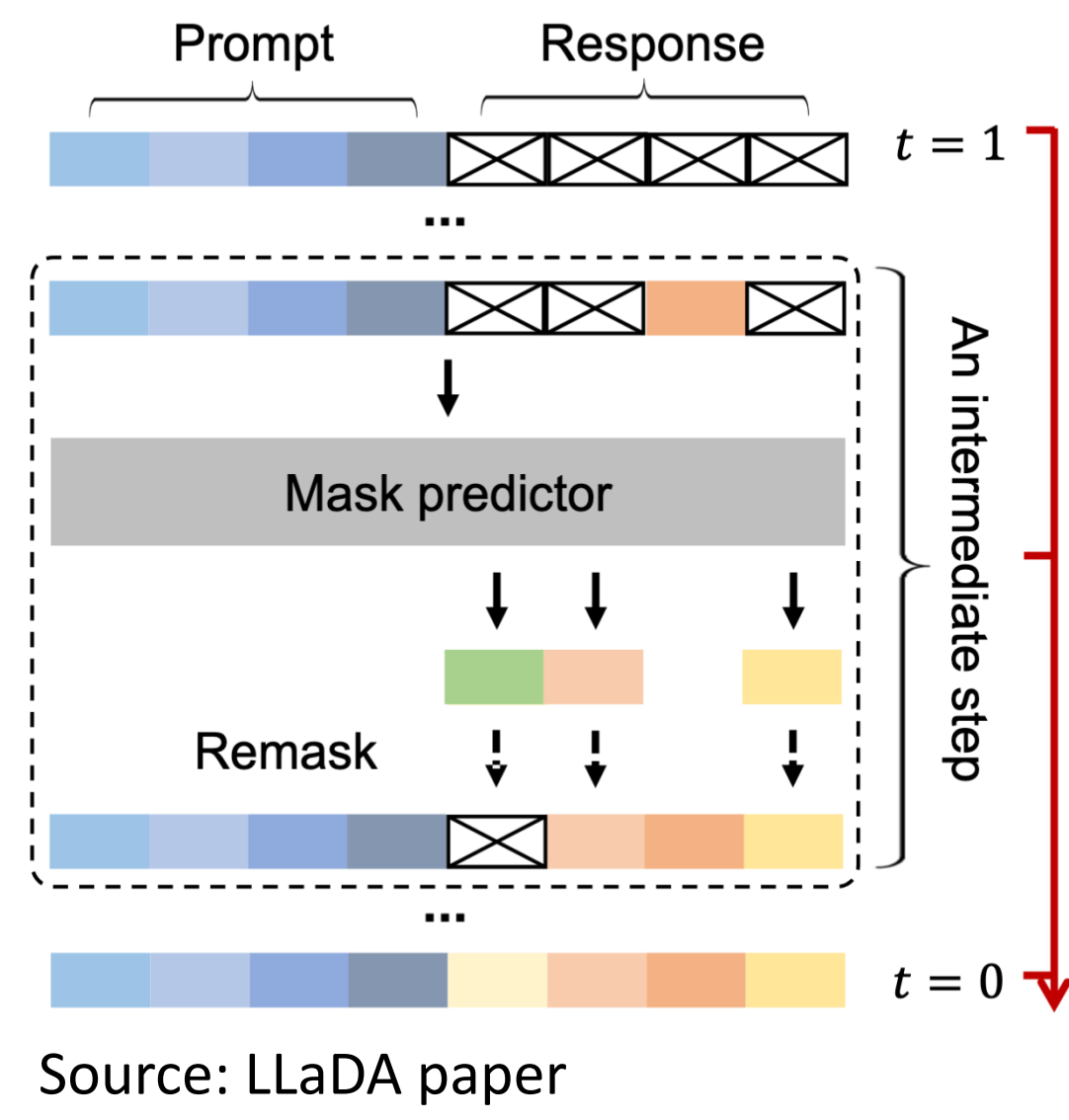


Author Page

Preliminary: Masked Generative Model (MGM)

