Post-Training Quantization for Diffusion Models

2026 하계 세미나 – 26.07.31

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
오채인

1) Lei Chen et al. "Q-DiT: Accurate Post-Training Quantization for Diffusion Transformers." CVPR 2025
2) Muyang Li et al. "SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models." ICLR 2025

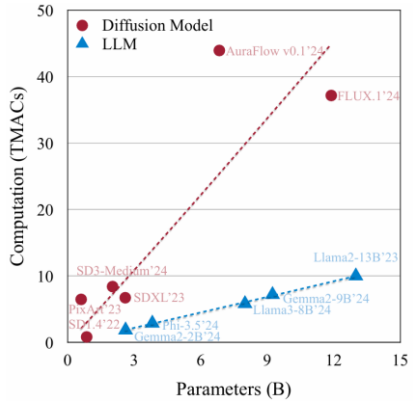
Outline

- Background
  - Diffusion model & Diffusion Transformer
  - Model quantization & Post-Training Quantization
  - Diffusion model quantization

- Papers
  - Q-DiT: Accurate Post-Training Quantization for Diffusion Transformers[1] [CVPR 2025]
  - SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models[2] [ICLR 2025]

Background

- Diffusion model & Diffusion Transformer
  - Diffusion model
    - Noise를 반복적으로 제거하며 이미지를 생성 (T-step denoising)
    - 한 장을 만드는 데 backbone을 수십 번 forward → 높은 inference cost
  - Diffusion Transformer (DiT)
    - Backbone이 UNet → Transformer로 이동 (PixArt-Σ, SD3, FLUX.1, Sora)
    - Timestep·condition은 adaLN으로 주입, 연산의 대부분이 linear layer
    - 모델 규모 급증: SD1.4 0.8B → SDXL 2.6B → FLUX.1 12B
      - 같은 파라미터 수의 LLM보다 연산량이 훨씬 빠르게 증가



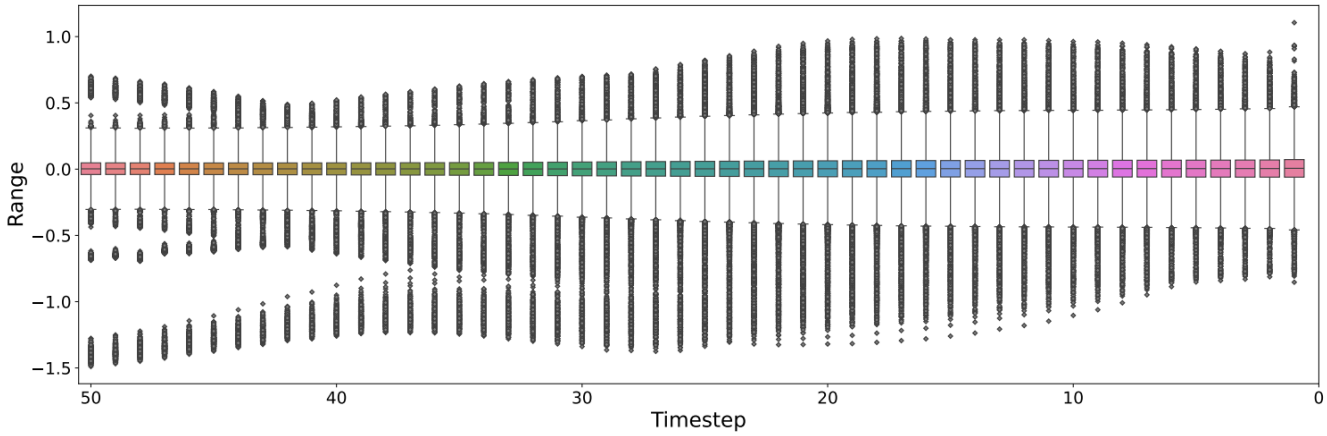
< Computation vs. parameters for LLMs and diffusion models >

Background

- Model quantization & Post-Training Quantization
  - Quantization
    - FP 값을 scale s로 나눠 low-bit 정수 grid에 매핑
      - $Q_X = \text{round}\left(\frac{X}{s_X}\right), \qquad s_X = \frac{\max(|X|)}{q_{\max}}$
      - s : scaling factor, q_max : 표현 가능한 최대 정수값
    - Granularity : per-tensor / per-channel / per-group
  - QAT vs. PTQ
    - QAT : 재학습으로 정확도 회복, 대형 모델에는 비용이 비현실적
    - PTQ : 소량의 calibration data만으로 quantization parameter 결정
  - WxAy 표기
    - W4A16(weight-only)은 주로 memory 절감
    - 연산 가속은 W와 A를 같은 bit로 맞출 때 유리

Background

- Diffusion model quantization
  - Diffusion quantization이 어려운 이유
    - Temporal 변동 : activation 분포가 timestep마다 크게 이동
      - ∴ 한 시점의 calibration이 전체 denoising 구간에 맞지 않음
    - Outlier : weight와 activation 모두 특정 channel에 큰 값이 집중
    - Compute-bound : weight-only quantization만으로는 실질적 가속이 제한적
    - 구조적 제약 : adaLN 때문에 offline rotation 계열 기법 적용 불가
    - 평가 지표 : layer MSE 감소가 생성 품질(FID) 향상을 보장하지 않음



