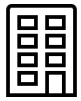
2026 동계 세미나

Sparse Attention for Video Diffusion



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

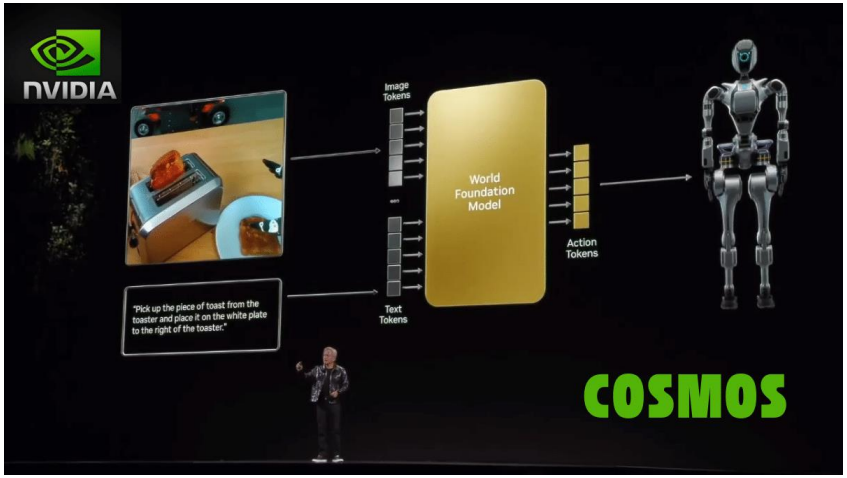
Presented by
유현우

Outline

- Background
  - 다양한 diffusion 모델 구조
  - Flash attention
  - Sparse VideoGen: Accelerating Video Diffusion Transformers with Spatial-Temporal Sparsity (ICML 2025)
- Radial Attention: O(nlog n) Sparse Attention with Energy Decay for Long Video Generation (NeurIPS 2025)
- Sparse VideoGen2: Accelerate Video Generation with Sparse Attention via Semantic-Aware Permutation (NeurIPS 2025)
- Conclusion

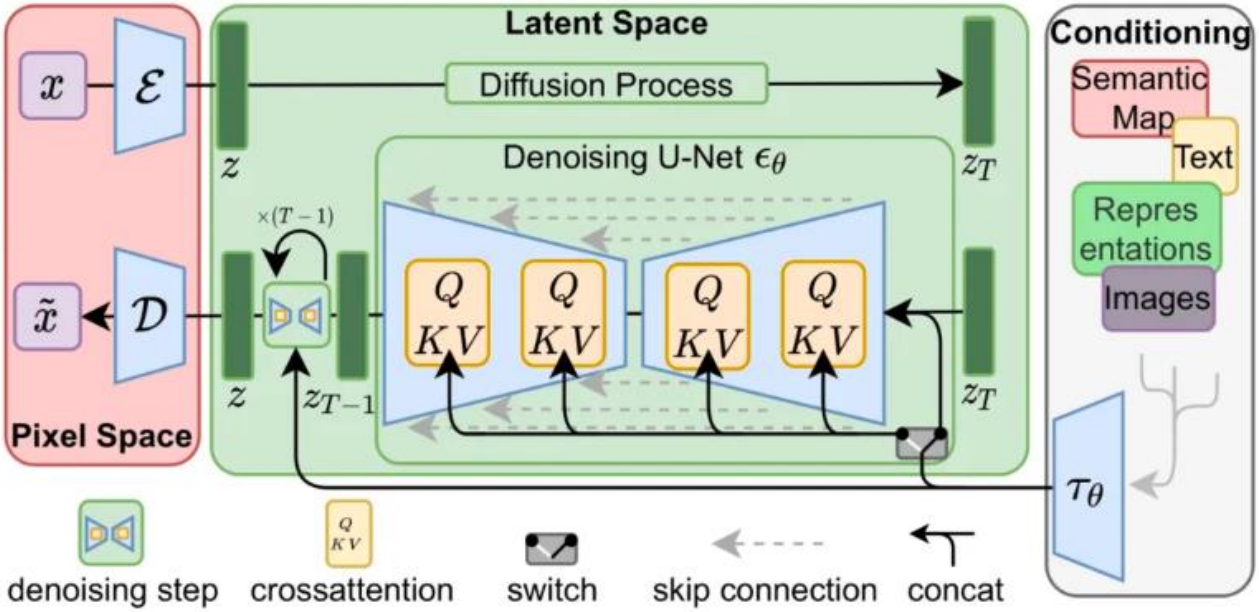
경량화에서 Video diffusion

- 최근 video diffusion을 대상으로 경량 기법을 적용한 많은 논문이 나오고 있음
  - Video를 타겟으로 하기 때문에 높은 연산량 발생
  - 최근 많은 주목을 받고 있는 world model 및 vision-language action(VLA) task가 video diffusion이 사용될 수 있음
    - World model : 현재 관찰 및 행동 값을 기반으로 미래 frame을 예측
    - VLA : 미래 장면을 예측하는 능력을 활용해서 action 학습



Diffusion mechanism

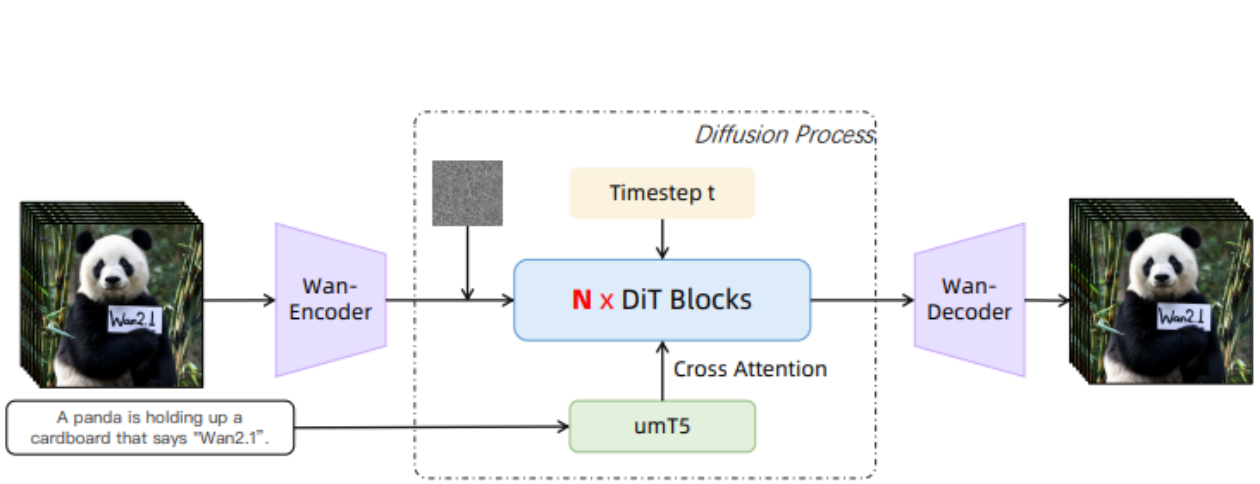
- Image를 encoder에 입력하여 latent vector를 추출
- Gaussian noise를 추가한 noised latent 를 모델에 입력
- Cross-attention 에 condition information을 주입시켜주는 방식 사용
- 추론 시에는 noised latent를 입력하고 diffusion step을 반복하는 구조



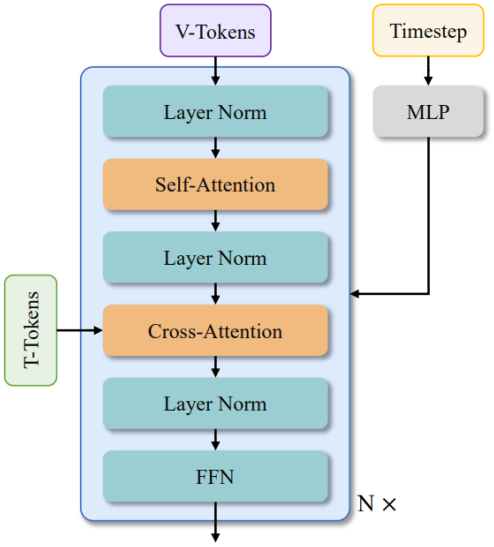
Stable diffusion architecture

Wan

- 전반적인 구조는 encoder-decoder 와 DiT block 으로 구성되어 있으며, text embedding 을 위한 모델이 있음
  - Attention 관련 경량화 접근은 주로 DiT에서 사용되는 attention을 타겟으로 함
- Wan block은 self-attention / cross-attention / FFN 로 구성
  - 이 구조는 즉 spatial 및 temporal 축에 존재하는 모든 token들을 sequence로 쭉 flatten 해서 attention을 수행하는 구조
    - Cross-attention의 key, value 가 text 로 사용



