서강대학교
SOGANG UNIVERSITY

2026 하계 세미나

Diffusion Transformer Acceleration

2026. 07. 16.

Presented By
김현준

Outline

- Background
  - Diffusion Transformer
  - Training-free acceleration
- Paper Review
  - Rethinking Token-Wise Feature Caching: Accelerating Diffusion Transformers With Dual Feature Caching [TIP 2026]
  - TAP: A Token-Adaptive Predictor Framework for Training-Free Diffusion Acceleration [CVPR 26]

Background

- Diffusion Transformer
  - Diffusion에서 Unet 구조 대신 latent를 patch로 쪼개 transformer block을 통해 전역적 feature 추출
  - Timestep, class label을 transformer block에 주입해 생성 과정 제어
  - DiT가 현재 latenet에 포함된 노이즈를 예측

$$-\hat{\epsilon}_t = \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}),$$

  - Sampler가 예측한 노이즈를 통해 Z_{t-1}을 예측

$$-\mathbf{z}_{t-1} \approx \frac{\mathbf{z}_t - \sqrt{\beta_t}\,\hat{\epsilon}_t}{\sqrt{1-\beta_t}}$$

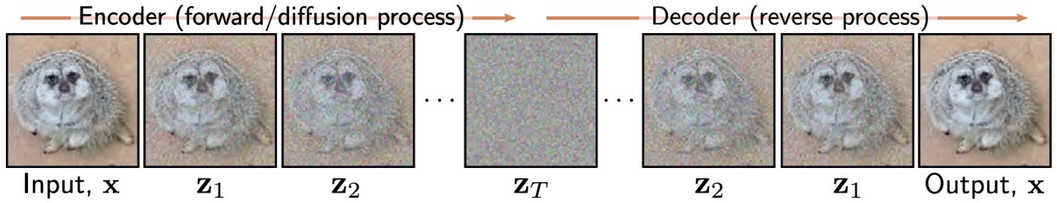
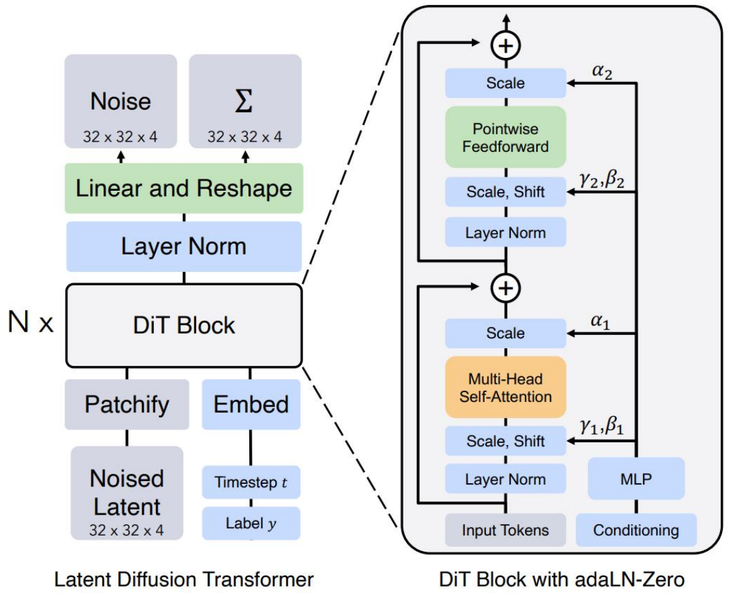
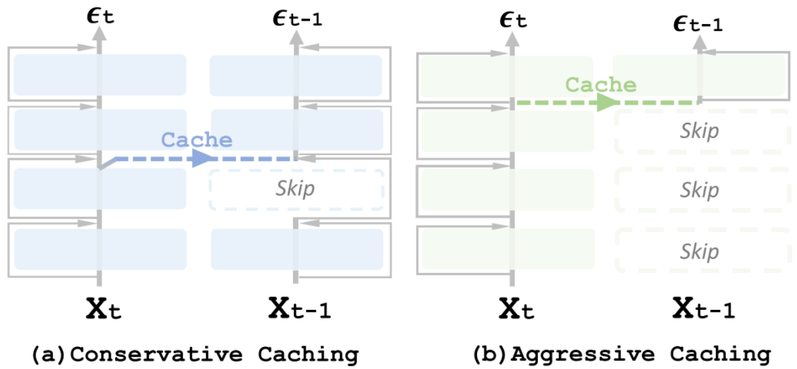


Figure 18.1 Diffusion models. The encoder (forward, or diffusion process) maps the input $\mathbf{x}$ through a series of latent variables $\mathbf{z}_1 \ldots \mathbf{z}_T$. This process is pre-specified and gradually mixes the data with noise until only noise remains. The decoder (reverse process) is learned and passes the data back through the latent variables, removing noise at each stage. After training, new examples are generated by sampling noise vectors $\mathbf{z}_T$ and passing them through the decoder.

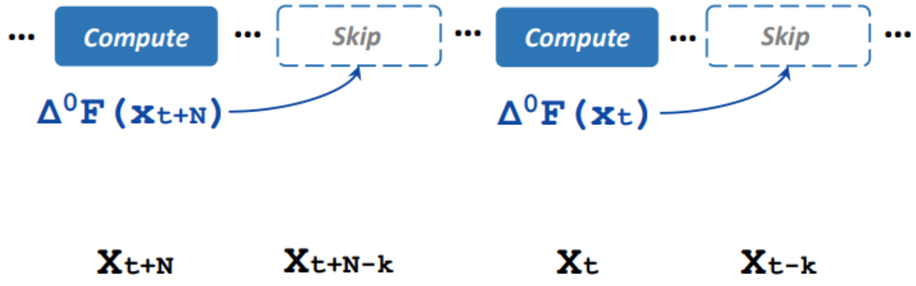


Background

- Training-free acceleration
  - DiT가 무거운 만큼 fine-tuning, 모듈 도입 등에 막대한 연산 비용 발생
    - 모델 확장성 제한
  - Caching-based
    - 과거 timestep의 feature를 그대로 재사용
    - $h_t \approx h_{t-1}$
  - Token-wise caching
    - 매 timestep 변화가 크다고 판단되는 중요 token만 새로 계산
  - Predictor-based
    - 과거 feature들의 변화 추세로 현재 feature를 예측
    - $h_t \approx h_{t-1} + \Delta h$



