2026 동계 세미나

Efficient Attention

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

Presented By
양진철

Outline

- Background

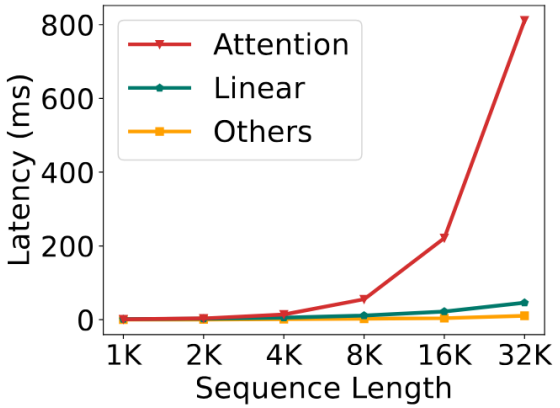
- Papers
  - SageAttention: Accurate 8-Bit Attention for Plug-and-play Inference Acceleration [ICLR 2025]

  - SpargeAttention: Accurate sparse attention accelerating any model inference [ICML 2025]

Background

- Attention
  - 긴 시퀀스 N 처리 시 메모리 사용량 증가 ($O(N^2)$)

$$\text{Attention}(Q,K,V) = \underbrace{\text{softmax}\left(\frac{\overbrace{QK^\top}^{\text{S}}}{\sqrt{d}}\right)}_{\text{P}} V \quad \text{where } Q,K,V \in \mathbb{R}^{N\times d}$$



  - I/O 병목
    - N x N 행렬을 HBM에 기록 및 읽어오는 과정에서 병목 현상 발생
      - 메모리 대역폭이 연산 속도에 비해 상대적으로 느림

- FlashAttention[1)]
  - Tiling & Online Softmax
    - N x N 행렬을 작은 block 단위로 쪼개서 계산
    - SRAM (on-chip memory)을 사용하여 HBM (off-chip memory) 최소화로 사용
    - Block별로 계산한 결과를 합쳐서 softmax 값 계산

$$S_{ij} = Q_i K_j^\top / \sqrt{d}, \quad (m_{ij}, \widetilde{P}_{ij}) = \tilde{\sigma}(m_{i,j-1}, S_{ij}),$$

현재 block score — Overflow 방지 — 최댓값 업데이트 — 이전 block 최댓값

Softmax 분모 업데이트

$$l_{ij} = \exp(m_{i,j-1} - m_{ij}) l_{i,j-1} + \text{rowsum}(\widetilde{P}_{ij}),$$

이전 분모 보정 — 현재 분모 추가

$$O_{ij} = \text{diag}\left(\exp(m_{i,j-1} - m_{ij})\right) O_{i,j-1} + \widetilde{P}_{ij} V_j$$

이전 결과 보정 — 현재 결과 추가 — 최종 결과 업데이트

Background

- Quantization
  - 모델 파라미터의 표현 정밀도를 낮추는 과정 (weights, activations)
    - Floating point (FP) value → INT value
  - Basic concepts

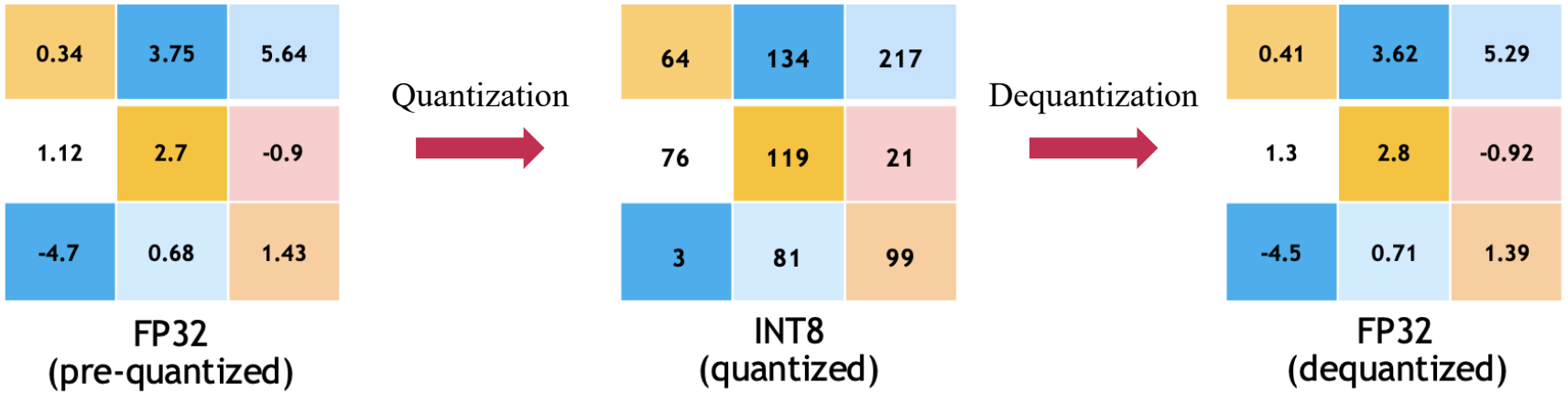
Quantization : $x_q = \text{clamp}(\left\lfloor \frac{x}{s} \right\rceil + z, 0, 2^b - 1)$

Dequantization : $\hat{x} = s \cdot (x_q - z)$

scale factor $s = \frac{\beta - \alpha}{2^b - 1}$

Zero-point $z = \left\lfloor -\frac{\min(x)}{s} \right\rceil$

  - Granularity
    - 얼마나 세부적으로 quantization을 진행하는지 (per-layer, -channel, -token ..)