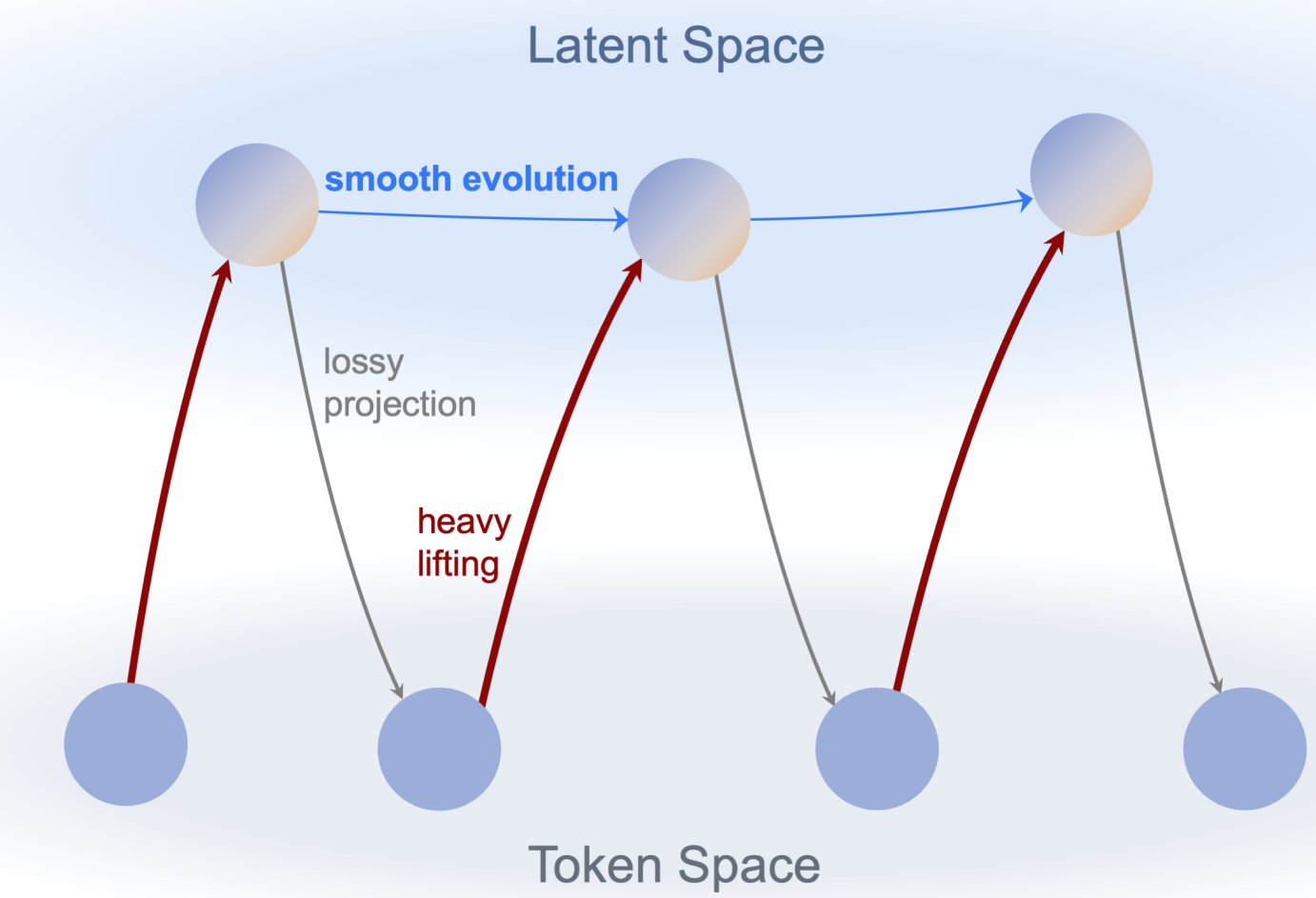
Generate in a discrete token space

- Begin with fully masked tokens x_0
- At step i , invoke a base model M to compute feature $f_i = M(x_{i-1})$
- Compute distribution p_i from f_i
- Sample from p_i for masked tokens
- Accept high-confidence samples; otherwise, remain masked