< Activation distribution across timesteps, DiT-XL/2 >

Q-DiT: Accurate Post-Training Quantization for Diffusion Transformers [CVPR 2025]

Q-DiT[1)]

- Introduction
  - 기존 diffusion PTQ 방식의 한계
    - 기존 방법은 대부분 UNet backbone을 가정하고 설계됨
      - e.g., PTQ4DM, Q-Diffusion, PTQD, TFMQ-DM
    - Reconstruction 기반 최적화는 대형 모델로 scale하기 어려움

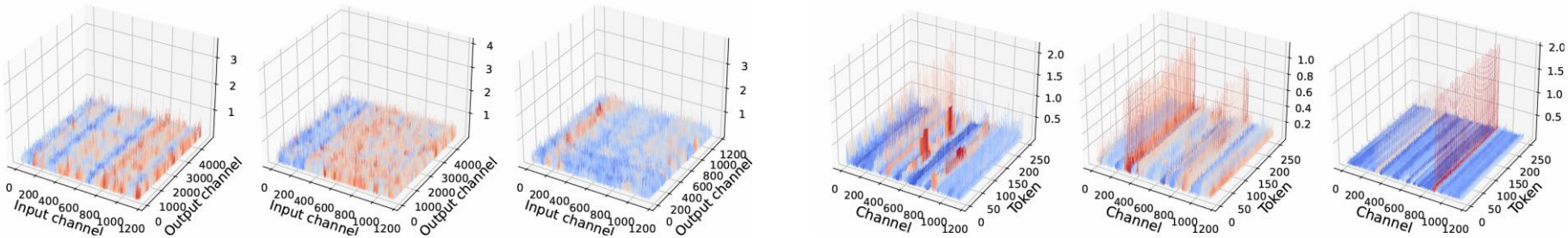
→ DiT의 고유한 distribution 특성을 반영한 PTQ가 필요

Model	Bit-width (W/A)	Method	Size (MB)	FID ↓	sFID ↓	IS ↑	Precision ↑
DiT-XL/2 256×256 (steps = 100)	16/16	FP	1349	12.40	19.11	116.68	0.6605
	6/8	PTQ4DM	508	17.86	25.33	92.24	0.6077
		RepQ-ViT	508	27.74	20.91	63.41	0.5600
		TFMQ-DM	508	22.33	27.44	72.74	0.5869
		PTQ4DiT	508	15.21	21.34	105.03	0.6440
		G4W+P4A	520	16.72	24.61	100.09	0.6123
		Ours	518	**12.21**	**18.48**	**117.75**	**0.6631**
	4/8	PTQ4DM	339	213.66	85.11	3.26	0.0839
		RepQ-ViT	339	224.14	81.24	3.25	0.0373
		TFMQ-DM	339	143.47	61.09	5.61	0.0497
		PTQ4DiT	339	28.90	34.56	65.73	0.4931
		G4W+P4A	351	25.48	25.57	73.46	0.5392
		Ours	347	**15.76**	**19.84**	**98.78**	**0.6395**

< DiT-XL/2, ImageNet 256 × 256, steps = 100 >

Q-DiT[1)]

- Observations of DiT Quantization
  - Observation 1
    - Input channel 방향의 큰 variance
      - Weight·activation 모두 output channel보다 input channel 축의 편차가 큼
      - 기존 diffusion PTQ는 output channel-wise quantization을 사용
        - ✓→ 분산이 큰 축과 quantization 축이 다름
      - 특정 channel에 activation outlier가 지속적으로 존재
        - ✓ Per-tensor를 쓰면 non-outlier 값의 오차가 크게 증가

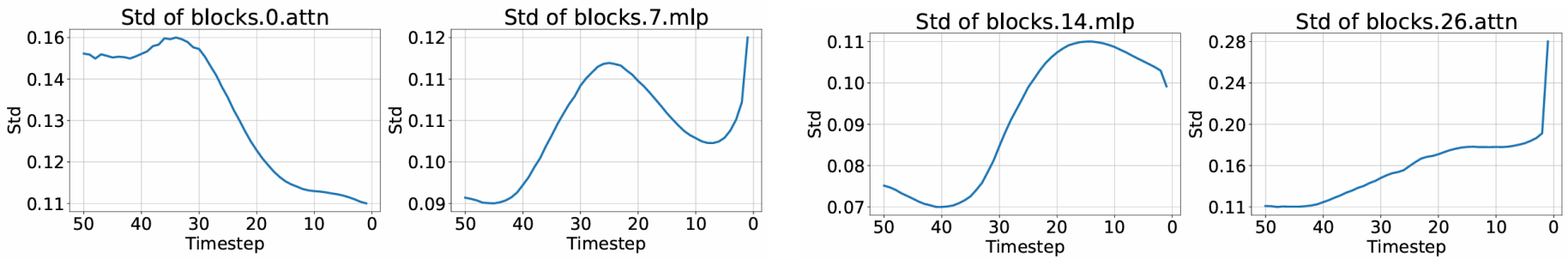


(a) **Weight** distribution (b) **Activation** distribution

< Weight / activation distributions of DiT-XL/2 >

Q-DiT[1)]