SageAttention: Accurate 8-Bit Attention for Plug-and-play Inference Acceleration [ICLR 2025]

1) Zhang, Jintao, et al. "Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration." *arXiv preprint arXiv:2410.02367* (2024).
2) Dao, Tri, et al. "Flashattention: Fast and memory-efficient exact attention with io-awareness." *Advances in neural information processing systems* 35 (2022): 16344-16359.

SageAttention[1)]

- Introduction
  - Goal
    - FlashAttention[2)]에 효율적인 quantization 방법 제안
  - Analysis
    - K matrix는 채널별로 상당한 outlier 존재 → quantization 손실 발생
    - P, V를 INT8로 quantization → 다양한 시나리오에서 P, V의 정확도 일관성을 보장 X

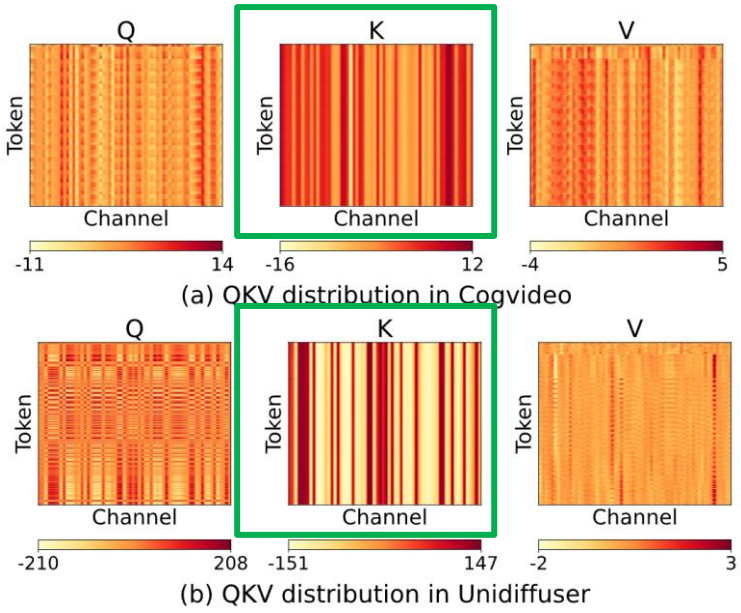


Table 2: **Average accuracy** using different data types across all layers of real models.

Q, K	$\widetilde{P}, V$	Cos Sim ↑	Relative L1 ↓	RMSE ↓
INT8	E4M3	99.94%	0.0345	3.53e-3
	E5M2	99.81%	0.0572	6.11e-3
	INT8	99.70%	0.1035	6.82e-3
E4M3	E4M3	99.81%	0.0607	5.93e-3
	E5M2	99.68%	0.0769	7.72e-3
	INT8	99.58%	0.1199	8.31e-3
E5M2	E4M3	99.37%	0.1107	1.09e-2
	E5M2	99.22%	0.1213	1.20e-2
	INT8	99.13%	0.1583	1.24e-2

Table 3: **Worst accuracy** using different data types across all layers of real models.

Q, K	$\widetilde{P}, V$	Cos Sim ↑	Relative L1 ↓	RMSE ↓
INT8	E4M3	76.36%	0.5899	0.4311
	E5M2	78.98%	0.4233	0.4371
	INT8	56.40%	0.7921	0.5405
	FP16	99.99%	0.0116	0.0091

SageAttention[1)]

- Smooth Matrix K
  - K는 채널 별로 outlier가 뚜렷한 패턴이 존재
    - 각 토큰의 K는 모든 토큰이 공유하는 큰 bias에 각 토큰별 작은 signal을 더한 형태
      - Outlier는 토큰들 사이의 값이 크게 변해서 생기는 게 아니라 큰 bias 때문에 생성
  - Quantization granularity for K
    - QK 연산: Q (N x d)와 K^T (d x N)의 곱셈 연산

$$\text{Result}_{ij} = \sum_{k=1}^{d} (Q_{ik} \times K_{jk})$$

- k는 1 ~ d까지 더해지므로 채널 차원으로 연산

    - Per-channel quantization (X): 채널마다 다른 scale (S)를 사용

$$\sum_{k=1}^{d} (Q_{ik} \times K_{jk} \times \mathbf{S_k})$$

- S_k가 k에 따라 변동되어 빠른 계산이 불가능 (inner axis)

    - Per-token quantization (O): 토큰 차원에 다른 scale (S)를 사용

$$\sum_{k=1}^{d} (Q_{ik} \times K_{jk} \times \mathbf{S_j}) \Rightarrow \mathbf{S_j} \times \sum_{k=1}^{d} (Q_{ik} \times K_{jk})$$

- S_j는 k와 무관하여 S_j는 QK 연산 이후에 곱셈 연산

SageAttention[1)]

- Smooth Matrix K
  - Bias 제거를 통한 K matrix 변형

$$\gamma(K) = K - \text{mean}(K)$$

    - Smoothing을 통해 K 값들을 0 근처로 이동시켜 데이터 분포가 고르게 변동
    - 연산이 느린 per-channel quantization을 사용하지 않아도 빠르고 높은 정확도 확보 가능
  - K matrix 변형에 의한 attention score

scalar

$$\sigma(q(K - \text{mean}(K)^\top)) = \sigma(qK^\top - q \cdot \text{mean}(K)) = \sigma(qK^\top)$$

    - q: 하나의 query token이므로 1 x d 크기의 벡터
    - mean(K): channel 마다 token의 평균이므로 1 x d 크기의 벡터