< Token-wise caching >



< Predictor-based acceleration>

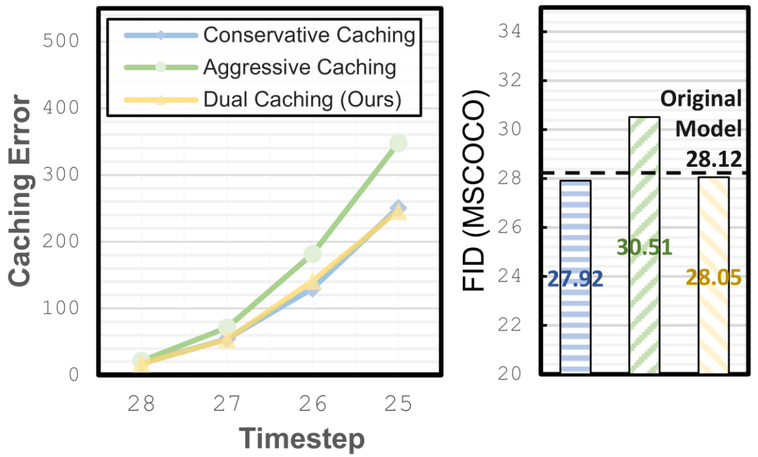
Paper Review

Rethinking Token-Wise Feature Caching: Accelerating Diffusion Transformers With Dual Feature Caching [TIP 2026]

Chang Zou, Shikang Zheng , Evelyn Zhang, Runlin Guo, Haohang Xu, Zhengyi Shi, Conghui He , Xuming Hu, and Linfeng Zhang

Introduction

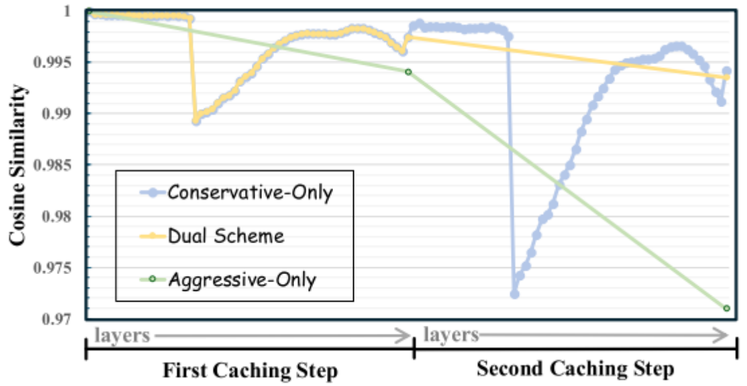
- Motivation
  - Token-wise caching의 가정
    - 중요 token은 매 timestep마다 계산해 갱신해야 함
    - Importance score로 선택된 token은 실제로 중요함
  - Insight
    - 중요 token은 보수적 caching 직후 timestep에는 굳이 필요하지 않다.
      - 기존 방법론에 의한 보수적 caching VS 공격적으로 마지막 layer 1개 제외 모두 caching
        - ✓첫 timestep에 대해서는 두 결과가 비슷함
      - Importance score 기반 selection 보다 random selection이 더 낫다.



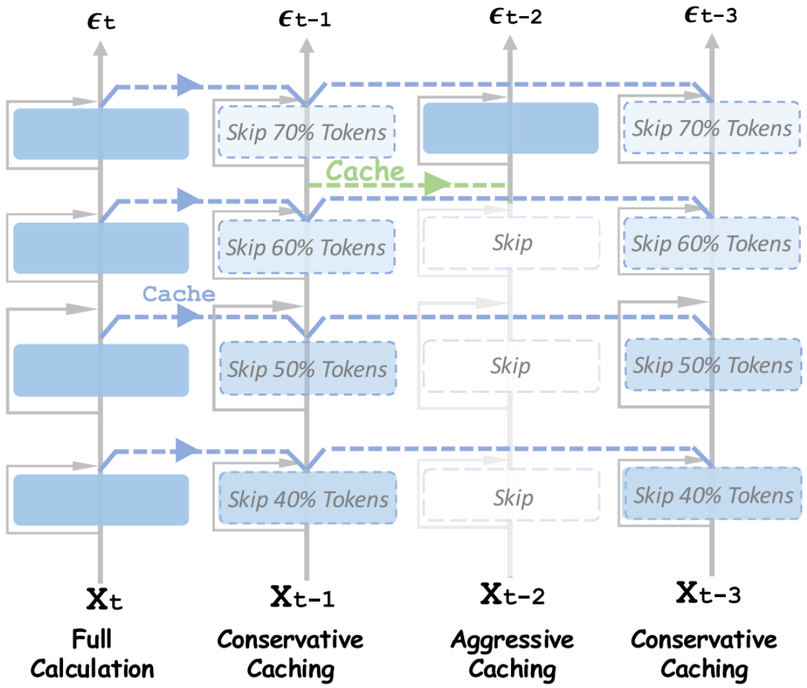
< Cashing Error Comparison>

Method

- Dual Feature Caching
  - Conservative 이후 aggressive를 했을 때 성능 저하가 적음
    - 두 방법을 번갈아가며 적용 (Dual)
  - Aggressive caching
    - 이전 layer의 input이 없으므로 이전 timestep의 전체 feature를 caching
  - Conservative caching
    - ToCa의 방법론에 따라 Residual만 caching



< Cosine Similarity of DiT residual >



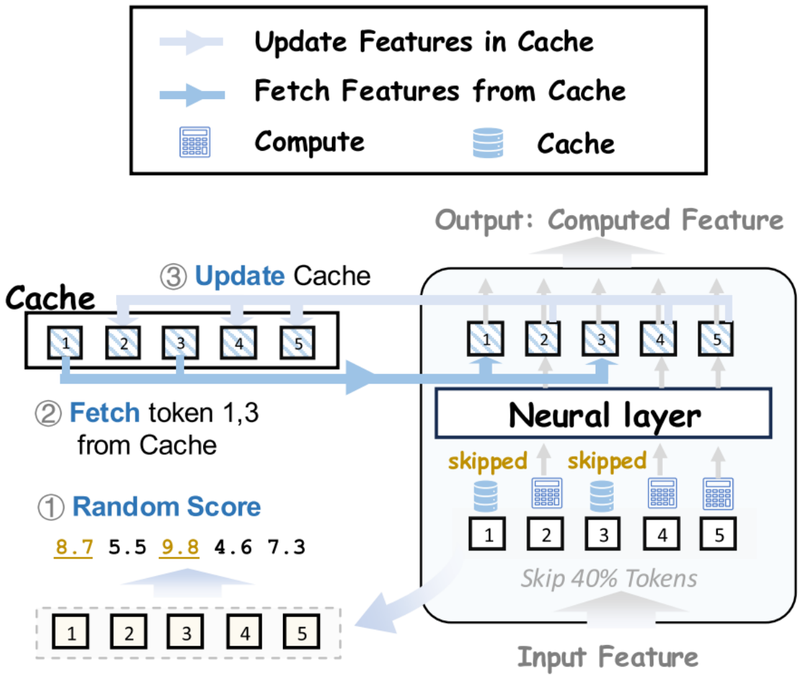
< Dual Feature Caching>

Methodology

- Random score selection
  - 10%의 token을 caching할 때 성능 비교
    - 1%는 random select, 9%를 Attention, K-norm, V-norm, Similarity가 최대, 최소일 때를 비교
      - ∴ 유사도가 먼 token을 제거하는 것이 좋다. (최소일 때 성능이 가장 좋기 때문)
    - Attention의 경우 이러한 방법 중 가장 성능이 높았으나, Random이 더 성능이 좋음
      - ∴ Attention base가 유사도가 먼 token들을 선택하지 못한다
      - ∴ Random의 경우 어느정도 보장되며 cost도 아낄 수 있음

