2026 하계 세미나

Efficient Diffusion Transformers

Sogang University

Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By

정윤성

Contents

- Background
- Classifier-Free Diffusion Guidance [arXiv:2207.12598 (2022)]
- TR-DQ: Time-Rotation Diffusion Quantization [AAAI 2026]

Background

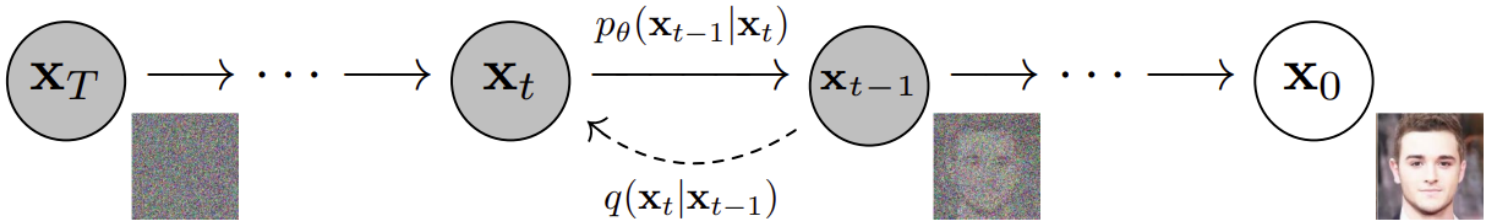
- Diffusion Model
  - 데이터에 노이즈를 조금씩 더했다가 그 과정을 거꾸로 제거하도록 학습하여, 순수 노이즈로부터 데이터를 생성하는 생성 모델
  - Forward process
    - 원본 x_0에 T step에 걸쳐 가우시안 노이즈를 점진적으로 추가
    - 마지막 x_T 는 사실상 순수 노이즈 N(0, I)
    - 한 step에 T번 더한 노이즈를 한 번에 쓸 수 있는 닫힌 식

$$x_t = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

✓ $\alpha_t = 1 - \beta_t$: 각 step에서 원본을 얼마나 남기는지

✓ $\overline{\alpha_t} = \prod \alpha_t$: t가 커질수록 0에 가까워지고 원본이 사라지며 노이즈만 남음

✓ ϵ : 매 step 더해지는 평균 0의 가우시안 노이즈



<Diffusion model process>

Background

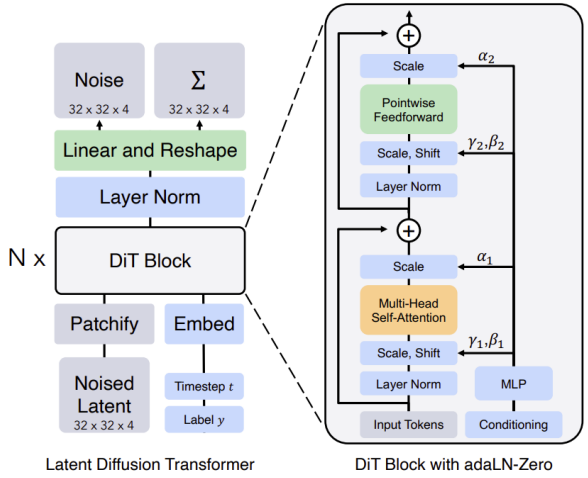
- Diffusion Model
  - Reverse process
    - x_T 즉 완전한 노이즈에서 시작해 한 step씩 노이즈를 제거하며 x_0인 이미지로 복원
    - 문제는 이 과정에서 x_t 를 x_{t-1} 로 되돌리는 정답은 하나가 아님
      - 한번에 x_0을 추정할 수 있지만 누적 노이즈를 한번에 예측하는 것은 정확도가 떨어짐
      - 따라서 x_{t-1}이나 x_0를 직접 예측하는 것이 아니라 노이즈 ϵ_θ를 예측
    - 신경망 $\epsilon_\theta(x_t, t)$ 가 현재 이미지 x_t 에 포함된 노이즈를 예측하도록 학습
      - Step 마다 노이즈의 양이 다르기 때문에 몇 번째 step인지 t를 입력으로 넣음
    - Training
      - 데이터에서 원본 x_0를 하나 뽑음
      - 무작위 step t를 고른 후 무작위 노이즈 ϵ로 x_t를 생성
      - $\epsilon_\theta(x_t, t)$로 노이즈를 예측 후 실제 ϵ와의 차이를 MSE로 최소화

Background

- Sampling
  - x_T를 순수 노이즈로 초기화 후 t = T에서 t = 1로 내려오며 아래 수식을 반복
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z$
    - 누적된 노이즈 ϵ_θ를 단번에 예측하는 건 너무 많은 해가 존재해서 부정확하기 때문에 한 번에 만들지 않고 여러 step을 거침
    - $\sigma_t z$: 소량의 새 노이즈
      - 신경망의 출력은 평균이기 때문에 다양성을 위해 추가

Background

- Diffusion Transformer (DiT)
  - 기존 ViT를 거의 그대로 사용, 단 분류기가 아닌 diffusion의 노이즈 예측기
    - 입력
      - ViT: image patch, [CLS] token / DiT: noised patch, timestep t, condition c
    - 출력
      - ViT: [CLS], class label / DiT: token별 noise ε
    - 정규화
      - ViT: standard LayerNorm / DiT: adaptive LayerNorm, γ·β를 condition에서 매 step 회귀



<Diffusion models with Transformers>

Background

- Diffusion Transformer (DiT)
  - 동작 순서
    - Patchify
      - latent를 patch로 분할해 token sequence 구성
      - positional embedding 더함
    - DiT block ×N
      - 각 block 구성
        - ✓Layer Norm, Multi-Head Self-Attention, Layer Norm, Feedforward Network
      - 각 sub-layer에 residual connection 적용
    - Conditioning
      - timestep embedding, condition embedding을 MLP에 통과
      - block별 scale, shift, gate parameter를 동적으로 생성
      - adaLN 방식으로 주입