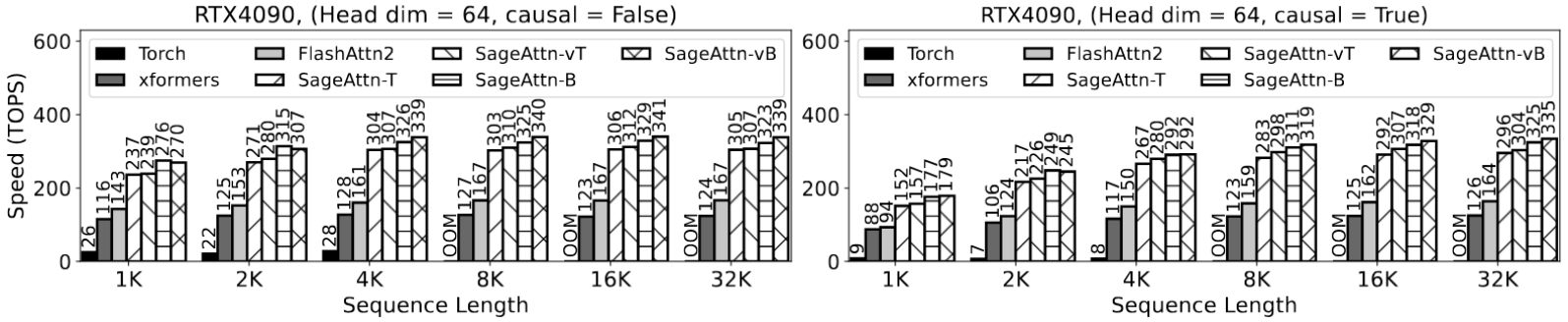
$$\sigma(x - c) = \frac{e^{x_i - c}}{\sum_j e^{x_j - c}} = \frac{e^{x_i} e^{-c}}{e^{-c} \sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = \sigma(x)$$

    - 따라서, K matrix 변형이 있어도 attention score는 동일

SageAttention[1)]

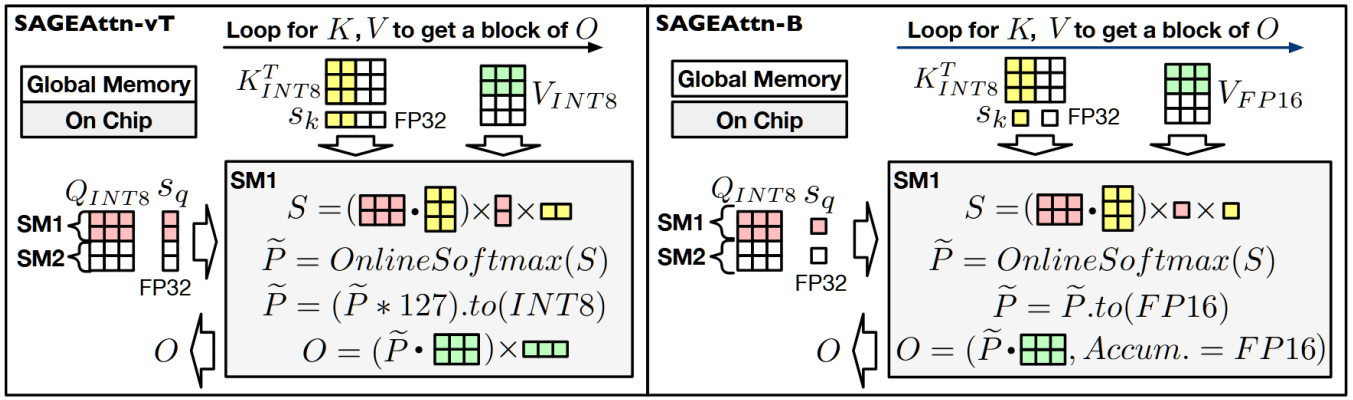
- Optimal Quantization
  - 다양한 data type, granularity에서 최적의 attention 연산 (Q, K, P, V)
    - Query: INT8 type, per-token (inner axis)
    - Key: INT8 type, per-token(+smoothing) (inner axis)
    - P: FP16 type, per-block
    - Value: FP16 type, per-channel

Kernel	$\psi_Q(Q), \psi_K(K)$	$\psi_P(P)$	$\psi_V(V)$
SAGEAttn-T	per-token, INT8	FP16, FP16 Accumulator	FP16, FP16 Accumulator
SAGEAttn-B (Algorithm 1)	per-block, INT8	FP16, FP16 Accumulator	FP16, FP16 Accumulator
SAGEAttn-vT (Figure 5(a))	per-token, INT8	per-block, INT8	per-channel, INT8
SAGEAttn-vB	per-block, INT8	per-block, INT8	per-channel, INT8



SageAttention[1)]

- Optimal Quantization
  - SAGEAttn-vT
    - Target: 추론 속도
      - SAGEAttn-B 대비 4% 더 빠름 , 정확도 검증이 완료된 레이어
    - Q, K (per-token, INT8), P (per-block, INT8), V (per-channel, INT8)
  - SAGEAttn-B
    - Target: 정확도와 범용성
      - FlashAttention2 대비 2.1x , 모든 모델
    - Q, K (per-block, INT8), P(FP16, FP16 Accumulator), V (FP16, FP16 Accumulator)



