TimeSformer + STORM

Bertasius, Wang & Torresani, ICML 2021
Liang et al., arXiv 2026

Yifan Wu, Huishu Xue, Erika Tay

CS 8803 VLM

Video VLMs and spatiotemporal reasoning

Sep 29, 2026

Georgia Tech

Outline

- Problem Statement
- Related Works
- Approach: seminal and frontier
- Experiments & Results
- What persisted and what changed?
- Limitations & Societal Implications
- Discussion Questions

Part 1 problem, related work, TimeSformer

Part 2 STORM

Part 3 what persisted and changed, limitations, discussion

Is Space-Time Attention All You Need for Video Understanding?

Gedas Bertasius [1] Heng Wang [1] Lorenzo Torresani [1 2]

Seminal: ICML 2021

STORM: Internalized Modeling for Spatial-Temporal Reasoning in Video-Language Models

Yiming Liang[1*] Yixiao Chen[2*] Yiyang Zhou[3†] Yixuan Wang[3] Shoubin Yu[3]
Andong Deng[4] Fuxiao Liu[5] Qin Zhang[6] Chen Chen[4] Mohit Bansal[3] Huaxiu Yao[3]
[1]Purdue [2]Harvard [3]UNC [4]UCF [5]NVIDIA [6]Physion Labs

Frontier: arXiv, May 2026

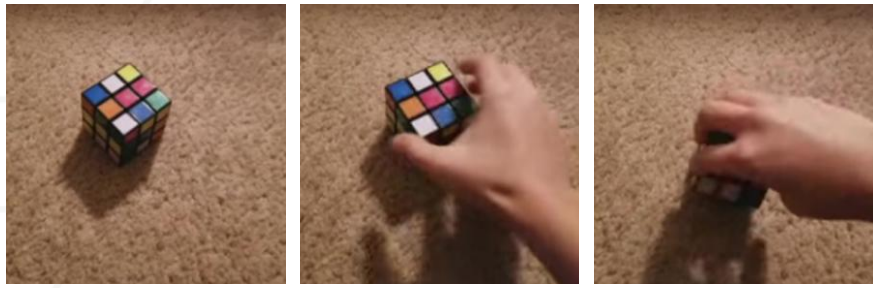
Georgia Tech

Part 1: TimeSformer

Georgia Tech

Problem Statement

Video models have to pay for TIME somewhere, compared to Image models



3 frames of one video clip

Every extra frame adds more to process, and relating frames to each other costs even more. Both papers are about keeping that cost under control.

TimeSformer, 2021

Can you recognize actions with attention alone and still afford it?

STORM, 2026

Can a video-language model reason about motion without slow test-time reasoning?

Three questions we'll ask of both papers

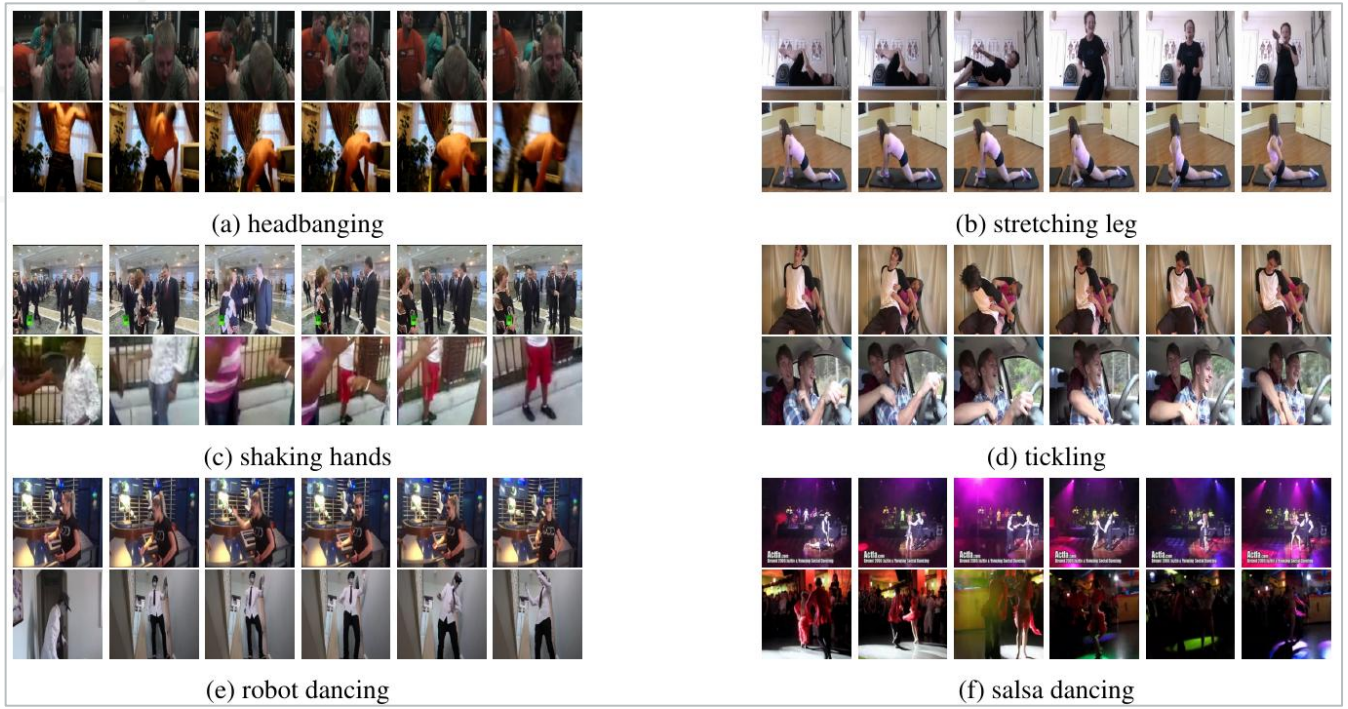
1 **Compute.** Where does the compute for time go?

2 **Supervision.** What actually tells the model about time?

3 **Evaluation.** Does the benchmark need time, or would single frames do?

