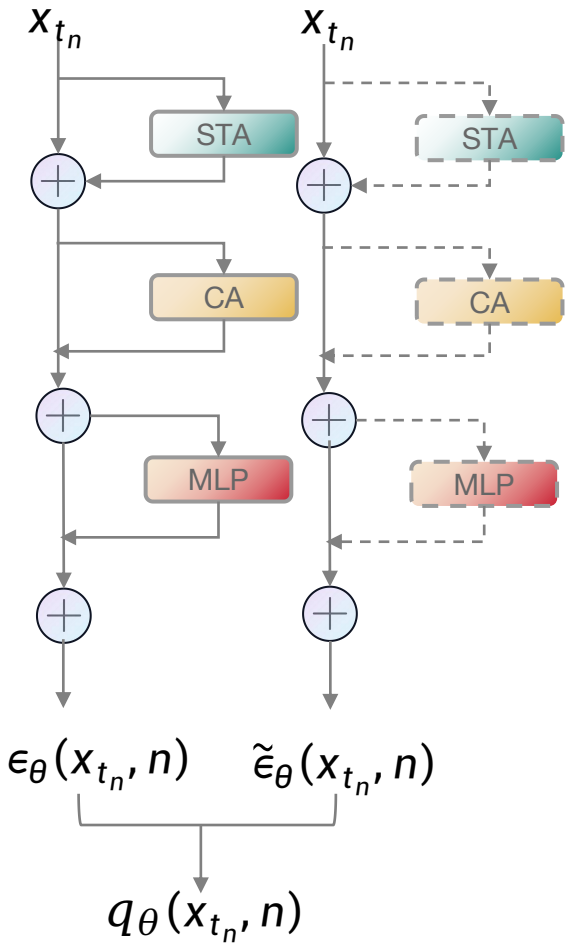
# **OmniCache**: A Trajectory-Oriented Global Perspective on Training-Free Cache Reuse for Diffusion Transformer Models

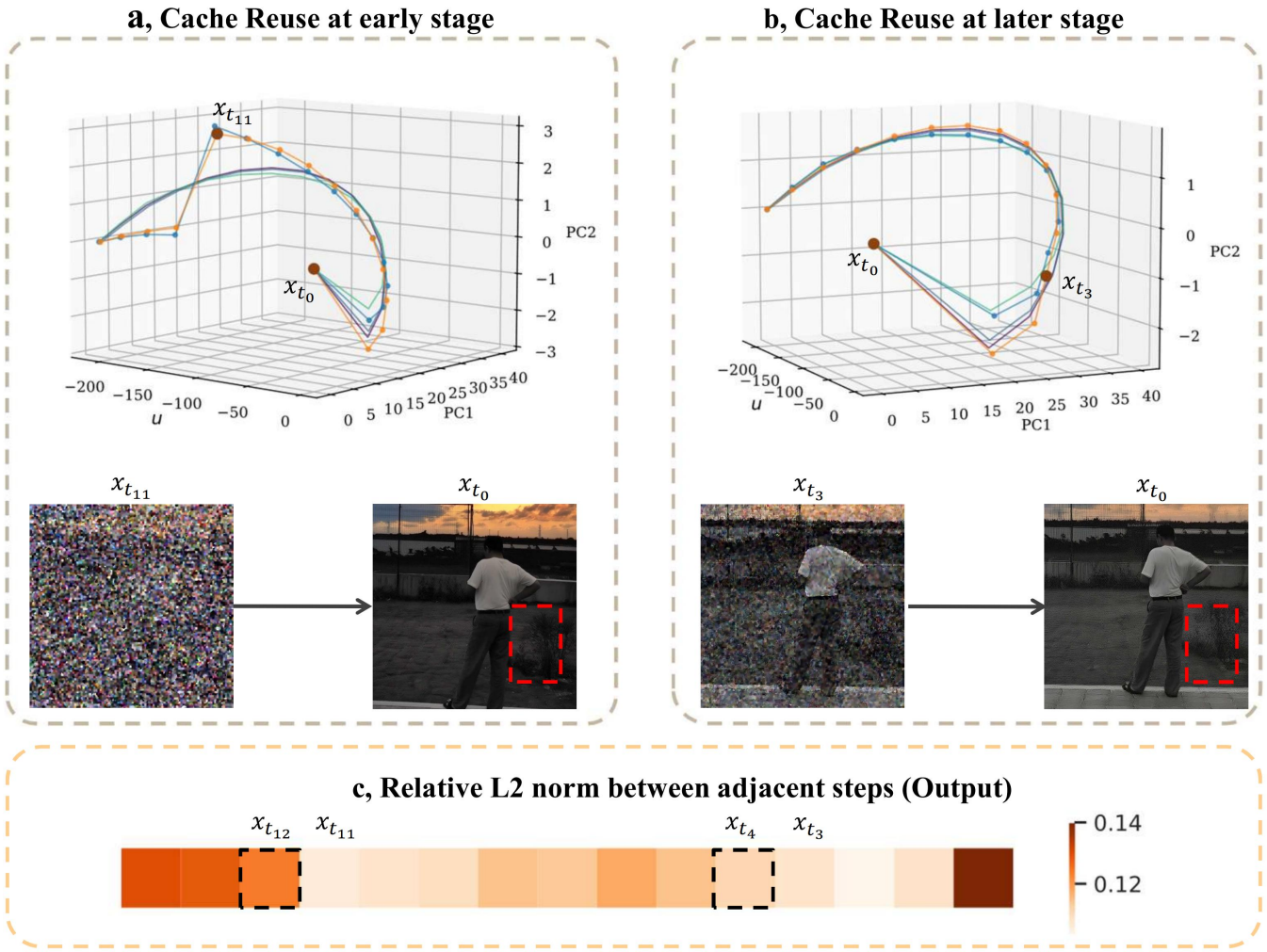
Huanpeng Chu[1], Wei Wu[2], Guanyu Feng[1], Yutao Zhang[1]
[1]Zhipu AI, [2]Nanjing University