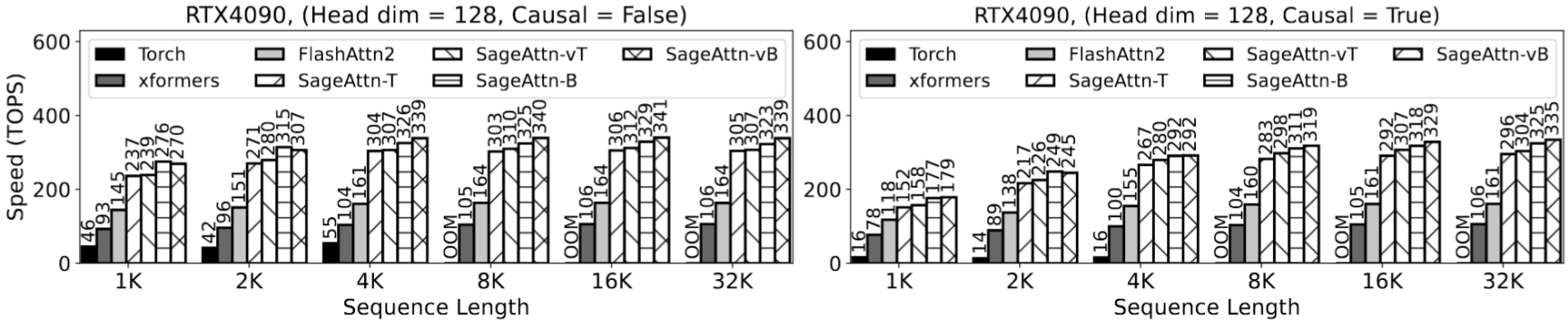
(a) SageAttention (per-token quantize Q,K; INT8 V) (b) SageAttention (per-block quantize Q,K; FP16 V)

SageAttention[1)]

- Experimental Results
  - Quantitative results - speed
    - SageAttention은 original attention 보다 약 2.83× 빠른 추론 속도 (RTX4090)

Model	Shape of Q, K, V	Original attention	SageAttention	Speedup
CogvideoX	(2, 30, 17776, 64)	163.37 (FlashAttn2)	**327.57**	**2.01x**
Llama2	(4, 32, 1536, 128)	130.99 (FlashAttn2)	**231.74**	**1.77x**
UltraPixel	(2, 32, 7285, 64)	152.03 (FlashAttn2)	**325.18**	**2.14x**
Unidiffuser	(4, 24, 1105, 64)	105.68 (xformers)	**246.93**	**2.34x**
TIMM	(12, 64, 197, 64)	18.910 (Torch)	**111.41**	**5.89x**

    - SageAttention은 341 TOPS(operations per second) 달성
    - FlashAttention2 보다 약 2.1×, xformers 보다 2.9x 빠른 추론 속도 (RTX4090)



SageAttention[1)]

- Experimental Results
  - Quantitative Results
    - Text, image, video generation 모델에서의 평가

Model	Attention	WikiText (Ppl.) ↓	Lambda (Acc.) ↑	MMLU (Acc.) ↑
Llama2	Full-Precision	5.823	0.886	0.46
	SageAttention	**5.824**	**0.887**	**0.46**

Model	Attention	CLIPSIM ↑	CLIP-T ↑	VQA-a ↑	VQA-t ↑	FScore ↑
CogvideoX	Full-Precision	0.1837	0.9976	68.962	75.925	3.7684
	SageAttention	**0.1836**	**0.9976**	**68.839**	**75.037**	**3.8339**

Model	Attention	FID ↓	sFID ↓	CLIP ↑	IR ↑
Unidiffuser	Full-Precision	163.33	145.08	0.3152	0.1609
	SageAttention	**166.49**	**143.18**	**0.3154**	**0.1521**
UltraPixel	Full-Precision	179.78	141.35	0.3132	0.6169
	SageAttention	**179.79**	**141.63**	**0.3131**	**0.6110**

Model	Attention	ImageNet (Acc.) ↑	Sketch (Acc.) ↑	ImageNet-r (Acc.) ↑
TIMM	Full-Precision	84.79%	45.32%	59.55%
	SageAttention	**84.74%**	**45.78%**	**60.32%**

Model	Attention	TextVQA (Acc.) ↑	POPE (Acc.) ↑	VQAv2 (Acc.) ↑
Llava1.6	Full-Precision	60.25%	86.45%	77.55%
	SageAttention	**60.09%**	**86.44%**	**77.47%**

SageAttention[1)]

- Experimental Results
  - Quantitative Results
    - Smooth K 적용

Quantization (Q, K)	Smoothing K	Llama WikiText ↓	CogVideo (Fscore) ↑	Unidiffuser (FID) ↓	UltraPixel (FID) ↓	TIMM ImageNet ↑
Full-Precision	-	5.823	3.768	163.33	179.78	84.79%
Per-token	✗	5.824	1.924	221.18	193.36	84.21%
	✓	5.824	**3.734**	**166.52**	**179.79**	84.74%
Per-block	✗	5.825	2.014	229.08	195.67	84.18%
	✓	5.824	**3.718**	**166.93**	**179.98**	84.76%
Per-tensor	✗	5.826	1.902	267.06	196.26	84.12%
	✓	5.824	**3.640**	**167.65**	**180.21**	84.69%
FlashAttn3 (with quant)		5.850	3.394	394.13	383.61	84.70%