Background

- Diffusion Transformer (DiT)
  - 동작 순서
    - Output head
      - final LayerNorm, Linear projection
      - token별 noise ε, variance 예측
      - 입력과 동일한 dense output
  - adaLN-Zero
    - ViT: 학습된 γ, β 고정 상수
    - DiT: timestep embedding과 condition embedding을 MLP에 통과시켜 γ, β, α를 매번 생성
      - scale γ, shift β, residual 반영 정도를 조절하는 gate α
      - α를 0으로 초기화하여 학습 초기에 각 block이 identity로 동작해 안정적으로 학습
      - weight가 step / 조건마다 다르게 동작해야 하므로 정규화 파라미터를 조건에서 만듦
  - 반복 실행
    - adaLN이 매 step 다른 scale·shift를 주입
    - 같은 weight가 step마다 전혀 다른 activation 분포를 만듦

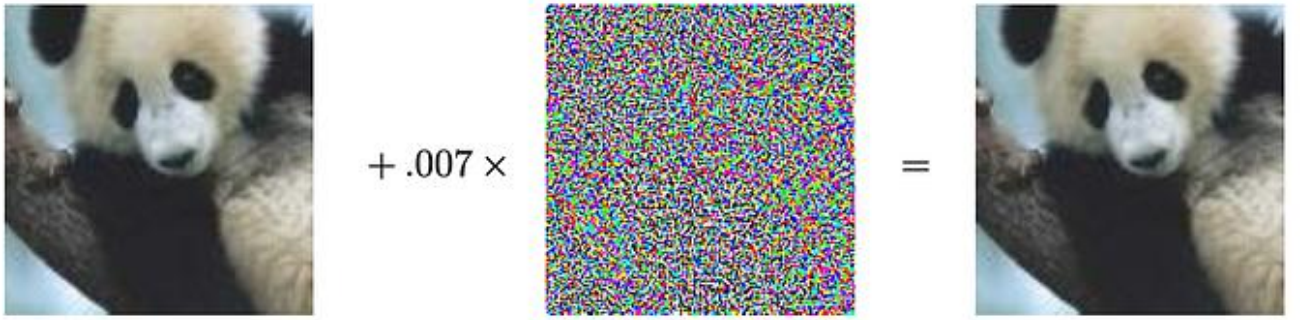
Classifier-Free Diffusion Guidance [arXiv:2207.12598 (2022)]

Introduction

- Conditional generation
  - 특정 강아지처럼 원하는 조건 c를 주고 그에 맞는 이미지를 생성하고 싶을 때 아래의 두 가지가 필요
    - 신경망에 조건을 반영
    - 얼마나 강하게 따를지 조절
  - 조건 c를 단순 신경망의 입력으로 넣으면 참고는 하지만 강하게 반영하지 않음
  - Classifier-Free Diffusion Guidance (CFG) 이전의 방법
    - 별도로 학습한 분류기의 gradient로 확실한 강아지 방향으로 샘플을 밀어줌
      - 분류기가 이 이미지를 강아지로 볼 확률을 높이는 방향 ($\nabla \log p(c|x)$)
      - guidance 강도를 키우면 다양성은 떨어지지만 개별 품질은 좋아짐

Introduction

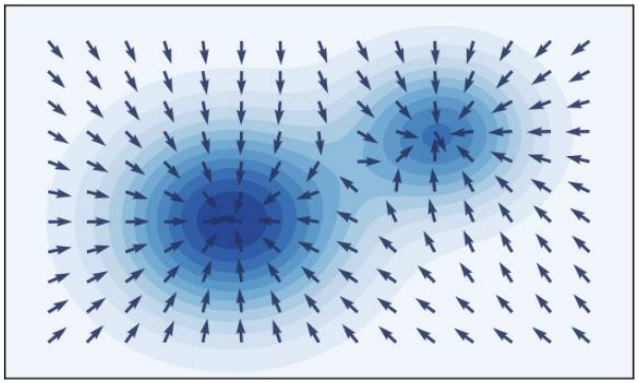
- Conditional generation
  - 이전 방법의 한계
    - 노이즈가 포함된 이미지 전용 분류기를 따로 학습해야 함
      - 학습 파이프라인이 복잡하고 기존의 분류기 사용이 불가능
    - 분류기 gradient에 의존
      - 분류기를 속이는 방향으로 샘플이 샐 수 있음
  - 따라서 별도의 분류기 없이 같은 효과를 주기 위해서 CFG를 도입



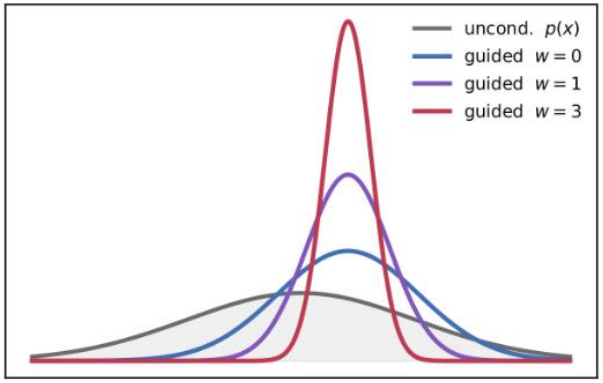
<분류기의 예측이 적은 노이즈로도 변하는 예시>

Method

- Score
  - $s(x) = \nabla_x \log p(x)$
    - p(x)는 학습 데이터가 이루는 분포이며 확률 밀도가 높을수록 진짜 같은 이미지
    - score는 그 밀도를 높이는 방향을 알려주는 벡터
  - 노이즈 예측 ϵ_θ가 사실상 score와 같은 정보이며 부호가 반대이고 스케일이 다름
    - $-\nabla_x \log q(x_t) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} * \epsilon_\theta(x_t, t)$
    - 즉 ϵ는 이미지에 노이즈가 추가되는 방향, - $\epsilon = score$는 이미지가 그럴듯해지는 방향
  - 노이즈가 적을수록 분포가 뾰족해 기울기 크기가 커짐