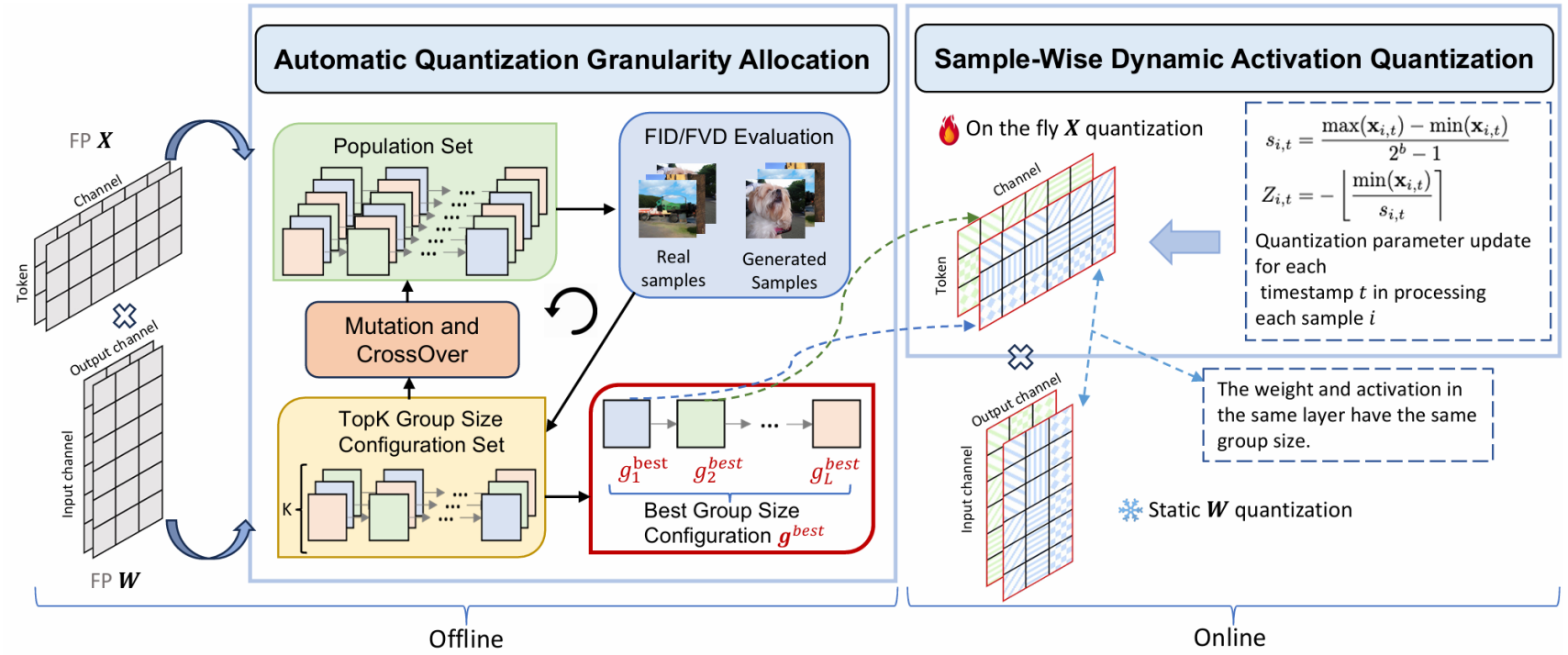
- Observations of DiT Quantization
  - Observation 2
    - Timestep에 따른 activation 분포 이동
      - Denoising이 진행될수록 activation range가 계속 변함
      - 변화 양상이 block·layer마다 서로 다름
      - 같은 timestep이라도 sample마다 shift가 다르게 나타남
    - 고정된 calibration 파라미터로는 대응 불가



< Std. of activations across timesteps for different blocks >

Q-DiT[1)]

- Overview
  - ① Automatic Quantization Granularity Allocation : Observation 1 대응
  - ② Sample-wise Dynamic Activation Quantization : Observation 2 대응
    - Weight는 GPTQ + asymmetric으로 static quantization, activation은 runtime 계산



< Overview of the proposed Q-DiT >

Q-DiT[1)]