Datasets: Kinetics-400 / 600

Most Kinetics classes can be recognized from a single frame



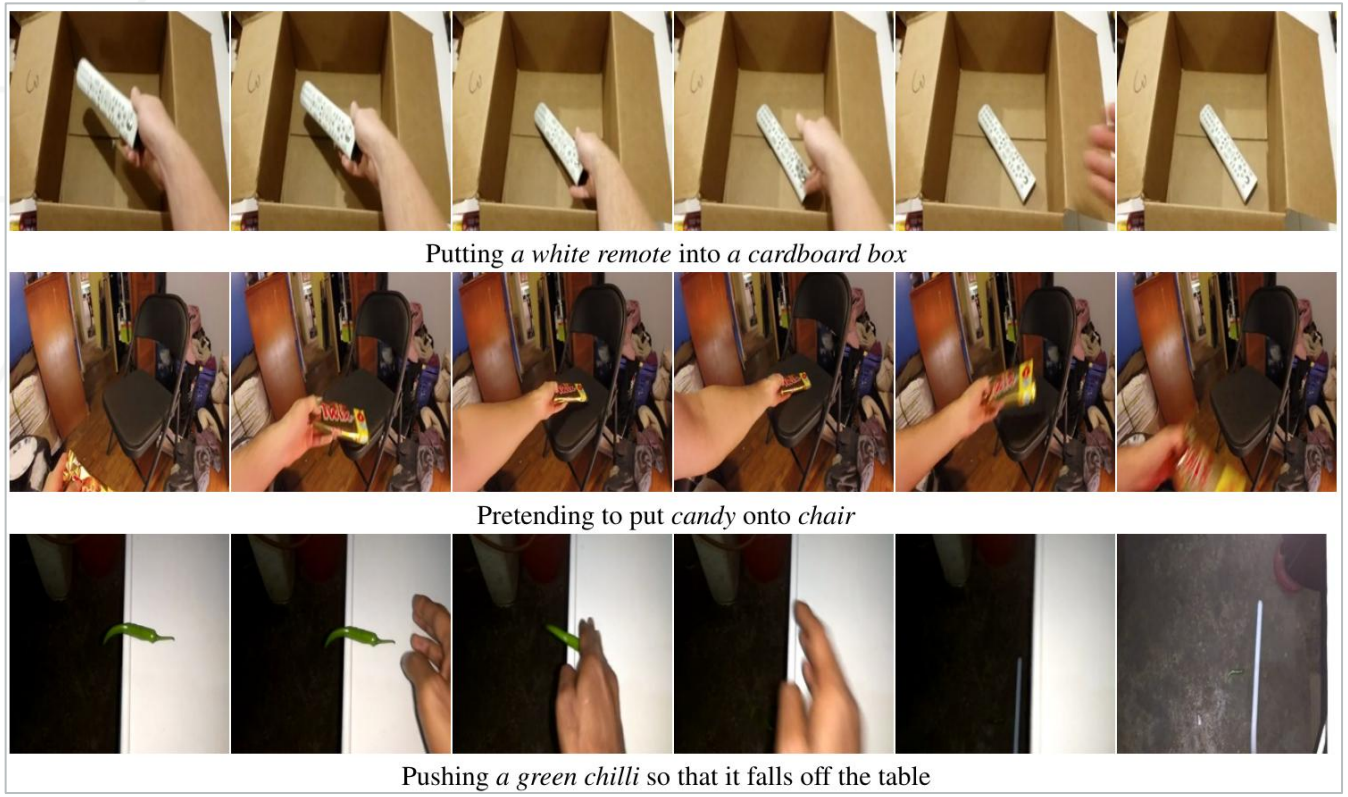
Kinetics paper (Kay et al. 2017), Fig. 1

At a glance	
Source	YouTube, about 10 s, one clip per video
Size	K400: 240K train / 20K val K600: 392K train / 30K val
Classes	400 / 600 human actions
Used for	Main results, ablations, cost

Georgia Tech

Datasets: Something-Something V2

In SSv2 the label depends on what the hand does over TIME



Something-Something paper (Goyal et al. 2017), Fig. 4

At a glance	
Source	Crowd workers act out a caption template
Size	220,847 clips: 168,913 train / 24,777 val / 27,157 test
Classes	174 templates, the worker picks the object
Used for	Attention schemes, ablations, state of the art

Georgia Tech

Datasets: Diving-48

Diving-48 classes differ only in how the diver moves



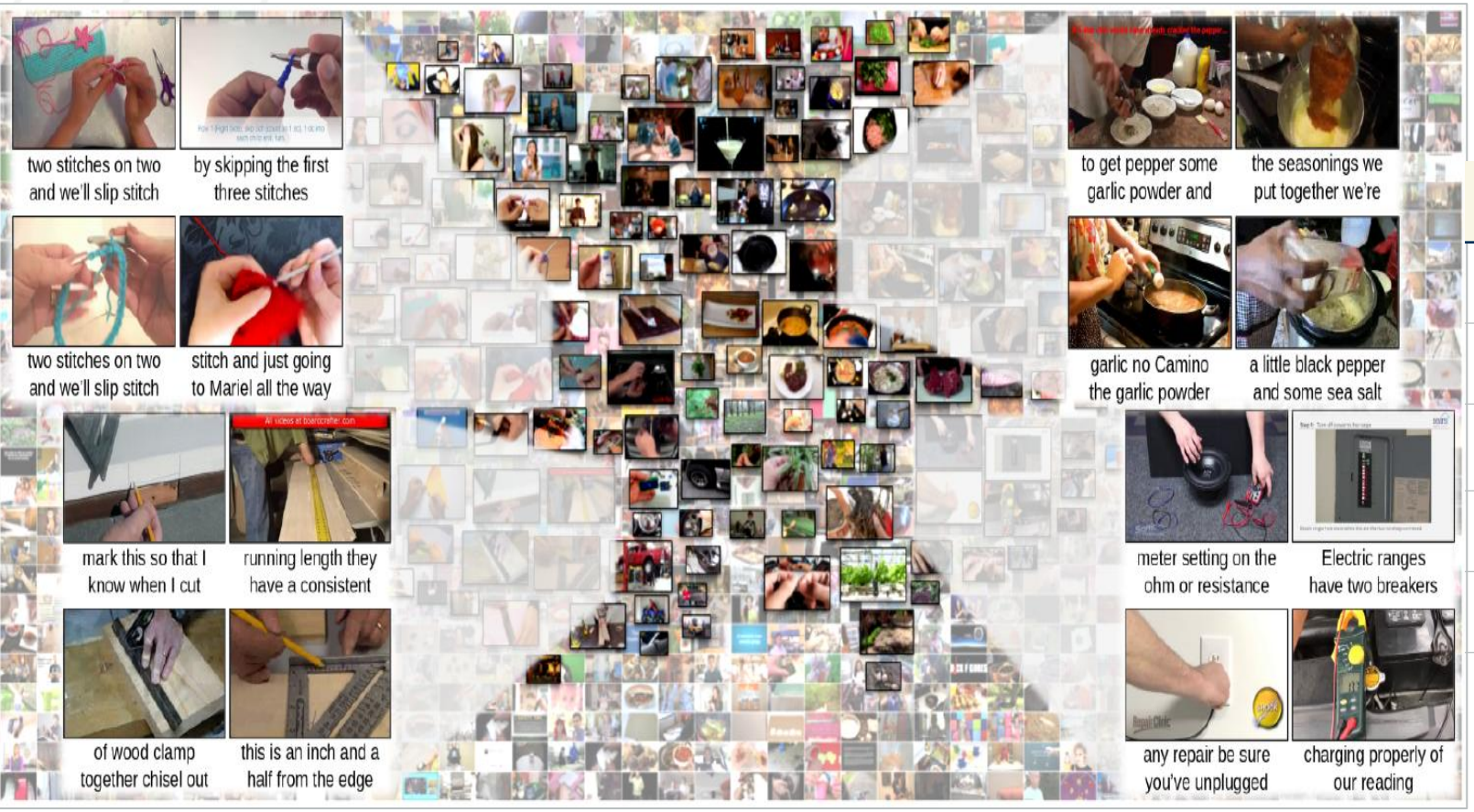
Diving-48 examples (Kanojia et al. 2019), Fig. 1