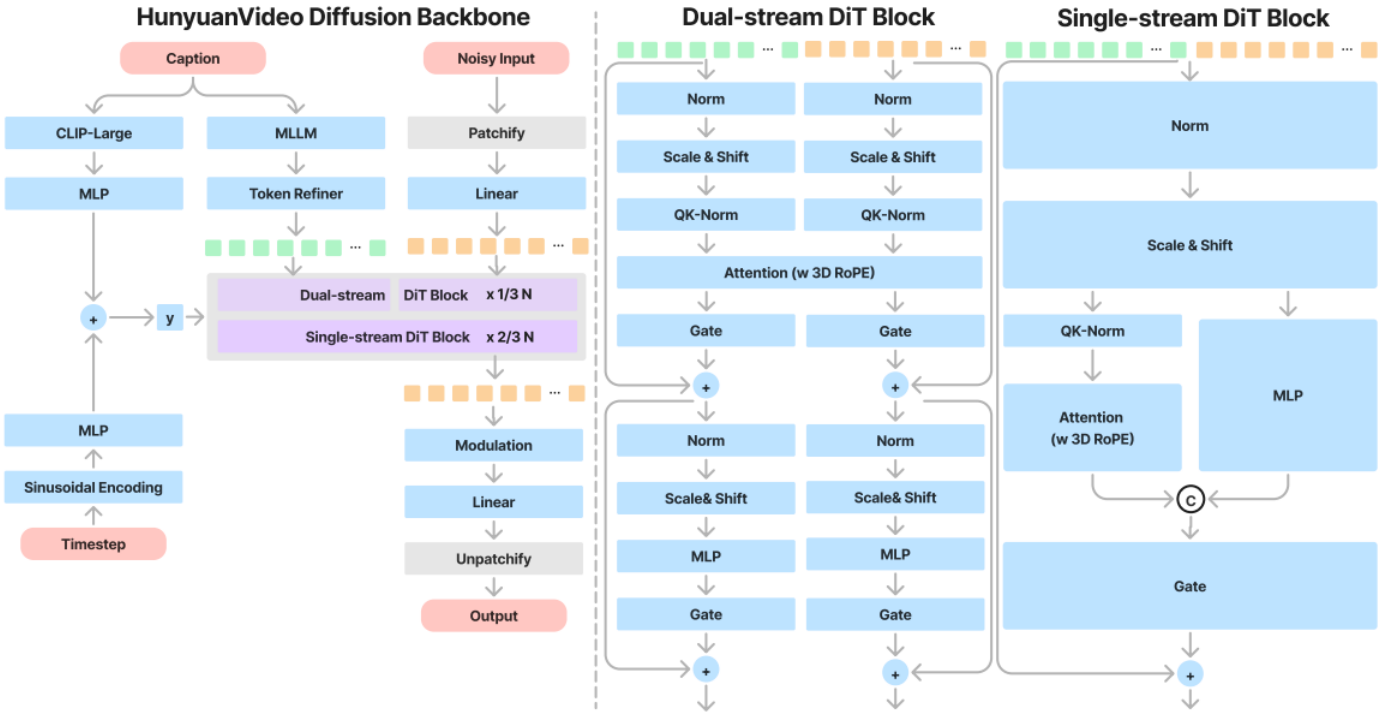
Wan architecture



Wan block

Hunyuan

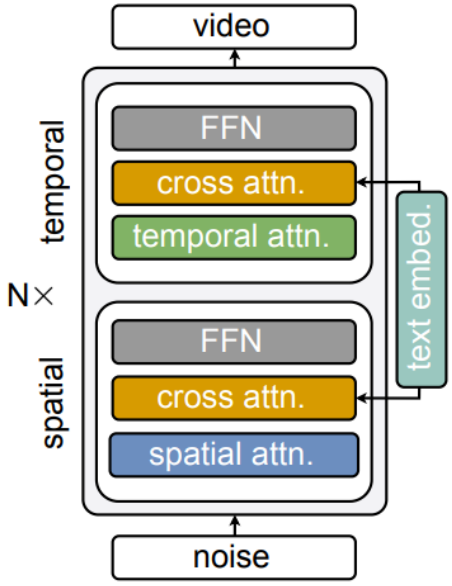
- Hunyuan은 cross-attention 없이 self-attention / FF 구조
  - Dual-stream에서 prompt 와 noise를 각각 embedding 한 후 concate해서 single stream에 입력하는 구조
  - 이를 통해 cross-attention 없이 prompt 의 정보를 conditioning 할 수 있음



Hunyuan architecture 및 block

Open-Sora

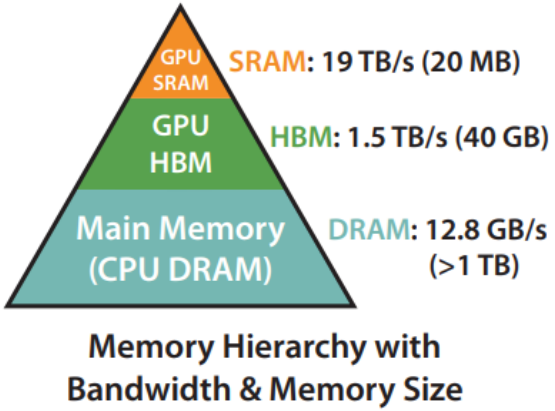
- Open-Sora는 Wan 및 Hunyuan과 다르게 spatial attention / temporal-attention /cross-attention / FF 구조
  - Attention을 수행할 때 spatial 축의 token간의 attention 을 먼저 수행해 준 뒤 temporal 축의 token간의 attention을 수행해주는 구조
  - 이를 통해 attention에 사용되는 token 개수를 줄임으로써 memory 측면에서 가벼운 모델



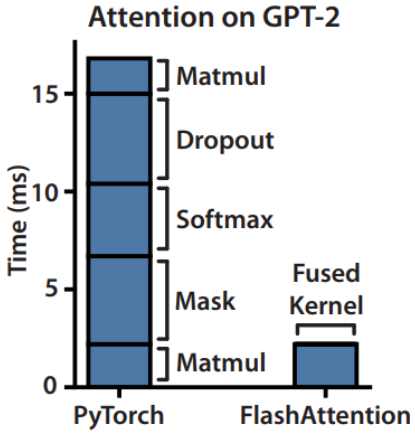
Open-Sora architecture

Flash Attention

- Fast and Memory-Efficient Exact Attention with IO-Awareness
  - Trade-off 없는 경량화
- 현재 GPU는 상대적으로 SRAM 연산은 빠르고, 오히려 read / write (IO 통신) 하는 과정에서 병목이 발생한다는 점을 지적
- HBM은 tensor를 저장하는 계층이고, SRAM은 연산을 수행
  - 따라서 GPU를 활용한 연산은 tensor를 HBM에서 SRAM으로 read하고 연산을 수행한 후, 다시 HBM으로 write하는 과정
- Attention 연산 $\text{Softmax}(QK^T)V$ 에서 matmul 이 차지하는 overhead가 적음

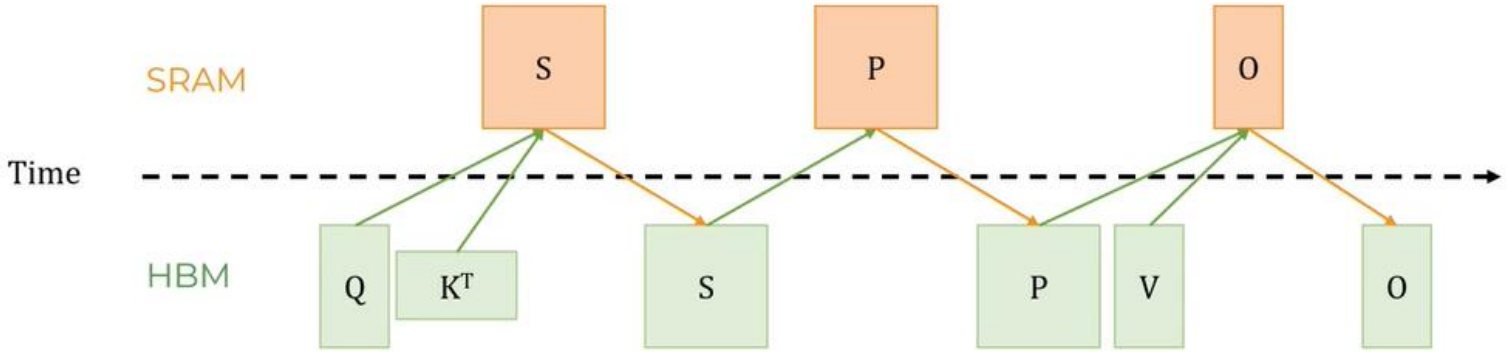


Memory Hierarchy with Bandwidth & Memory Size



Flash Attention

- 기존의 attention 은 Softmax(QK^T)V 에서 N^2 스케일의 matrix에 대해서 SRAM-HBM을 IO 하는 구조가 비효율적
- Matrix S,P를 HBM에 올리므로 메모리 측면에서 비효율적



Algorithm 0 Standard Attention Implementation

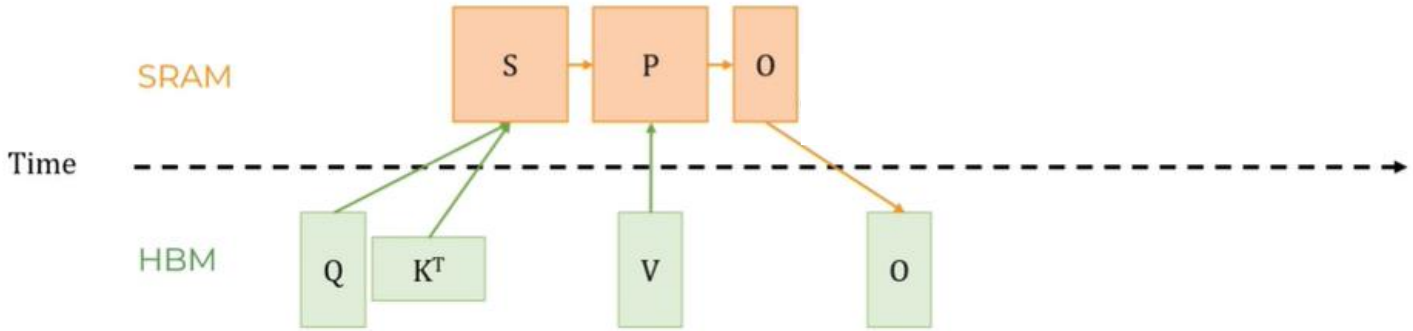
Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.
4: Return $\mathbf{O}$.

기존 attention 과정

Flash Attention

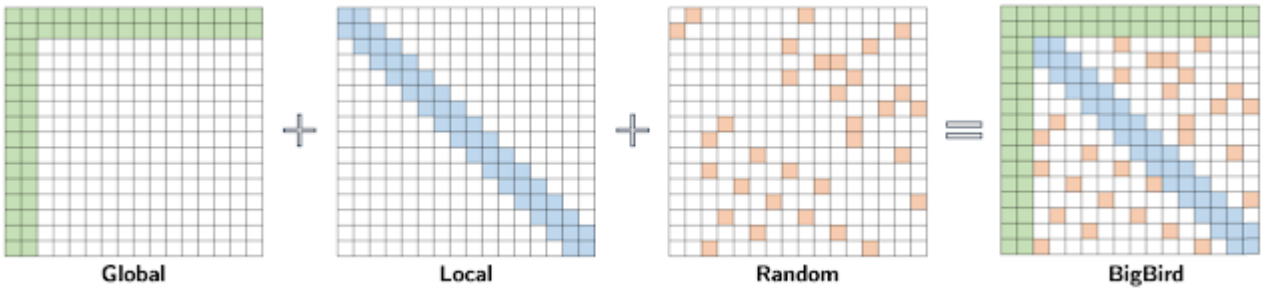
- Flash attention은 이러한 Softmax(QK^T)V 과정을 SRAM에서 한번에 연산하여 overhead를 줄이는 접근
- 따라서 attention map에 대한 정보에 접근할 수 없음



Flash attention의 개념도

Sparse Attention 이란?