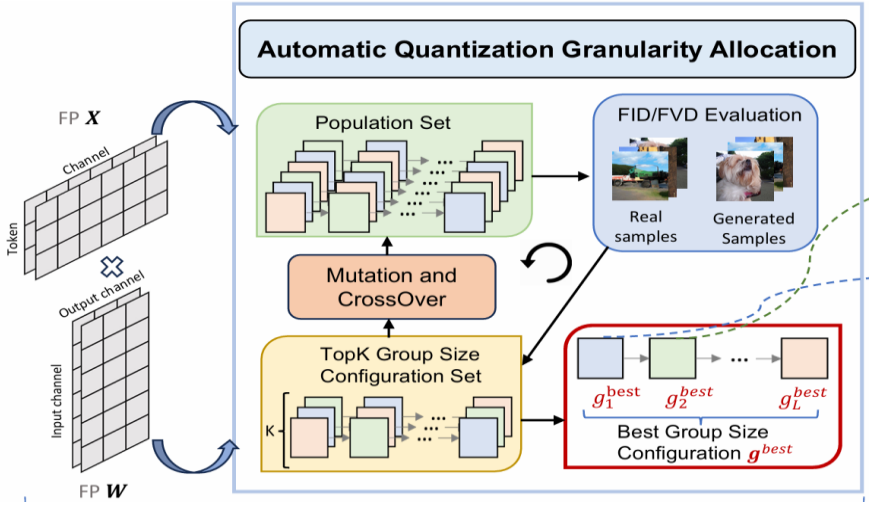
- Methods
  - Fine-grained Group Quantization
    - Input channel-wise quantization은 정확하지만 비효율적
      - 중간 rescaling이 반복되어 low-precision 연산의 장점이 사라짐
    - Group quantization: tensor-wise 및 channel-wise의 절충안
      - 각 group이 하나의 scale과 zero-point를 공유
  - Non-monotonicity in group size
    - 직관: group size ↓ → quantization error ↓
      - 실제로는 성립하지 않음

256×256			512×512		
Group	FID ↓	sFID ↓	Group	FID ↓	sFID ↓
128	17.87	20.45	96	20.76	21.97
96	19.97	21.42	64	20.90	22.58

< Quantization results with varying group sizes >

Q-DiT[1)]

- Methods
  - ① Automatic granularity allocation
    - Layer MSE 기반 mixed-precision 할당의 한계
      - DiT에서는 낮은 recon. MSE가 높은 생성 품질을 보장하지 않음
    - 생성 품질 L(g)를 목적함수로 직접 사용
      - $g^* = \arg\min L(g), \quad \text{s.t. } B(g) \le N_{\text{bitops}}$
        - ✓$L(g)$: image FID 또는 video FVD, $B(g)$: 해당 group-size 구성의 BitOps
    - BitOps 제약을 만족하는 구성을 evolutionary search로 탐색



< Automatic Quantization Granularity Allocation overview >

Q-DiT[1)]

- Methods
  - ② Sample-wise dynamic activation quantization
    - 기존 timestep-aware 방식의 한계
      - Timestep별 quantization parameter 저장으로 메모리 증가
    - **Q-DiT**: sample·timestep별 quantizer를 on-the-fly 계산
      - 각 activation group $x_{i,t}$의 min–max로 scale과 zero-point 산출

$$\checkmark s_{i,t} = \frac{\max(x_{i,t}) - \min(x_{i,t})}{2^b - 1} \quad , \quad Z_{i,t} = -\left\lfloor \frac{\min(x_{i,t})}{s_{i,t}} \right\rceil$$

      - 별도 parameter 저장 없이 현재 activation 분포를 직접 반영
      - Min-max 연산은 이전 operator와 fuse하여 overhead 완화

Q-DiT[1)]

- Experiments
  - Experimental Setup
    - Image generation
      - DiT-XL/2, ImageNet 256×256 / 512×512, DDIM 50 · 100 steps
      - 10K samples per setting, with/without CFG
      - Metrics: FID, sFID, IS, Precision
    - Video generation
      - STDiT3 (Open-Sora)
      - VBench with 16 evaluation dimensions
    - Baselines
      - PTQ4DM, TFMQ-DM, RepQ-ViT, PTQ4DiT, G4W+P4A
      - Asymmetric W/A quantization; GPTQ weight quantization with default group size 128

Q-DiT[1)]

- Experiments
  - Quantitative Results
    - W6A8 : FID 5.32 vs FP 5.31 — 사실상 무손실
    - W4A8 : FID 6.40으로 FP 대비 +1.09, PTQ4DiT(7.75)·G4W+P4A(7.66)보다 우수
    - Video : W4A8에서 VBench 16개 지표 중 15개에서 baseline을 상회

Model	Bit-width (W/A)	Method	Size (MB)	FID ↓	sFID ↓	IS ↑	Precision ↑
DiT-XL/2 256×256 (steps = 100 cfg = 1.5)	16/16	FP	1349	5.31	17.61	245.85	0.8077
	6/8	PTQ4DM	508	8.41	25.56	196.73	0.7622
		RepQ-ViT	508	10.77	18.53	163.11	0.7264
		TFMQ-DM	508	8.87	23.52	194.08	0.7737
		PTQ4DiT	508	5.34	18.48	209.90	**0.8047**
		G4W+P4A	520	6.41	19.52	225.48	0.7705
		Ours	518	**5.32**	**17.40**	**243.95**	0.8044
	4/8	PTQ4DM	339	215.68	86.63	3.24	0.0741
		RepQ-ViT	339	226.60	77.93	3.61	0.0337
		TFMQ-DM	339	141.90	56.01	6.24	0.0439
		PTQ4DiT	339	7.75	22.01	190.38	0.7292
		G4W+P4A	351	7.66	20.76	193.76	0.7261
		Ours	347	**6.40**	**18.60**	**211.72**	**0.7609**

< Image generation, steps = 100, cfg = 1.5 (Tab.2) >

Q-DiT[1)]

- Experiments
  - Qualitative Results
    - G4W+P4A는 구조 왜곡과 artifact가 뚜렷
    - Q-DiT는 W4A8에서도 형태와 texture를 안정적으로 유지