Score	Efficient Attention	Max♠	Min♡
Attention	✘	0.9798	0.9707
K-norm	✔	0.9783	0.9726
V-norm	✔	0.9747	0.9798
Similarity	✔	0.9715	0.9811
Random	✔	0.9806	
Original Model	✔	0.9898	

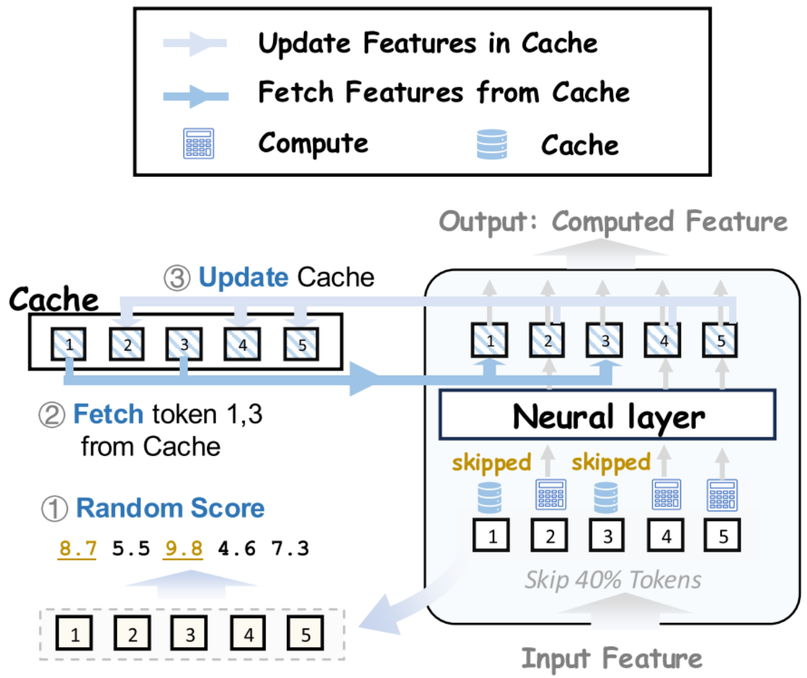
< Token Selection Methods comparison>



< Conservative Caching with Random Score >

Methodology

- Random score selection
  - 각 token 별로 난수를 생성해 정렬 후 비율에 따라 token 선택
  - Residual이 없는 token은 update, 있는 token은 유지
  - 각 layer 별 token pruning과 같은 효과



< Conservative Caching with Random Score >

Experiments

- Model configuration
  - Text to Image (FLUX.1-dev, HiDream, PixArt-α)
  - Text to Video (OpenSora)
  - Class-Conditional Image Generation (DiT-XL/2)
- Metric
  - Text to Image
    - DrawBench 200개
      - ImageReward (생성 이미지의 전반적인 시각적 품질과 사람의 선호도 평가)
      - GenEval (생성 이미지가 입력 텍스트의 객체, 개수, 속성 및 관계의 반영도 평가)
  - Text to Video
    - Vbench
  - Class-Conditional Image Generation
    - ImageNet 1000개 class 에서 50장 씩 평가
    - FID-50k (전체 분포), sFID (공간적 특징), Precision, Recall

Experiments

- Quantitative Result
  - 생성 품질
    - DuCa는 3.45× FLOPs 가속에서도 ImageReward 0.9896을 유지하며, ToCa 대비 품질 저하를 크게 줄임
  - 텍스트–이미지 정렬
    - 최대 4.05× 연산 압축에서도 GenEval 성능을 거의 손실 없이 유지
  - 실제 추론 속도
    - FlashAttention과 호환되어 2.93× latency 가속을 달성하며, ToCa의 1.59×를 압도

Method FLUX.1 [37]	Efficient Attention [36]	Acceleration				Image Reward ↑ DrawBench	Geneval↑ Overall
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
[dev]: 50 steps	✔	17.20	1.00×	3719.50	1.00×	0.9898	0.6752
60% **steps**	✔	10.49	1.64×	2231.70	1.67×	0.9739	0.6700
50% **steps**	✔	8.80	1.95×	1859.75	2.00×	0.9429	0.6655
40% **steps**	✔	7.11	2.42×	1487.80	2.62×	0.9317	0.6606
Δ-DiT ($N = 2$)	✔	11.86	1.45×	2480.01	1.50×	0.9444	0.6658
Δ-DiT ($N = 3$)	✔	8.68	1.98×	1686.76	2.21×	0.8721	0.6413
TeaCache	✔	7.41	2.32×	1114.44	2.63×	0.9344	0.6722
FORA [8]	✔	7.08	2.43×	1320.07	2.82×	0.9227	0.6594
ToCa [9]	✘	10.80	1.59×	1126.76	3.30×	0.9731	0.6750
DuCa ($N = 4, R = 90\%$)	✔	6.22	2.77×	1190.24	3.13×	<u>0.9895</u>	<u>0.6750</u>
DuCa ($N = 5, R = 90\%$)	✔	<u>5.88</u>	<u>2.93×</u>	<u>1078.34</u>	<u>3.45×</u>	**0.9896**	0.6747
DuCa ($N = 6, R = 90\%$)	✔	**5.14**	**3.35×**	**917.32**	**4.05×**	0.9654	**0.6756**