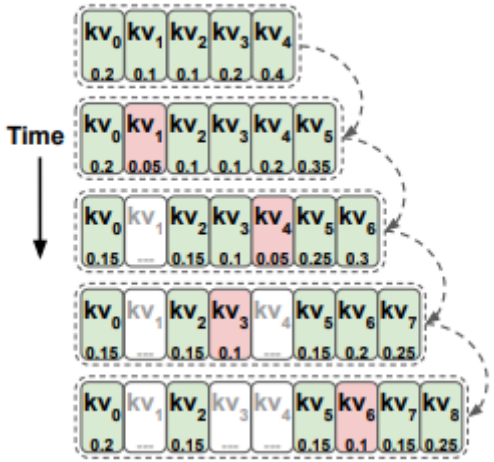
- Sparse attention 은 attention에 masking을 취해서 연산 이득을 보게 함
  - Score = QK^T
  - Score += mask
  - Attn = softmax(scores)
  - Out = Attn V
- 이 과정에서 masking 된 영역은 key, value에 적용돼서 연산에 호출이 안되는 방식



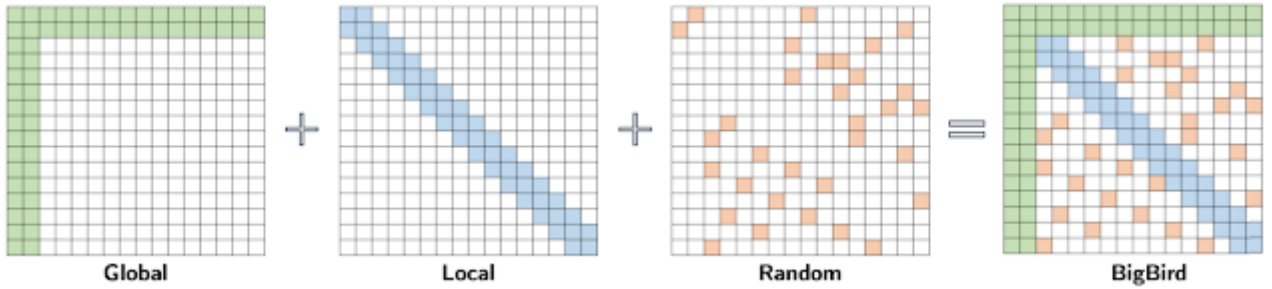
Sparse attention 방법

Key, value token pruning 과 다른 점?

- Key, value token pruning은 query와 관련 없이 key, value token을 일괄적으로 줄임
- Key, value token pruning은 각 query token에 adaptive하게 줄이는 방식이 아님
  - 첫 번째 query token 에서 안필요하던 KV token이 두 번째 query token에서는 필요할 수 있음
- Sparse attention 은 각 query token에 필요한 key, value token을 선별해서 사용하는 방식



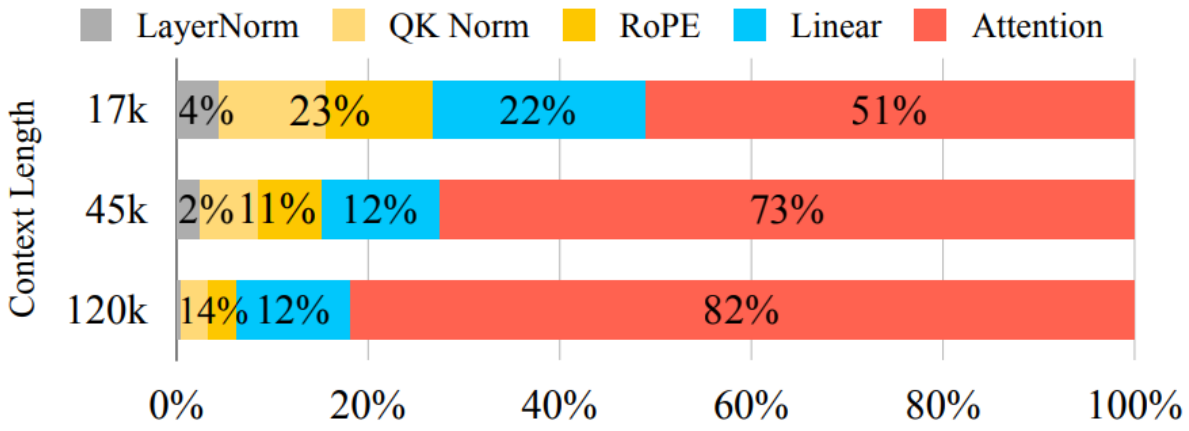
KV token pruning



Sparse attention 방법

Sparse Video Gen(SVG) v1

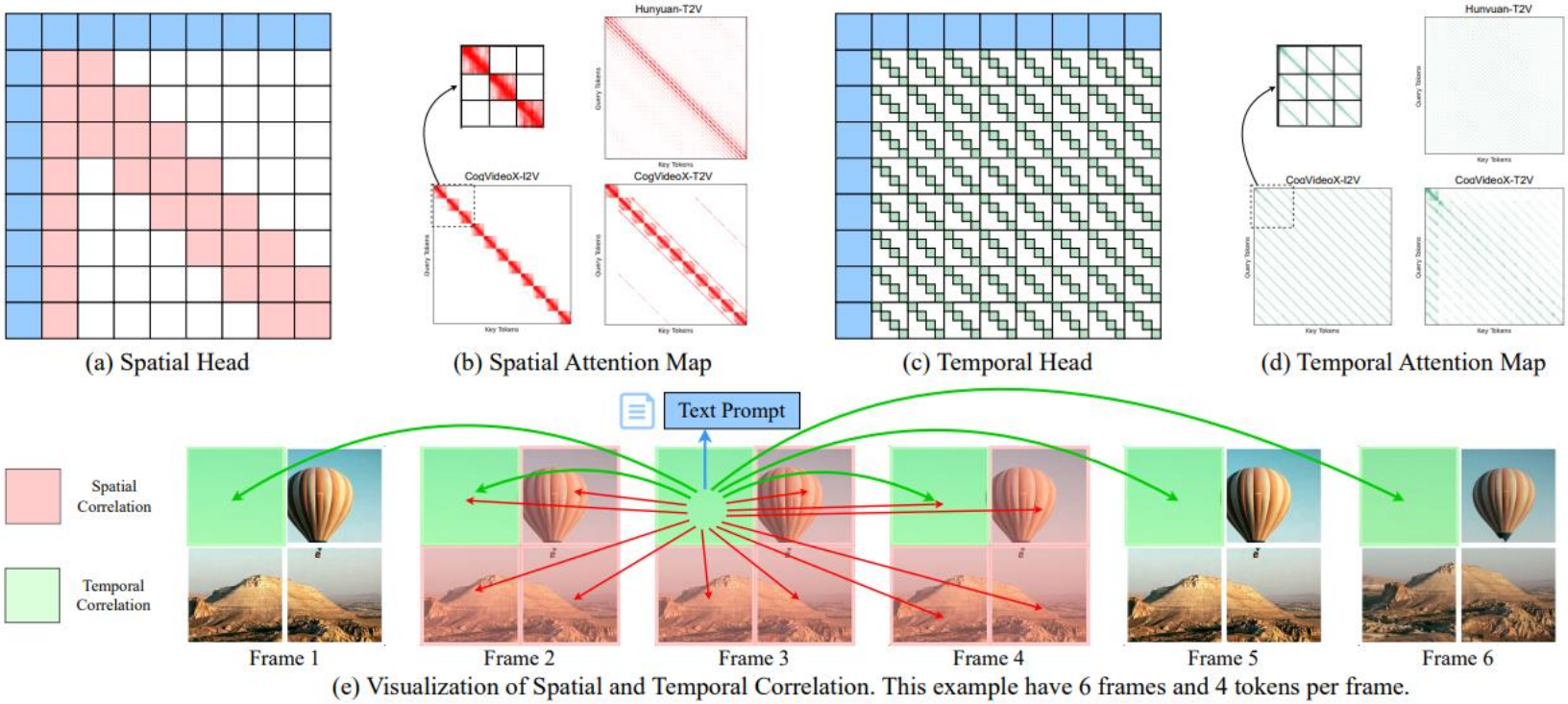
- Video generation에서는 특히 많은 token 수로 인해 quadratic complexity를 갖는 attention 연산의 overhead가 특히 크게 발생
  - Attention map의 분포를 확인해보면 소수의 token이 큰 값을 차지하고 있음
  - 따라서 sparse attention을 통한 training-free attention 경량화 방법을 제안하는데 목적



Context length에 따른 attention 의 overhead

Sparse Video Gen(SVG) v1

- Attention map의 패턴을 spatial head와 temporal head로 분류
  - Spatial head는 같은 frame (or 인접한 frame) 내부에 있는 token에 집중
  - Temporal head는 모든 temporal 축의 동일한 spatial 위치의 token에 집중



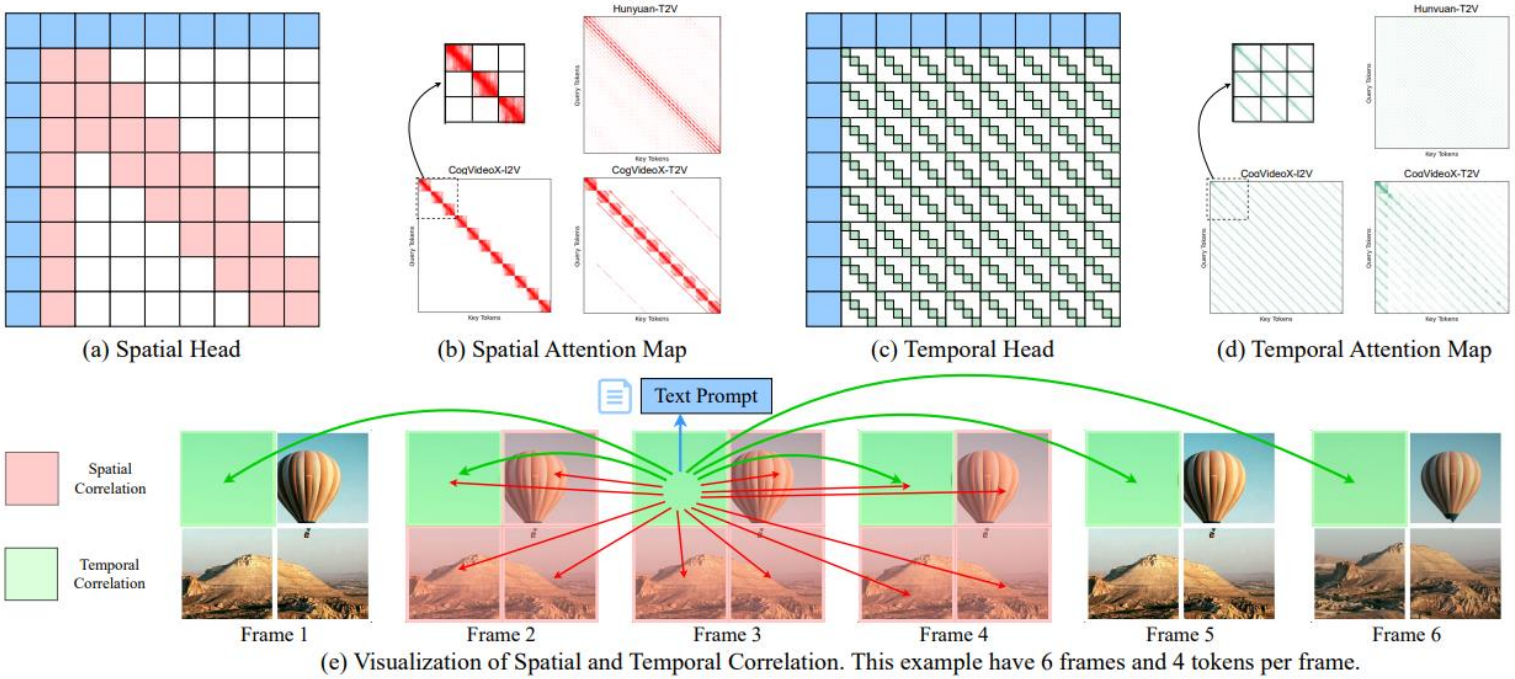
(a) Spatial Head (b) Spatial Attention Map (c) Temporal Head (d) Temporal Attention Map

(e) Visualization of Spatial and Temporal Correlation. This example have 6 frames and 4 tokens per frame.