< Samples with W4A8 on ImageNet 256 × 256 / 512 × 512 (Q-DiT Fig. 5) >

Q-DiT

- Discussion
  - Contributions
    - DiT의 channel-wise spatial variance와 temporal variance를 각각 고려
    - 생성 품질 기반의 automatic group-size allocation 제안
    - W6A8에서 near-lossless 성능, W4A8에서도 높은 생성 품질 유지
  - Limitations
    - Evolutionary search의 높은 탐색 비용
    - Activation은 8-bit로 유지되어 W4A4 연산을 지원하지 못함
    - BitOps만 제시하며 실제 latency 및 hardware speedup은 검증하지 않음

SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models [ICLR 2025]

SVDQuant[1)]

- Introduction
  - Why W4A4
    - Diffusion model은 compute-bound
      - LLM은 weight loading 병목으로 weight-only quantization도 효과적
      - Diffusion은 batch 1에서도 연산량이 커 W4A16의 가속 효과가 제한적
  - Weight와 activation의 bit-width를 함께 낮춰야 함
    - W4A16 연산은 높은 precision으로 처리되어 INT4 연산 이점을 충분히 활용하지 못함
    - W4A4는 저비트 연산을 직접 활용하여 memory와 latency를 동시에 절감

Prompt: *a cyberpunk cat holding a huge neon sign that says "SVDQuant is lite and fast", wearing fancy goggles and a black leather jacket.*

< FLUX.1-dev : NF4 W4A16 대비 LPIPS·latency 모두 우위 >

SVDQuant[1)]

- Methods
  - Quantization Error Decomposition
    - Linear layer의 출력과 4-bit 양자화 출력
      - b : batch size, m : input channel, n : output channel
        - ✓ $X \in \mathbb{R}^{b \times m}, \quad W \in \mathbb{R}^{m \times n} \quad \Rightarrow \quad Y = XW \quad \longrightarrow \quad Q(X)Q(W)$
      - 양자화 오차
        - ✓ $E(X, W) = \|XW - Q(X)Q(W)\|_F$
        - ✓ $E(X, W) \leq \underbrace{\|X\|_F \, \|W - Q(W)\|_F}_{\text{Weight quantization error}} + \underbrace{\|X - Q(X)\|_F (\|W\|_F + \|W - Q(W)\|_F)}_{\text{Activation quantization error}}$
      - 오차를 줄이기 위해 아래 네 항을 동시에 줄여야 함
        - ✓ $\|X\|_F, \quad \|W\|_F, \quad \|X - Q(X)\|_F, \quad \|W - Q(W)\|_F$

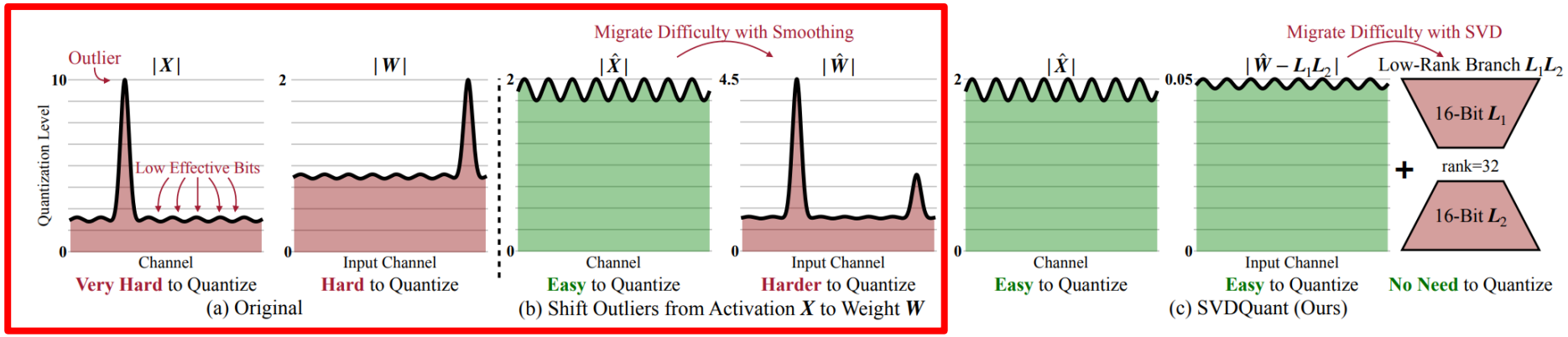
SVDQuant[1)]

- Methods
  - Step 1. Smoothing — activation outlier를 weight로 이동
    - 채널별 smoothing factor $\lambda \in \mathbb{R}^m$ 로 activation과 weight를 반대 방향으로 rescaling

$$\hat{X} = X\,\mathrm{diag}(\lambda)^{-1}, \quad \hat{W} = \mathrm{diag}(\lambda)\,W \quad \Rightarrow \quad XW = \hat{X}\hat{W}$$

