INT v.s. FP: A Comprehensive Study of Fine-Grained Low-bit Quantization Formats

저자: Mengzhao Chen[1,2], Meng Wu[3], Hui Jin[2], Zhihang Yuan[2], Jing Liu[2], Chaoyi Zhang[2], Yunshui Li[2], Jie Huang[2], Jin Ma[2], Zeyue Xue[1], Zhiheng Liu[1], Xingyan Bin[2]†, Ping Luo[1]†

소속 : [1] The University of Hong Kong, [2] ByteDance Seed, [3] PicoHeart

비고 : † Corresponding authors

Introduction

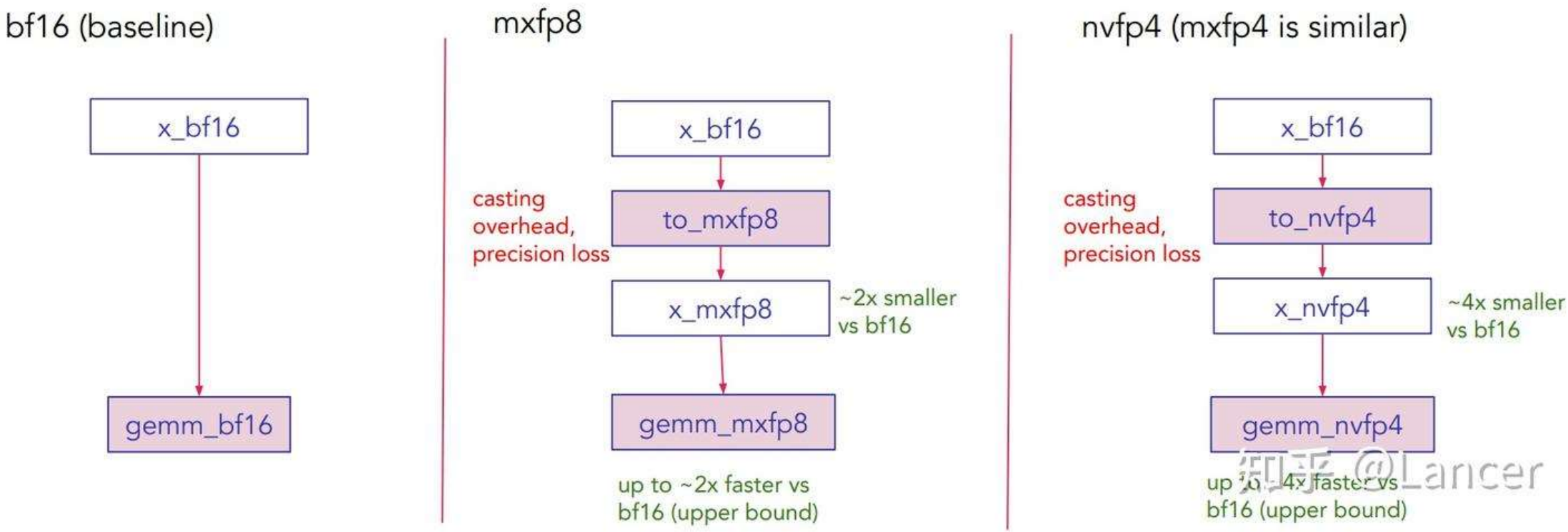
- 대규모 언어 모델(LLM)과 **양자화의 중요성**
  - LLM 확산으로 계산량·메모리 요구량 폭증 → 효율적 배포를 위해 양자화는 필수 기술
  - Transformer 기반 LLM에서는 **강한 Outlier 활성화 분포가** 저정밀도에서 큰 문제를 유발
  - Outlier의 폭넓은 동적 범위를 처리하기 위해 업계는 **FP8·FP4 등의 저정밀 부동소수점(FP)** 채택 가속화
    - 예: NVIDIA Blackwell 아키텍처는 FP가 INT보다 Outlier 대응에 안정적이라 강조
- **현재 논의의 한계**
  - 업계는 **FP 우위를 당연시**하지만 FP vs INT의 장단점을 **양자화 단위(granularity)별로 체계적으로 비교한 연구 부족**