Attention map 패턴 분석

Sparse Video Gen(SVG) v1

- SVG을 적용할 때 고려해야 할 점
  - Sparse pattern이 입력 prompt 에 따라 동적으로 변함
    - 따라서 Inference 중에 sparse pattern을 식별해서 적용해야 함
  - Temporal head sparse pattern은 연산될 token들이 널리 퍼져있어 GPU 친화적이지 않음



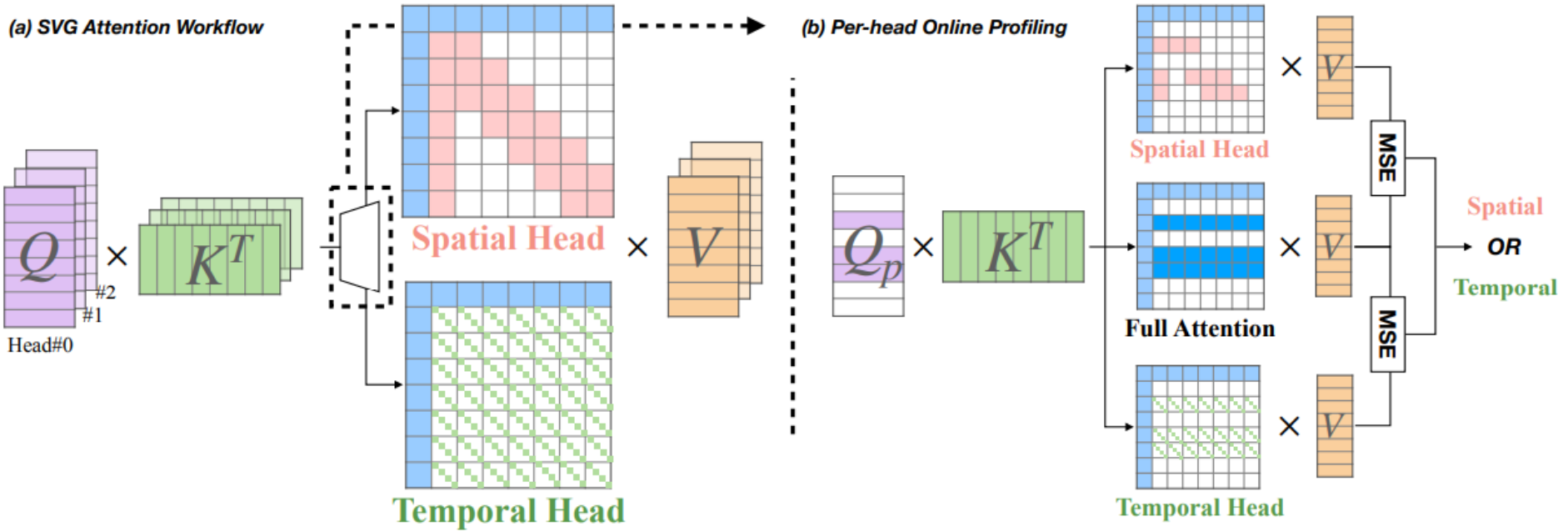
(a) Spatial Head (b) Spatial Attention Map (c) Temporal Head (d) Temporal Attention Map

(e) Visualization of Spatial and Temporal Correlation. This example have 6 frames and 4 tokens per frame.

Attention map 패턴 분석

Sparse Video Gen(SVG) v1

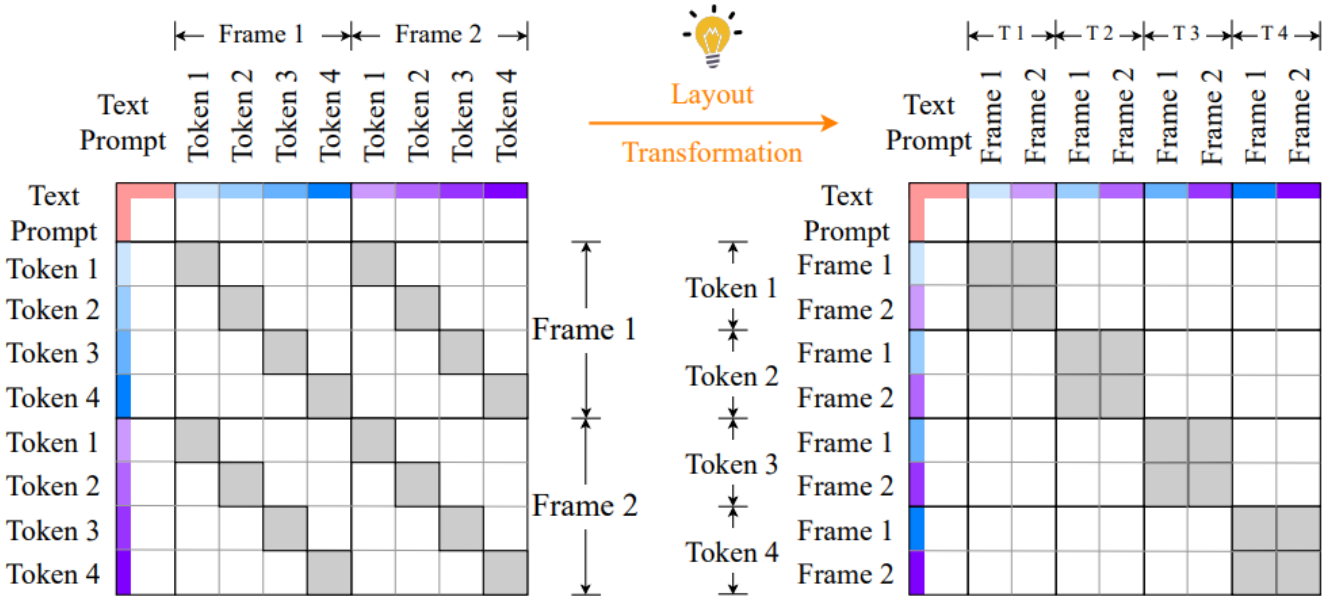
- SVG는 token의 random sampling 을 통해 spatial head와 temporal head를 고르는 전략을 제안
  - 1%의 query token을 random sampling 해서 full / spatial / temporal attention을 수행시킨 뒤 full attention과의 MSE로 sparse pattern 결정



Proposed SVG

Sparse Video Gen(SVG) v1

- GPU 연산을 위해서는 특정 단위(block)의 tensor를 불러와서 연산함
  - 따라서 token이 개별 단위로 있으면 그 token을 포함한 필요없는 token까지 불러와서 연산되기 때문에 비효율적으로 동작
- 다른 frame의 동일 위치 token 끼리 가깝게 위치하도록 변환



Layout transformation for temporal head

Sparse Video Gen(SVG) v1

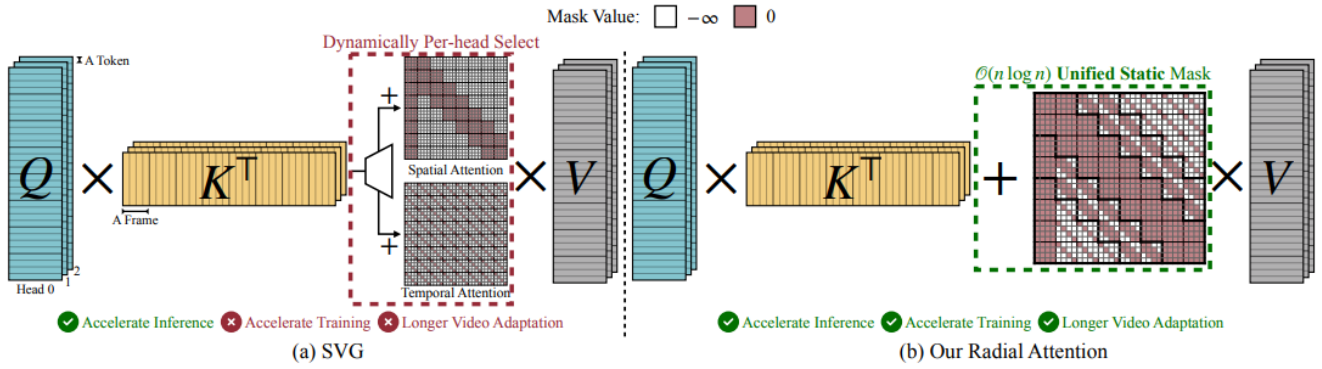
- Experimental results
  - Base 모델과 비교하여 PSNR, SSIM, LPIPS 등의 metric과 vbench의 imagequal 및 subconsist 에서 성능이 잘 유지되는 결과를 보임
  - 그와 동시에 최대 2배 이상의 latency 개선을 이룸

Table 1. Quality and efficiency benchmarking results of SVG and other baselines.