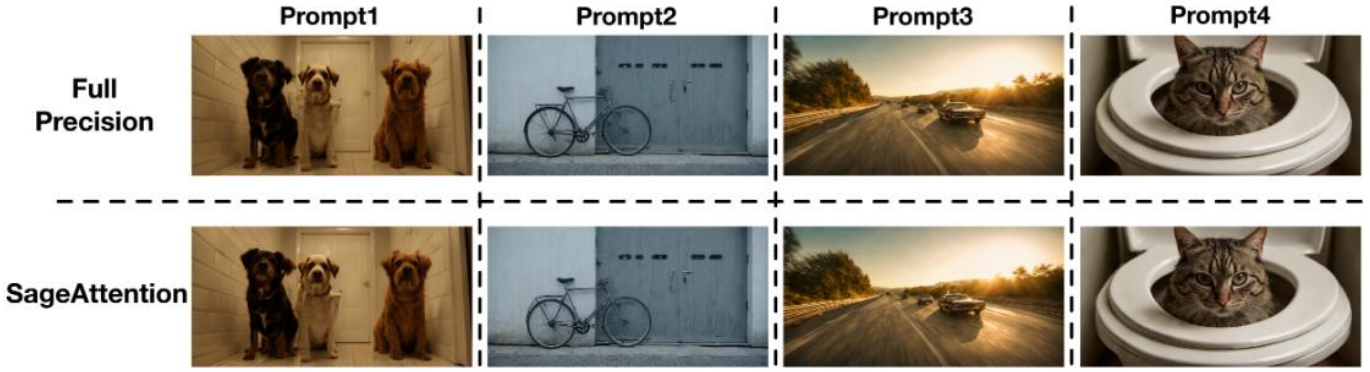
    - Overhead smooth K
      - 약 0.2% 감소로 미미한 수준

Model	Smooth K	TOPS ↑
CogvideoX	✗	327.57
	✓	**327.52**
UltraPixel	✗	325.18
	✓	**324.56**

SageAttention[1)]

- Experimental Results
  - Qualitative Results
    - An example of SageAttention on video generation (CogvideoX)

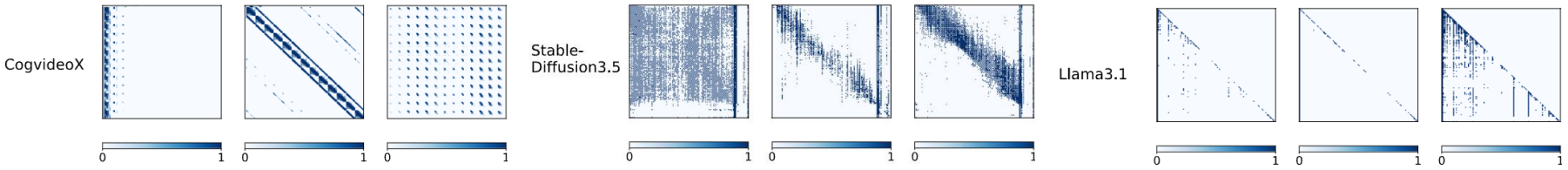
    - More image generation examples of UltraPixel



SpargeAttention: Accurate sparse attention accelerating any model inference [ICML 2025]

SpargeAttention[1)]

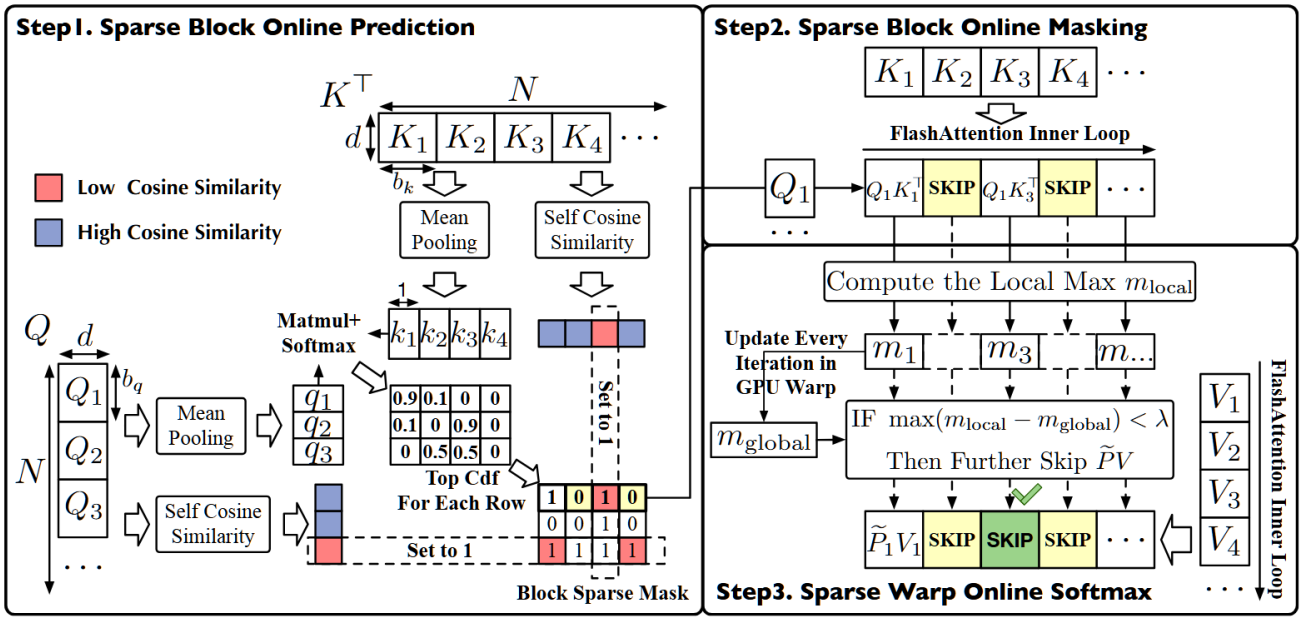
- Introduction
  - Goal
    - Sparse FlashAttention 연산을 성능 손실 없이 효율적으로 동작하게 하기 위한 방법 제안
      - Sparse FlashAttention
        - ✓ FlashAttention의 tiling 기법을 사용하면서 mask를 통해 불필요한 계산하지 않아 빠른 추론 속도
  - Limitation
    - Universality
      - 기존 방법론들은 특정 task에서만 sparse attention이 적용되어 일반화가 필요
    - Usability
      - 기존 방법의 Efficiency & overhead 문제