    → 특히 **미세 블록(fine-grained) 환경**에서 두 형식의 성능·효율 트레이드오프는 아직 명확히 규명되지 않음

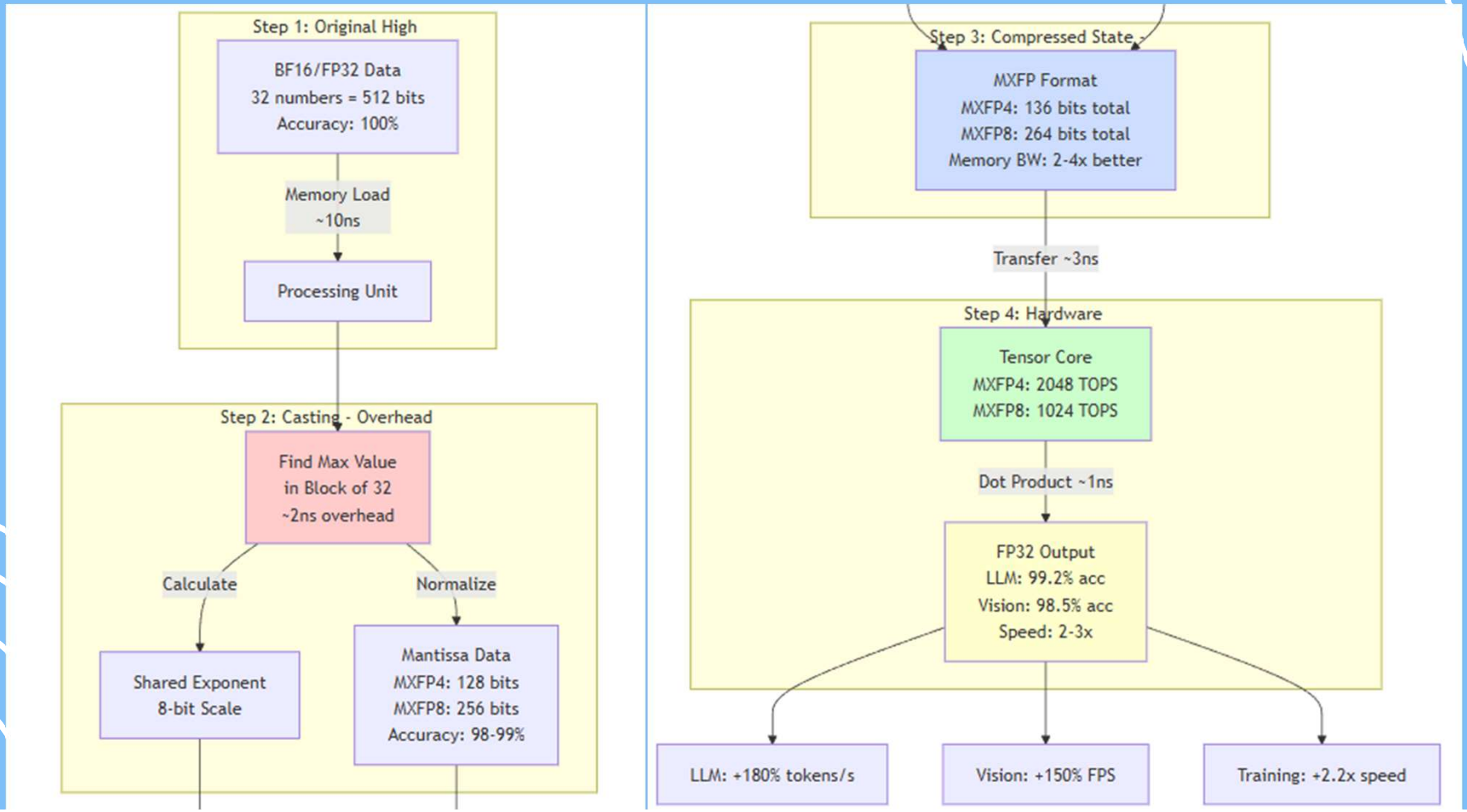
Introduction

- **실증적 발견**
  - 그러나 블록 단위가 작아질수록:
    - 블록 내부의 동적 범위가 축소
    - **INT의 균일 정밀도(uniform precision) 효과가 크게 강화**
    - 결국 **INT 형식이 FP 형식을 능가하는 '교차 지점(crossover point)' 존재**
  - Microscaling(MX, 32 elements) 및 NVIDIA(NV, 16 elements) 방식 모두에서 동일한 현상 관찰
- **정량적 비교를 위해 대응 관계 재정의**
  - 기존 FP 형식(MXFP8, MXFP6, MXFP4, NVFP4)에 상응하는 INT 형식(MXINT8, MXINT6, MXINT4, NVINT4)을 새롭게 정의 및 비교

Introduction



Introduction



Introduction

- **FP 및 INT의 Quantization SNR(QSNR) 이론·통계 프레임워크 제안**
  - 양자화 형식·블록 크기·비트 폭 간의 성능 트레이드오프를 수학적으로 직접 비교
  - 교차 지점(crossover point) 정량 규명
- **실험적 우위 입증**
  - **MXINT8**이 inference 및 low-bit training 모두에서 **MXFP8을 일관적으로 능가**
  - **NVINT4 + Hadamard rotation**이 **NVFP4보다 우수**
  - **대칭 클리핑(symmetric clipping)**을 새롭게 도입해 **MXINT8 low-bit training에서 성능 손실 최소화**
- **하드웨어 비용 분석**
  - 동일한 처리량 기준에서 **fine-grained INT가 FP 대비 칩 면적·에너지 측면에서 훨씬 우수**
- **종합 결론**
  - **현재의 FP 중심 하드웨어 설계 방향은 최적이 아님**
  - **차세대 AI 가속기 설계에서는 미세 블록 기반 INT 형식이 정확도·효율 측면에서 더 바람직한 선택지**

양자화 기초 및 Granularity (Quantization Basics)

- 양자화(Quantization)의 정의:
  - 고정밀도 텐서 **X**를 저비트로 변환하는 과정
  - **INT Quant**

$$\mathbf{X}_q = \text{clip}\left(\left\lfloor \frac{\mathbf{X}}{s} \right\rceil, Q_{\text{min}}, Q_{\text{max}}\right) \cdot s, \qquad Q_{\text{min}} = -2^{b-1}, \qquad Q_{\text{max}} = 2^{b-1} - 1.$$

  - **FP Quant**

$$\mathbb{C}_{\text{FP}} = \begin{cases} (-1)^s \times (1.m)_2 \times 2^{e-\text{bias}} & \text{if } e \neq 0 \text{ (Normal)}, \\ (-1)^s \times (0.m)_2 \times 2^{1-\text{bias}} & \text{if } e = 0,\ m \neq 0 \text{ (Subnormal)}, \end{cases} \tag{2}$$

$$\mathbf{X}_q = \text{Nearest}\left(\frac{\mathbf{X}}{s}, \mathcal{C}_{FP}\right) \cdot s, \tag{3}$$

INT Quant

- 비트수: 8-bit signed
  - $Q_{min} = -2^{b-1} = -128, Q_{max} = 2^{b-1} = 128$
  - 스케일: $s = 0.01$
  - 입력 값: $X = -2.5$

$$\mathbf{X}_q = \text{clip}\left(\left\lfloor \frac{\mathbf{X}}{s} \right\rceil, Q_{\text{min}}, Q_{\text{max}}\right) \cdot s, \tag{1}$$

$$Q_{\text{min}} = -2^{b-1}, \qquad Q_{\text{max}} = 2^{b-1} - 1.$$

INT Quant

1. 스케일로 나누기
   - $\frac{x}{s} = \frac{-2.5}{0.01} = -250$
2. 반올림
   - round(-250) = -250
3. 정수 범위로 클리핑
   - $Q_{min} = -128, Q_{max} = 128$
   - Clip(-250, -128, 127) = -128
4. 다시 실수영역으로 복원 (dequantize)
   - $x_q^{INT8} = -128 \cdot 0.01 = -1.28$