# Motivation



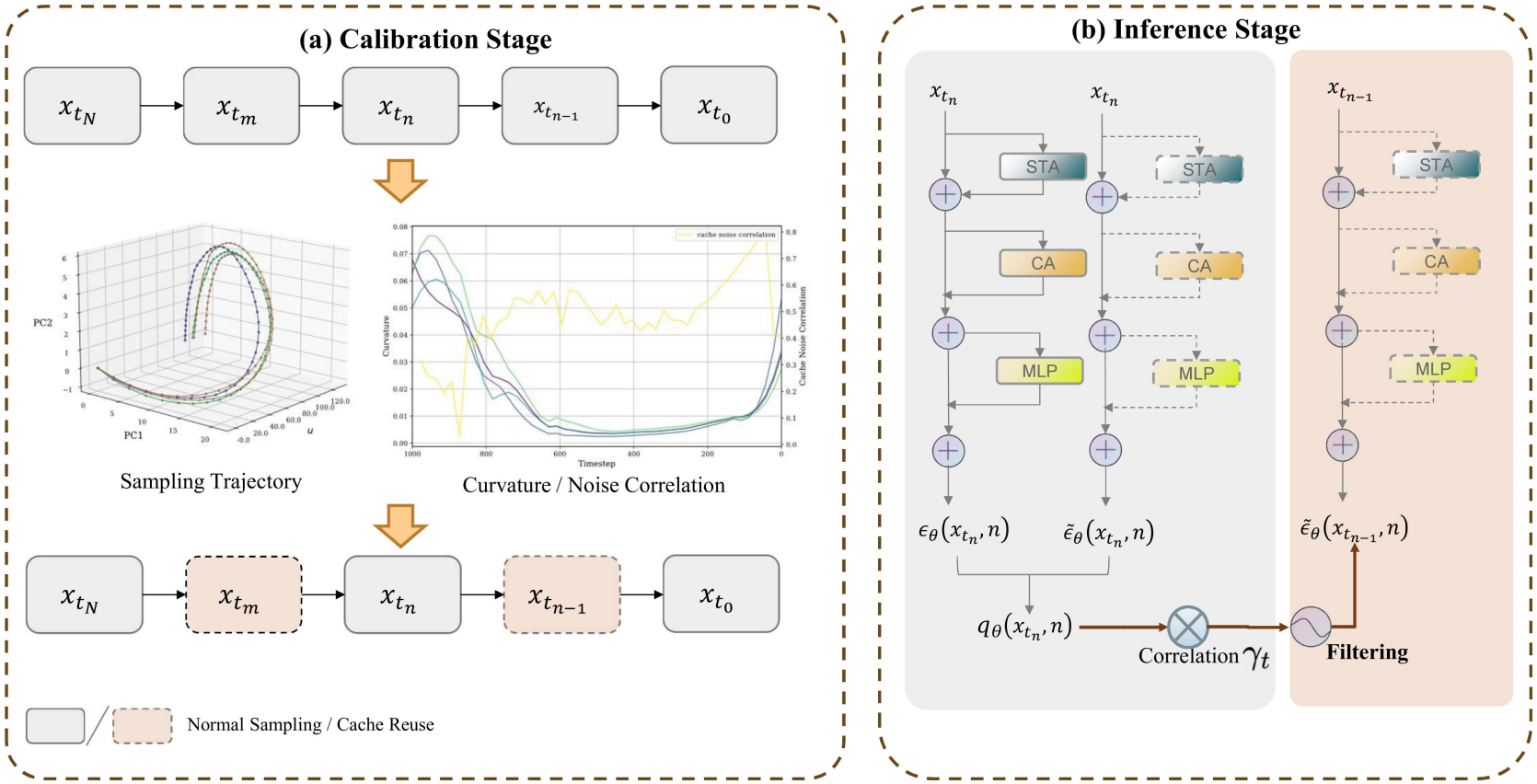

- Diffusion Transformers are slow: many steps, heavy per-step compute.
- Prior accelerations: fewer steps (solvers/distillation) or lighter nets (quantization/sparsity).
- Existing cache-reuse triggers mostly late via local similarity → unstable quality, fragile on distilled models.

# Motivation




Existing similarity-driven cache reuse methods predominantly target the latter half of sampling, at which point the denoiser's corrective capacity is limited, making cache-induced errors difficult to remedy.