<Score field>



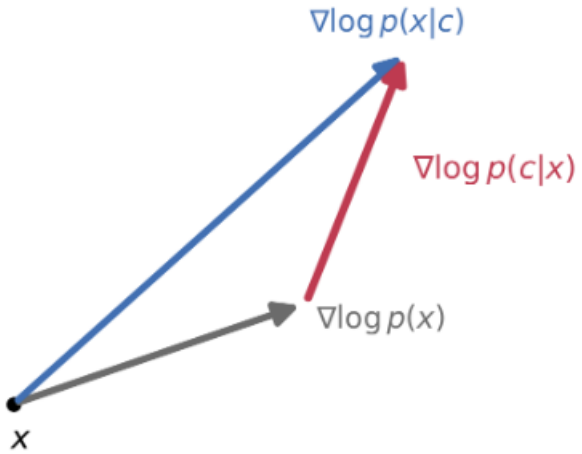
<Guidance sharpens the conditional>

Method

- Score
  - 모든 이미지의 확률을 계산하지 않아도 되는 이유
    - p(x) 자체는 직접 계산하지 않으며 필요한 것은 확률의 기울기인 score
    - MSE로 노이즈를 맞추도록 학습하면 ϵ_θ는 $E[\varepsilon \mid x_t]$ 를 근사
      - 어떤 함수 f를 MSE로 학습시키면, 즉 $E[(Y - f(X))^2]$를 최소화하면, 최적해는 항상 $f(X) = E[Y \mid X]$
      - 조건부 평균이 제곱오차를 최소화하는 값이기 때문
    - Tweedie 관계에 의해 이 값이 $\nabla_x \log q(x_t)$ 와 스케일만 다른 값이 됨
    - Denoising 학습의 부산물로 score가 얻어지며 이를 denoising score matching라고 함
    - 확률은 몰라도 되고 방향만 알면 생성이 가능
      - 생성 과정이 방향만 요구하고 계산 불가능한 항인 정규화 상수 Z가미분에서 사라짐
  - 조건을 준다는 것은 진짜 같은 이미지가 아니라 진짜 같은 강아지 이미지 방향을 안다는 것

Method

- Bayes 정리로 조건 방향을 분리
  - $p(x|c) = \frac{p(c|x)p(x)}{p(c)}$
  - 양변에 log: $\log p(x|c) = \log p(c|x) + \log p(x) - \log p(c)$
  - 양변을 x로 미분:$\nabla_x \log p(x|c) = \nabla_x \log p(c|x) + \nabla_x \log p(x)$
    - 이 때 $p(c)$는 x와 아무 상관 없는 값이기 때문에 미분 시 0
  - 조건 방향 기준으로 정리하면 $\nabla_x \log p(c|x) = \nabla_x \log p(x|c) - \nabla_x \log p(x)$
- 따라서 진짜 강아지 이미지 방향은 조건부 score – 무조건 score



<Condition direction>

Method

- CFG
  - 하나의 네트워크가 두 역할을 겸하기 때문에 분류기가 필요 없음
    - 조건 입력 자리에 c 또는 null condition ∅ 를 받는 단일 noise predictor $\epsilon_\theta(x_t, c)$를 학습
    - 동일 네트워크가 두 score를 모두 제공
      - $\epsilon_\theta(x_t, c)$ 는 conditional score
      - $\epsilon_\theta(x_t, \emptyset)$ 는 unconditional score
    - null condition ∅ 는 조건 없음을 뜻하는 learnable null embedding
      - 텍스트의 경우 빈 문자열 또는 special token
  - 학습 목표: 표준 diffusion loss와 동일, condition dropout만 추가
    - 매 학습 스텝마다 조건 c를 확률 p_{uncond}로 ∅로 교체 (보통 10~20%)
    - 대부분의 배치는 c가 들어가 조건부 능력을 학습
    - 가끔 ∅ 가 들어가 무조건 능력을 학습
    - 두 모델을 따로 학습하지 않고 하나의 네트워크와 하나의 loss로 동시 학습
    - 샘플링 때 같은 네트워크를 c와 ∅ 로 두 번 호출해 두 예측을 확보

Method

- CFG
  - Classifier guidance가 별도 classifier로 구하던 $\nabla \log p(c \mid x_t)$ 를, CFG는 학습된 두 예측의 차이로 획득
  - Guidance 강도 조절 유도
    - guided score = unconditional score + (1+w) × condition direction
      - $\nabla \log \tilde{p} = \nabla \log p(x_t) + (1+w) \cdot \nabla \log p(c \mid x_t)$
    - 조건 확률을 (1+w)제곱으로 sharpening
      - $\tilde{p}(x_t \mid c) \propto p(x_t \mid c) \cdot p(c \mid x_t)^w$
    - noise 형태로 변환 시 최종 공식과 일치
      - $\tilde{\epsilon} = (1+w) \cdot \epsilon_\theta(c) - w \cdot \epsilon_\theta(\emptyset)$

<Sampling>

Method

- Guidance Scale
  - 샘플링 단계에서 두 예측을 혼합
    - $\widetilde{\epsilon_\theta}(x_t, c) = (1 + w)\epsilon_\theta(x_t, c) - w\epsilon_\theta(x_t, \emptyset) = \epsilon_\theta(x_t, c) + w[\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t, \emptyset)]$
    - 조건 방향 즉 차이를 w배 더 밀어줌
  - CFG 논문과 현재 diffusion 논문의 표기 차이
    - CFG w: w = 0이면 순수 조건부, 아무 이미지가 아니라 강아지는 나오지만 증폭이 없음
      - w ∈ {0, 0.1, …, 4} sweep
    - Stable Diffusion, TR-DQ s: $\tilde{\epsilon} = \epsilon_\theta(\emptyset) + s \cdot [\epsilon_\theta(c) - \epsilon_\theta(\emptyset)]$
      - s = 1이 조건부, s = 7.5가 강한 guidance
    - 관계: s = 1 + w
      - W나 s를 키우면 조건부 분포가 뾰족해지고 선명해지지만 다양성이 감소