      ✓ Smoothing 전 후 연산 결과가 동일

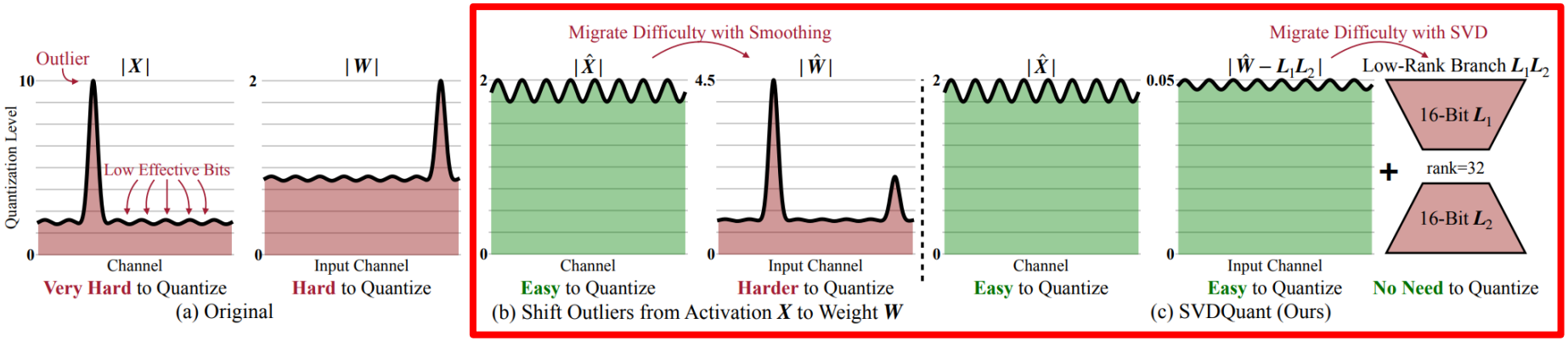
    - Activation 분포는 완화되지만 weight outlier가 증가
    - 따라서 smoothing만으로는 W4A4 양자화 오차를 해결할 수 없음



< Overview of SVDQuant (Fig. 3) >

SVDQuant[1)]

- Methods
  - Step 2. Weight를 low-rank 성분과 residual로 분리
    - Smoothed weight를 rank-r 성분과 나머지로 나눔
      - $\widehat{W} = L_1 L_2 + R, \quad L_1 \in \mathbb{R}^{m \times r}, \quad L_2 \in \mathbb{R}^{r \times n}, \quad R = \widehat{W} - L_1 L_2$
      - $XW = \hat{X}\widehat{W} = \hat{X}L_1 L_2 + \hat{X}R \approx \hat{X}L_1 L_2 + Q(\hat{X})Q(R)$
    - $L_1 L_2$는 FP16, R은 INT4로 quantization
      - 대부분의 연산은 R이 담당



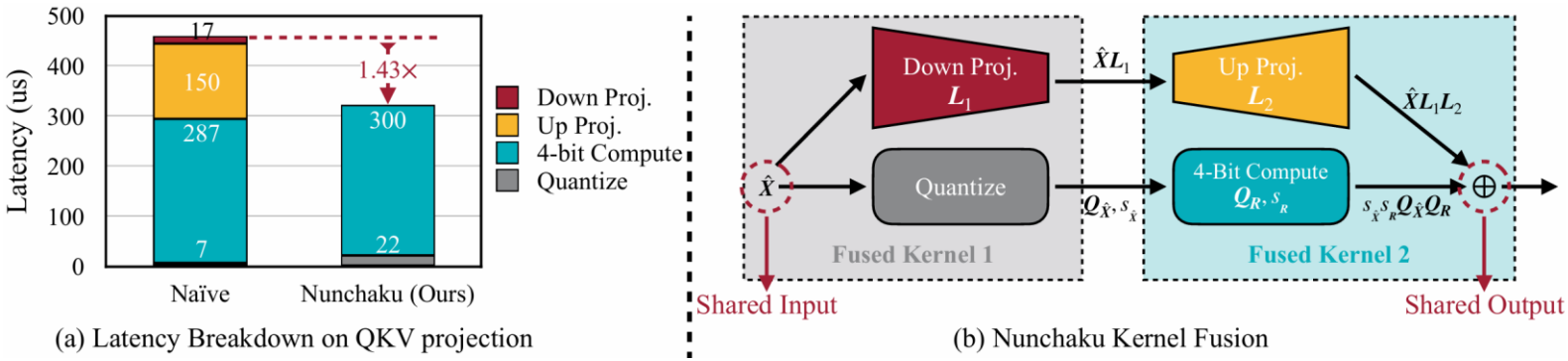
< Overview of SVDQuant (Fig. 3) >

SVDQuant[1)]