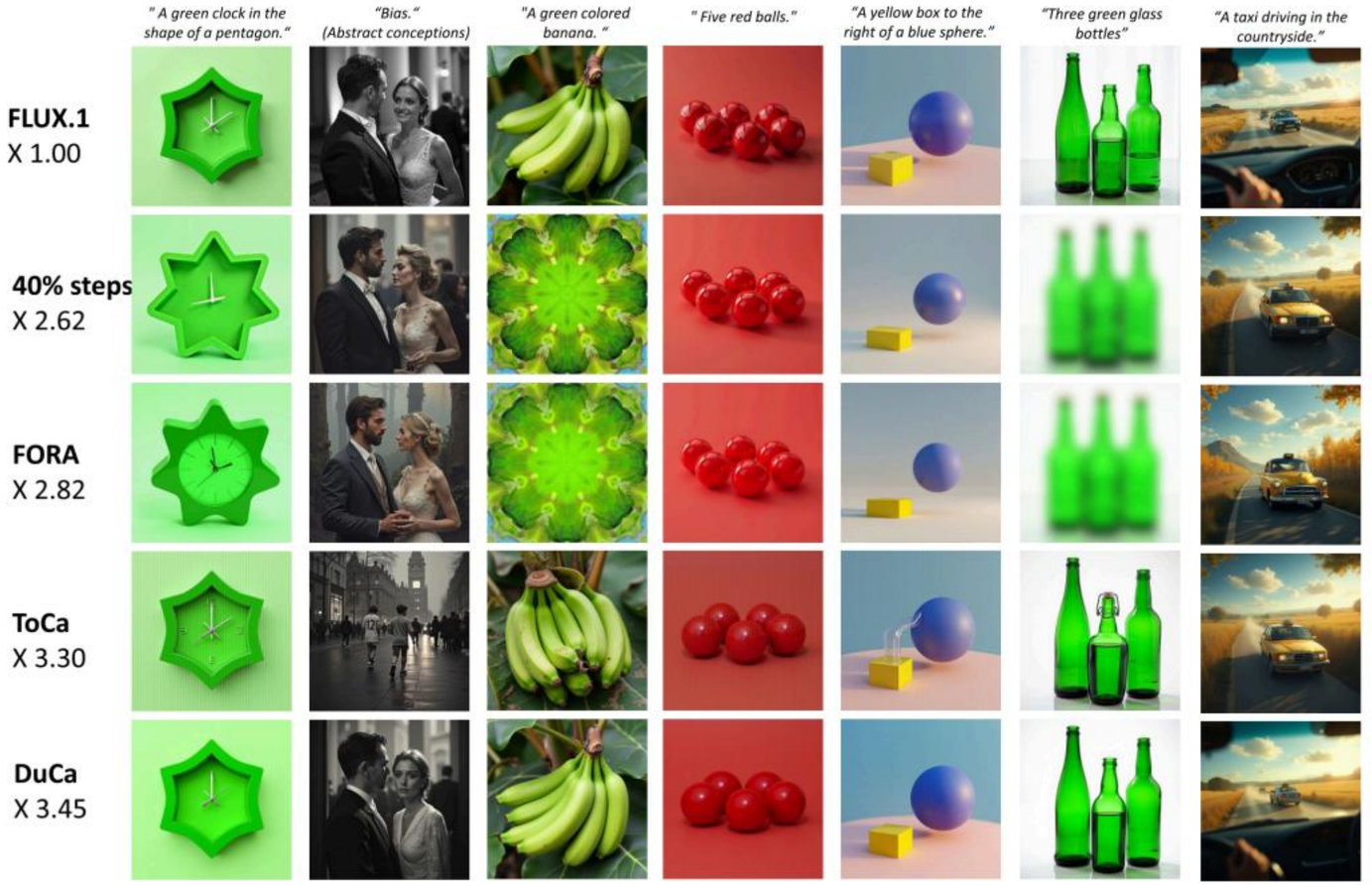
< Quantitative Results on ImageReward for FLUX>

Method HiDream-l1	Acceleration			Image Reward ↑ DrawBench
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	
[full]:50 steps	48.81	×1.00	7780.0	1.182
25 steps	24.41	×2.00	3890.0	1.128
ToCa	31.22	×1.56	2540.1	1.111
PAB	36.33	×1.34	5962.0	0.8876
DuCa(a)	19.88	×2.46	2873.4	1.152
DuCa(b)	15.36	×3.18	2197.3	1.090

< Quantitative Results For HIDREAM-L1>

Experiments

- Qualitative Result



< Visualization results for different acceleration methods on FLUX.1-dev model>

1) Zhu, Haowei, et al. "Tap: A token-adaptive predictor framework for training-free diffusion acceleration." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2026.

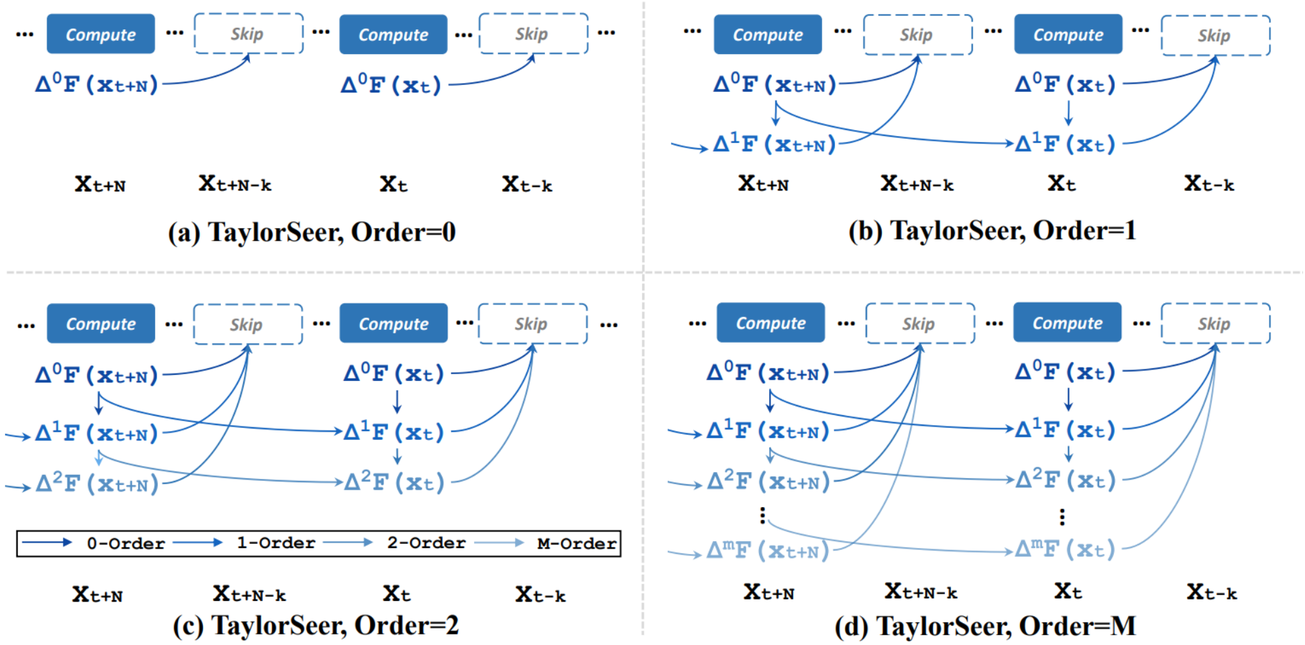
Paper Review

TAP: A Token-Adaptive Predictor Framework for Training-Free Diffusion Acceleration [CVPR 26]