# Method




## Trajectory-Guided Cache Reuse

- View sampling as a path from noise→data; compute a simple curvature score per step.
- Select reuse steps by low curvature, not late-only heuristics. Allocate a fixed reuse budget across early/mid phases, enforce a minimum gap.
- What we reuse: DiT attention + MLP activations needed for the denoiser;

# Method

## Cache-induced Noise Correction

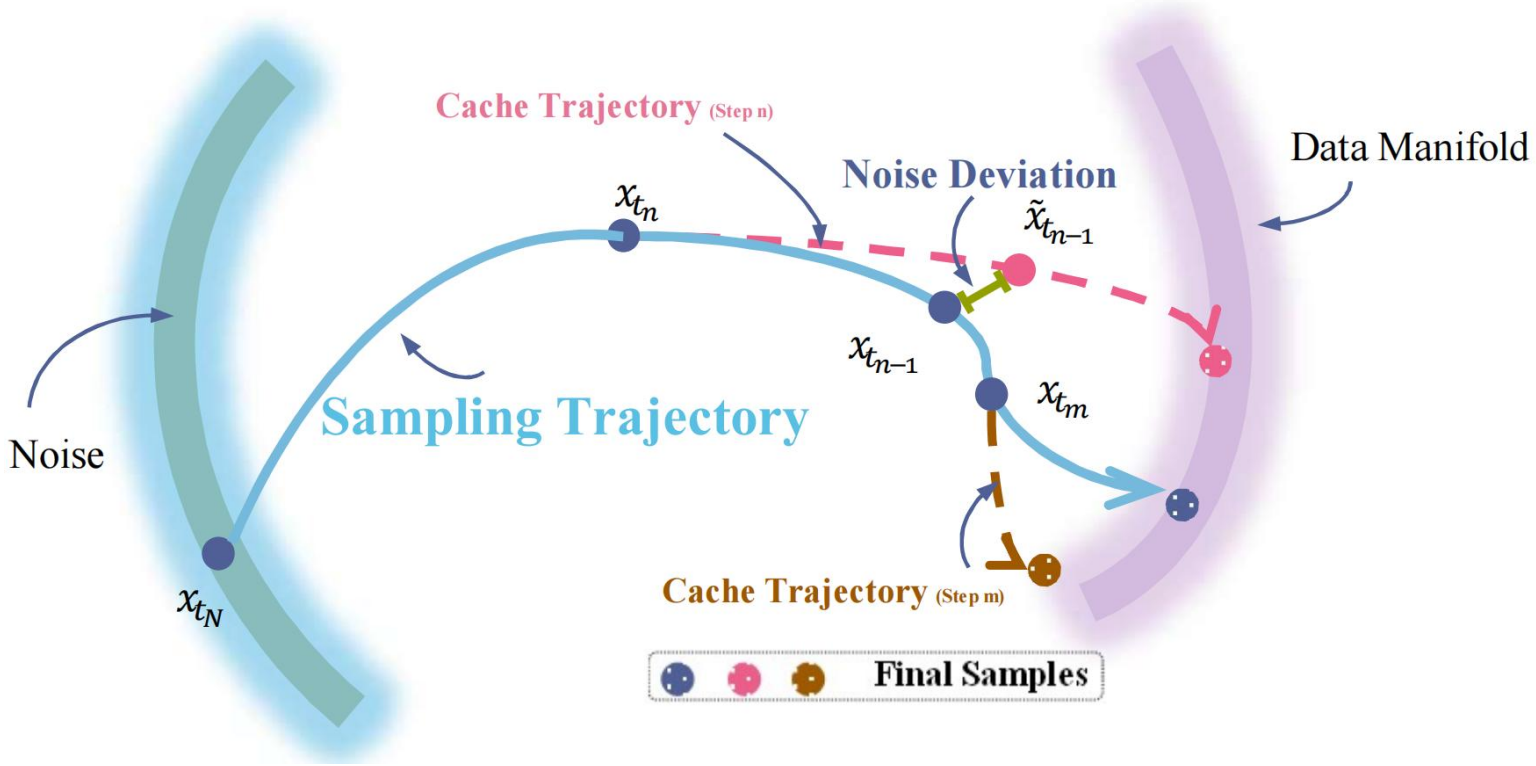
- The sampling process during cache reuse