Type	Method	Quality					Efficiency		
		PSNR ↑	SSIM ↑	LPIPS ↓	ImageQual ↑	SubConsist ↑	FLOPS ↓	Latency ↓	Speedup ↑
I2V	CogVideoX-v1.5 (720p, 10s, 80 frames)	-	-	-	70.09%	95.37%	147.87 PFLOPs	528s	1x
	DiTFastAttn (Spatial-only)	24.591	0.836	0.167	70.44%	95.29%	78.86 PFLOPs	338s	1.56x
	Temporal-only	23.839	0.844	0.157	70.37%	95.13%	70.27 PFLOPs	327s	1.61x
	MInference	22.489	0.743	0.264	58.85%	87.38%	84.89 PFLOPs	357s	1.48x
	PAB	23.234	0.842	0.145	69.18%	95.42%	105.88 PFLOPs	374s	1.41x
	Ours	**28.165**	**0.915**	**0.104**	70.41%	95.29%	74.57 PFLOPs	237s	**2.23x**
T2V	CogVideoX-v1.5 (720p, 10s, 80 frames)	-	-	-	62.42%	98.66%	147.87 PFLOPs	528s	1x
	DiTFastAttn (Spatial-only)	23.202	0.741	0.256	62.22%	96.95%	78.86 PFLOPs	338s	1.56x
	Temporal-only	23.804	0.811	0.198	62.12%	98.53%	70.27 PFLOPs	327s	1.61x
	MInference	22.451	0.691	0.304	54.87%	91.52%	84.89 PFLOPs	357s	1.48x
	PAB	22.486	0.740	0.234	57.32%	98.76%	105.88 PFLOPs	374s	1.41x
	Ours	**29.989**	**0.910**	**0.112**	63.01%	98.67%	74.57 PFLOPs	232s	**2.28x**
T2V	HunyuanVideo (720p, 5.33s, 128 frames)	-	-	-	66.11%	93.69%	612.37 PFLOPs	2253s	1x
	DiTFastAttn (Spatial-only)	21.416	0.646	0.331	67.33%	90.10%	260.48 PFLOPs	1238s	1.82x
	Temporal-only	25.851	0.857	0.175	62.12%	98.53%	259.10 PFLOPs	1231s	1.83x
	MInference	23.157	0.823	0.163	63.96%	91.12%	293.87 PFLOPs	1417s	1.59x
	Ours	**29.546**	**0.907**	**0.127**	65.90%	93.51%	259.79 PFLOPs	1171s	1.92x
	Ours + FP8	**29.452**	**0.906**	**0.128**	65.70%	93.51%	259.79 PFLOPs	968s	**2.33x**

Radial attention

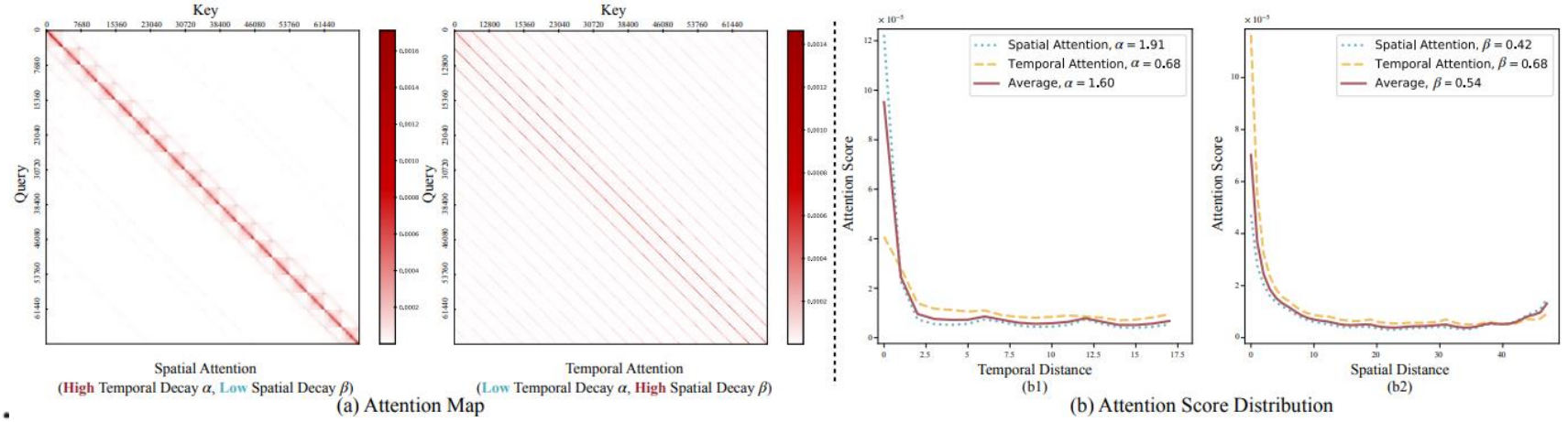
- 기존 방법의 한계
  - 학습시에 SVG와 같이 spatial 및 temporal head로 구분해버리면, 최적화 과정에서 잘못 분류 되었을 경우 악영향을 미칠 수 있음
  - SVG는 일부 token을 random sampling해서 head를 판단하는데, 이로인해 long video로 갈수록 예측 정확도가 떨어질 가능성이 큼
    - 또한 long video로 갈수록 attention map의 분포가 다양해짐
- 제안하는 방법
  - Radial attention 은 spatial 및 temporal를 통합한 static mask 형태를 제안
    - 공간적 또는 시간적 거리가 멀수록 attention score 가 감소하는 결과 기반으로 디자인
    - Long video로 갈수록 이 mask 형태가 효과적임



SVG와 제안하는 radial attention

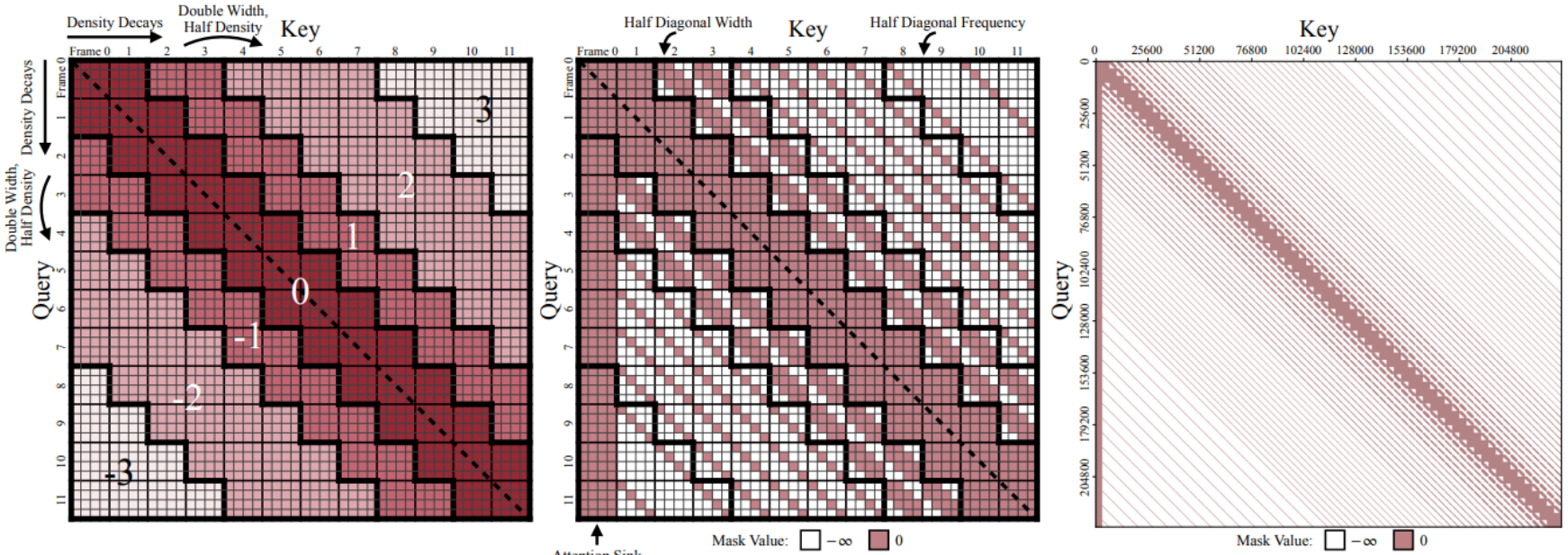
Radial attention

- SVG에서 분류한 spatial attention 과 temporal attention 을 visualization
- 이 두 attention map과 여러 head를 평균낸 average에 대해서 temporal 및 spatial distance에 따른 attention score를 분석
  - 각 query token에 대해서 평균
- Query와 key token 사이가 멀어질수록 attention score가 exponential 하게 감소하는 결과
  - 이 논문에서는 spatiotemporal energy decay라고 부름
- 따라서 이 분석을 기반으로 spatial 및 temporal distance가 멀어질수록 sparsity를 증가시키는 구조 개발



Radial attention

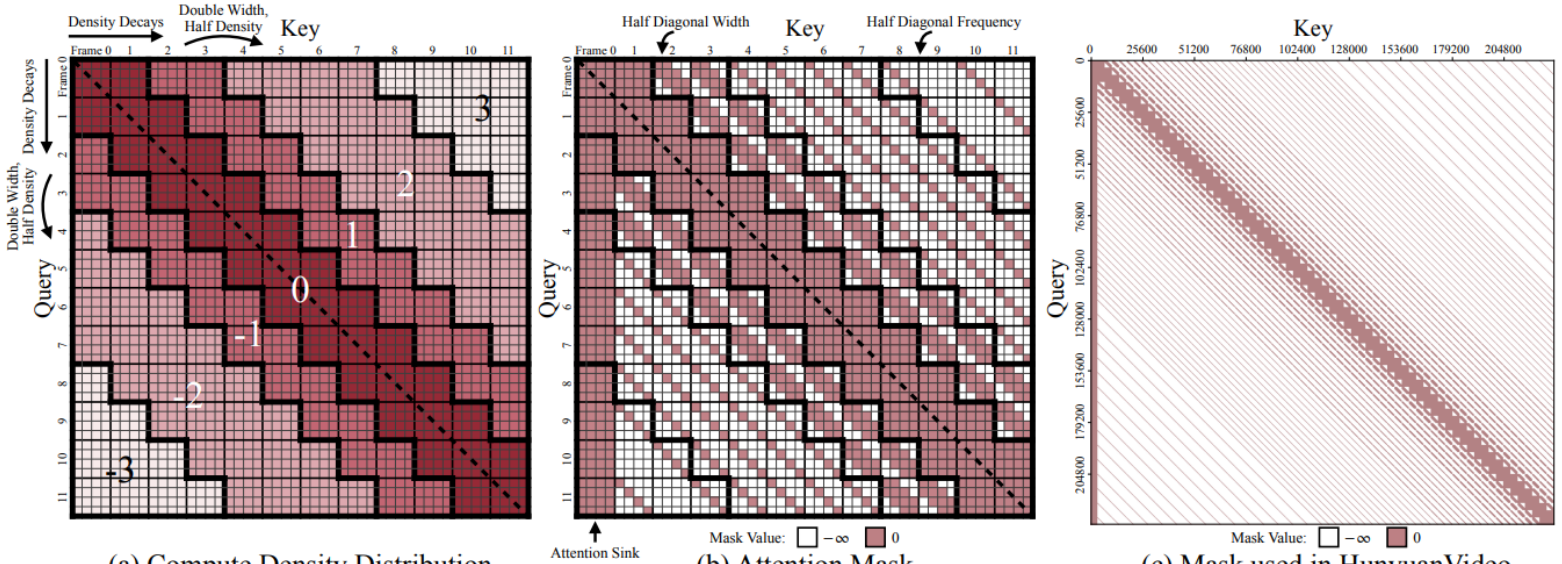
- Temporal density : $(1/2)^{\lfloor \log_2(\max(|i-j|,1)) \rfloor}$
  - Density는 i번째 frame이 j번째 frame을 볼 때, 몇 %를 계산할지에 대한 값
  - $d = |i-j| \quad r = \lfloor \log_2(\max(d,1)) \rfloor$ 와 같이 둔다면 frame 거리가 2배 커질수록 r이 1 증가
  - 이 r 값이 증가할수록 sparsity 가 2배씩 증가