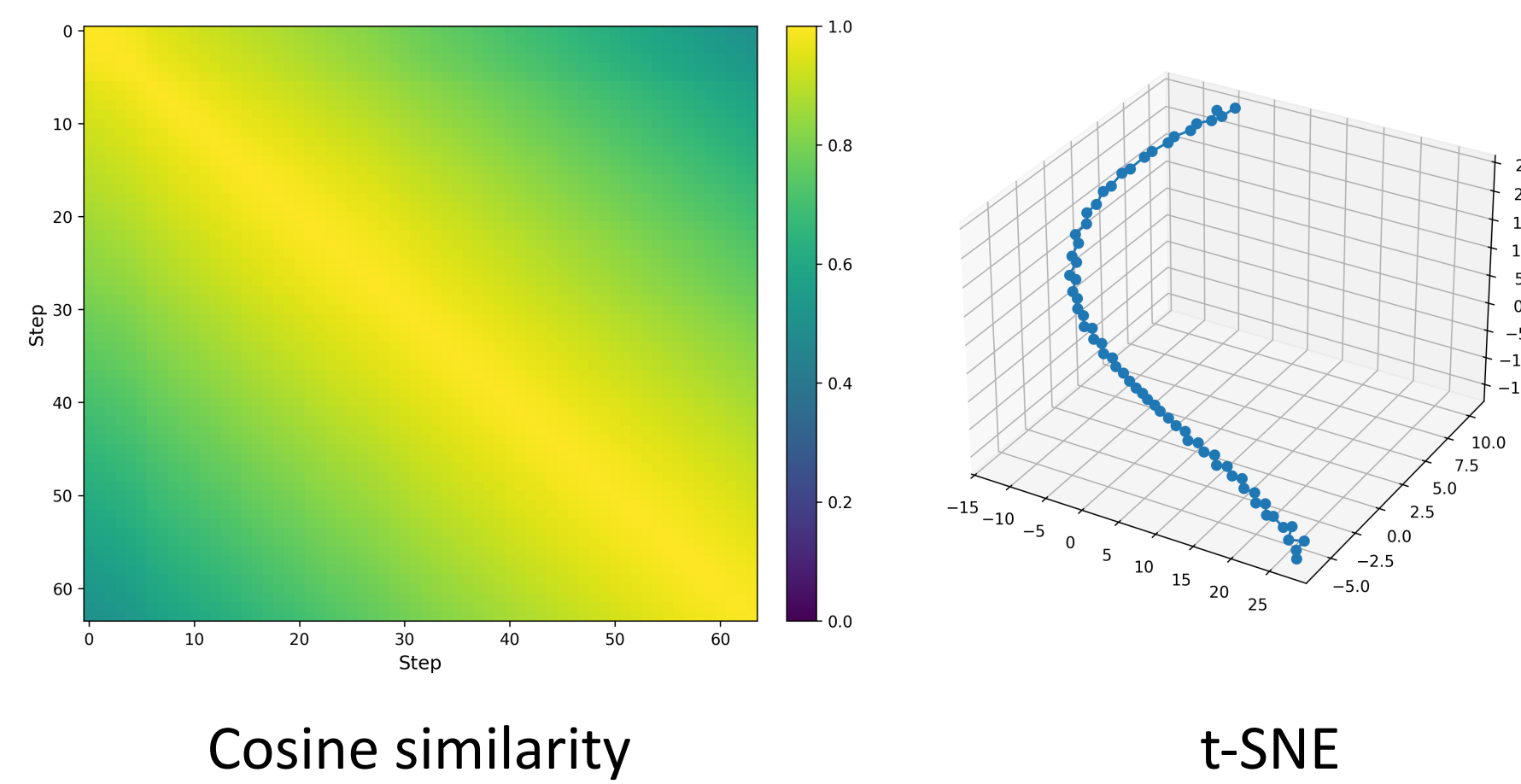
Insight: Avoid Waste, Operate in Latent Space

- Projection from latent space (f_i) to token space (x_i) is **lossy**
- Lifting from token space (x_i) to latent space (f_{i+1}) is **heavy**
- Directly model the latent transition instead