At a glance	
Source	Competitive diving videos
Size	18,404 clips: 16,067 train / 2,337 test
Classes	48 dives, each a mix of takeoff, somersaults, twists and flight position
Scene	Same kind of pool and background, so the scene does not help
Used for	State-of-the-art comparison (Table 7)

Georgia Tech

Datasets: HowTo100M

HowTo100M videos last minutes, far longer than one TimeSformer clip



At a glance	
Source	Narrated instructional YouTube videos
Full set	1.22M videos, 136M clips, 23.6K tasks
Subset used	120K videos, 1,059 tasks 85K train / 35K test
Length	About 6.5 min per video on average
Used for	Long-video test (Table 8)

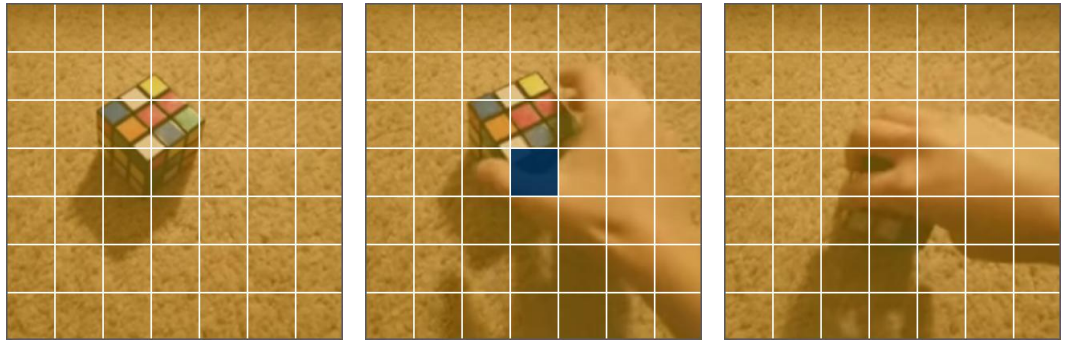
HowTo100M paper (Miech et al. 2019), Fig. 2

Georgia Tech

Problem Statement: what made it hard

3D CNNs were local and slow to train. Full attention over video was too expensive.

- 3D CNNs (I3D, SlowFast, X3D) only see a small neighborhood, so relating distant moments takes many layers.
- Slow to train: 3,840 GPU-hours for SlowFast.
- A clip of 8 frames at 224 px, cut into 16 x 16 patches, gives 1,568 patches. Full attention compares each with all the others, and at 32 frames it no longer fits in GPU memory.



frame t - 1 frame t frame t + 1

1 query vs 1,569 keys with full attention (real grid is 14x14)

Georgia Tech

Related Works: video models before 2021

TimeSformer keeps two ideas from 3D CNNs: reuse ImageNet, split space and time

Model	What it did	What TimeSformer takes from it
I3D (2017)	Inflated ImageNet 2D filters into 3D, trained on Kinetics	Start from image weights (here: ViT)
R(2+1)D (2018)	Split each 3D conv into a space part and a time part	The same split, done with attention
Non-local (2018)	Added attention blocks on top of a CNN	Attention as the whole model, not an add-on
SlowFast (2019)	Two streams: few frames in detail, many frames for motion	Main baseline (needs 30 test views)
X3D (2020)	Scaled a small 3D CNN along frames, resolution, width	Best Kinetics number before (80.4)

Georgia Tech

Related Works: from TimeSformer to STORM

By 2026 video goes into an LLM, and that is where STORM starts

Year	Work	Why it matters here
2021	TimeSformer, ViViT, MViT	Attention-based video backbones
2022	VideoMAE	Masked-video pretraining reduces the need for labeled data
2023	Video-ChatGPT, Video-LLaVA	Frame features fed into an LLM for open-ended video QA
2024	Qwen2-VL, Coconut	M-RoPE with time, height and width positions, latent reasoning in LLMs
2025	Qwen2.5-VL, Video-R1, Wan 2.1, Video-Thinker, LongVT	Absolute-time positions, RL for video reasoning, open video generation, test-time tools
2026	STORM	Latent steps trained with generated thought videos

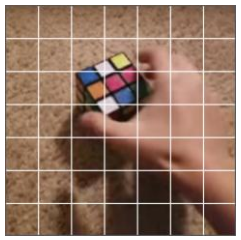
Georgia Tech

Approach: TimeSformer

ViT-Base with one change: the set of tokens each query may attend to



Input clip: 8 frames, 224 px



16 x 16 patches, N = 196

1,568 tokens
+ CLS, 12 blocks

Input embedding (Eq. 1)

$$\mathbf{z}_{(p,t)}^{(0)} = E\mathbf{x}_{(p,t)} + \mathbf{e}_{(p,t)}^{pos}$$

Output (Eq. 10), then MLP

$$\mathbf{y} = \mathrm{LN}\left(\mathbf{z}_{(0,0)}^{(L)}\right) \in \mathbb{R}^{D}$$

e(p, t) is a learned position for patch p in frame t, the only explicit source of temporal order.

From ViT: 16 x 16 patches as tokens, 196 per 224 px frame. The authors credit most of the efficiency to this tokenization, not to sparse attention.