Haowei Zhu, Tingxuan Huang, Xing Wang, Tianyu Zhao, Jiexi Wang, Weifeng Chen, Xurui Peng, Fangmin Chen, Junhai Yong, Bin Wang

Introduction

- Taylor predictor
  - 기존 prediction 방식
    - 가까운 timestep의 변화량은 미미하기 때문에 다음 timestep도 결과가 비슷할 것이다
      - 가까운 timestep에 대해 DiT의 결과 $\hat{\epsilon}_t$를 그대로 t-1 timestep의 결과물이라 근사
    - Feature-caching과 달리 $\hat{\epsilon}_t$ 그 자체를 예측해 다음 timestep의 DiT block을 Skip 함
  - Taylor predictor는 taylor 근사를 통해 더 나은 근사를 하는 방식

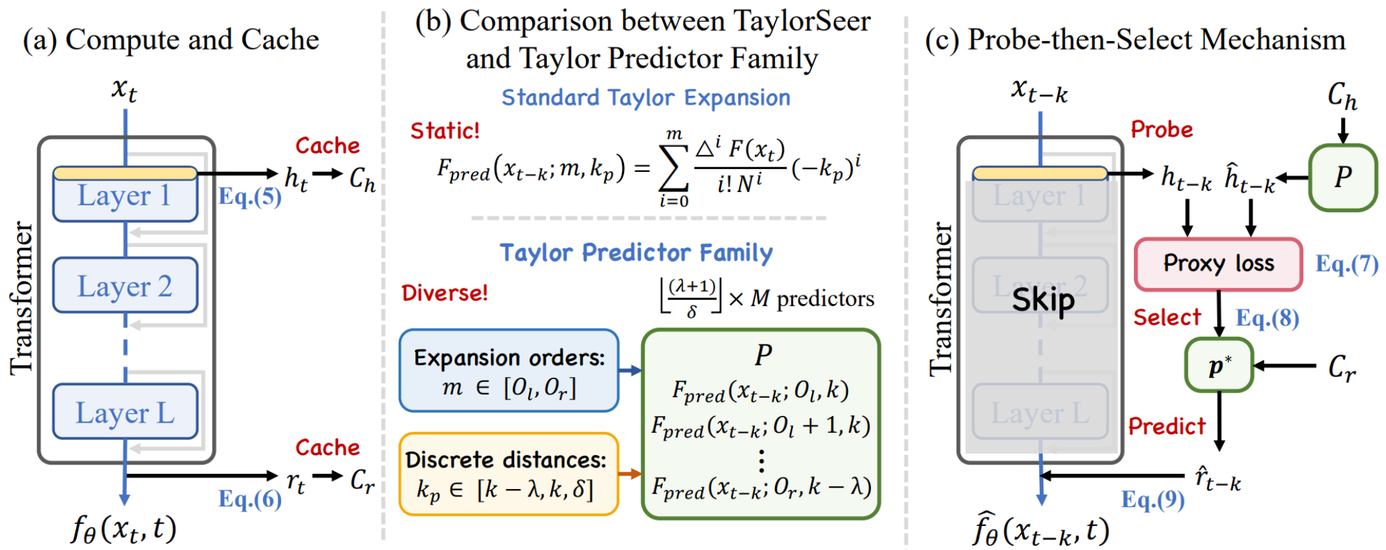


(a) TaylorSeer, Order=0 (b) TaylorSeer, Order=1

(c) TaylorSeer, Order=2 (d) TaylorSeer, Order=M

Methodology

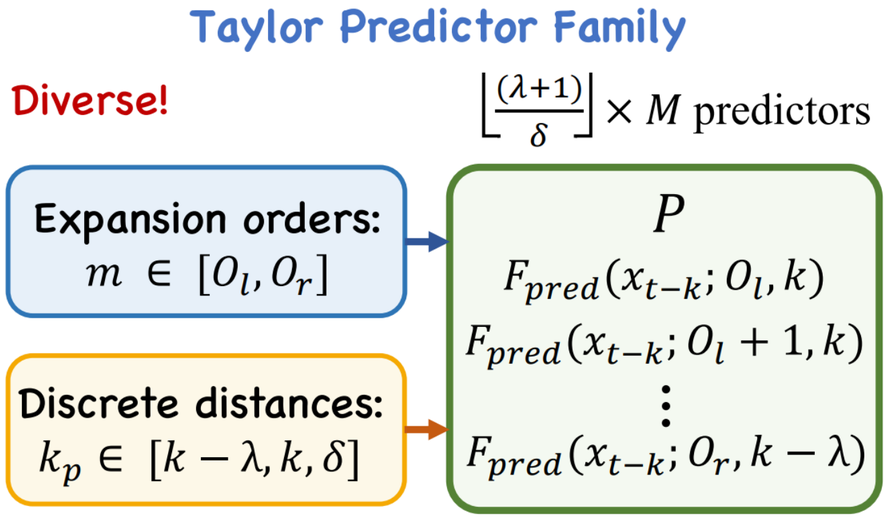
- TAP
  - Token마다 timestep에 따른 feature 변화 속도와 temporal dynamics가 다름
    - 모든 token에 똑같은 predictor는 비효율적이다.
  - 각 토큰의 변화 양상에 따라 다른 Predictor를 사용
  - 정확도 및 효율성을 크게 향상