- Methods
  - Step 3. 남는 오차는 residual branch의 양자화 오차
    - 근사 출력과 원래 출력의 차이를 정리하면 low-rank 항이 소거
      - $\left\|\hat{X}\widehat{W} - \left(\hat{X}L_1L_2 + Q(\hat{X})Q(R)\right)\right\|_F = \left\|\hat{X}R - Q(\hat{X})Q(R)\right\|_F = \boldsymbol{E}(\hat{\boldsymbol{X}}, \boldsymbol{R})$
    - 근사 출력과 원래 출력의 차이를 정리하면 low-rank 항이 소거
      - $\hat{\boldsymbol{X}}$은 smoothing으로 이미 outlier 정리된 상태
      - 남은 변수는 $\|\mathrm{R}\|_F$, $\|\mathrm{R} - Q(\mathrm{R})\|_F$
  - Step 4. Residual의 크기를 줄이는 문제로 환원
    - Residual의 최대 절댓값이 전체 크기에 비례한다고 가정
      - $\mathbb{E}[\max(|R|)] \leq c \cdot \mathbb{E}[\|R\|_F]$
    - 이때 residual의 양자화 오차는 다음과 같이 bound
      - $\mathbb{E}[\|R - Q(R)\|_F] \leq \frac{c\sqrt{\mathrm{size}(R)}}{q_{\max}} \cdot \mathbb{E}[\|R\|_F]$
      - $\min_{L_1,L_2} \left\|\widehat{W} - L_1L_2\right\|_F$

SVDQuant[1)]

- Methods
  - Nunchaku kernel fusion
    - 문제 : low-rank branch를 따로 돌리면 오버헤드가 큼
      - rank 32에서도 약 57% latency 증가 (4-bit branch의 약 50% 수준)
        - ✓연산량은 작지만 입출력 activation 크기는 그대로
    - 해결 : 공유 텐서를 기준으로 kernel을 합침
      - Down proj. L_1은 quantize kernel과 입력을 공유 → Fused Kernel 1
      - Up proj. L_2는 4-bit compute kernel과 출력을 공유 → Fused Kernel 2
    - 오버헤드 5~10%로 축소, kernel 호출 수 절반



< Latency breakdown & Nunchaku kernel fusion (Fig. 6) >

SVDQuant[1)]

- Experiments
  - Experimental setup
    - Models — UNet과 DiT backbone을 모두 포함
      - DiT : FLUX.1-dev / -schnell (12B), PixArt-Σ, SANA-1.6B
      - UNet : SDXL, SDXL-Turbo
    - Data
      - Calibration : COCO Captions prompts
      - Evaluation : MJHQ-30K, sDCI 에서 각각 5K prompts
    - Metrics
      - 품질 : FID, ImageReward / 16-bit 대비 유사도 : LPIPS, PSNR
    - Baselines
      - NF4 (W4A16 weight-only), ViDiT-Q, MixDQ, TensorRT

SVDQuant[1)]

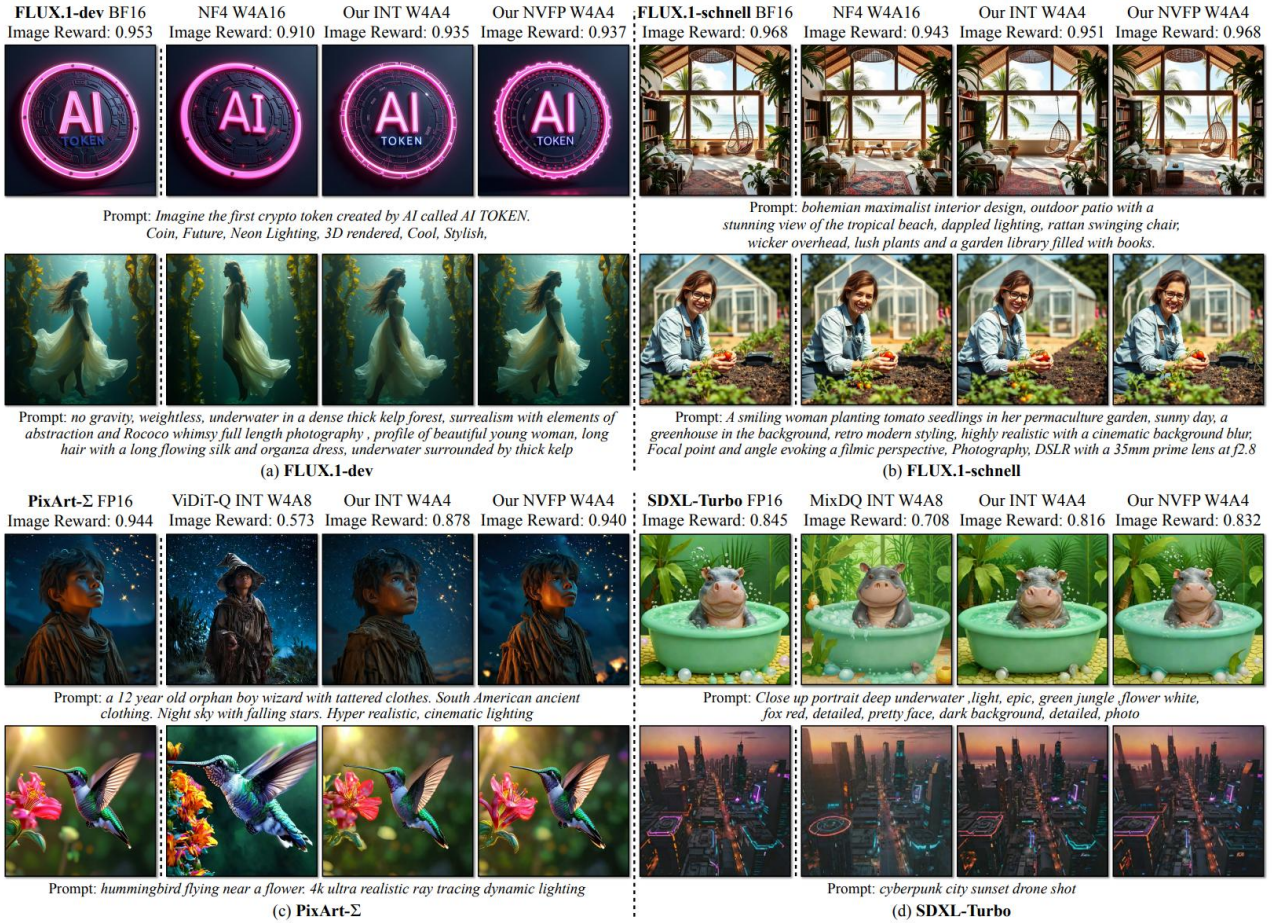
- Experiments
  - Quality results
    - W4A4가 weight-only W4A16보다 오히려 성능이 좋음