Model details	
Input	Clip of 8 RGB frames, 224 x 224, stride 32 (about 8.5 s). 196 patches of 16 x 16 per frame
Output	Scores over C classes (400, 600, 174, 48 or 1,059) from the CLS token: LN, then a 1-hidden-layer MLP
Backbone	ViT-Base: 12 blocks, D = 768, 12 heads. 1,568 tokens plus CLS
Init	ImageNet-1K or 21K. Temporal attention copied from the spatial weights
Inference	3 views (1 clip x 3 crops), scores averaged. CNN baselines use 30
Variants	HR: 16 frames at 448 px. L: 96 frames at 224 px (about 12 s)

Georgia Tech

Approach: TimeSformer

Five ways to make attention affordable: restrict what each patch attends to

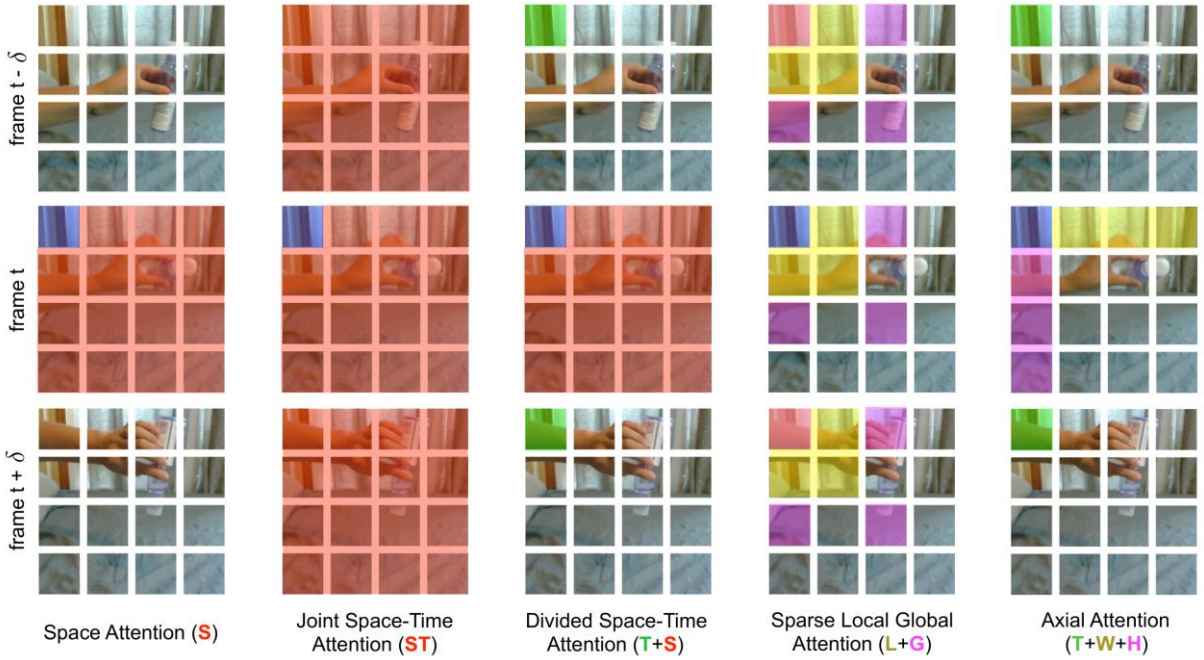


Fig. 2: blue = the patch being updated, colored = the patches it attends to

Scheme	Each patch attends to	Keys
Space (S)	its own frame only	197
Joint (ST)	every patch in every frame	1,569
Divided (T+S)	**the same spot in all frames, then its own frame**	**9 + 197**
Local-global (L+G)	a local window, then a sparse sample of the clip	392 + 196
Axial (T+W+H)	along time, then width, then height	8 + 14 + 14

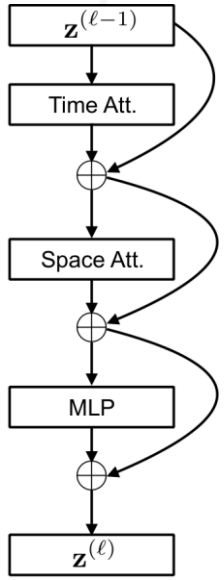
Highlighted: divided attention, the scheme TimeSformer uses

Takeaway: Space-only is an image model, joint is the expensive upper bound. The other three look for a middle ground.

Georgia Tech

Approach: TimeSformer

Divided attention: look across time first, then across space



Divided Space-Time Attention (T+S)

Fig. 1: one block



Divided Space-Time Attention (T+S)

Fig. 2: green = time, red = space

1 Time step same location p in all 8 frames, plus CLS: 9 keys

$$\boldsymbol{\alpha}_{(p,t)}^{(\ell,a)\text{time}} = \text{SM}\left(\frac{\mathbf{q}_{(p,t)}^{(\ell,a)\top}}{\sqrt{D_h}} \cdot \left[\mathbf{k}_{(0,0)}^{(\ell,a)} \left\{\mathbf{k}_{(p,t')}^{(\ell,a)}\right\}_{t'=1,\ldots,F}\right]\right)$$

What happened at this spot over time?

2 Space step all 196 patches of the same frame t, plus CLS: 197 keys

$$\boldsymbol{\alpha}_{(p,t)}^{(\ell,a)\text{space}} = \text{SM}\left(\frac{\mathbf{q}_{(p,t)}^{(\ell,a)\top}}{\sqrt{D_h}} \cdot \left[\mathbf{k}_{(0,0)}^{(\ell,a)} \left\{\mathbf{k}_{(p',t)}^{(\ell,a)}\right\}_{p'=1,\ldots,N}\right]\right)$$

What is around me in this frame?

Each step has its own W_Q, W_K, W_V and a residual connection. An MLP follows.

Georgia Tech

Approach: TimeSformer

What divided attention buys, and what it gives up

Joint: keys per patch

1,569

Divided: keys per patch

206

9 + 197

Cost per layer: joint $O(N^2F^2)$, divided $O(NF(N + F))$, so it grows linearly with the number of frames F.

What it gives up

A patch never attends directly to a different location in another frame. Motion across locations has to pass through both steps, which later work (trajectory attention in Motionformer) targets.