(a) Compute Density Distribution (b) Attention Mask (c) Mask used in HunyuanVideo

Radial attention

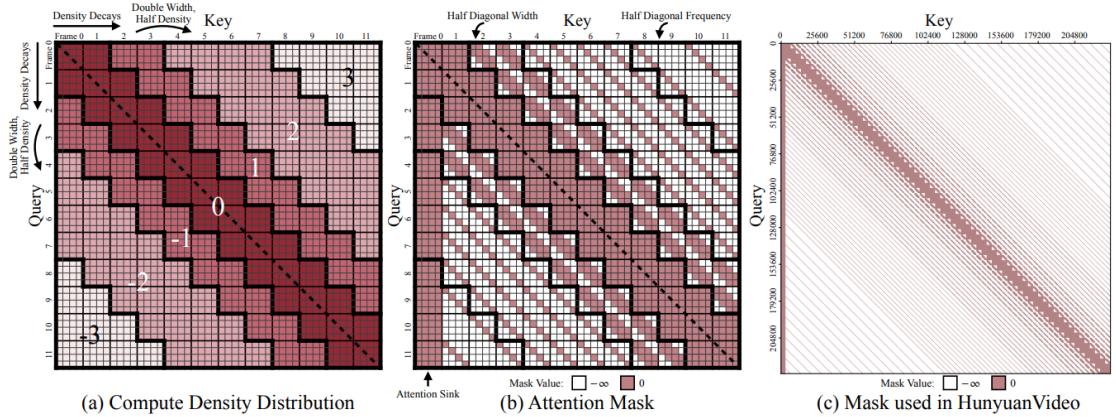
- Spatial density $\left\lfloor \frac{s}{2^{\lfloor \log_2 \max(|i-j|,1) \rfloor}} \right\rfloor$
  - Diagonal width
    - S 는 frame 내 token 개수
  - 만약 diagonal width가 1보다 작아지면 다음의 조건을 만족 $|i-j| \bmod \left\lceil \frac{2^{\lfloor \log_2 \max(|i-j|,1) \rfloor}}{s} \right\rceil = 0$
  - $|i-j| \bmod m = 0$ 은 $|i-j|$가 0, m, 2m, 3m, … 일 때만 block을 남기고, 그 외의 block은 계산 안함



(a) Compute Density Distribution (b) Attention Mask (c) Mask used in HunyuanVideo

Radial attention

- Complexity
  - Band r에 속하는 block 개수 $N_{\text{blocks}}(r) \approx f \cdot 2^r$
  - 그 block 내부에서 연산되는 token 수 (f :frame, s : frame 내부 token 수)
    - Diagonal width $w(r) \approx \frac{s}{2^r}$
    - 따라서 block 하나에서 연산량 $N_{\text{entries/block}}(r) \approx s \cdot w(r) \approx s \cdot \frac{s}{2^r} = \frac{s^2}{2^r}$
  - 특정 band 에서 연산되는 연산량 $(f \cdot 2^r) \times \left(\frac{s^2}{2^r}\right) = fs^2$
  - Band 의 개수 $\log f$
  - Total $\text{total} \approx (fs^2) \cdot \log_2 f$ $fs^2 \log f = (fs) \cdot s \log f = n \cdot s \log f$
  - Long video는 해상도 고정에 frame이 늘어난다고 하면 $O(n \log f) = O(n \log n)$



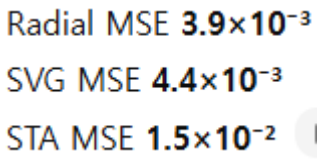
(a) Compute Density Distribution (b) Attention Mask (c) Mask used in HunyuanVideo

Radial attention

- 실험 결과
  - Radial attention은 기존의 training-free sparse attention 방법들과 비교하여 우수한 성능
  - SVG와 비교했을 때는 비슷한 성능을 보임
  - 하지만 attention output과의 MSE를 계산했을 때 radial attention이 더 작은 MSE를 보임

Model	Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Vision Reward (↑)	PFLOPs	Latency (s)	Speedup
Hunyuan Video (117 frames)	Original	–	–	–	0.141	612	1649	–
	STA (FA3)	26.7	0.866	0.167	0.132	331	719	2.29×
	PA	22.1	0.764	0.256	0.140	339	1002	1.65×
	SVG	27.2	**0.895**	**0.114**	**0.144**	340	867	1.90×
	Ours	**27.3**	0.886	**0.114**	0.139	339	876	1.88×
Wan2.1-14B (69 frames)	Original	–	–	–	**0.136**	560	1630	–
	STA (FA3)	22.9	0.830	0.171	0.132	322	812	2.01×
	PA	22.4	0.790	0.176	0.126	324	978	1.67×
	SVG	23.2	0.825	0.202	0.114	324	949	1.71×
	Ours	**23.9**	**0.842**	**0.163**	0.128	323	917	1.77×

Training-free inference에서 성능 비교



Attention output에 대한 MSE

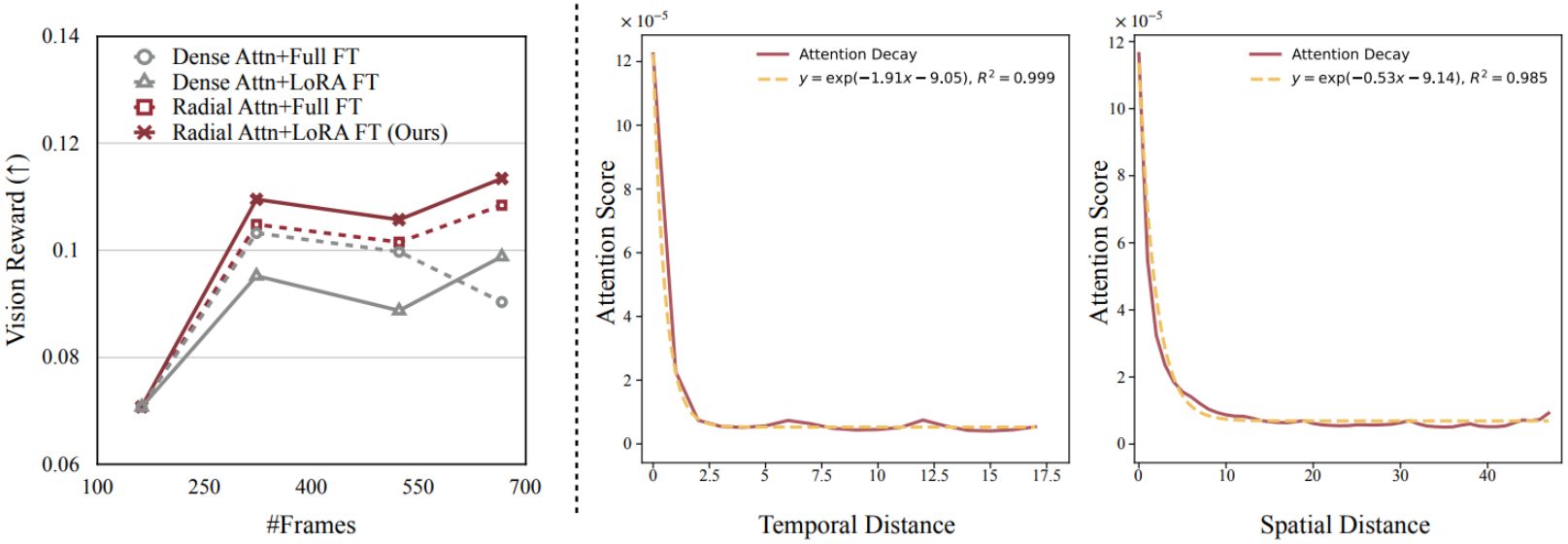
Radial attention

- Long video 실험 결과

Model	#Frames	Method	Sparsity	Training Time (h)	Training Speedup	Inference Time (s)	Inference Speedup	Vision Reward (↑)	VBench		
									S.C.	A.Q.	I.Q.
Hunyuan Video	125 (1×)	Original	0.00%	–	–	225	–	0.119	0.959	0.643	0.672
	253 (2×)	Original	0.00%	–	–	797	1.00×	0.122	0.953	0.603	0.611
		RIFLEx	0.00%	–	–	797	1.00×	**0.128**	0.969	0.622	0.614
		Spatial	80.5%	16.0	2.81×	335	2.38×	0.054	**0.979**	0.607	0.670
		Temporal	80.7%	16.2	2.78×	338	2.36×	0.104	0.963	0.620	0.658
		Long LoRA	80.6%	16.6	2.71×	363	2.20×	0.112	0.958	0.620	**0.685**
		PA [45]	80.4%	16.7	2.69×	334	2.39×	0.109	0.967	0.608	0.653
		SANA	–	12.8	3.52×	285	2.80×	-0.205	0.907	0.300	0.442
		Full	0.00%	45.0	1.00×	797	1.00×	0.124	0.955	0.616	0.648
		Ours	80.8%	16.2	**2.78**×	339	**2.35**×	0.126	0.968	**0.623**	0.663
	509 (4×)	Original	0.00%	–	–	2895	1.00×	0.054	0.988	0.545	0.451
		RIFLEx	0.00%	–	–	2895	1.00×	0.037	**0.989**	0.539	0.456
		Spatial	88.3%	20.7	4.52×	755	3.83×	0.112	0.922	0.598	0.664
		Temporal	88.2%	21.1	4.44×	774	3.74×	0.083	0.972	0.597	0.646
		Long LoRA	88.4%	20.9	4.48×	803	3.61×	0.130	0.936	0.618	**0.689**
		PA [45]	88.2%	21.8	4.29×	766	3.78×	0.128	0.950	0.590	0.648
		Full	0.00%	93.6	1.00×	2895	1.00×	0.133	0.977	0.590	0.635
		Ours	88.3%	21.4	**4.37**×	781	**3.71**×	**0.134**	0.973	**0.623**	0.672
Mochi 1	163 (1×)	Original	0.00%	–	–	112	–	0.071	0.973	0.623	0.672
	331 (2×)	Original	0.00%	–	–	302	1.00×	0.040	0.937	0.551	0.466
		Spatial	76.1%	8.57	1.75×	186	1.62×	0.088	0.935	0.596	0.595
		Temporal	76.3%	8.54	1.76×	189	1.60×	0.075	0.936	0.591	0.593
		Long LoRA	76.0%	9.07	1.65×	210	1.44×	0.095	0.950	0.596	**0.630**
		PA [45]	77.8%	8.53	1.76×	183	1.65×	0.101	0.946	0.610	0.626
		SANA	–	8.22	1.82×	166	1.82×	-0.201	0.905	0.334	0.568
		Full	0.00%	15.0	1.00×	302	1.00×	0.095	0.923	0.610	0.594
		Ours	76.4%	8.43	**1.78**×	185	**1.63**×	**0.110**	**0.951**	**0.615**	0.602
	667 (4×)	Original	0.00%	–	–	992	1.00×	-0.091	0.916	0.383	0.322
		Spatial	85.2%	17.4	2.83×	382	2.60×	0.091	0.930	0.611	0.585
		Temporal	85.4%	17.6	2.80×	393	2.52×	0.028	0.931	0.556	0.536
		Long LoRA	86.0%	19.0	2.59×	426	2.33×	0.086	0.944	0.584	0.543
		PA [45]	86.5%	17.3	2.84×	381	2.60×	0.107	0.956	**0.633**	**0.650**
		Full	0.00%	49.2	1.00×	992	1.00×	0.099	0.934	0.613	0.613
		Ours	85.5%	17.4	**2.83**×	386	**2.57**×	**0.113**	**0.958**	0.618	0.638
Wan2.1 -14B	81 (1×)	Original	0.00%	–	–	1630	–	0.135	0.973	0.623	0.672
	161 (2×)	Original	0.00%	–	–	5735	1.00×	0.109	0.946	0.598	0.614
		Full	0.00%	28.0	1.00×	5735	1.00×	**0.150**	0.966	0.590	**0.689**
		Ours	73.6%	14.5	**1.93**×	2847	**2.01**×	0.145	**0.981**	**0.607**	0.677