Method

- Analysis
  - CFG를 사용하면 가장 전형적인 강아지 쪽으로 확률이 몰림
    - 데이터 밀도가 높으면서 동시에 class 판별이 확실한 영역에 확률 질량이 집중
  - 비용
    - 매 step 조건부·무조건 두 번의 forward 필요하기 때문에 추론이 2배로 느림
    - 두 예측의 차이를 s배 증폭하면, 그 차이에 낀 양자화 오차도 함께 증폭됨
      - Guided 출력의 오차: $e_g = e_u + s \cdot (e_c - e_u) = e_u + s \cdot e_\Delta$
      - 차이 오차 e_Δ는 크기는 작아도 s로 곱해져 지배적, 오차 항에 $s^2 \cdot \| e_\Delta \|^2$ 이 생김

<왼쪽 이미지: w = 0, 오른쪽 이미지 w = 3>

Experiment

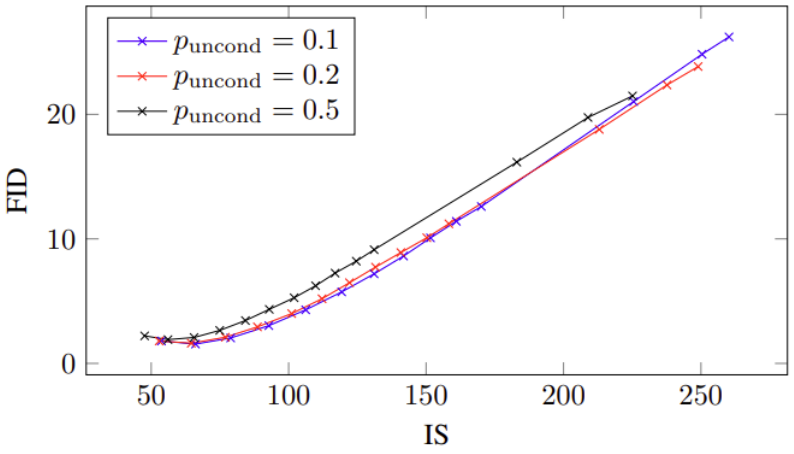
- Dataset
  - class-conditional ImageNet 64×64, 128×128, class label을 조건 c로 사용
- Model
  - 선행 diffusion 모델과 동일 구조를 조건부+무조건 공동 학습
    - 무조건 확률 $p_{uncond} = 0.1$
- Guidance sweep
  - w ∈ {0, 0.1, …, 4}, 각 w마다 5만 장 생성해 지표 측정
- 지표
  - FID↓: 실제 분포와의 거리
  - IS↑: Inception Score, class 뚜렷함

Results

- Quantitative

Model	FID (↓)	IS (↑)
ADM (Dhariwal & Nichol, 2021)	2.07	-
CDM (Ho et al., 2021)	**1.48**	67.95
Ours	$p_{\text{uncond}} = 0.1/0.2/0.5$	
$w = 0.0$	1.8 / 1.8 / 2.21	53.71 / 52.9 / 47.61
$w = 0.1$	1.55 / 1.62 / 1.91	66.11 / 64.58 / 56.1
$w = 0.2$	2.04 / 2.1 / 2.08	78.91 / 76.99 / 65.6
$w = 0.3$	3.03 / 2.93 / 2.65	92.8 / 88.64 / 74.92
$w = 0.4$	4.3 / 4 / 3.44	106.2 / 101.11 / 84.27
$w = 0.5$	5.74 / 5.19 / 4.34	119.3 / 112.15 / 92.95
$w = 0.6$	7.19 / 6.48 / 5.27	131.1 / 122.13 / 102
$w = 0.7$	8.62 / 7.73 / 6.23	141.8 / 131.6 / 109.8
$w = 0.8$	10.08 / 8.9 / 7.25	151.6 / 140.82 / 116.9
$w = 0.9$	11.41 / 10.09 / 8.21	161 / 150.26 / 124.6
$w = 1.0$	12.6 / 11.21 / 9.13	170.1 / 158.29 / 131.1
$w = 2.0$	21.03 / 18.79 / 16.16	225.5 / 212.98 / 183
$w = 3.0$	24.83 / 22.36 / 19.75	250.4 / 237.65 / 208.9
$w = 4.0$	26.22 / 23.84 / 21.48	**260.2** / 248.97 / 225.1