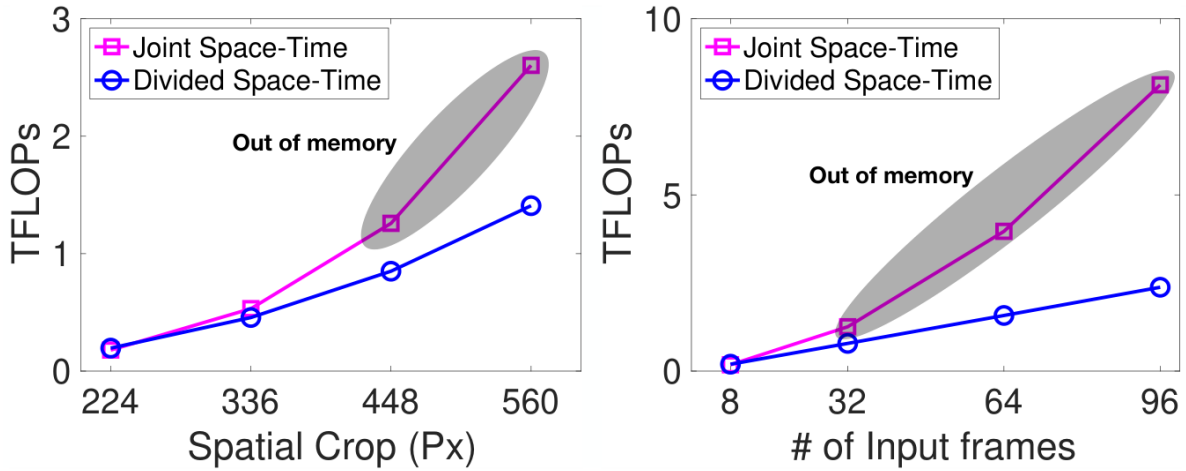


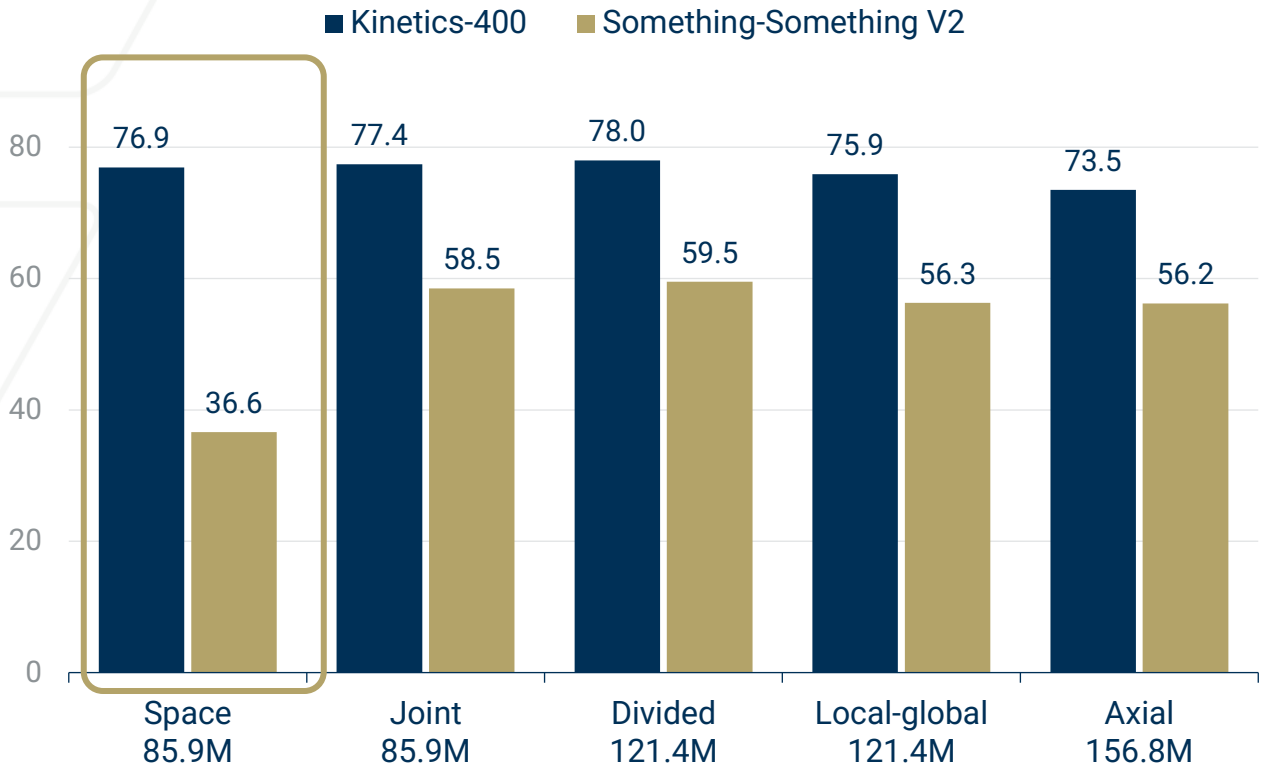
Fig. 3: joint attention runs out of memory at 448 px or 32 frames

Takeaway: At 8 frames the cost is similar. The real gain is headroom: it is what lets TimeSformer-L use 96 frames.

Georgia Tech

Experiments & Results: TimeSformer

Divided attention is the most accurate, and space-only shows which dataset needs time



- Divided: 78.0 / 59.5, best on both.
- Joint: 77.4 / 58.5, close behind but far more expensive.
- Axial: most parameters, lowest accuracy.
- Space-only (circled): 1.1 behind on K400, 22.9 behind on SSv2.

Takeaway: Kinetics can't tell a temporal model from a frame classifier (space-only: -1.1). SSv2 can (-22.9).

Georgia Tech

Experiments & Results: what does it look at?