$$\tilde{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\tilde{\epsilon}_\theta(x_t, t)\right) + \sigma_t z, \quad z \in N(0, I)$$
$$= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\left(\epsilon_\theta(x_t, t) + q_\theta(x_t, t)\right)\right) + \sigma_t z.$$

- Cache-induced noise exhibits correlation and can therefore be corrected.

$$q_\theta(x_{t-1}, t-1) \approx q_\theta(x_t, t) - \frac{d\, q_\theta(x_t, t)}{dt} + \mathcal{O}(\Delta t^2)$$
$$\approx \gamma_{t-1}\, q_\theta(x_t, t),$$

# Method



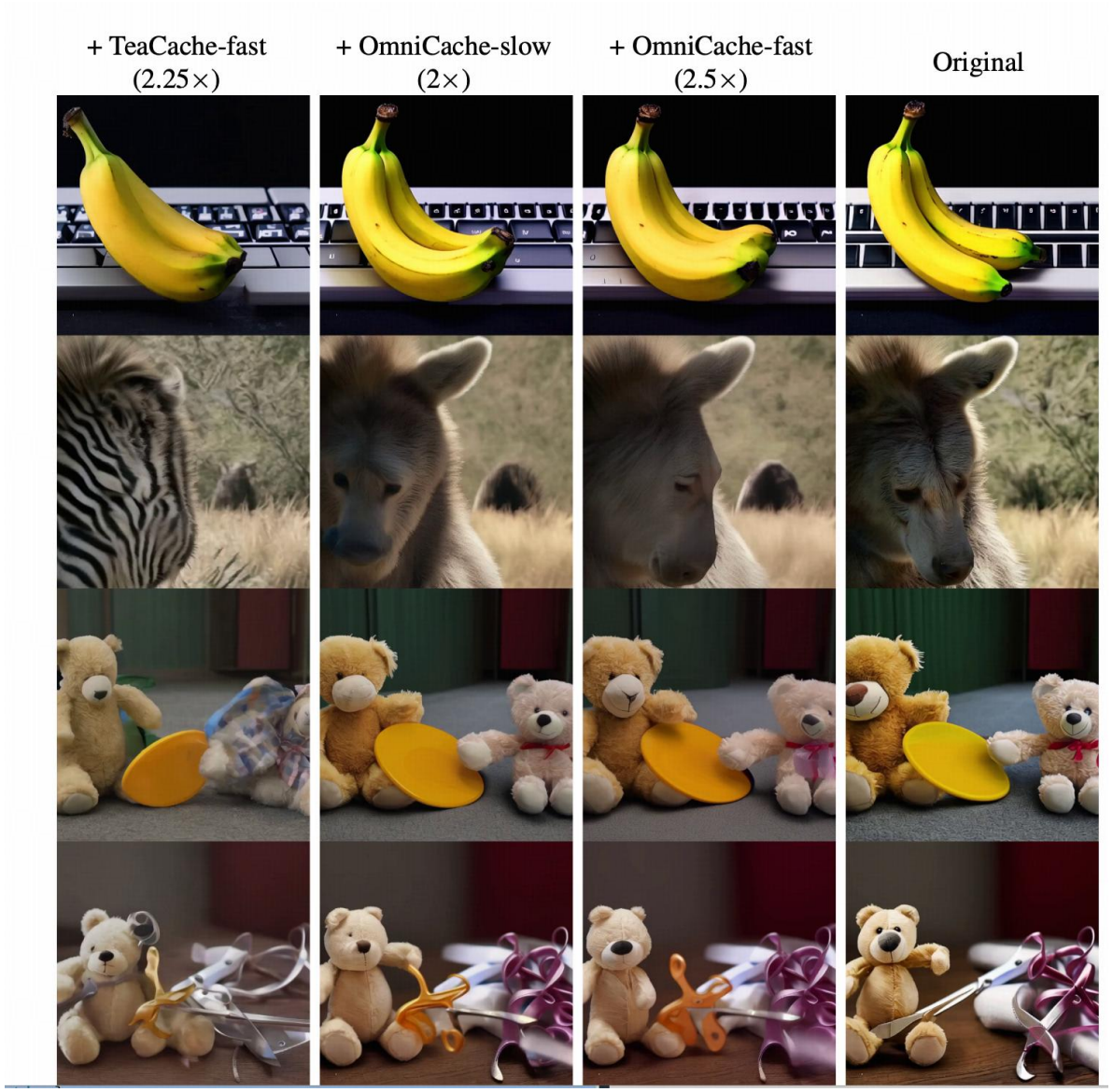

**Cache-induced Noise Correction**

# Experiment Result

| Method | VBench (%) ↑ | PSNR ↑ | LPIPS ↓ | SSIM ↑ | FLOPs (T) | Speedup |
|---|---|---|---|---|---|---|
| Open-Sora | 79.22 | – | – | – | 3230.24 | 1.00× |
| + Δ-DiT | 78.21 | 11.91 | 0.5692 | 0.4811 | 3166.47 | – |
| + T-GATE | 77.61 | 15.50 | 0.3495 | 0.6760 | 2818.40 | 1.10× |
| + PAB-fast | 76.95 | 23.58 | 0.1743 | 0.8220 | 2558.25 | 1.34× |
| + PAB-slow | 78.51 | 27.04 | 0.0925 | 0.8847 | 2657.70 | 1.20× |
| + ToCa(R = 85%) | 78.34 | – | – | – | 1394.03 | 2.36× |
| + TeaCache-fast | 78.48 | 19.10 | 0.2511 | 0.8415 | 1640.00 | 2.25× |
| + OmniCache-slow | 78.83 | 22.37 | 0.1553 | 0.8180 | 1615.12 | 2.00× |
| + OmniCache-fast | 78.50 | 21.27 | 0.1841 | 0.7930 | 1292.10 | 2.50× |