Backbone	Model	Precision	Method	MJHQ				sDCI			
				Quality		Similarity		Quality		Similarity	
				FID (↓)	IR (↑)	LPIPS (↓)	PSNR(↑)	FID (↓)	IR (↑)	LPIPS (↓)	PSNR (↑)
DiT	FLUX.1 -dev (50 Steps)	BF16	–	20.3	0.953	–	–	24.8	1.02	–	–
		INT W8A8	Ours	20.4	0.948	0.089	27.0	24.7	1.02	0.106	24.9
		W4A16	NF4	20.6	0.910	0.272	19.5	24.9	0.986	0.292	18.2
		INT W4A4	Ours	**19.9**	0.935	0.223	21.0	**24.2**	**1.01**	0.240	19.7
		NVFP W4A4	Ours	20.4	**0.937**	**0.208**	**21.4**	24.7	**1.01**	**0.218**	**20.2**
	FLUX.1 -schnell (4 Steps)	BF16	–	19.2	0.938	–	–	20.8	0.932	–	–
		INT W8A8	Ours	19.2	0.966	0.120	22.9	20.7	0.975	0.133	21.3
		W4A16	NF4	18.9	0.943	0.257	18.2	20.7	0.953	0.263	17.1
		INT W4A4	Ours	**18.3**	0.951	0.258	18.3	**20.1**	**0.979**	0.260	17.2
		NVFP W4A4	Ours	19.0	**0.968**	**0.227**	**19.0**	20.5	**0.979**	**0.226**	**18.1**
	PixArt-Σ (20 Steps)	FP16	–	16.6	0.944	–	–	24.8	0.966		
		INT W8A8	ViDiT-Q	**15.7**	0.944	0.137	22.5	**23.5**	**0.974**	0.163	20.4
		INT W8A8	Ours	16.3	**0.955**	**0.109**	**23.7**	24.2	0.969	**0.129**	**21.8**
		INT W4A8	ViDiT-Q	37.3	0.573	0.611	12.0	40.6	0.600	0.629	11.2
		INT W4A4	ViDiT-Q	412	-2.27	0.854	6.44	425	-2.28	0.838	6.70
		INT W4A4	Ours	19.2	0.878	0.323	17.6	25.9	0.918	0.352	16.5
		NVFP W4A4	Ours	**16.6**	**0.940**	**0.271**	**18.5**	**22.9**	**0.971**	**0.298**	**17.2**
	SANA -1.6B (20 Steps)	BF16	–	20.6	0.952	–	–	29.9	0.847	–	–
		INT W4A4	RTN	20.5	0.894	0.339	15.3	28.6	0.807	0.371	13.8
		INT W4A4	Ours	**19.3**	0.935	0.220	17.8	**28.1**	**0.846**	0.242	16.2
		NVFP W4A4	RTN	19.7	0.932	0.237	17.3	29.0	0.829	0.265	15.6
		NVFP W4A4	Ours	20.0	**0.955**	**0.177**	**19.0**	29.3	**0.846**	**0.196**	**17.3**

< MJHQ 기준 성능 비교 (Tab. 1) >

SVDQuant[1)]

- Experiments
  - Qualitative results



< Qualitative results on MJHQ >

SVDQuant[1)]

- Discussion
  - Contributions
    - Smoothing의 한계를 low-rank branch로 우회하는 새로운 4-bit paradigm
    - W4A4에서 품질 유지 + 실제 GPU에서 측정된 speedup을 함께 달성
    - UNet·DiT 공통 적용, INT4/NVFP4 지원, LoRA 재양자화 불필요
  - Limitations
    - 완전한 4-bit는 아님
      - 16-bit low-rank branch가 항상 필요
    - Nunchaku 없이는 오버헤드로 속도 이득이 사라짐 → engine 의존성
    - 모델 규모가 작을수록 불리
    - Rank 증가에 따른 trade-off는 본문이 아닌 appendix에서만 다룸

1) Lei Chen et al. "Q-DiT: Accurate Post-Training Quantization for Diffusion Transformers." CVPR 2025
2) Muyang Li et al. "SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models." ICLR 2025

감사합니다