< Pipeline of TAP >

Methodology

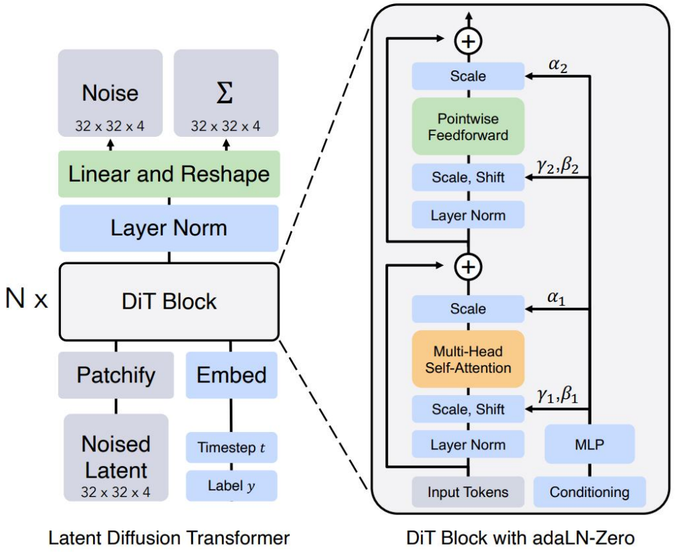
- Taylor Predictor Family 구성
  - λ : Maximum prediction horizon
  - δ : Step size between prediction−horizon candidates
  - M : Number of available Taylor orders
  - TAP에서는 $M = 3, \lambda = 4, \delta = 1$로 고정
    - 0,1,2 차 taylor series
    - 0, 1, 2, 3, 4의 K
      - e.g. K가 3이면 predictor는 3 timestep 이후의 결과값을 예측
    - 총 15개의 predictor로 family를 생성



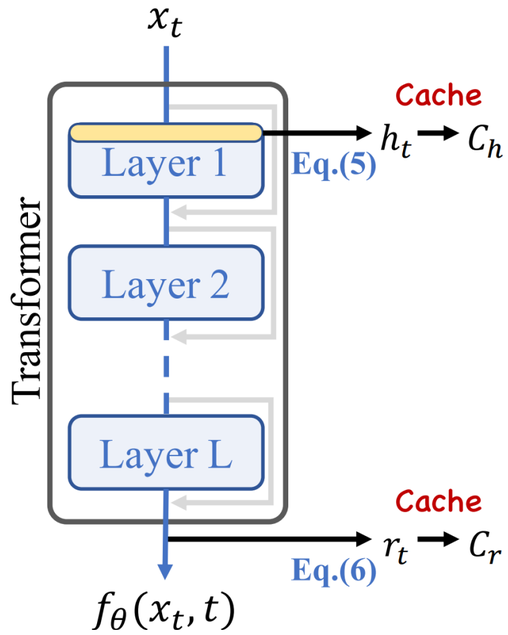
< Taylor Predictor Family >

Methodology

- Compute and Cache
  - 초기 layer는 생성 품질에 중요
    - Full computing on 3-timestep, FLUX.1-schnell은 2-timestep
  - N-timestep마다 full computing
    - 이때 MHSA 입력 전 값(h_t)를 계산 후 Cache
      - 이걸 본 논문에서는 probe라고 함
      - 기존 연구에서 shallow-layer activation이 전체 feature를 잘 나타낸다고 함
    - DiT block 최종 결과물 residual(r_t)을 Cache



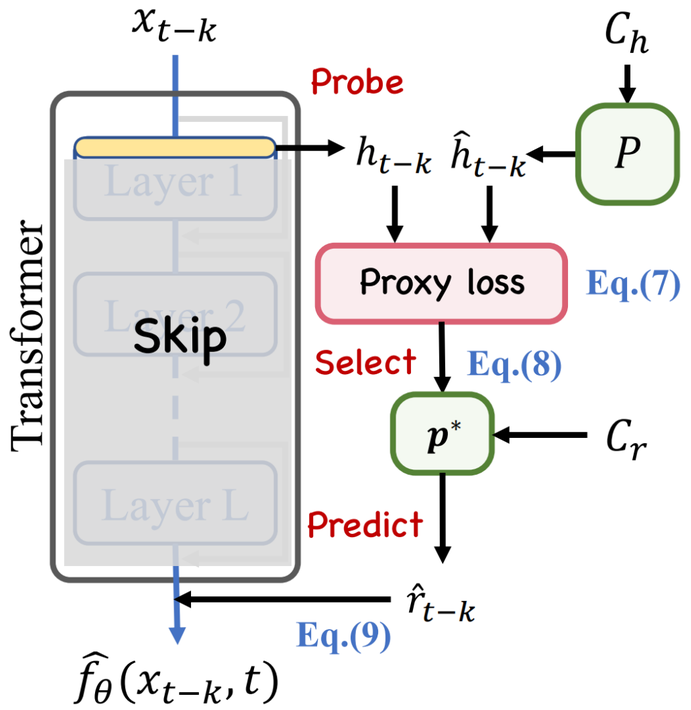
< Diffusion Transformer Architecture>



< Compute and Cache >

Methodology

- Predictor selection
  - Proxy loss estimation
    - 각 token에 대해 caching 한 probe를 15개의 predictor에 통과시켜 Full computing과 비교
      - 이때 skip block에서도 probe 계산까지는 full computing
      - 본 논문에서는 cosine distance를 사용
  - 가장 distance가 가까운 predictor를 선택 후 cache에 있는 DiT 결과물을 입력
    - 출력으로 현재 timestep의 최종 결과물을 근사