| Latte | 77.40 | – | – | – | 3439.47 | 1.00× |
| + Δ-DiT | 52.00 | 8.65 | 0.8513 | 0.1078 | 3437.33 | – |
| + T-GATE | 75.42 | 19.55 | 0.2612 | 0.6927 | 3059.02 | 1.11× |
| + PAB-fast | 73.13 | 17.16 | 0.3903 | 0.6421 | 2576.77 | 1.33× |
| + PAB-slow | 76.32 | 19.71 | 0.2699 | 0.7014 | 2767.22 | 1.24× |
| + AdaCache-fast | 76.26 | 17.70 | 0.3522 | 0.6659 | 1010.33 | 2.74× |
| + AdaCache-fast (w/ MoReg) | 76.47 | 18.16 | 0.3222 | 0.6832 | 1187.31 | 2.46× |
| + AdaCache-slow | 77.07 | 22.78 | 0.1737 | 0.8030 | 2023.65 | 1.59× |
| + TeaCache-fast | 76.69 | 18.62 | 0.3133 | 0.6678 | 1120.00 | 3.28 × |
| + OmniCache-slow | 77.24 | 22.48 | 0.1955 | 0.7903 | 1719.74 | 2.00× |
| + OmniCache-fast | 77.09 | 21.06 | 0.2463 | 0.7575 | 1375.79 | 2.50× |

Table 1. Quantitative evaluation of text-to-video generation quality. Here, we compare our method, OmniCache, with several training-free, cache-based DiT acceleration approaches on multiple video baselines. It can be observed that our OmniCache-fast scheme strikes a favorable balance between the acceleration ratio and actual performance metrics, incurring only minimal performance degradation with a 2.5× speedup. Corresponding visualizations can be found in Appendix.

# Visualization



# Thank You!