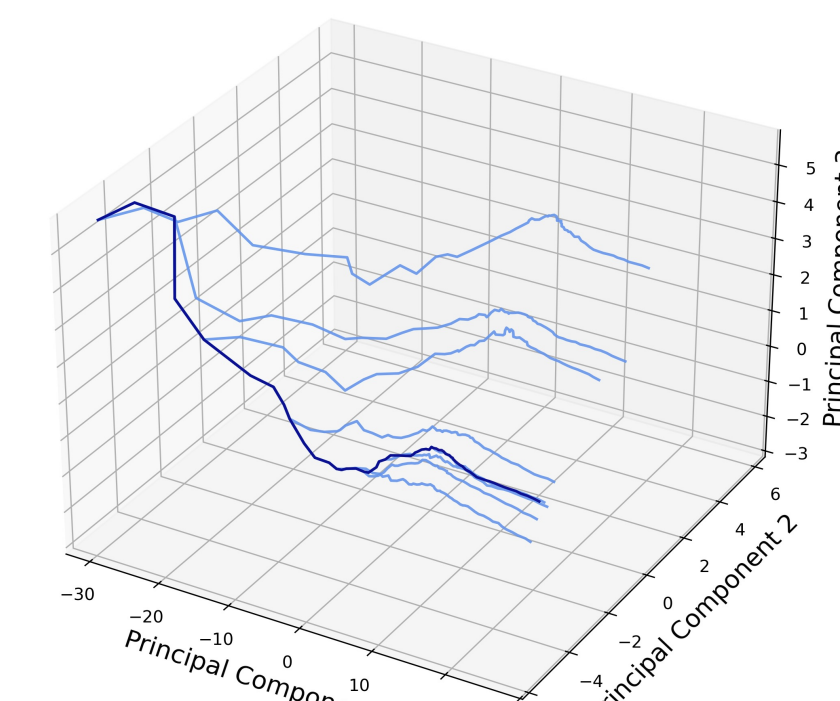
Motivation 1: Smoothness

- Observation 1: Feature trajectory is smooth
- Assumption 1: It is feasible to predict the next feature from the current feature with light computation

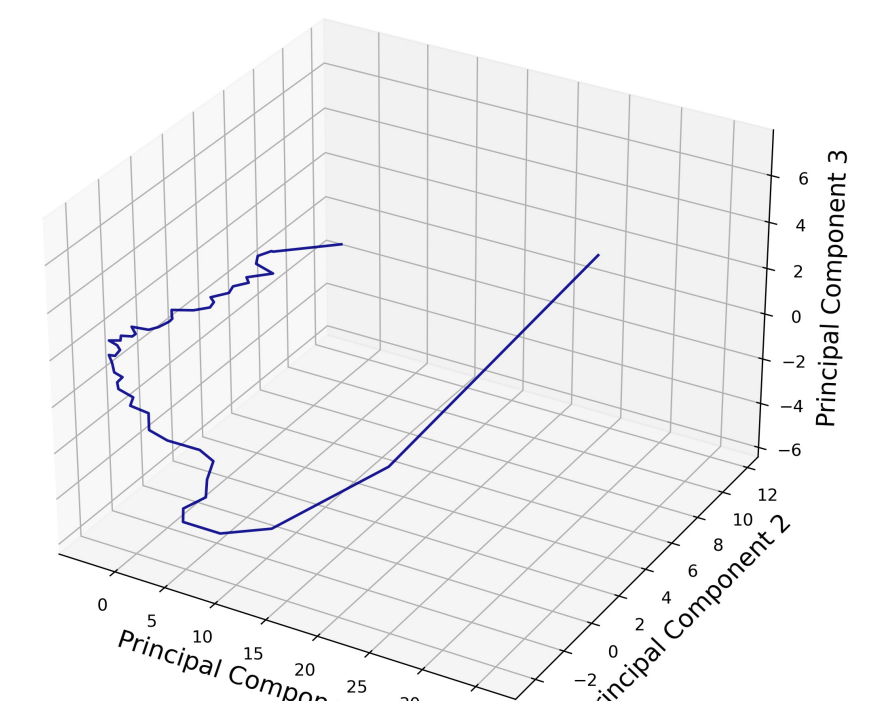


Motivation 2: Heteronomy

- Observation 2: Feature trajectory is controlled by sampled tokens
- Assumption 2: It is a must to account for info of sampled tokens