< Probe-then-Select Mechanism>

Experiments

- Quantitative Results

Table 1. **Quantitative comparison in text-to-image generation** for FLUX.1-dev and FLUX.1-schnell (a distilled version). Best results are highlighted in **bold**, and second-best are underlined.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	ImageReward↑	CLIP↑	PSNR↑	SSIM↑	LPIPS↓
50 steps	42.18	1.00×	3726.87	1.00×	0.95	30.63	-	-	-
40% steps	17.08	2.47×	1490.75	2.50×	**0.93**	**30.84**	**15.60**	**0.69**	**0.40**
30% steps	13.14	3.21×	1118.06	3.33×	0.92	30.70	14.81	0.66	0.44
20% steps	**8.52**	**4.95**×	**745.37**	**5.00**×	0.89	30.28	13.96	0.62	0.50
FORA ($N = 4$)	12.37	3.41×	969.93	3.84×	0.87	30.45	14.28	0.64	0.47
TeaCache ($l = 1.1$)	12.33	3.42×	913.98	4.08×	0.93	30.47	15.19	0.65	0.47
TaylorSeer ($N = 5, O = 2$)	12.02	3.51×	895.32	4.16×	0.95	30.79	16.73	0.71	0.36
SpeCa ($N = 2, M = 8$)	**11.68**	**3.61**×	**845.41**	**4.41**×	0.96	30.73	16.09	0.68	0.40
TAP ($N = 5$)	11.82	3.57×	895.08	4.16×	**0.97**	**30.86**	**17.43**	**0.71**	**0.34**
FORA ($N = 5$)	10.12	4.17×	746.29	4.99×	0.89	30.55	13.82	0.61	0.50
TeaCache ($l = 1.4$)	11.16	3.78×	779.83	4.77×	0.92	30.44	14.79	0.63	0.51
TaylorSeer ($N = 6, O = 2$)	**10.02**	**4.21**×	746.29	4.99×	0.94	30.76	15.88	0.67	0.42
SpeCa ($N = 4, M = 8$)	10.76	3.92×	755.32	4.93×	0.93	30.57	15.72	0.66	0.43
TAP ($N = 6$)	10.21	4.13×	**746.02**	**5.55**×	**0.96**	**30.92**	**16.76**	**0.68**	**0.39**
FORA ($N = 7$)	7.91	5.33×	597.26	6.24×	0.80	30.42	13.43	0.60	0.55
TeaCache ($l = 2.0$)	8.59	4.91×	604.02	6.17×	0.66	30.07	14.23	0.60	0.58
TaylorSeer ($N = 8, O = 2$)	8.17	5.16×	597.26	6.24×	0.91	30.62	14.72	0.61	0.50
TAP ($N = 8$)	8.40	5.02×	596.99	6.24×	**0.99**	**31.19**	**16.11**	**0.64**	**0.44**
TAP ($N = 10$)	**6.21**	**6.79**×	**522.48**	**7.13**×	0.91	30.69	15.94	0.63	0.49
FLUX-Schnell (4 steps)	4.46	1.00×	278.41	1.00×	0.88	32.03	-	-	-
TaylorSeer ($N = 3, O = 2$)	2.61	1.71×	139.23	2.00×	0.86	32.08	22.67	0.72	0.32
TAP ($N = 3$)	**2.55**	**1.75**×	**139.12**	**2.00**×	**0.88**	**32.36**	**26.26**	**0.86**	**0.15**

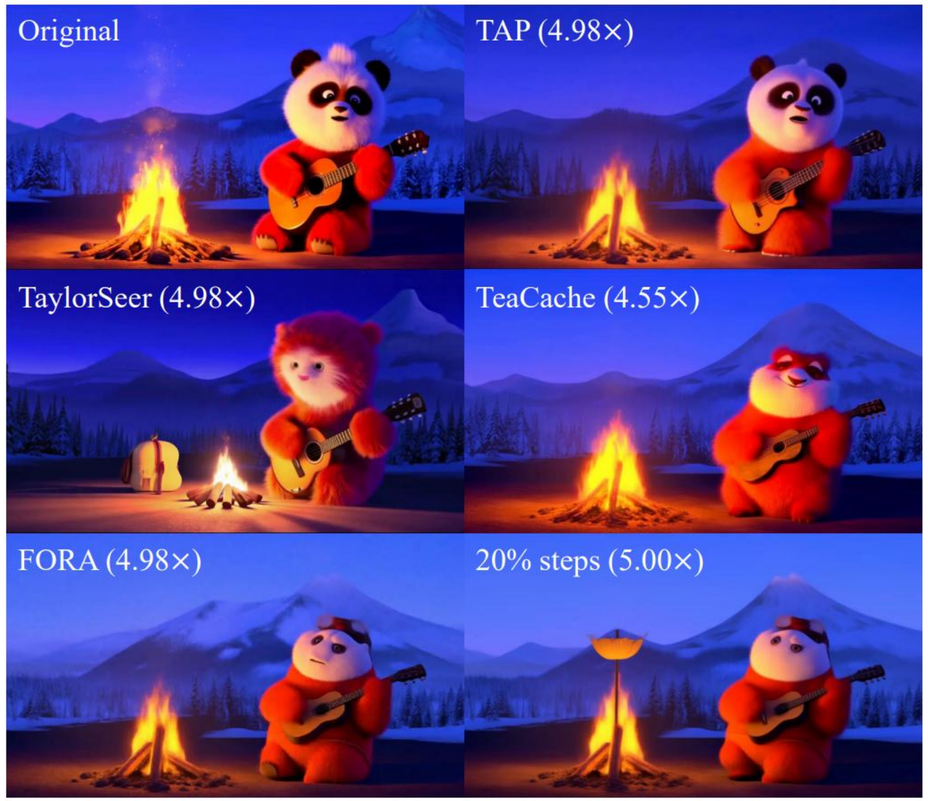
Table 2. **Quantitative comparison in text-to-image generation** for Qwen-Image and Qwen-Image-Lightning (a distilled version). Best results are highlighted in **bold**, and second-best are underlined.