FP Quant

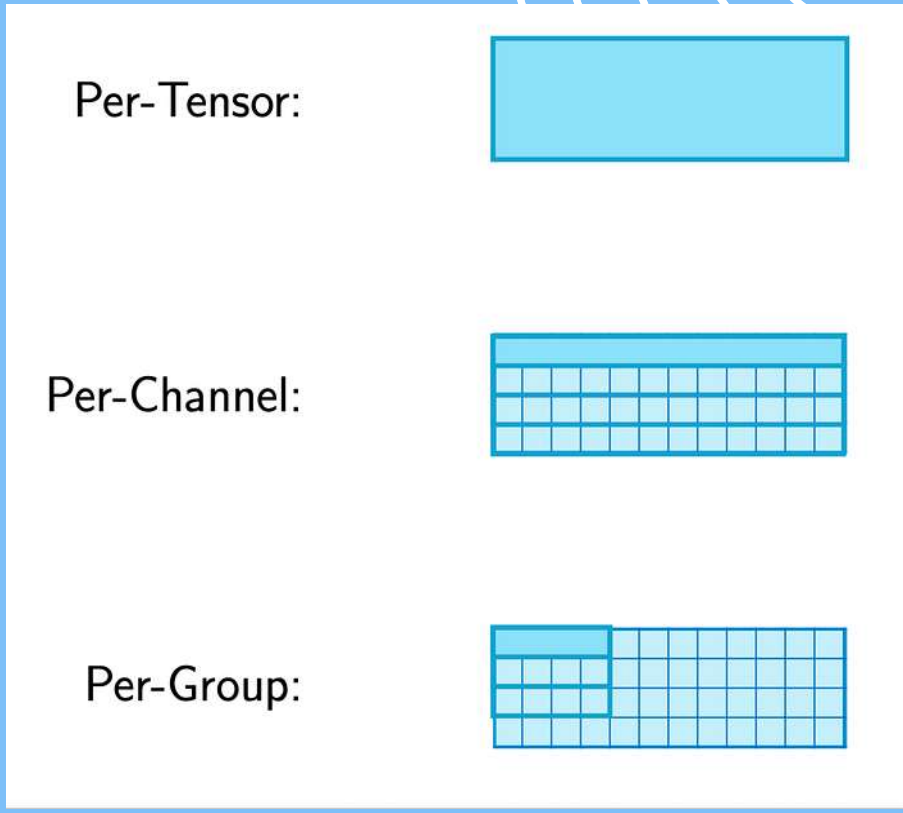
1. 동일하게 스케일로 나누기
   - $\frac{x}{s} = \frac{-2.5}{0.01} = -250$
2. FP8(E4M3)에서 표현 가능한 값들(일부만 예시)
   - $c_{FP} \ni \cdots, -224, -240, \ldots$
3. -250에서 가장 가까운 값 찾기
   - $Nearest(c_{FP}) = -240$
4. 다시 스케일을 곱해 복원 (dequantize)
   - $x_q^{INT8} = -240 \cdot 0.01 = -2.4$

양자화 기초 및 Granularity (Quantization Basics)

- Granularity (양자화 단위):
  - 스케일(s)을 얼마나 세밀하게 적용하는가?
  - Per-tensor: 텐서 전체 1개 스케일 (단순하지만 부정확)
  - Per-channel: 채널당 1개 (일반적)
  - Block-k (핵심):

    → 저비트에서 정확도 핵심.

    텐서를 1 X k 조각으로 나눠 블록마다 스케일 적용



최신 저비트 블록 포맷 (Microscaling Formats)

- OCP Microscaling (MX) 포맷:
  - **특징:** 블록 크기 **32**, 공유 스케일 **UE8M0** 사용
    - **UE8M0**: 지수 8비트, 가수 0비트로 구성된 8비트 unsigned floating-point 형식
  - **FP 변형 (NVIDIA 지원)**: MXFP8 (E4M3), MXFP6, MXFP4
  - *Note:* 가수(Mantissa) 비트가 많은 포맷(E4M3, E2M3)이 성능에 유리
- NVIDIA (NV) 포맷:
  - **특징**: 블록 크기 **16**, 스케일 **E4M3** + 2차 스케일(Per-tensor) 도입
  - **FP 변형**: NVFP4 (더 세밀한 블록)
- 본 연구의 접근 (Fair Comparison):
  - FP에 대응하는 **INT 변형**을 정의하여 직접 비교 수행
  - **New INT Formats**: MXINT8, MXINT6, MXINT4, NVINT4

최신 저비트 블록 포맷 (Microscaling Formats)