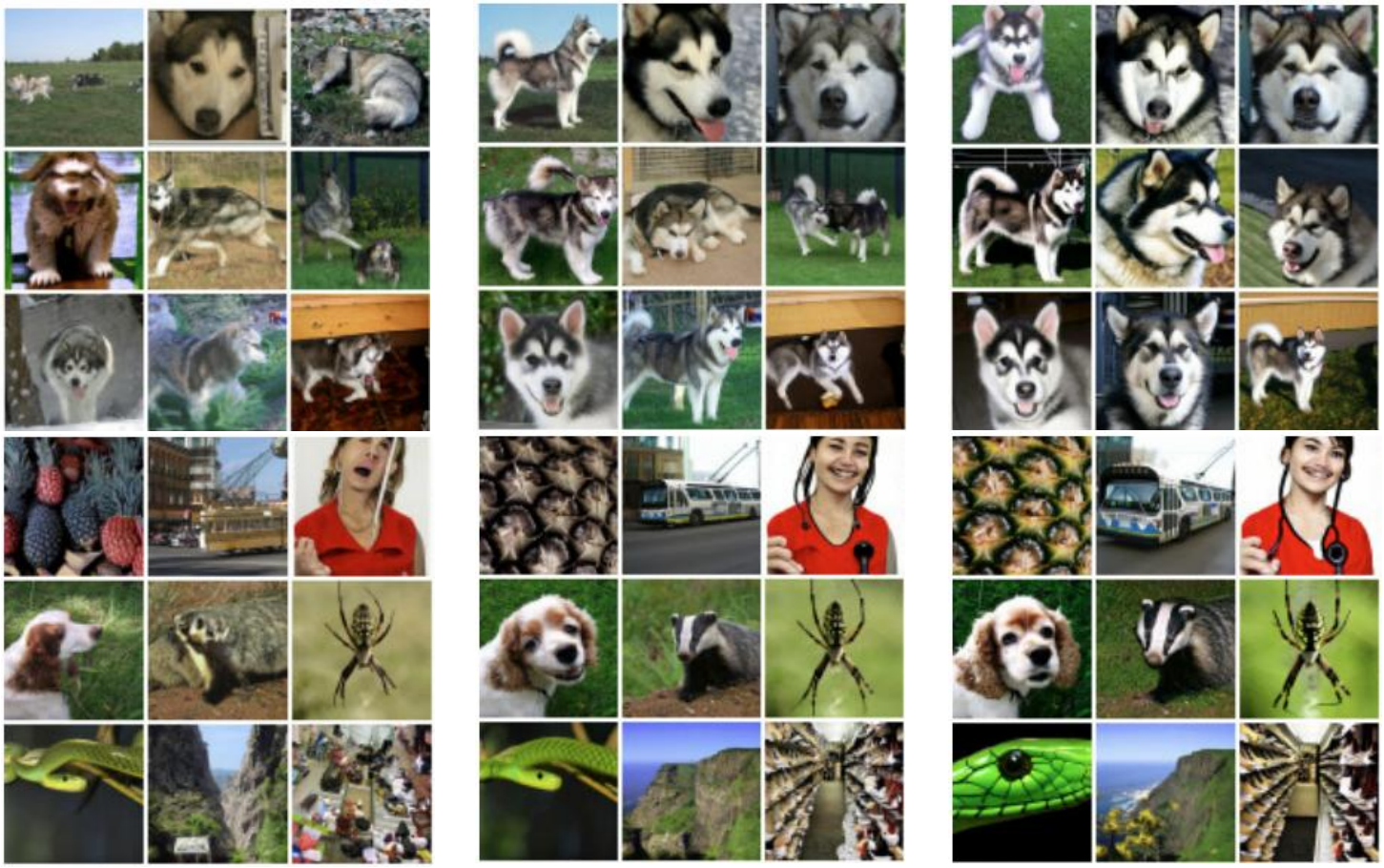
<ImageNet 64x64 results>



< IS/FID curves over guidance strengths >

Results

- Qualitative



<w = 0, 1, 2일 때의 생성 이미지>

Conclusion

- CFG는 조건부, 무조건 score를 하나의 모델로 학습하고, 샘플링 때 차이를 w배 증폭해 조건을 강하게 반영
  - 현재 생성 모델에서 많이 사용되는 이유
    - 학습 비용이 거의 없으며 condition dropout 한 줄 수준의 단순함
    - 별도 분류기나 추가 모델이 불필요
    - 추론 시 scale을 통해서 품질과 다양성을 조절 가능
- Tradeoff
  - guidance를 키우면 fidelity↑, diversity↓
- 한계
  - 두 번 추론이기 때문에 느리고 차이 증폭이 low bit quant를 어렵게 만듦
  - TR-DQ는 이 CFG의 연산 중복을 attention-sharing으로 압축
    - 하지만 CFG가 만드는 양자화 오차 자체는 다루지 않음

TR-DQ: Time-Rotation Diffusion Quantization [AAAI 2026]

Introduction

- Diffusion 모델은 이미지·비디오 생성에서 GAN을 능가하는 품질을 내지만, 네트워크 구조가 복잡해 생성 과정의 비용이 큼
  - 메모리 사용량이 많고, 매 time-step마다 연산이 필요해 latency가 길어서 quantization과 같은 방법으로 경량화를 해야함
- 기존 diffusion quant 기법들의 한계
  - 주로 weight quant에만 집중하고, 샘플링을 진행하며 time-step마다 activation 분포가 크게 달라진다는 점을 무시
    - Diffusion은 노이즈 제거를 여러 step 반복하는데, step마다 이 값의 크기·분포가 크게 달라짐
  - 아무리 처리해도 사라지지 않는 유난히 큰 massive activation을 제대로 다루지 못해, low bit로 누르면 품질이 급격히 저하
  - CFG로 인한 연산 낭비도 기존 연구들이 다루지 않음
    - 이 부분에 대한 압축 여부를 아무도 확인하지 않았음

Introduction

- Contribution
  - time-step + rotation 기반 최적화를 결합한 TR-DQ 제안
    - Quant하기 어려운 activation을 회전 행렬을 적용해서 weight 쪽으로 shift하고, activation과 weight 모두 smoothing을 통해서 quant하기 쉬운 분포로 만듦
  - 직교 행렬은 $R \cdot R^T = I$ 이므로 XW 값이 수학적으로 완전히 보존됨
    - 즉, 값은 그대로 두고 outlier 에너지만 여러 채널로 분산
  - 하나의 global 회전 행렬을 time-step 분포에 맞춰 time-dependent 회전 행렬로 확장
    - step마다 다른 회전 적용
  - CFG, non-CFG 경로의 attention 민감도 유사성을 분석
    - attention merging quant로 추가 압축
  - 이미지·비디오 생성 품질을 유지하면서 속도 향상 및 메모리 절감

Method

- Diffusion의 layer 대부분은 선형 연산이기 때문에 uniform quant를 사용해서 activation은 per-token, weight는 per-channel로 quantization

$$X_{\text{int}} = \text{clamp}\left(\text{round}\left(\frac{X}{s}\right) + z, 0, 2^b - 1\right)$$