< patterns of attention map P in video, image, and language generation models >

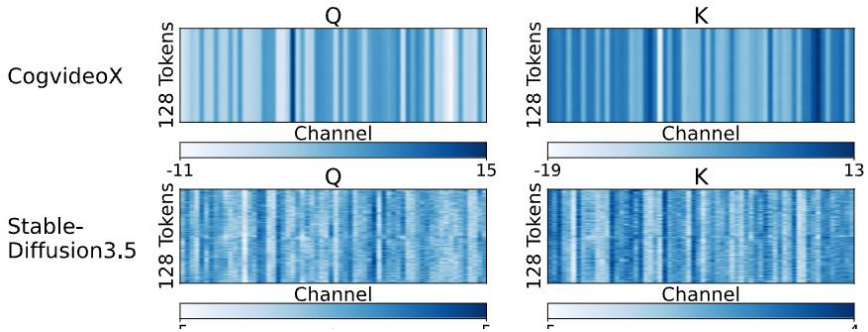
SpargeAttention[1)]

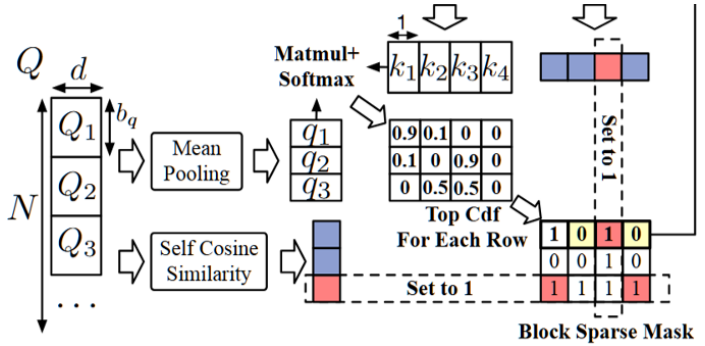
- Overview of SpargeAttention
  - Sparse Block Online Prediction & Masking
    - Attention map에서 sparse block을 예측하는 빠르고 정확한 방법 제안
    - QK와 PV의 곱을 skip 하는 방식
  - Sparse Warp Online Softmax
    - PV 곱을 추가적으로 skip 하는 sparse online softmax 방법 제안



SpargeAttention[1)]

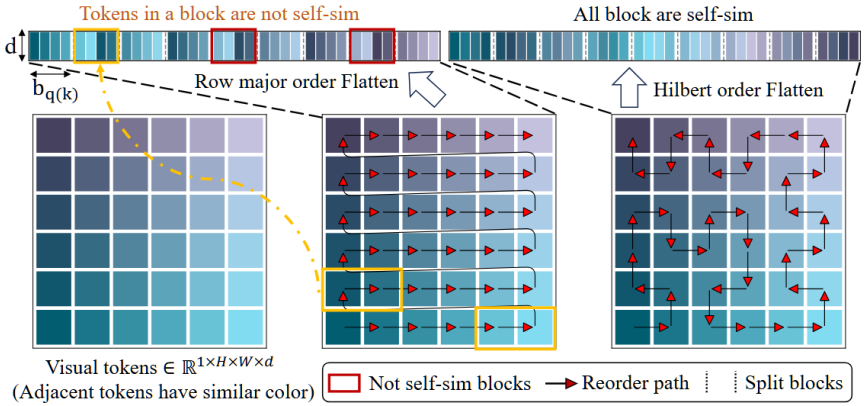
- Proposed Method
  - Sparse Block Online Prediction & Masking
    - Selective Token Compression for Sparse Prediction
      - Key idea
        - ✓ 다양한 모델에서 attention의 Q, K matrix에서 인접한 토큰들의 높은 유사성
        - ✓ 이는 토큰들을 대표하는 단일 토큰으로 통합 가능
    - Attention map에 sparse block을 식별하는 pattern-free online prediction 방법 제안
      - Q, K에서 high self-similarity가 보이는 block들을 하나의 토큰으로 압축
      - 압축된 Q, K로 attention map 계산하여 높은 score에 대해서만 선택적으로 계산
        - ✓ Cosine similarity가 낮은 부분의 행, 열 부분을 1로 설정





SpargeAttention[1)]

- Proposed Method
  - Sparse Block Online Prediction & Masking
    - Hilbert Curve Permutation
      - 공간적으로 가까운 토큰들을 한 block에 모이도록 순서 재배치
      - Block 안의 토큰들이 서로 비슷할수록 압축 토큰이 block을 더 잘 대표하고 이를 통해 TopCdf에서 더 많은 block을 제거



    - TopCdf
      - Attention score에서 일정 비율 (tau)에 도달할 때까지 큰 순서대로 고르는 방법

```
def Top_Cdf(P[i], tau):
    sorted_P, idx = torch.sort(P[i], descending=True)
    cusum_P = torch.cumsum(sorted_P, dim=0)
    mask = cusum_P <= tau * P[i].sum()
    M_i = torch.zeros_like(mask)
    M_i[idx] = mask
    return M_i
```

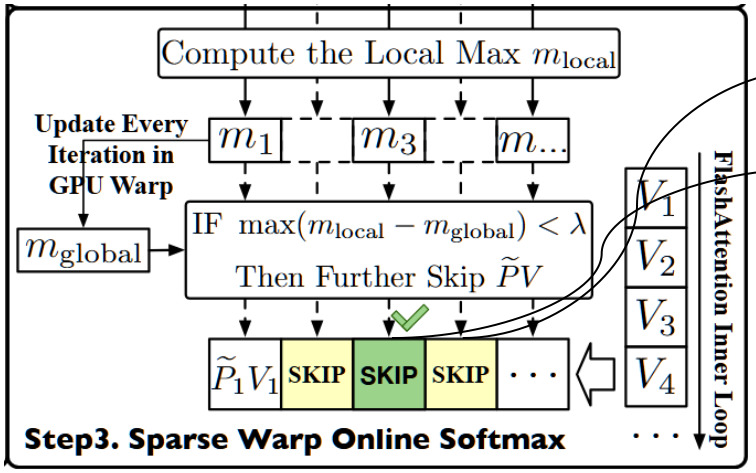
SpargeAttention[1)]