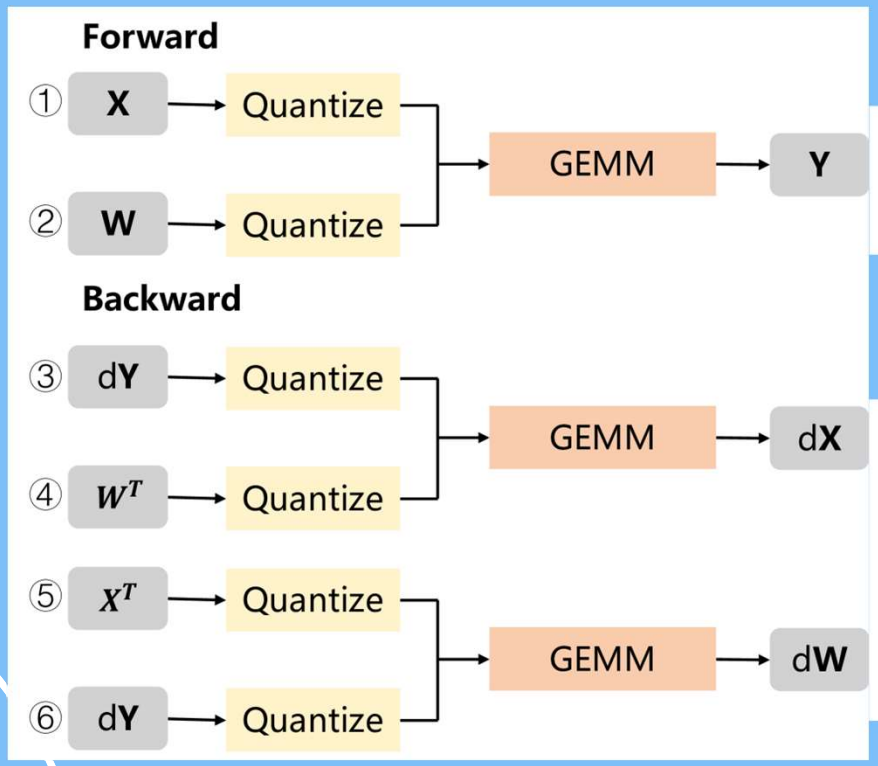
표 1: 저비트(low-bit) 형식의 명칭, 표현 가능한 값 범위, 그리고 스케일 구성 요소

Format	Block Size	Max Value	Min Value	Dynamic Range	Scale-1	Scale-2
MXFP8 (E4M3)	32	±448	$\pm 2^{-9}$	1.75×2^{17}	UE8M0	–
MXINT8	32	127	1	127	UE8M0	–
MXFP6 (E2M3)	32	±7.5	±0.125	60	UE8M0	–
MXINT6	32	±31	±1	31	UE8M0	–
MXFP4 (E2M1)	32	±6	±0.5	12	UE8M0	–
MXINT4	32	±7	±1	7	UE8M0	–
NVFP4	16	±6	±0.5	12	E4M3	FP32
NVINT4	16	±7	±1	7	E4M3	FP32

저비트 연산 흐름 (Quantization Compute Flow)

- Forward(순전파)와 Backward(역전파) 모두 저비트로 수행
  - Forward (2회): 입력(X)과 가중치(W) 양자화 → Y 계산
  - Backward (4회): 그래디언트(dY)와 전치된 가중치(w^T), 입력(x^T) 등 양자화 → dX, dW 계산
- Total 6 Quantizations:
  - ① X, ② W, ③ dY, ④ W^T, ⑤ X^T, ⑥ dY^T
- GEMM(General Matrix Multiply, 일반행렬곱) 연산 가속을 위해 각 텐서는 Reduction 차원(Axis)을 따라 블록 양자화됨

저비트 연산 흐름 (Quantization Compute Flow)



$$\mathbf{Y} = \underbrace{\text{Quantize}(\mathbf{X})}_{①}\underbrace{\text{Quantize}(\mathbf{W})}_{②}.$$

$$d\mathbf{X} = \underbrace{\text{Quantize}(\mathbf{dY})}_{③}\underbrace{\text{Quantize}(\mathbf{W}^T)}_{④},$$

$$d\mathbf{W} = \underbrace{\text{Quantize}(\mathbf{X}^T)}_{⑤}\underbrace{\text{Quantize}(\mathbf{dY}^T)}_{⑥}.$$