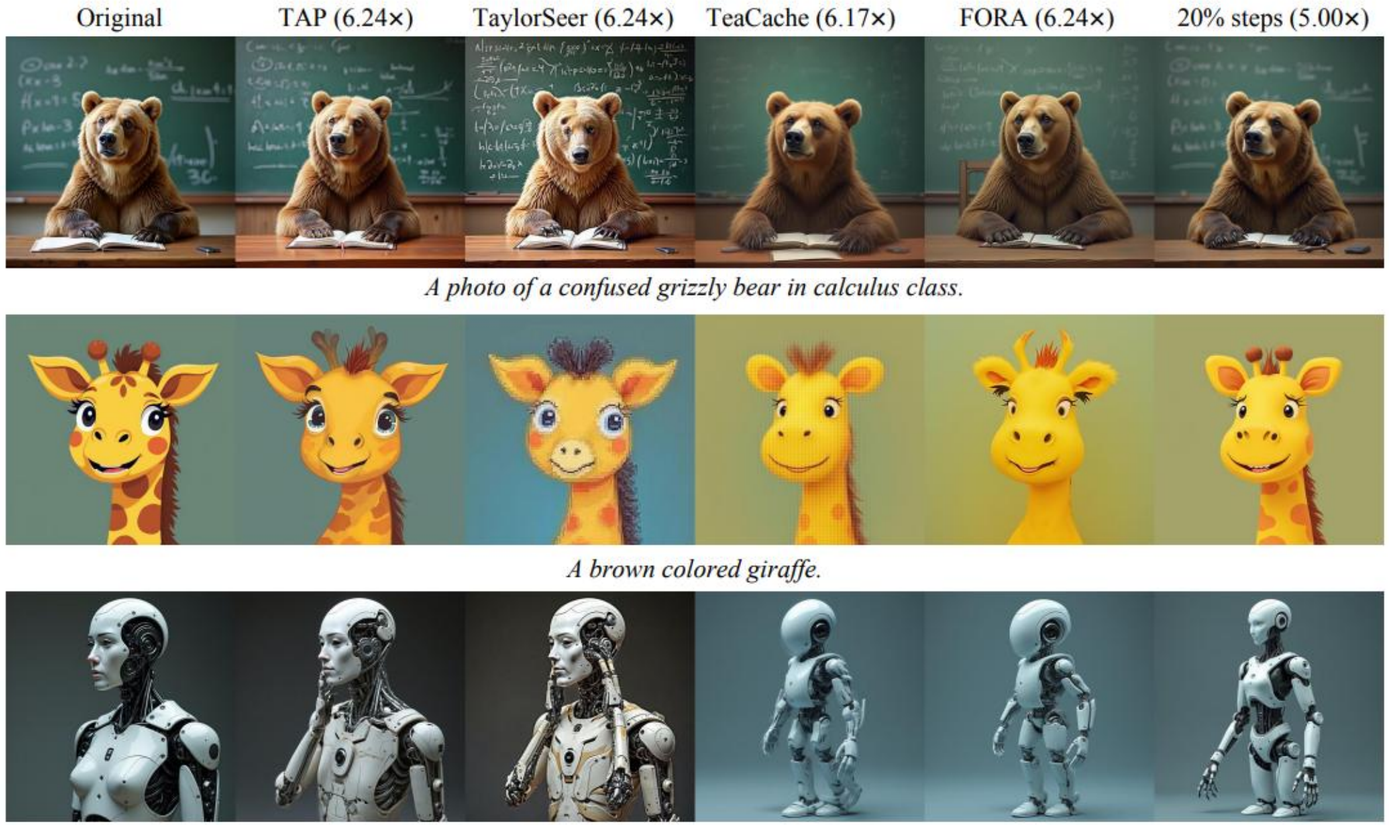
Method	Acceleration			Quality Metrics		Perceptual Metrics		
	Latency (s) ↓	FLOPs (T) ↓	Speed ↑	ImageReward ↑	CLIP ↑	PSNR ↑	SSIM ↑	LPIPS ↓
50 steps	127.10	12917.56	-	1.23	33.74	-	-	-
50% steps	63.54	6458.78	2.00×	**1.20**	**33.77**	**18.35**	**0.77**	**0.28**
20% steps	**25.54**	**2583.51**	**5.00**×	0.73	32.24	13.45	0.57	0.57
FORA ($N = 3$)	48.59	4392.93	2.94×	0.92	32.25	14.12	0.59	0.50
TeaCache ($l = 0.8$)	52.64	3621.18	3.57×	1.18	33.52	19.07	0.79	0.27
TaylorSeer ($N = 4, O = 2$)	44.37	3617.96	3.57×	1.18	33.44	18.02	0.76	0.30
TAP ($N = 4$)	**40.96**	**3617.43**	**3.57**×	**1.23**	**33.80**	**20.13**	**0.81**	**0.22**
FORA ($N = 4$)	38.41	3359.99	3.84×	0.70	31.98	13.13	0.53	0.58
TeaCache ($l = 1.0$)	47.84	3276.45	3.94×	1.16	33.47	18.13	0.73	0.34
TaylorSeer ($N = 5, O = 2$)	40.70	3101.30	4.17×	1.16	33.37	16.18	0.70	0.39
TAP ($N = 5$)	**34.21**	**3100.76**	**4.17**×	**1.19**	**33.55**	**18.72**	**0.78**	**0.27**
Qwen-Image-Lightning (8 steps)	7.46	560.96	-	1.30	33.06	-	-	-
TaylorSeer ($N = 5, O = 2$)	**4.68**	280.54	2.00×	1.21	33.03	17.59	0.70	0.27
TAP ($N = 5$)	4.80	**280.50**	**2.00**×	**1.27**	**33.50**	**21.11**	**0.77**	**0.23**

Method	Latency (s) ↓	FLOPs (T)	VBench (%) ↑
50 steps	121.32	$5481.50_{(1.00\times)}$	66.61
50% steps	**60.96**	$\mathbf{2740.75}_{(2.00\times)}$	66.38
20% steps	24.80	$1096.30_{(5.00\times)}$	**64.16**
FORA ($N = 5$)	31.68	$1101.72_{(4.98\times)}$	63.87
TeaCache ($\ell = 0.4$)	32.33	$1205.42_{(4.55\times)}$	65.13
TeaCache ($\ell = 0.5$)	**25.53**	$\mathbf{991.96}_{(5.53\times)}$	64.28
Taylor ($N = 6, O = 2$)	33.67	$1101.72_{(4.98\times)}$	64.89
TAP ($N = 6$)	31.91	$1101.47_{(4.98\times)}$	**65.46**

Experiments

- Qualitative Results

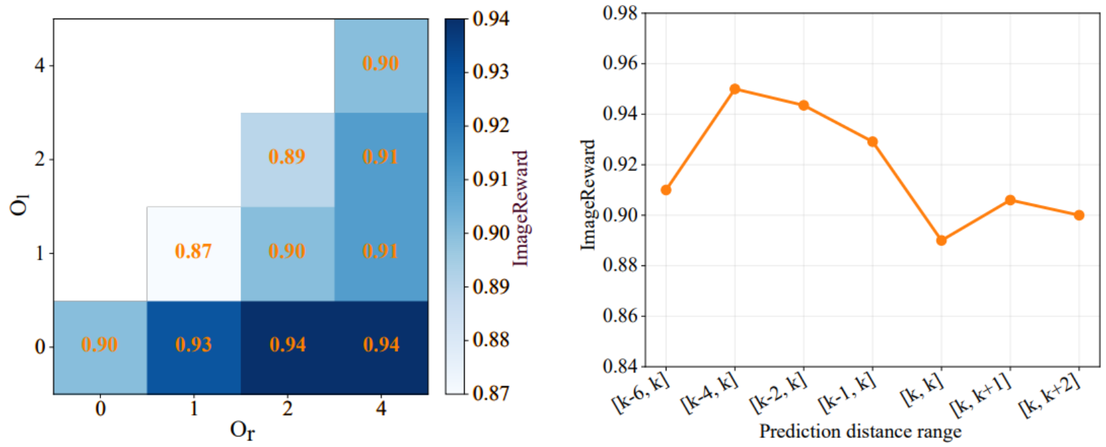




< Comparison with SOTA >

Experiments

- Ablation Study
  - Taylor predictor family의 구성에 따른 효과
    - Family를 확장하며 성능향상을 얻지만, Saturation
      - 본 논문의 세팅이 Saturation 직후 세팅
    - 세분화 실험 진행 시 0.005 ImageReward의 미미한 효과
      - δ : Step size between prediction–horizon candidates
        - ✓δ $1 \rightarrow 0.1$



(a) Effect of prediction order. (b) Effect of predictor distance.

< Ablation on Taylor predictor family>

감사합니다