Radial attention

- Full fine-tuning 및 LoRA fine-tuning 결과에서 dense attention 보다 좋은 결과를 보임
- 지수함수 y = exp(-ax +b) 를 regression으로 피팅해본 실험 결과 attention score가 spatiotemporal distance에 따라 지수적으로 decay되는 가정이 맞는 결과를 확인할 수 있음

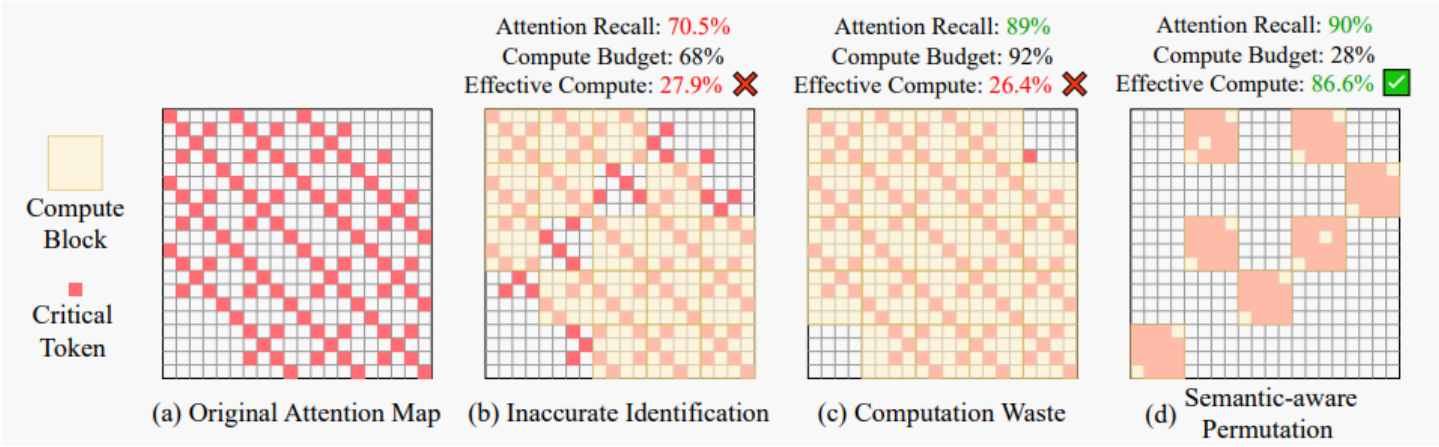


(a) LoRA Effectiveness Results

(b) Regression Analysis

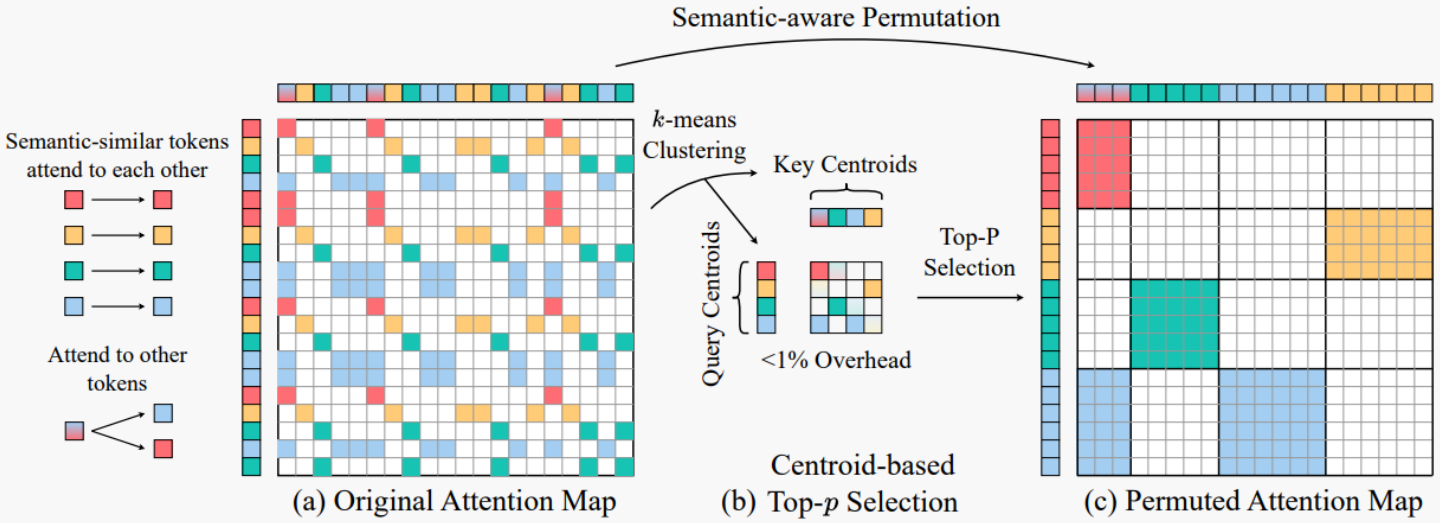
Sparse Video Gen(SVG) v2

- 기존의 한계
  - 기존 방법들은 semantic 기준이 아니라 position을 기준으로 sparse pattern을 정함
    - 기존 방법들은 연속된 token들을 block으로 cluster 하고, block 단위에서 attention score를 근사하여 sparse pattern을 정함
    - 이는 흩어져 있는 중요 token을 정밀하게 찾는데 비효율적
  - 중요 token이 흩어져 있어 block 단위 처리에 최적화된 GPU에서 불필요한 계산 발생
    - Nvidia GPU tensor core의 경우 16x16x8 크기의 minimum matrix multiplication shape 갖고 있음
    - QK 연산을 위해 16개 단위의 token이 필요하다는 뜻
    - 따라서 16개 중 일부만 중요 token이더라도 tensor core를 활용하려면 block 전체를 연산해야 함



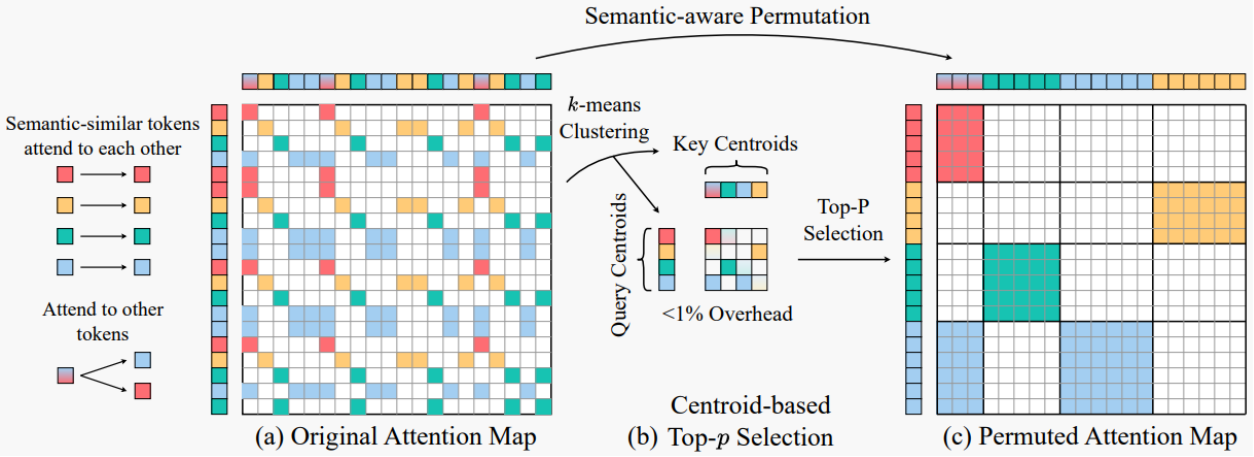
Sparse Video Gen(SVG) v2

- 제안하는 방법
  - 각 head와 layer의 Q,K,V token에 대해 k-means clustering을 수행
    - 이후 같은 cluster 의 token들을 연속적으로 배치하여 의미적으로 일관된 block을 형성
  - 각 cluster 의 centroid 를 기준으로 attention score를 근사
    - 확률 누적합이 top-P 가 될 때까지 뽑음 (dynamic allocation 전략)
    - 이를 통해 성능 보존을 어느정도 보존할 수 있음



Sparse Video Gen(SVG) v2

- 제안하는 방법
  - 각 head와 layer의 Q,K,V token에 대해 k-means clustering을 수행
    - Cluster는 iteration을 돌면서 수렴시키는 매커니즘이므로 latency가 크게 발생
      - 경우에 따라 전체 overhead 의 50%를 차지하기도 함
      - 하지만 diffusion은 diffusion step에 따라 token representation이 유사하다는 특성이 있음
      - 따라서 초반 diffusion step에서 얻은 centroid 정보를 이후 step에서 활용하는 방법 사용
    - 이후 같은 cluster 의 token들을 연속적으로 배치하여 의미적으로 일관된 block을 형성