스케일링 및 대칭 클리핑

- 1. Scale Factor: "Round-Down" vs "Round-Up"
  - **Standard AbsMax Scale:**

$$s = \frac{\text{AbsMax}(\mathbf{X})}{Q_{max}},$$

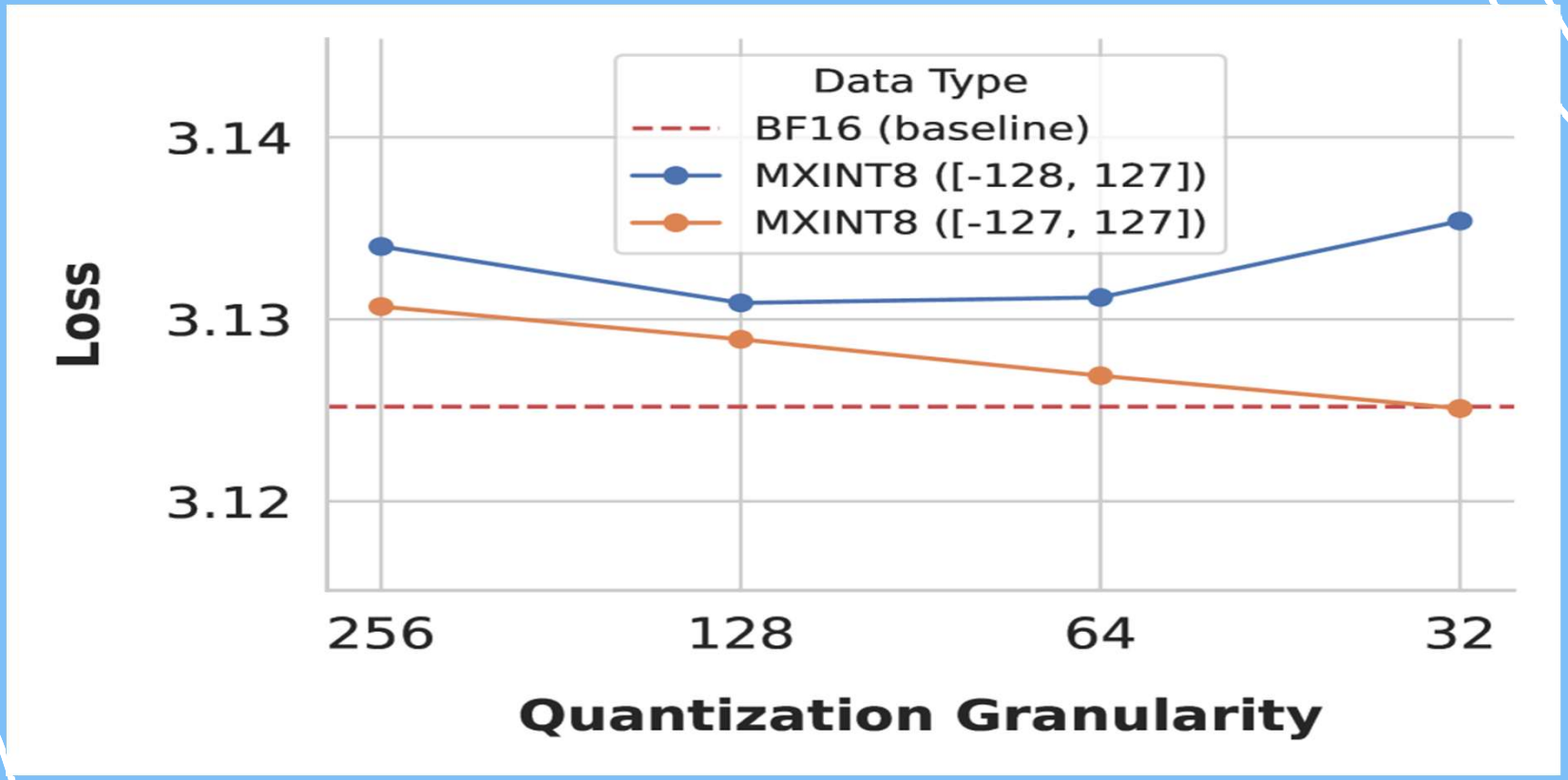
  - The Issue (OCP Style): 로그 스케일 변환 시 내림(Floor, $\lfloor \cdot \rfloor$) 적용 → 스케일이 작아져 Overflow 발생위험

$$s' = 2^{\text{clip}\left(\lfloor \log_2(\text{AbsMax}(\mathbf{X})) \rfloor - \lfloor \log_2(Q_{max}) \rfloor, -127, 127\right)}$$

  - Our Proposal (Round-Up): 올림(Ceil, $\lceil \cdot \rceil$)적용 → 스케일을 키워 Clipping Error 방지

$$s' = 2^{\text{clip}\left(\lceil \log_2(s) \rceil, -127, 127\right)}$$

스케일링 및 대칭 클리핑


Data Type
BF16 (baseline)
MXINT8 ([-128, 127])
MXINT8 ([-127, 127])
3.14
3.13
3.12
Loss
256
128
64
32
Quantization Granularity

이론적 분석 - QSNR 모델링 (Theoretical QSNR)

1. Why QSNR? (QSNR을 사용하는 이유)
   - **공통 척도 필요:** 구조가 다른 INT(균일 간격)와 FP(비균일 간격)의 **수치적 충실도(Numerical Fidelity)**를 직접 비교하기 위함
   - 정의: 원본 신호(X) 대비 양자화 잡음(X-X_q)의 비율 (dB 단위)

$$QSNR = -10 \log_{10} \left(\frac{\|X - X_q\|^2}{\|X\|^2} \right)$$

   - **의미:** QSNR이 높을수록 양자화된 벡터가 **원본의 크기와 방향을 더 잘 보존**함을 의미함

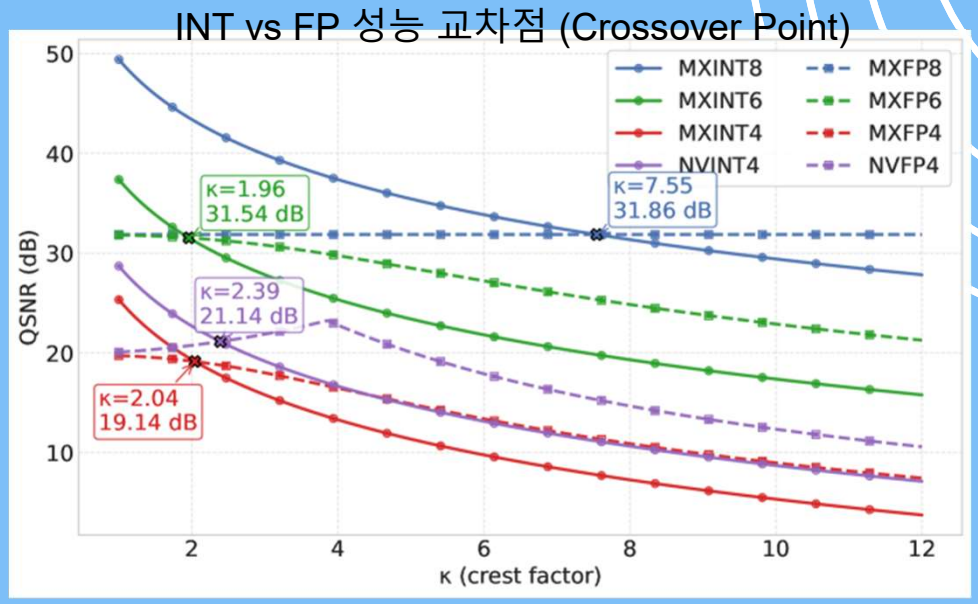
이론적 분석 - QSNR 모델링 (Theoretical QSNR)

2. 핵심 변수: Crest Factor (k)
   - 정의: $k = \frac{\max|X|}{\sigma}$ (데이터의 아웃라이어 강도)
   - **INT의 특성**: k가 커지면(Outlier 증가) 성능이 급격히 하락 ($-20log_{10}k$)
   - FP의 특성: 지수(Exponent) 덕분에 k에 강건하지만, **Mantissa 비트 수(M)**가 성능 상한을 제한함
- 유도된 공식 (Simplified):
   - $INT: QSNR \approx C_1 \cdot b - 20log_{10}k$ (Outlier에 민감)
   - $FP: QSNR \approx C_2 \cdot M$ (Mantissa에 의존, Outlier에 강건)

구체적인 예시 (왜 성능 상한이 걸리는가?)