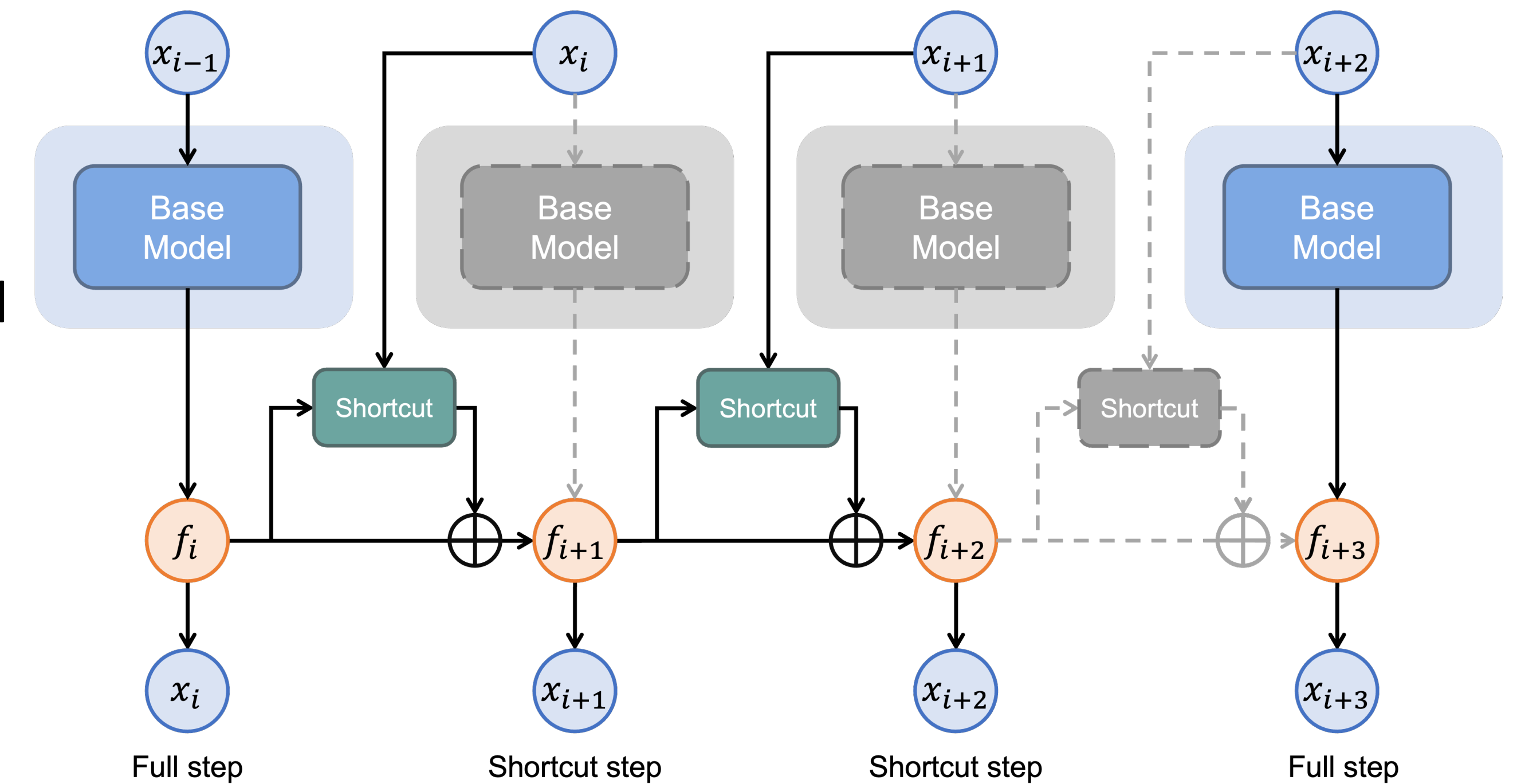
Masked Generation:
Randomness in intermediate steps,
controlled trajectory



Diffusion with ODE sampling:
Randomness only in start point,
self-contained trajectory