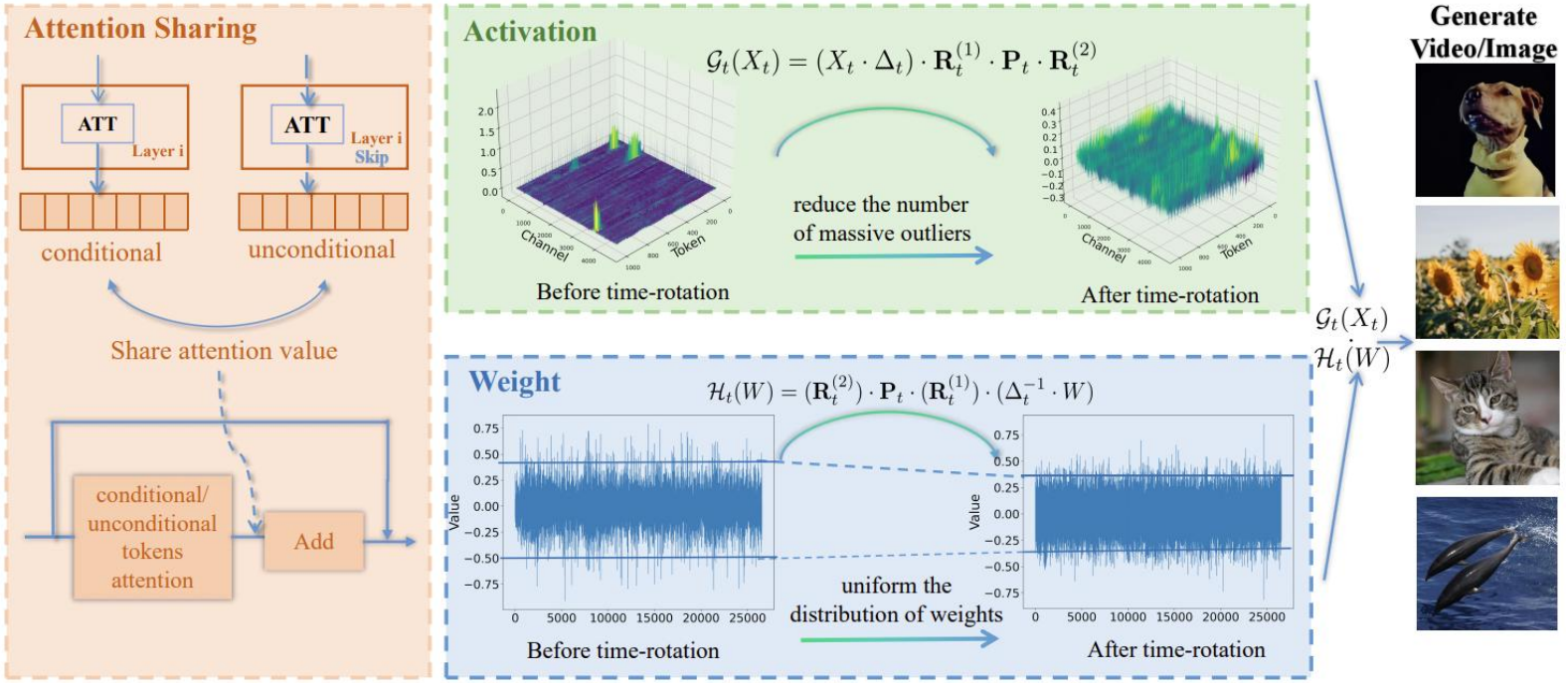
$$s = \frac{\max - \min}{2^b - 1}, \quad z = \text{round}\left(-\frac{\min}{s}\right)$$

∴ S = range, z = zero point offset, b = bit num

- SmoothQuant
  - Scaling으로 Quant 난이도를 activation에서 weight로 이동
    - $Y = (X \cdot \text{diag}(\Delta)^{-1}) \cdot (\text{diag}(\Delta) \cdot W)$, $\Delta_j = \max|X_j|^{\alpha} / \max|W_j|^{(1-\alpha)}$
  - 일부 outlier는 줄어들지만 massive outlier처럼 극단값이 여전히 존재하고, diffusion의 경우 ViT에 비해 weight 분포가 불규칙해짐
    - Activation과 weight의 balancing을 조절하는 것이 필요

Method

- Rotation 기반 balancing
  - DuQuant를 확장한 직교 회전 행렬 R로 activation의 massive outlier 수를 줄이고, activation와 weight 분포를 모두 고르게 만들어 group-wise quant를 쉽게 함
    - DuQuant는 outlier를 회전 및 치환으로 재배치해 LLM을 quant한 기존 방법