- FP4 (E2M1) 포맷을 예로 들어보겠습니다. 가수(M)가 1비트입니다.이 포맷은 숫자를 표현할 때 유효숫자를 딱 2개(1.0 또는 1.5)밖에 못 가집니다.
- 상황: 데이터가 1.2345라고 가정합니다.
- 지수(Exponent): 크기(10^0 근처)는 잘 맞춥니다. (k 해결)
- 가수(Mantissa): 그런데 눈금이 부족해서 1.0 아니면 1.5로만 저장해야 합니다.
  - 결국 1.2345는 1.0이나 1.5로 강제 변환되면서 엄청난 오차가 생깁니다.
  - 이 오차는 지수를 아무리 잘 조절해도 줄일 수 없는 태생적인 한계입니다.
- 결론:FP는 큰 컵(Exponent)을 가지고 있어서 물(Outlier)을 흘리지 않고 잘 담는다. 하지만 컵 안에 눈금(Mantissa)이 거의 없어서, 물이 정확히 몇 ml인지는 대충밖에 모른다.

성능 교차점 분석 (Crossover Point Analysis)



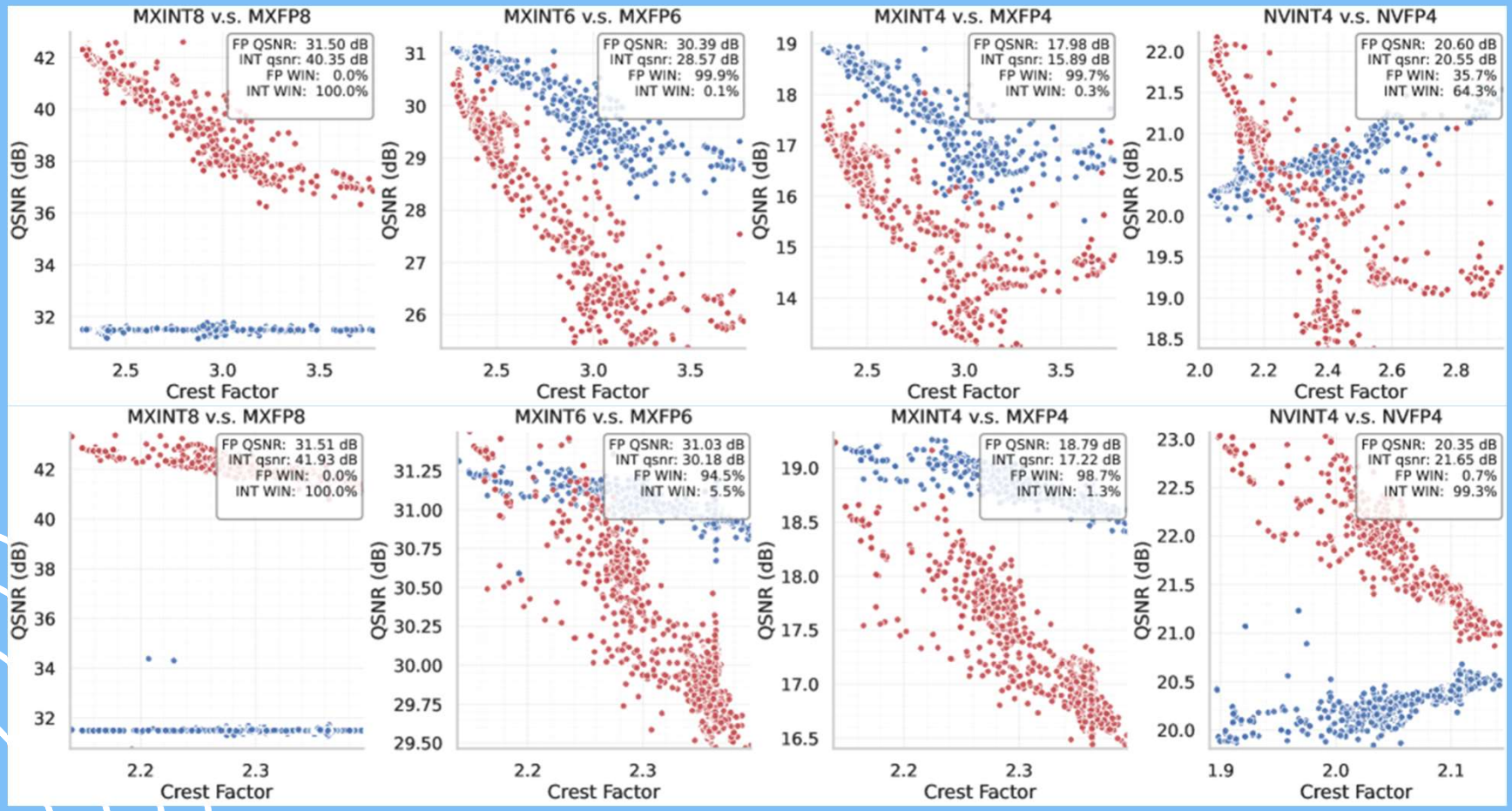
- Understanding the Graph (Figure 3)
  - INT (실선): k가 작을수록 QSNR 급상승.
  - FP (점선): k에 상관없이 일정 수준 유지.