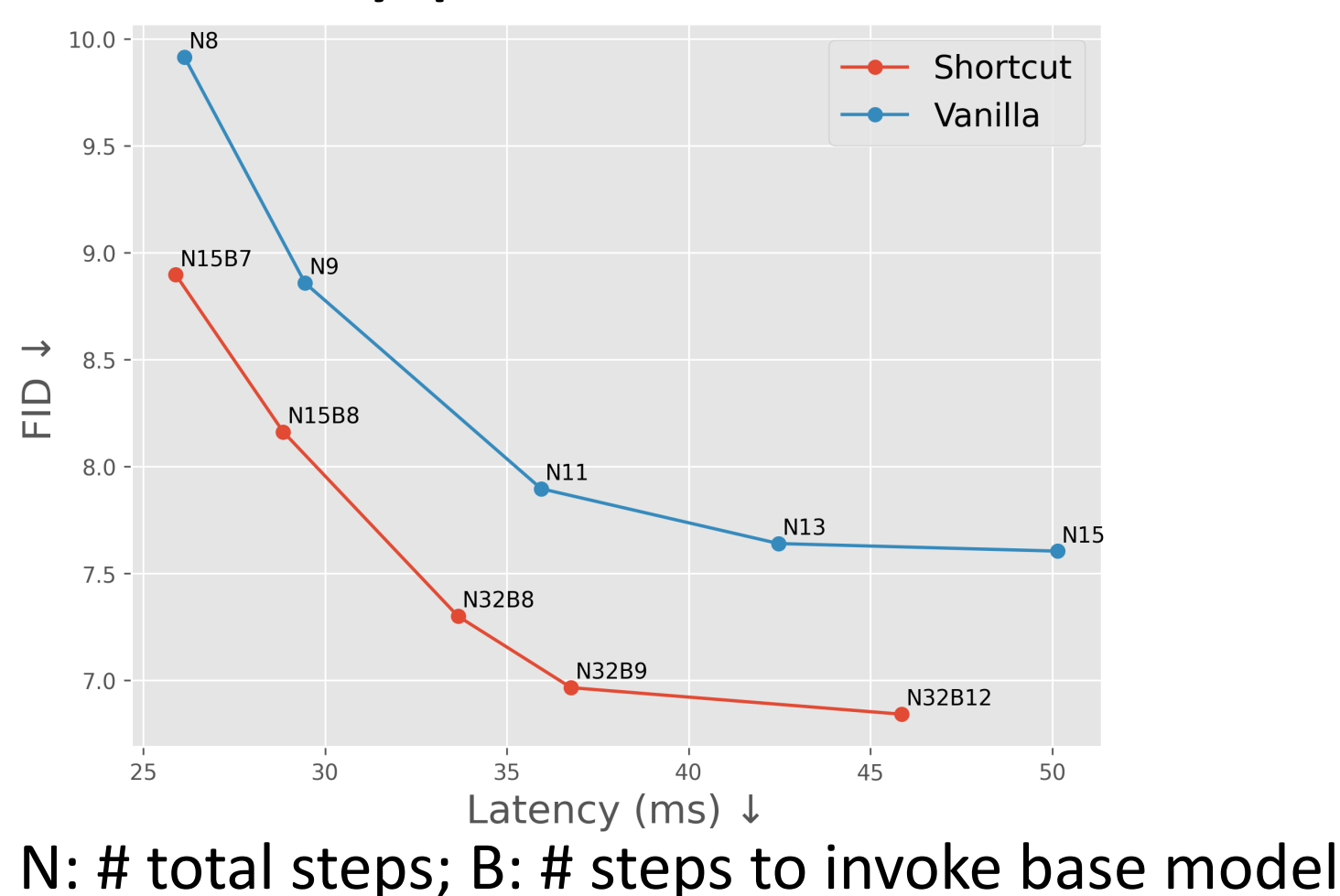
Method: MIGM-Shortcut

- Construct a lightweight “**shortcut model**” S_θ to compute the next feature from the current feature and sampled tokens
- S_θ finds a smooth shortcut in the latent space to skip the heavy base model
- Formulated as state-space model: latent as state, tokens as observation
- State transition: $f_{i+1} = f_i + S_\theta(f_i, x_i) + \epsilon$
- Observation: vanilla sampling procedure $x_{i+1} \sim K(x_i, f_{i+1})$
- Inference: Periodically replace the base model with the shortcut model
- Predefine a budget of #steps to invoke base model and evenly distribute it
- Training: Collect samples (f_i, x_i, f_{i+1}) and train by Mean Square Error