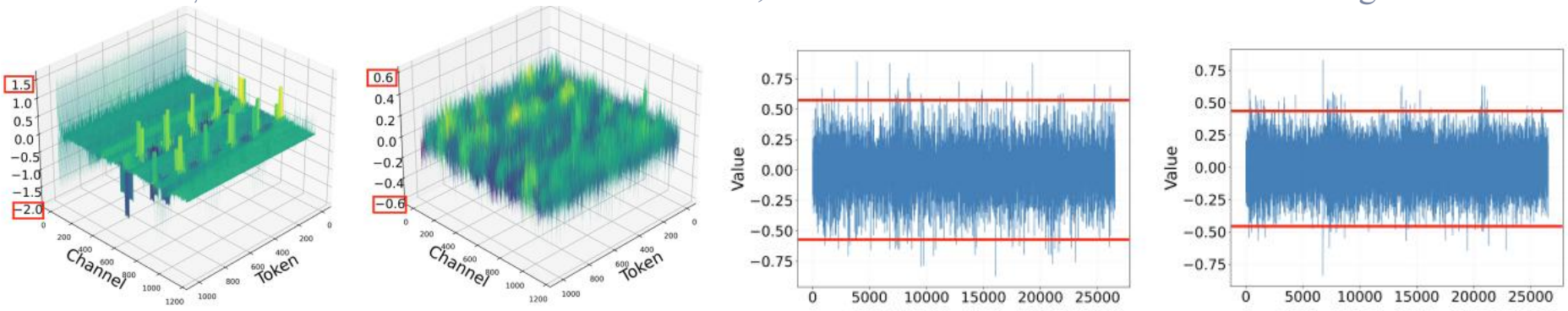
<Pipeline>

Method

- Rotation 기반 balancing
  - 회전 행렬
    - $R_1 = C_1 \cdot Q \cdot C_2$
      - C_1 (스위칭 행렬): 최대 outlier가 있는 열을 첫 번째 열과 교환
      - Q: 랜덤 초기화된 직교 회전 행렬, 단 첫 행은 균일 분포
        - ✓C_1 변환 후 첫 열의 outlier를 완화하는 역할
      - C_2: C_1의 역연산, 두 열을 원위치로 되돌려 activation 구조 보존
    - Greedy 방식으로 outlier를 최소화하는 여러 회전을 순차 적용
    - 전체 R을 직접 만들면 연산·메모리 부담이 큼
      - block-diagonal 형태로 근사: R = BlockDiag(R_b1, …, R_bK)
        - ✓각 블록 $2^n \times 2^n$, 블록 수 K = C_in/2^n

Method

- Zigzag Permutation
  - 1차 블록 회전은 블록 안으로는 outlier를 줄이지만, 블록 사이 분포는 여전히 불균형하기 때문에 2차 블록 회전에 불리
  - zigzag permutation을 사용해서 이를 해결
    - activation이 큰 채널부터 첫 블록에 배정하고 그 다음 큰 채널을 다음 블록에 배정
    - 마지막 블록에 닿으면 방향을 뒤집어 다시 오름차순으로 배분
    - 어느 한 블록에도 크거나 작은 activation이 몰리지 않도록 블록 간 균형을 맞춤
      - 이후 2차 블록 회전으로 한 번 더 매끄럽게 만듦
    - 최종 balancing 식: $Y = X \cdot W = [(X \cdot \Delta)\, R^{(1)} \cdot P \cdot R^{(2)}] \cdot [(R^{(2)})^{T} \cdot P^{T} \cdot (R^{(1)})^{T}\, (\Delta^{-1} \cdot W)]$
      - 앞부분: 매끄럽게 만드는 activation, 뒷부분: Activation에 맞게 보정된 Weight



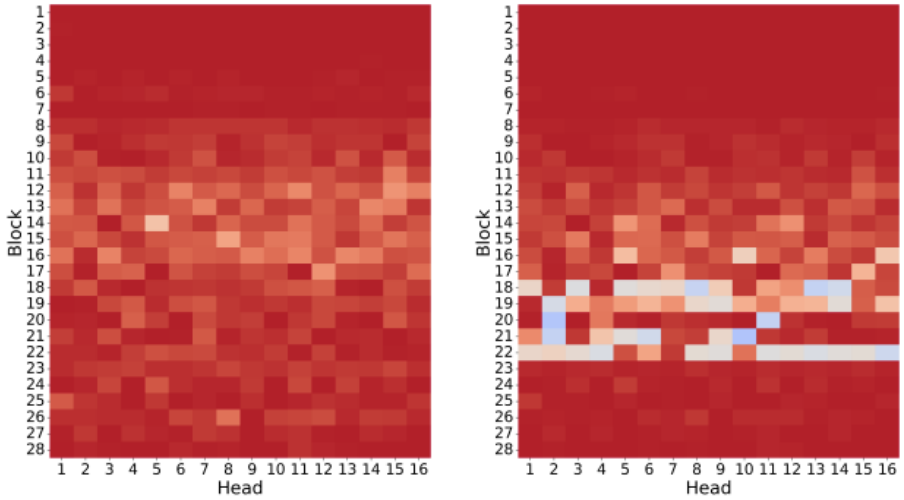
<Time Rotation 전과 후의 activation과 weight 분포 변화>

Method

- Time-Steps Awareness
  - diffusion은 time-step마다 activation, outlier 분포가 크게 다름
    - 모든 step에 하나의 고정된 파라미터(R, P, Δ) 를 쓰면 생성 품질이 크게 훼손됨
  - 이 논문의 해결 방안
    - Dynamic quantization
      - Quant 파라미터를 미리 고정하지 않고 각 step에서 min, max만 runtime 계산
    - Time-Rotation
      - time-step별로 calibration한 R, P, Δ를 미리 구해 두고, denoising 중 현재 t에 맞는 것을 lookup해 적용
  - 수식
    - $F_t(X_t, W) = G_t(X_t) \cdot H_t(W)$
      - $G_t(X_t)=(X_t \cdot \Delta_t) \cdot R^{(1)}{}_t \cdot P_t \cdot R^{(2)}{}_t$, $H_t(W)=(R^{(2)}{}_t)^T \cdot P_t^T \cdot (R^{(1)}{}_t)^T \cdot (\Delta_t^{-1} \cdot W)$, $t \in \{1,\ldots,N\}$
  - Prior, greedy, permutation까지 포함하기 때문에 단순 회전에 비해서 불균형 데이터를 더 잘 처리함

Method

- Attention-Sharing
  - CFG는 조건부·무조건부 두 번의 denoising을 요구하기 때문에 추론이 느림
  - diffusion transformer의 특정 블록에서 조건부, 무조건부 경로의 multi-head self-attention 값이 서로 매우 유사
    - 단, time-step 차이 때문에 모든 step·모든 층이 유사하진 않음
    - 초반부와 후반부 블록의 attention이 전 step에 걸쳐 특히 유사함
  - 유사한 블록에 대해 조건부/무조건부가 attention 계산을 공유해서 중복 연산을 제거하여 속도 향상