- Crossover Point Found:
  - Case 1 (8-bit): k < 7.55이면 INT8 > FP8.
    - 실제 LLM 텐서는 k ≈ 2.96 → INT8 압승.
  - Case 2 (4-bit): 기본적으로는 FP4 > INT4.
    - However: Hadamard Rotation 적용 시 k 감소 → INT4 > FP4 역전 발생.
- Conclusion: Block-wise Quantization(= 작은 k) 환경에서는 INT가 FP보다 우월하다.

실제 데이터 분석 (Real-Data Analysis)

- Crest Factor (k) 분석 (Boxplot):
  - Coarse-grained (Per-channel): k ≈ 11.97 >>교차점 → FP 우세
  - Fine-grained (Block-wise): k ≈ 2.96
    - 2.96 < 7.55 → MXINT8 승리.
    - 2.96 > 1.96 → MXINT6/4 패배
  - Rotation Effect: k를 2.96 → 2.39로 감소시킴
    - **NVINT4가 NVFP4를 역전**하는 계기 마련.
  - QSNR 측정 결과:
    - **MXINT8 vs MXFP8**: 평균 40.35dB vs 31.50dB → INT8 압승.
    - **NVINT4 vs NVFP4**: Rotation 적용 시 INT4 승리 (21.65dB > 20.35dB).

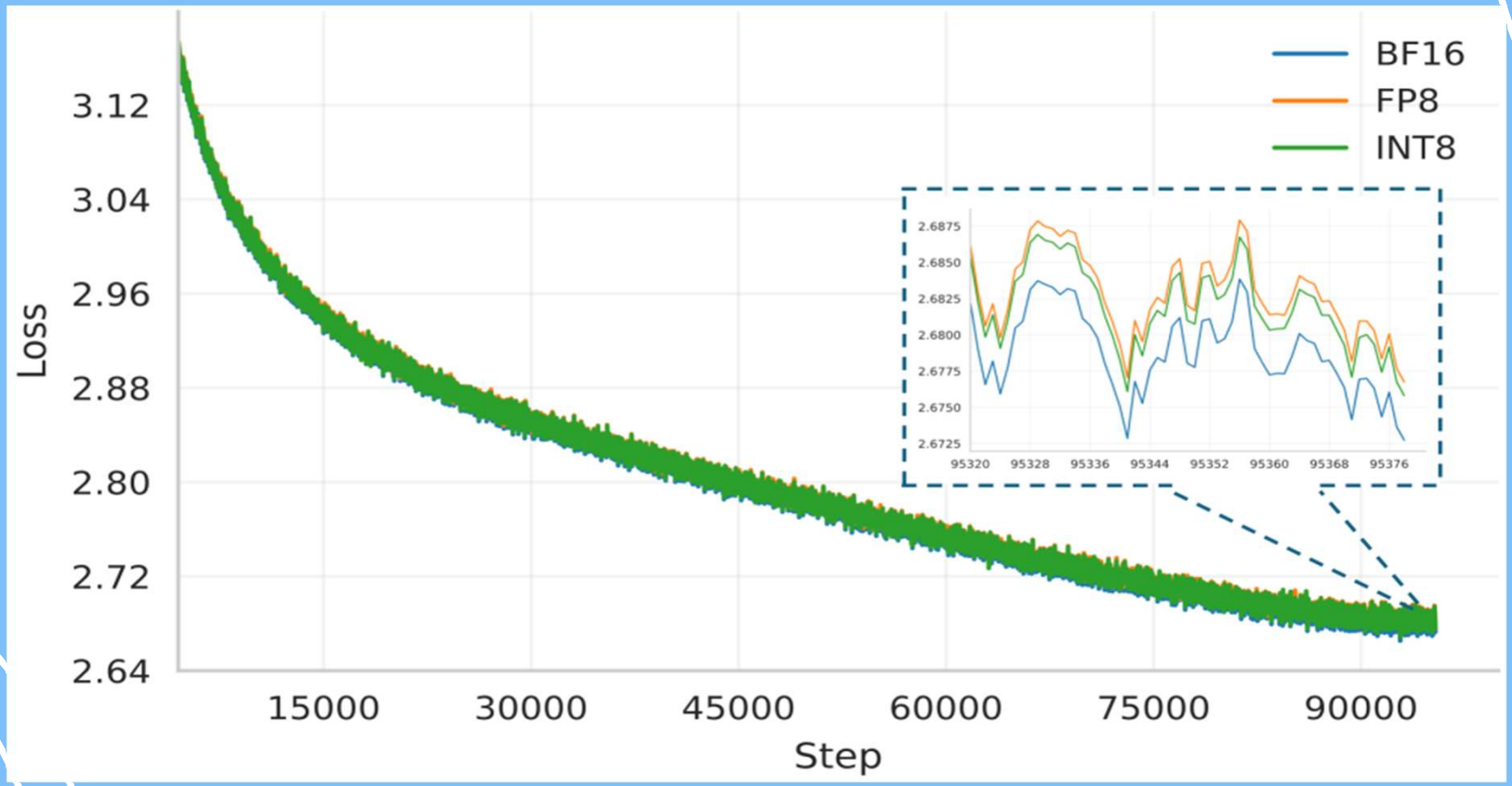
실제 데이터 분석 (Real-Data Analysis)



추론 및 학습 성능

- Inference Results (12 Models):
  - **Original:** MXINT8 (12/12 Win). Others lose.
  - **w/ Rotation: MXINT8 & NVINT4 (12/12 Win).**
- Training Results (Llama-3 1B/3B):
  - Loss Curve: MXINT8 ≈ BF16 (Fully overlaps).
  - **Accuracy:** MXINT8 achieves BF16-level accuracy on downstream tasks
  - **Comparison:** MXINT8 shows slightly lower loss (-0.001) than MXFP8

추론 및 학습 성능


BF16
FP8
INT8
Loss
3.12
3.04
2.96
2.88
2.80
2.72
2.64
15000
30000
45000
60000
75000
90000
Step
2.6875
2.6850
2.6825
2.6800
2.6775
2.6750
2.6725
95320
95328
95336
95344
95352
95360
95368
95376