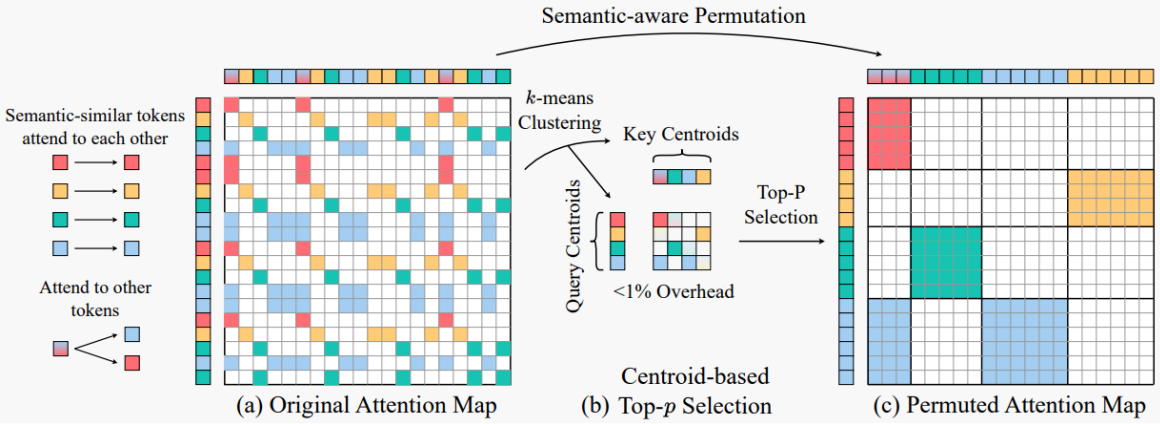
Proposed SVG2

Sparse Video Gen(SVG) v2

- 제안하는 방법
  - 각 cluster 의 centroid 를 기준으로 attention score를 근사
    - 확률 누적합이 top-P 가 될 때까지 뽑음 (dynamic allocation 전략)
    - 이를 통해 성능 보존을 어느정도 보존할 수 있음

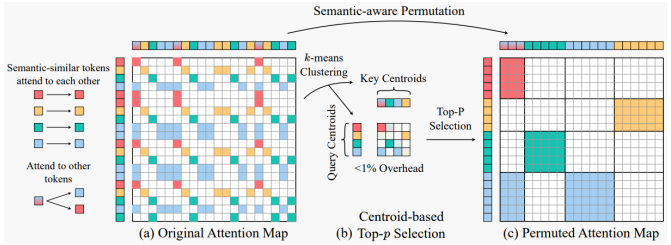
$$S_{ij} = \frac{\texttt{centroid}(Q_i) \cdot \texttt{centroid}(K_j)^T}{\sqrt{d_k}} \quad (1) \qquad P'_{ij} = \frac{|K_j| \exp(S_{ij})}{\sum_{k=1}^{C_k} |K_k| \exp(S_{ik})}$$

- Q_i : I 번째 clustered query token 집합 , K_j : j 번째 clustered query token 집합
- K cluster 크기가 크다는 뜻은 그 cluster 에서 실제 attention score 누적합이 크다는 뜻
  - 따라서 Softmax를 취할 때 cluster 크기에 대한 가중치를 취해줌

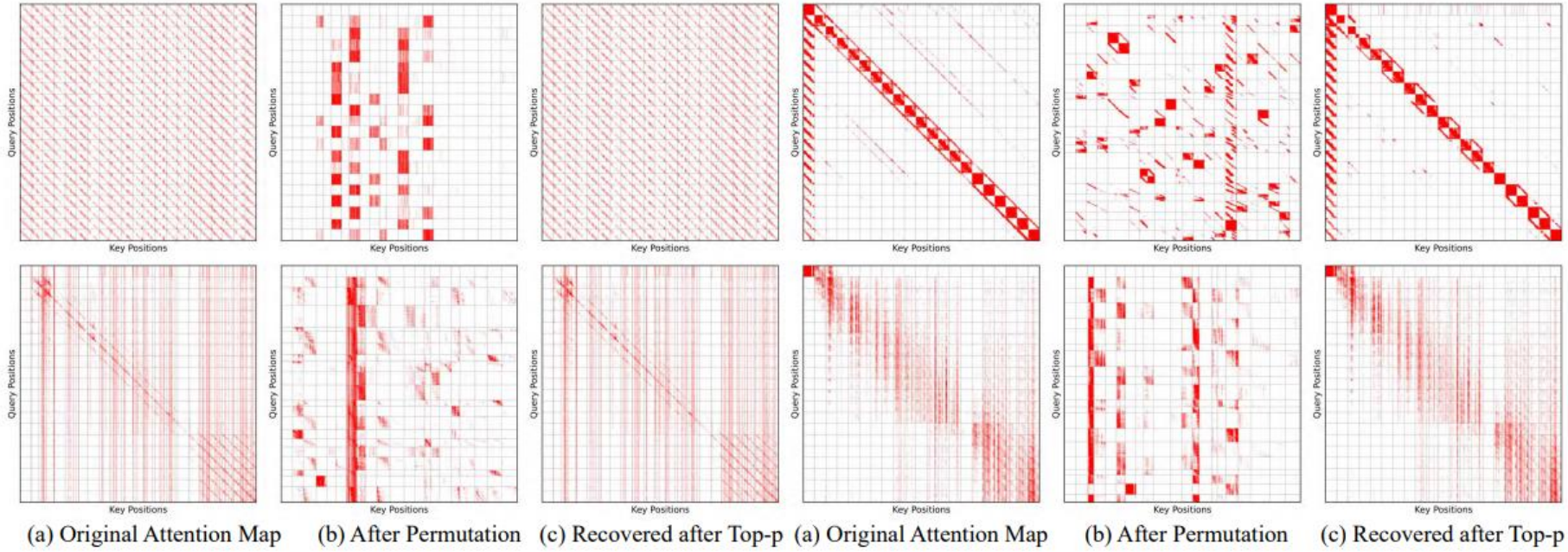


Proposed SVG2

Sparse Video Gen(SVG) v2



- Original attention map은 full attention score
- After permutation 은 clustering을 기준으로 재배열 했을 때의 시각화
  - 이 때 block 단위의 표현이 잘될수록 clustering이 semantic 기준으로 잘 됐다는 뜻
- Recovered after top-p는 이후 top-p selection 을 하고 역 permutation을 수행했을 때의 시각화



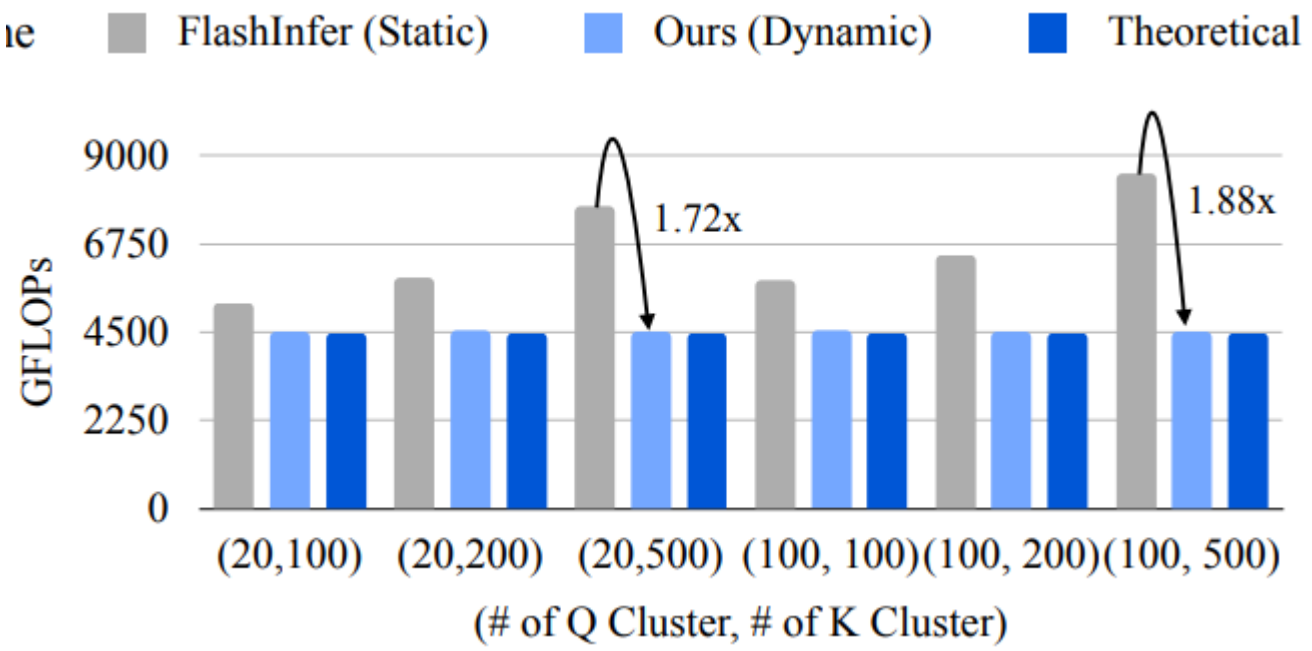
Sparse Video Gen(SVG) v2

- 실험 결과
  - WAN 및 Hunyuan에 대해서 평가
    - SVG 와 비교하여 더 높은 성능 유지와 latency 개선을 볼 수 있음

Model	Config	PSNR ↑	SSIM ↑	LPIPS ↓	VBench ↑	Density ↓	FLOP ↓ ↑	Speedup ↑
Wan 2.1	*14B, 720P, Image-to-Video*	-	-	-	0.841	100%	526.76 PFLOPs	1×
	SpargeAttn	21.181	0.665	0.333	-	38.99%	366.80 PFLOPs	1.47×
	SVG	24.059	0.813	0.174	0.836	30.25%	343.88 PFLOPs	1.56×
	Ours	26.562	0.861	0.138	0.838	31.28%	346.59 PFLOPs	1.58×
	Ours-Turbo	24.510	0.812	0.179	0.836	14.13%	301.62 PFLOPs	1.84×
Wan 2.1	*14B, 720P, Text-to-Video*	-	-	-	0.846	100%	658.46 PFLOPs	1×
	SpargeAttn	20.519	0.623	0.343	0.820	42.03%	468.46 PFLOPs	1.44×
	SVG	22.989	0.785	0.199	0.837	30.25%	429.86 PFLOPs	1.58×
	Ours	25.808	0.854	0.138	0.842	29.51%	427.43 PFLOPs	1.60×
	Ours-Turbo	23.682	0.789	0.196	0.838	12.87%	372.89 PFLOPs	1.89×
Hunyuan	*13B, 720P, Text-to-Video*	-	-	-	0.850	100%	612.38 PFLOPs	1×
	SpargeAttn	27.892	0.884	0.151	-	42.62%	399.16 PFLOPs	1.53×
	XAttention	28.892	0.898	0.120	0.839	39.32%	386.90 PFLOPs	1.56×
	SVG	29.157	0.905	0.120	0.845	29.86%	351.75 PFLOPs	1.91×
	SVG + FP8	29.033	0.902	0.121	0.843	29.86%	351.75 PFLOPs	2.3×
	Ours	30.452	0.910	0.117	0.852	25.45%	335.36 PFLOPs	2.30×
	Ours + FP8	30.389	0.908	0.118	0.851	25.45%	335.36 PFLOPs	2.55×

Sparse Video Gen(SVG) v2

- QK cluster에 따른 실험을 통해 GFLOPs 분석



Conclusion

- Video diffusion 을 위한 sparse attention 분석
- Sparse attention 의 static 및 dynamic 방법 조사