<조건부, 무조건부 상황에서 multi-head self-attention의 heat map>

Experiment

- NVIDIA A800 (80G)
- 데이터셋
  - 이미지 생성: PixArt-α, COCO 데이터셋으로 평가
  - 비디오 생성: Open-SORA, UCF-101(101개 행동 클래스 비디오)로 평가
  - 공통: sampling STEP = 20, CFG scale = 4.5
- Quant bit
  - 이미지 W8A8, W4A8 / 비디오 W8A8, W6A6, W4A8
- 평가 지표
  - 이미지: FID ↓ : 실제 이미지 분포와의 거리, CLIPScore ↑ : 텍스트와 이미지 일치, ImageReward ↑ : 사람 선호도
  - 비디오: VBench ↑ : 영상 생성 종합 점수, CLIPSIM / CLIP-Temp ↑ : 텍스트 정합 / 시간 일관성, VQA-A / VQA-T ↑ : 미적 / 기술적 화질, Flow-score ↓ : 프레임 간 흐름 안정성

Results

- Quantitative

Method	Bit-width (W/A)	Imaging Quality	Aesthetic Quality	Motion Smooth.	Dynamic Degree	BG. Consist.	Subject Consist.	Scene Consist.	Overall Consist.
-	16/16	63.68	57.12	97.01	56.94	96.13	92.28	40.51	26.21
Q-Diffusion [23]	8/8	60.38	55.15	94.44	68.05	94.17	87.74	36.62	25.66
Q-DiT [5]	8/8	60.35	55.80	93.64	68.05	94.70	86.94	32.34	26.09
PTQ4DiT [39]	8/8	56.88	55.53	95.89	63.88	96.02	91.26	34.52	25.32
ViDiT-Q [48]	8/8	61.48	56.95	96.14	61.11	95.84	90.24	38.22	26.06
TR-DQ	8/8	61.82	57.44	96.63	55.38	96.11	91.14	39.78	26.18
TR-DQ+AS	8/8	60.38	57.10	96.26	50.27	95.71	91.58	38.50	25.99
Q-DiT [5]	4/8	23.30	29.61	97.89	4.166	97.02	91.51	0.00	4.985
PTQ4DiT [39]	4/8	37.97	31.15	92.56	9.722	98.18	93.59	3.561	11.46
ViDiT-Q[48]	4/8	59.01	55.37	95.69	48.33	95.23	88.72	36.19	25.94
TR-DQ	4/8	59.88	56.20	96.57	51.83	96.65	90.74	32.46	26.17
TR-DQ+AS	4/8	57.69	55.02	96.74	47.78	96.58	91.03	32.25	25.34

<Video quantitative result>

Methods					Bit-width	CLIPSIM	CLIP-Temp	VQA-	VQA-	Δ Flow
Smooth	R_1	P	R_2	T-R	(W/A)			Aesthetic	Technical	Score.
-	-	-	-	-	16/16	0.1797	0.9988	63.40	50.46	-
✓	-	-	-	-	4/8	0.1739	0.9985	44.12	21.19	0.675
✓	✓	-	-	-	4/8	0.1755	0.9941	50.42	42.12	0.421
✓	✓	-	✓	-	4/8	0.1745	0.9972	52.38	45.67	0.342
✓	✓	✓	✓	-	4/8	0.1741	0.9985	54.94	47.97	0.219
✓	✓	✓	✓	✓	4/8	0.1815	0.9990	59.86	55.56	0.130

<Ablation study>

Results

- Quantitative

Method	Bit-width (W/A)	FID(↓)	CLIP(↑)	IR(↑)
-	16/16	73.34	0.258	0.901
Q-Diffusion [23]	8/8	96.54	0.239	0.186
Q-DiT [5]	8/8	73.60	0.256	0.854
PTQ4DiT [39]	8/8	127.9	0.217	-1.216
ViDiT-Q [48]	8/8	75.98	0.232	0.859
TR-DQ	8/8	75.12	0.249	0.887
TR-DQ+AS	8/8	75.57	0.233	0.863
Q-Diffusion [23]	4/8	91.95	0.228	-0.224
Q-DiT [5]	4/8	475.8	0.127	-2.277
PTQ4DiT [39]	4/8	171.9	0.177	-2.064
ViDiT-Q [48]	4/8	76.65	0.243	0.837
TR-DQ	4/8	75.53	0.252	0.851
TR-DQ+AS	4/8	75.76	0.246	0.847

<Image quantitative result>

Bit-width (W/A)	A800		A100	
	Memory	Latency	Memory	Latency
16/16	1.00×	1.00×	1.00×	1.00×
8/8 (ViDiTQ)	1.98×	1.70×	2.00×	1.74×
8/8 (TR-DQ)	1.97×	1.69×	1.97×	1.69×
8/8 (TR-DQ+AS)	2.17×	1.89×	2.17×	1.91×
4/8 (ViDiTQ)	2.41×	1.36×	2.42×	1.38×
4/8 (TR-DQ)	2.46×	1.38×	2.47×	1.42×
4/8 (TR-DQ+AS)	2.58×	1.41×	2.59×	1.44×

<Efficiency Comparison>

Results

- Qualitative

<Image generation result>

Conclusion

- Low-bit diffusion quant문제 정의
  - 제거되지 않는 massive activation
  - timestep마다 크게 달라지는 activation 분포
  - CFG로 인한 중복 연산
- 해결 방법
  - rotation으로 양자화가 어려운 activation을 weight로 이전
  - timestep마다 R, P, Δ를 적응적으로 조정
  - 유사한 block의 attention을 공유
- 이미지와 비디오 생성 품질을 유지하면서 속도개선 및 메모리 확보
- 한계
  - CFG가 만드는 양자화 오차의 증폭 자체는 미해결

감사합니다.