Experiment: on MaskGIT

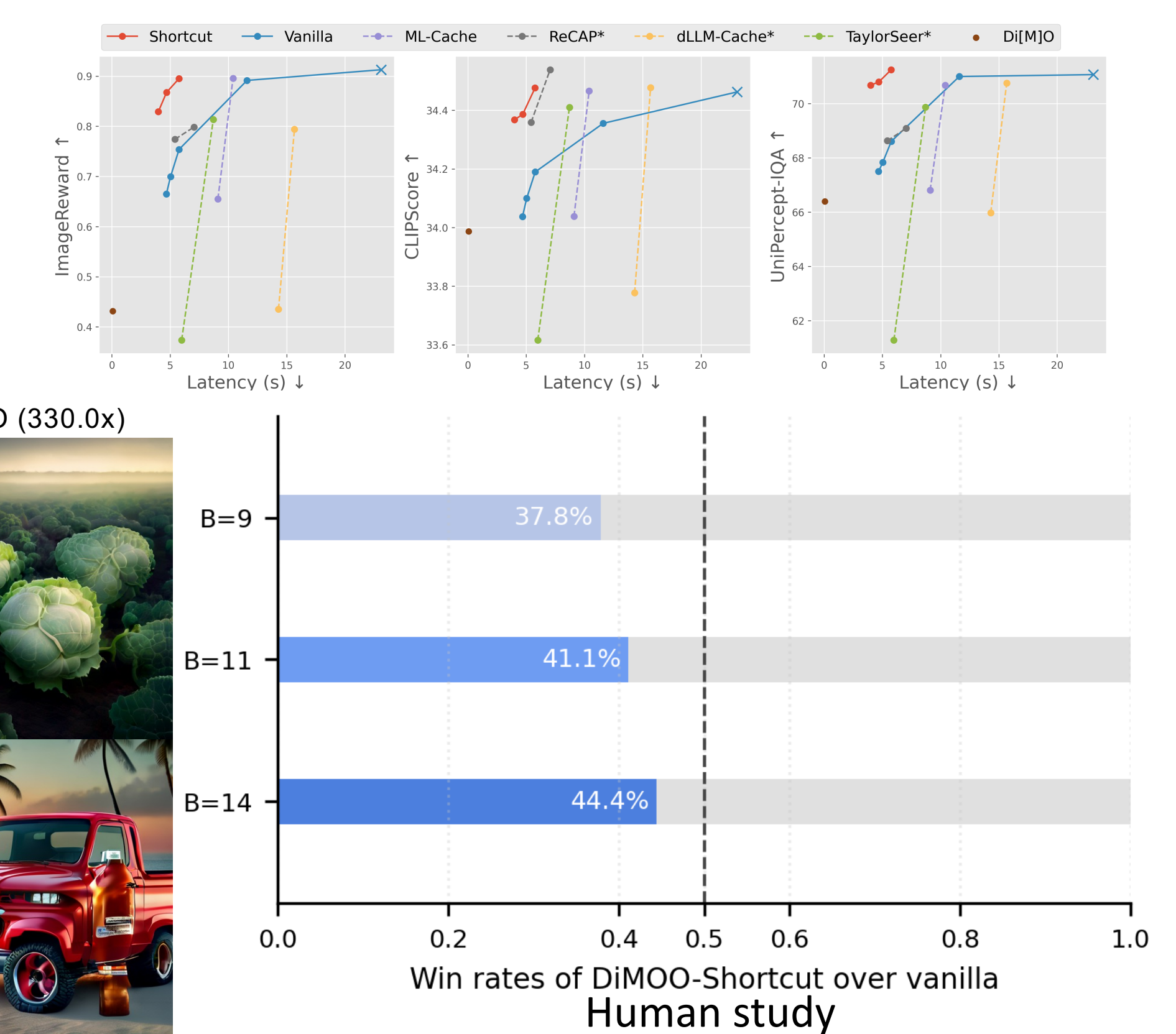
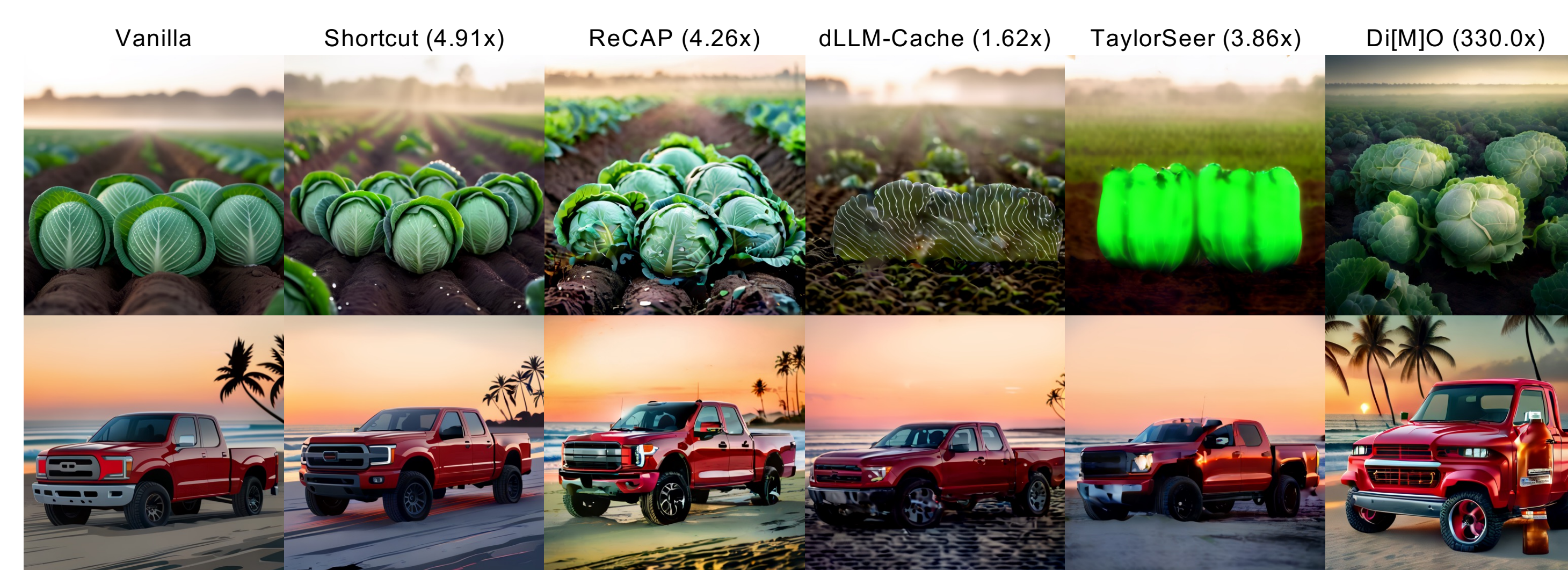
- 8.6 M parameters (1/20 of the base model)
- Train for 20 GPU hours
- Greatly push Pareto frontier