- Proposed Method
  - Sparse Warp Online Softmax
    - Key idea
      - Online softmax 과정 중 attention map에서 작은 값을 추가로 식별 가능
      - P의 값이 0에 충분히 가까우면 PV는 생략 가능

$$\widetilde{P}_{ij} = \exp(S_{ij} - m_{i,j})$$

$$O_{ij} = \mathrm{diag}\left(\exp(m_{i,j-1} - m_{ij})\right) O_{i,j-1} + \widetilde{P}_{ij} V_j$$

✓ 현재 block의 최댓값 S 가 지금까지의 최댓값 m 보다 작으면 PV 생략



Step2에서 skip 된 block

Step3에서 skip 된 block

- Warp : GPU가 데이터를 처리하는 32개 스레드의 최소 묶음 단위;
- 하드웨어 가속 극대화를 위한 최적화

SpargeAttention[1)]

- Experimental Results
  - Quantitative Results
    - End-to-end metrics across text, image, and video generation model

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**WikiText (Ppl.)** ↓	**Longbench** ↑	**InfiniteBench** ↑	**NIAH** ↑
Llama3.1 (128K)	Full-Attention	156.9	6.013	38.682	0.6594	0.907
	Minference (0.5)	140.1	10.631	28.860	0.5152	0.832
	FlexPrefill (0.5)	240.6	6.476	38.334	0.6460	0.858
	Minference (0.3)	115.7	6.705	34.074	0.6532	0.870
	FlexPrefill (0.42)	206.9	6.067	38.334	0.6581	0.878
	SpargeAttn (0.54)	**708.1**	**6.020**	**39.058**	**0.6638**	**0.909**

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**CLIPSIM** ↑	**CLIP-T** ↑	**VQA-a** ↑	**VQA-t** ↑	**FScore** ↑
CogvideoX (17K)	Full-Attention	166.0	0.1819	0.9976	80.384	75.946	5.342
	Minference (0.5)	264.6	0.1728	0.9959	70.486	62.410	2.808
	FlexPrefill (0.6)	175.3	0.1523	0.9926	1.5171	4.5034	1.652
	Minference (0.3)	196.9	0.1754	0.9964	77.326	63.525	3.742
	FlexPrefill (0.45)	142.0	0.1564	0.9917	7.7259	8.8426	2.089
	SpargeAttn (0.46)	**507.9**	**0.1798**	**0.9974**	**78.276**	**74.846**	**5.030**
Mochi (22K)	Full-Attention	164.2	0.1725	0.9990	56.472	67.663	1.681
	Minference (0.5)	202.4	0.1629	0.9891	6.668	50.839	0.653
	FlexPrefill (0.48)	191.3	0.1667	0.9898	0.582	0.0043	✗
	Minference (0.3)	147.7	0.1682	0.9889	14.541	42.956	0.833
	FlexPrefill (0.4)	171.7	0.1677	0.9909	2.941	0.7413	✗
	SpargeAttn (0.47)	**582.4**	**0.1720**	**0.9990**	**54.179**	**67.219**	**1.807**

Model (seq_len)	**Attention** (Sparsity)	**CLIPSIM** ↑	**CLIP-T** ↑	**VQA-a** ↑	**VQA-t** ↑	**FScore** ↑	**Latency** ↓
Open-Sora-Plan (38K)	Full-Attention	0.1650	0.9994	81.40	80.60	0.847	629s
	SpargeAttn (0.34)	0.1686	0.9985	77.59	76.91	0.839	**393s**

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**FID** ↓	**CLIP** ↑	**IR** ↑
Flux (4.5K)	Full-Attention	158.2	166.103	31.217	0.8701
	Minference (0.5)	151.8	180.650	30.235	0.4084
	FlexPrefill (0.48)	47.7	443.928	18.3377	-2.2657
	Minference (0.3)	118.9	170.221	31.001	0.7701
	FlexPrefill (0.41)	40.9	405.043	19.5591	-2.2362
	SpargeAttn (0.38)	**280.3**	**163.982**	**31.448**	**0.9207**
Stable-Diffusion3.5 (4.5K)	Full-Attention	164.2	166.101	32.007	0.9699
	Minference (0.5)	186.4	348.930	18.3024	-2.2678
	FlexPrefill (0.37)	23.1	350.497	18.447	-2.2774
	Minference (0.3)	150.3	337.530	18.099	-2.2647
	FlexPrefill (0.35)	22.7	348.612	18.147	-2.2756
	SpargeAttn (0.31)	**293.0**	**166.193**	**32.114**	**0.9727**

SpargeAttention[1)]

- Experimental Results
  - Quantitative Results
    - End-to-end speedup
      - Original 과 비교해서 약 1.6x 빠른 추론 속도