하드웨어 비용 및 결론

- Hardware Efficiency (vs FP):
  - **MXINT8:** Energy **-37%**, Area **-21%**.
  - **NVINT4:** Energy **-66%**, Area **-62%**.
- Conclusion:
  - Myth Busted: "FP is always better for outliers" is false in fine-grained quantization
  - Winner: MXINT8 (Accuracy + Efficiency)
  - Proposal: Prioritize Fine-grained INT formats for future AI accelerators

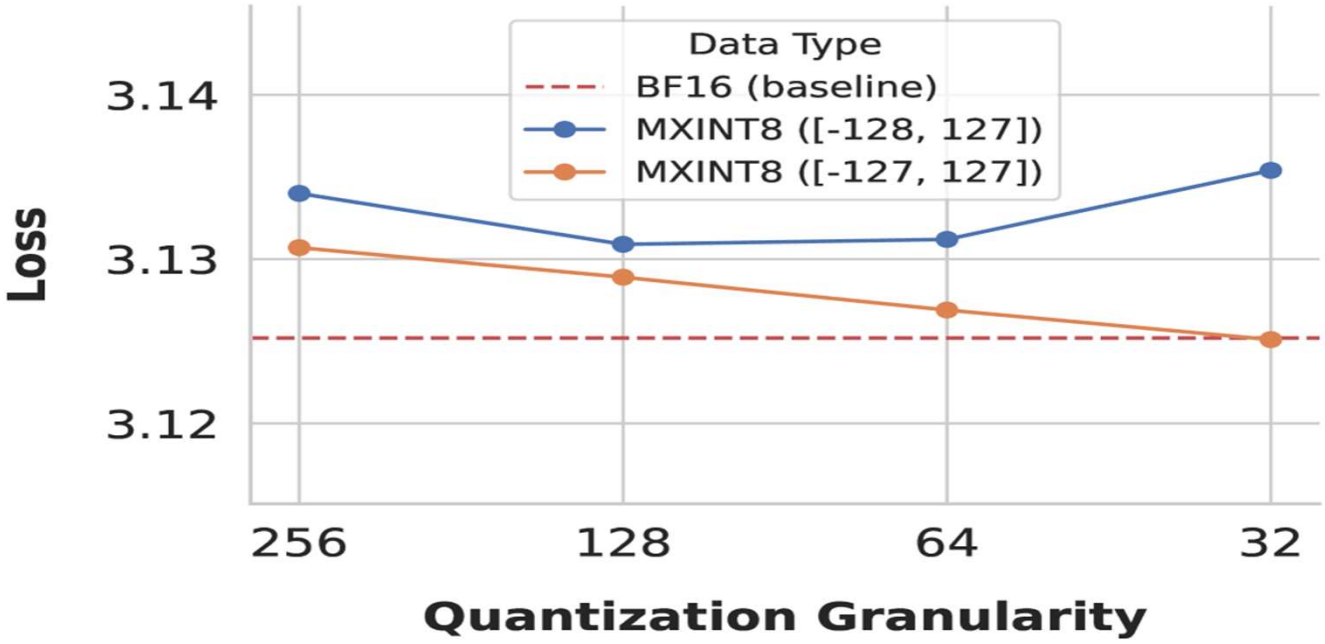
하드웨어 비용 분석 (Hardware Cost Analysis)

- . Single-Format Efficiency (표 5 참조):
  - **MXINT8:** MXFP8 대비 에너지 **37% 절감**
  - **NVINT4:** NVFP4 대비 에너지 **38% 절감**
- 2. Mixed-Format Efficiency (Blackwell Style):
  - Setup: NVIDIA Blackwell처럼 8-bit와 4-bit를 모두 지원하며, 처리량 비율을 1:2로 설정
  - Result: "MXINT8 + NVINT4" 구성이 FP 조합 대비 면적 34% 추가 절감
- 3. Why? (이유)
  - INT 파이프라인의 회로 재사용(Circuit Reuse) 구조가 FP보다 훨씬 단순하기 때문.
  - FP는 지수/가수 정렬, 정규화 등 복잡한 로직이 필요함.

	Single Format				Mixed Format	
	MXFP8	MXINT8	NVFP4	NVINT4	MXFP8+NVFP4	MXINT8+NVINT4
Energy	1x	**0.63x**	0.55x	**0.34x**	1x	**0.75x**
Area	1x	**0.79x**	0.54x	**0.38x**	1x	**0.66x**

결론 및 제언

- Final Verdict:
  - Coarse-grained: FP Wins
  - Fine-grained (Block-wise): INT Wins (MXINT8 > MXFP8)
  - 4-bit: NVINT4 > NVFP4 (with Hadamard Rotation)
- Call to Action:
  - Challenge the current FP-centric trends (e.g., Blackwell).
  - Proposal: Shift towards Fine-grained INT Co-design for next-gen AI accelerators.



1. FP는 변환 오버헤드와 정밀도 손실이 있고, LLM에서의 outlier를 완벽히 해결하지 못합니다.
2. LLM의 실제 κ≈3에서는 MXINT8이 MXFP8을 압도합니다. 4-bit에서도 rotation 적용 시 NVINT4가 역전합니다."
3. 정확도가 더 좋은 데다가 칩 크기와 전력까지 20~60% 절감. FP 기반 가속기보다 INT 기반 아키텍처가 더 경제적입니다."

	Single Format				Mixed Format	
	MXFP8	MXINT8	NVFP4	NVINT4	MXFP8+NVFP4	MXINT8+NVINT4
Energy	1x	**0.63x**	0.55x	**0.34x**	1x	**0.75x**
Area	1x	**0.79x**	0.54x	**0.38x**	1x	**0.66x**