It follows the hand, then the object



Fig. 7: top = SSv2 frames, bottom = where attention goes

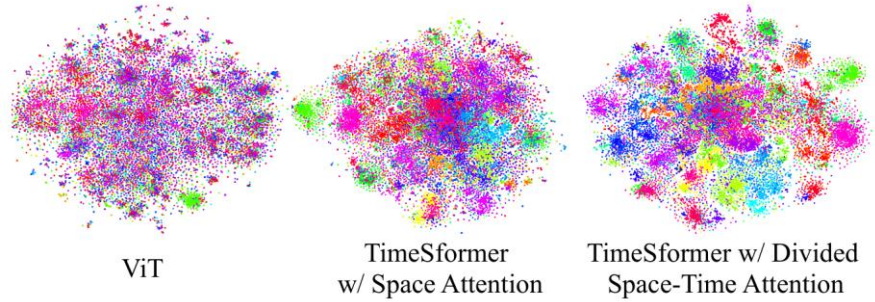


Fig. 8: SSv2 videos as dots, one color per class

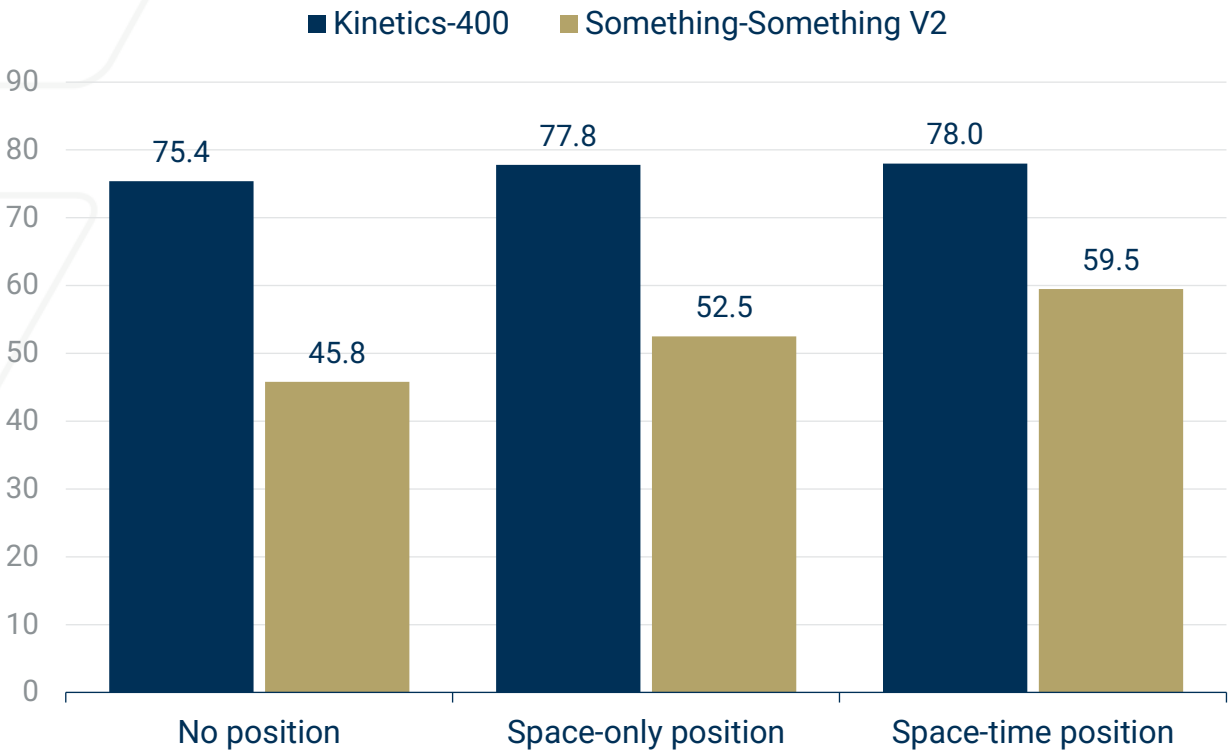
- Attention stays on the hand, and on the object once the hand leaves
- Divided attention gives the cleanest class groups (right)

Takeaway: With attention across time, videos of the same action group together more cleanly than with ViT or space-only.

Georgia Tech

Ablations: TimeSformer

What tells the model about time? Mostly the position embedding.



Position embedding ablation (Table 4), top-1 %

Bigger image pretraining (Table 3)

ImageNet-21K instead of 1K: K400 75.8 to 78.0, SSv2 59.5 to 59.5. More images help appearance, not time.

Other ablations (from the text)

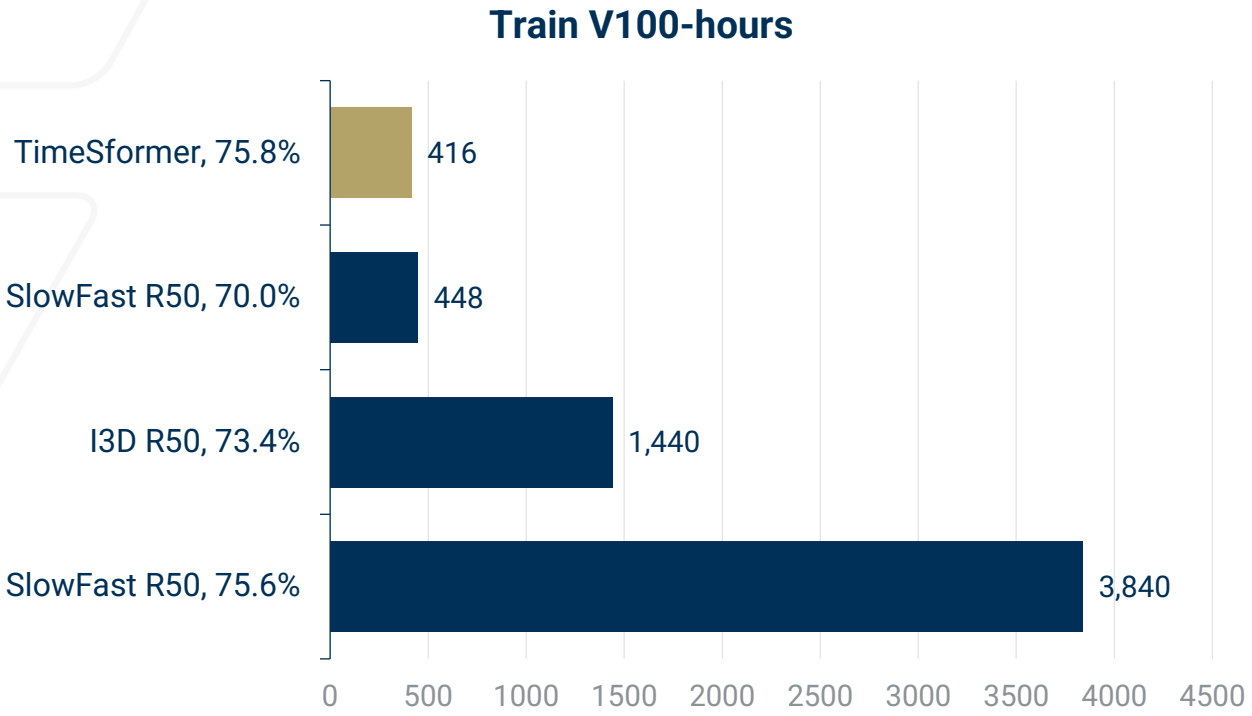
- Space before time: 0.5% worse. In parallel: 0.4% worse.
- ViT-Large about 1% worse, ViT-Small about 5% worse.
- Patches of 32 instead of 16: about 3% worse.

Takeaway: Time position is worth +7.0 on SSv2, ImageNet-21K +0.0. Order of time and space barely matters.

Georgia Tech

Experiments & Results: TimeSformer

Cheaper to train than 3D CNNs, mostly thanks to ImageNet initialization



Training cost on K400 in V100-hours, with top-1 accuracy (Table 2)

Why training is cheap

Trained from scratch, TimeSformer reaches only 64.8%. The ImageNet start does most of the work.

Why inference is cheap

0.59 TFLOPs covers 3 views, while the CNN numbers cover 30. On the 20K validation videos: TimeSformer-L 2.6 h, SlowFast 14.88 h.

Our read: most of the ~9x saving is ImageNet init (64.8% from scratch), plus 3 views instead of 30.