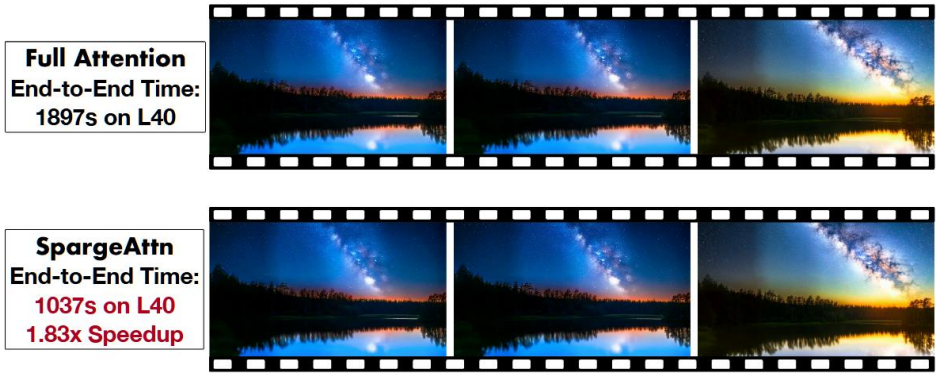
Model	GPU	Original	SageAttn	SpargeAttn
CogvideoX	RTX4090	87 s	68 s	**53 s**
Mochi	L40	1897 s	1544 s	**1037 s**
Llama3.1 (24K)	RTX4090	4.01 s	3.53 s	**2.6 s**
Llama3.1 (128K)	L40	52 s	42s	**29.98 s**

    - Overhead of sparse block prediction
      - Sequence length가 커질수록 overhead는 미미한 수준

Sequence Len	Prediction (ms)	Full Attention (ms)	Overhead
8k	**0.251**	6.649	3.78%
16k	**0.487**	26.83	1.82%
32k	**0.972**	106.68	0.911%
64k	**2.599**	424.24	0.612%
128k	**8.764**	1696.2	0.516%

SpargeAttention[1)]

- Experimental Results
  - Qualitative Results
    - Mochi on L40 GPU, with no video quality loss

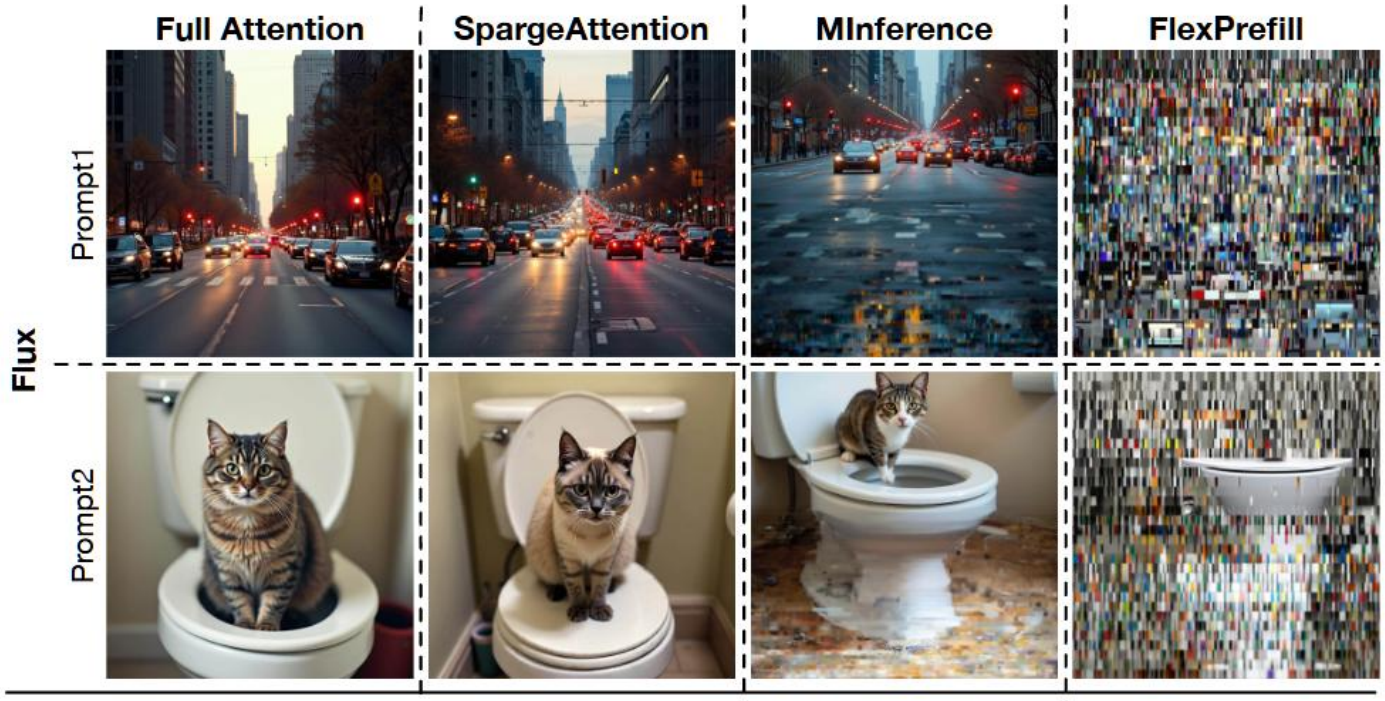


    - CogvideoX

1) Zhang, Jintao, et al. "SpargeAttention: Accurate and Training-free Sparse Attention Accelerating Any Model Inference." *Forty-second International Conference on Machine Learning*.
2) Jiang, Huiqiang, et al. "Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention." *Advances in Neural Information Processing Systems* 37 (2024): 52481-52515.
3) Lai, Xunhao, et al. "Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference." *arXiv preprint arXiv:2502.20766* (2025).

SpargeAttention[1)]

- Experimental Results
  - Qualitative Results
    - Comparison examples on Flux and Stable-Diffusion3.5
      - Minference[2)]: 특정 패턴을 미리 정의하고, 그 패턴이 나타나는 부부만 골라서 계산
      - FlexPrefil[3)]: 중요도가 낮은 토큰들을 연산에서 제외하여 속도 향상



Thank you