N: # total steps; B: # steps to invoke base model

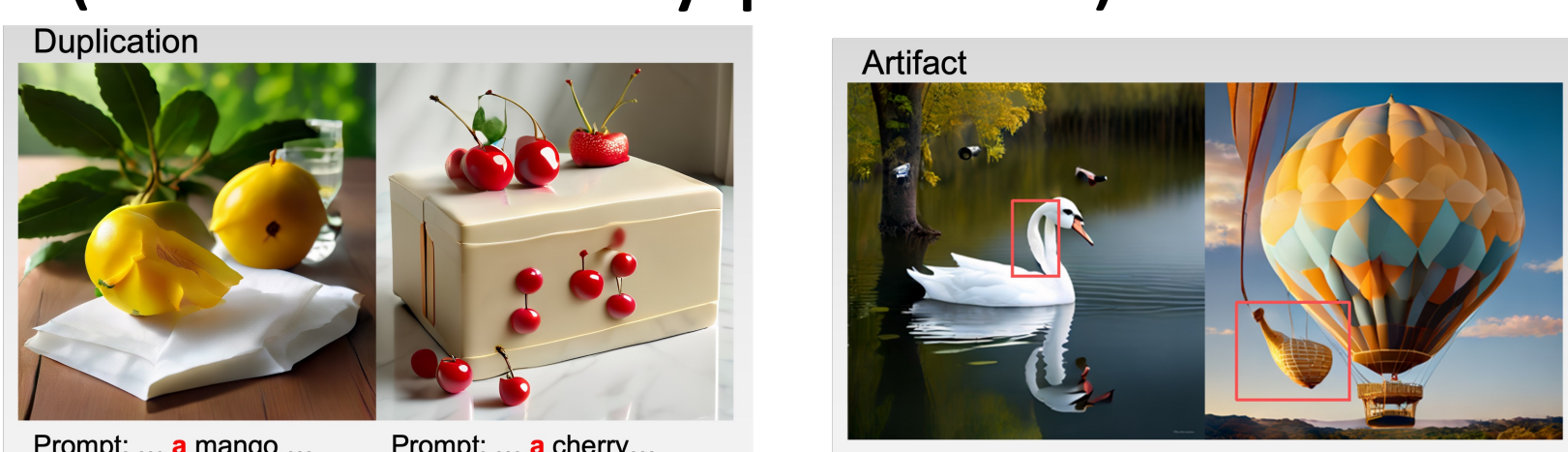
Experiment: text-to-image on Lumina-DiMOO

- 220 M parameters (**1/37** of the base model)
- Train for 48 GPU hours
- Accelerate **4-6 times** while maintaining quality



Difficulty of Reducing # Steps

- MGMs approximate joint distribution $p(x_1, \dots, x_n)$ with factorized product $p(x_1) \dots p(x_n)$
- Approximation error increases when # steps is reduced (multi-modality problem)



- Shortcut model reduces per-step cost instead

Ablation Studies

- Model complexity should strike a balance
 - Too high: waste computation
 - Too low: fail to capture subtle dynamics
- Necessity of sampling info
 - Without sampling info, forced to learn blurry output