Georgia Tech

Experiments & Results: TimeSformer

Against the strongest 3D CNNs: ahead on three benchmarks, behind on SSv2

Benchmark	Best prior in paper	TimeSformer	-HR	-L
K400	80.4 (X3D-XXL)	78.0	79.7	80.7
K600	81.9 (X3D-XL)	79.1	81.8	82.2
SSv2	65.2 (bLVNet)	59.5	62.2	62.4
Diving-48	77.6 (SlowFast*)	74.9	78.0	81.0

top-1 %. *reproduced SlowFast (the old Diving-48 labels had an issue)

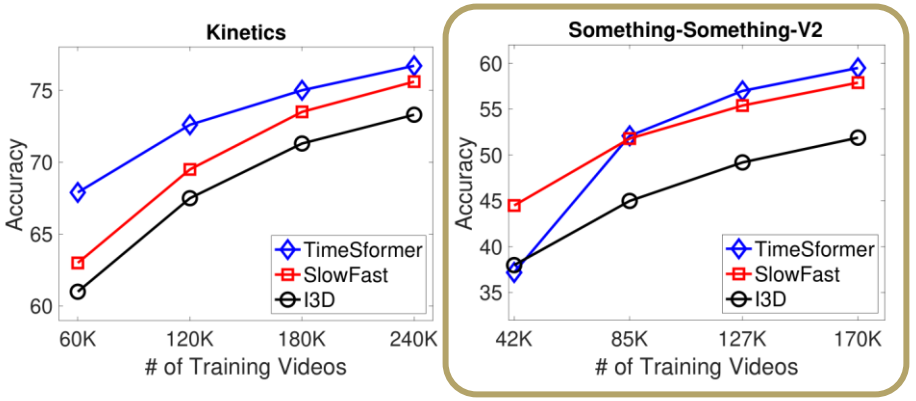


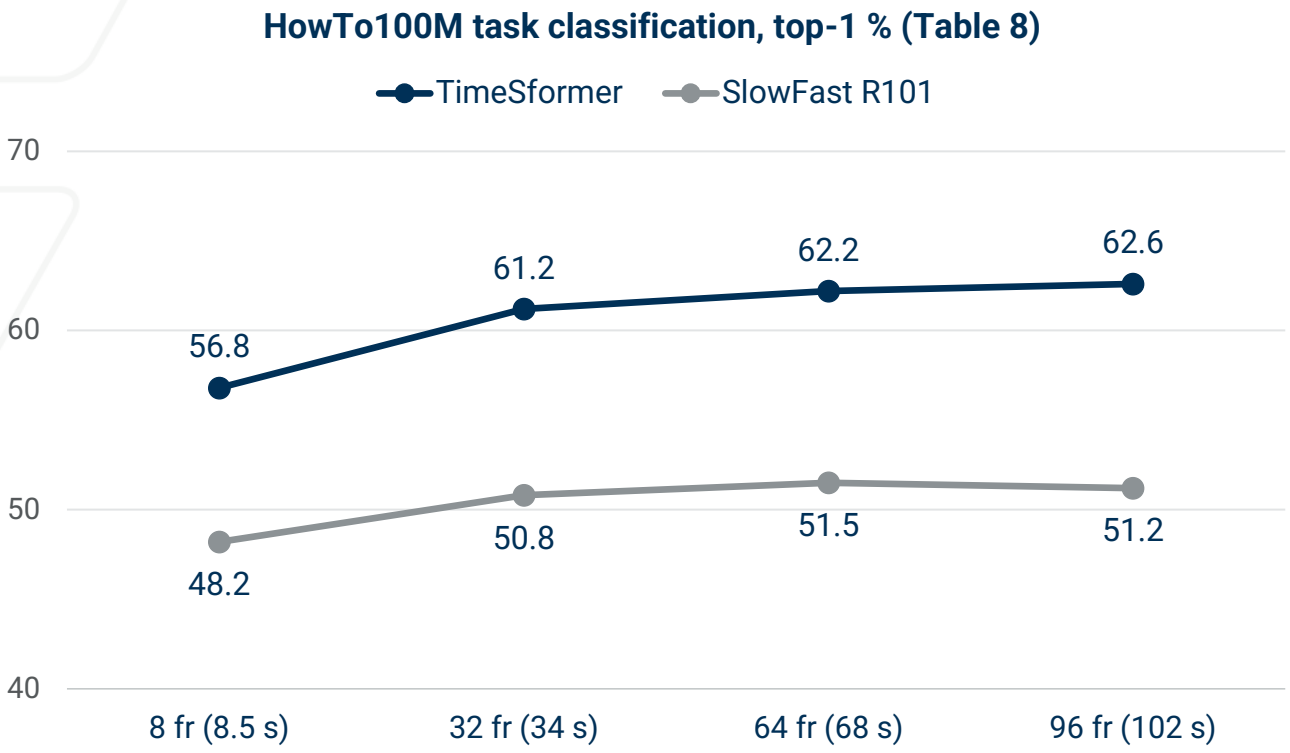
Fig. 4: on SSv2 it only wins with 75% or more of the data

Takeaway: Learning motion from labels alone takes more data than learning appearance.

Georgia Tech

Experiments & Results: TimeSformer

More frames help, with diminishing returns



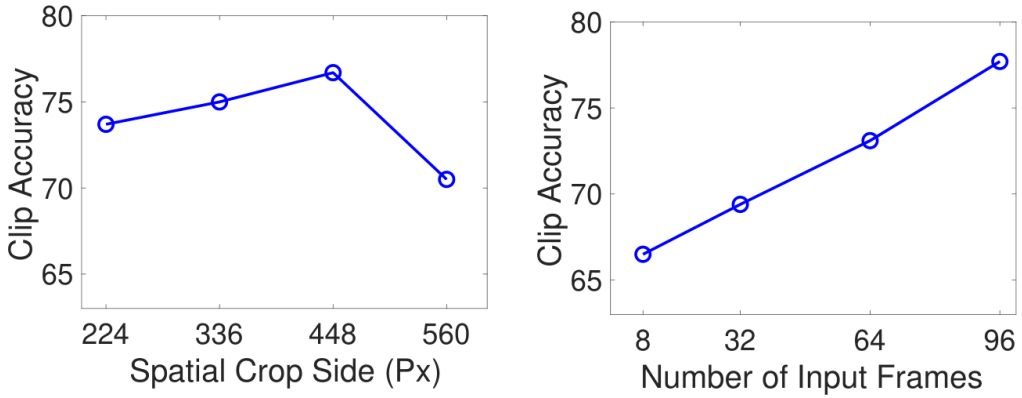


Fig. 5 (Kinetics)

- Beats SlowFast by 8.6 to 11.4 points at every clip length
- Each video is still cut into 4 to 48 clips and averaged

Takeaway: 32 to 96 frames adds only 1.4, and the model never sees the whole video. Long video stays open.

Georgia Tech

Part 1 recap: TimeSformer

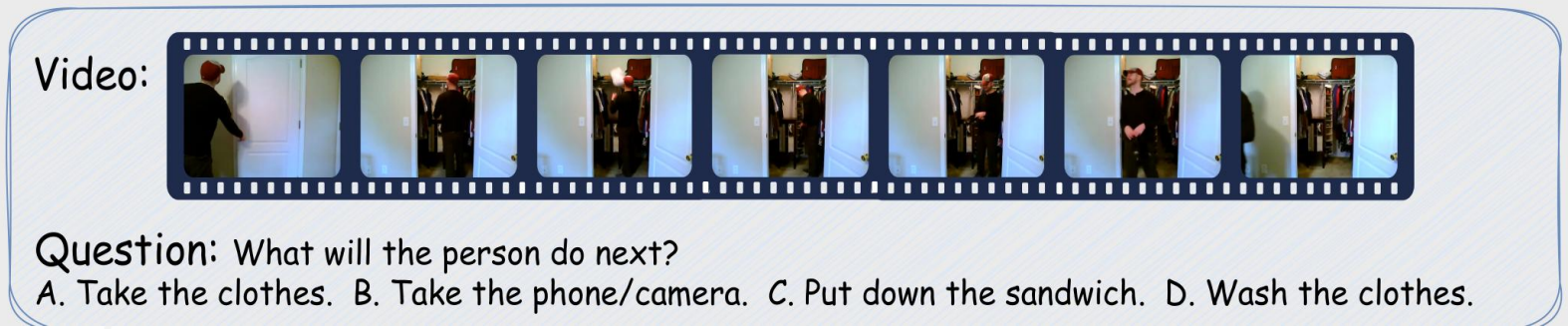
Our three questions, answered for 2021

Question	TimeSformer	Evidence
Where does the compute for time go?	Into every encoder block. Divided attention makes 96 frames fit.	Joint attention runs out of memory at 32 frames (Fig. 3)
What tells the model about time?	Class labels plus one time position embedding. Needs lots of data.	Time position: +7.0 on SSv2 (Table 4). SSv2 needs 75% or more of the data (Fig. 4)
Does the benchmark need time?	Kinetics barely, SSv2 a lot.	Space-only: -1.1 on K400, -22.9 on SSv2 (Table 1)

Part 2: STORM: Internalized Modeling for Spatial-Temporal Reasoning in Video-Language Models

Georgia Tech

Problem Statement: STORM



Example in Fig. 4. Which option would you pick?

To answer, the model must track

- **Motion:** he walks to the closet and opens it
- **Temporal order:** what already happened, so what comes next
- **Evolving visual states:** what is picked up or put down

The paper's question: can intermediate visual evidence be used during training to induce an internal latent reasoning process, while keeping inference as a single bounded decoding procedure?

We judge the answer on two things: accuracy and inference cost.

Georgia Tech

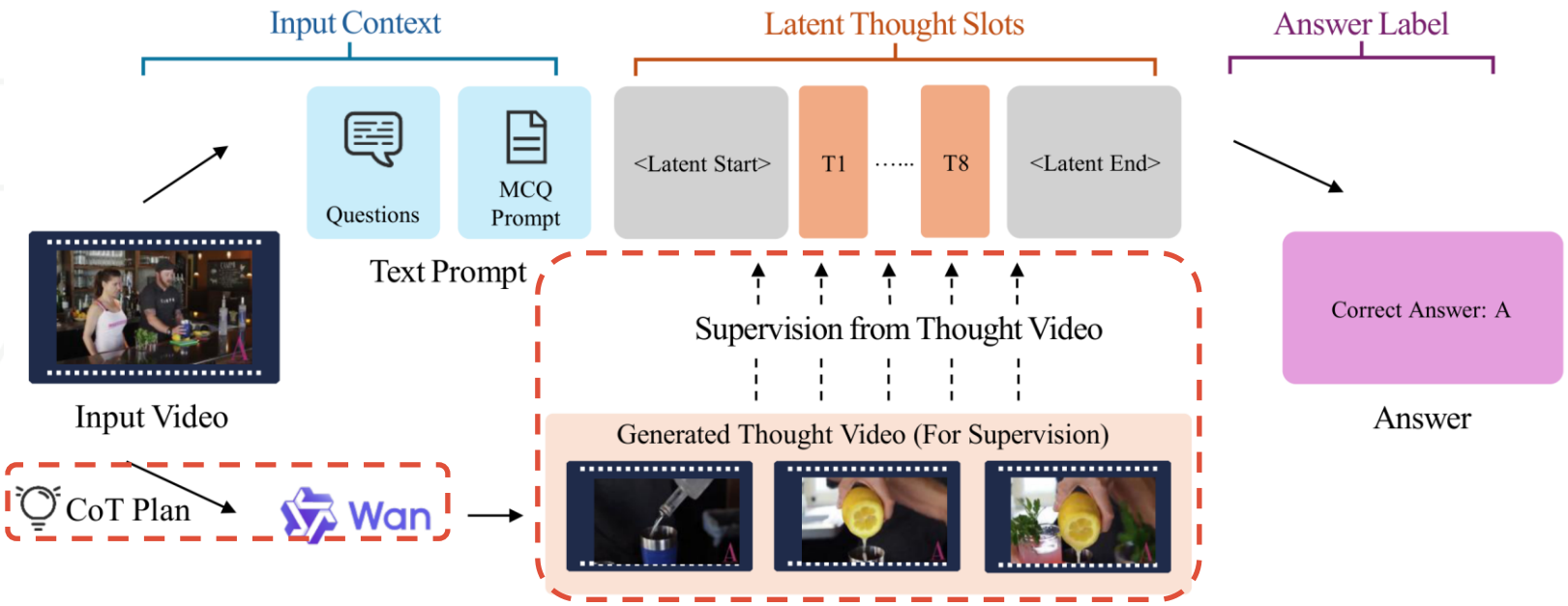
Motivation: STORM

A fundamental trade-off in video reasoning

- **Textual chain-of-thought:** structures reasoning but struggles to represent fine-grained temporal and visual evidence.
- **Revisiting frames or using tools:** provides visual evidence but adds latency, complexity, and potential failures.
- **Core trade-off:** preserving rich visual-temporal information versus inference efficiency and robustness.
- **STORM's approach:** internalize visual-temporal reasoning in continuous hidden states, avoiding repeated processing of explicit evidence.

Key Idea: STORM

The key idea is to expose the model to rich dynamic evidence during training, but to require it to use only compact hidden-state rollouts at inference time.



Red dashed boxes: used to create training supervision

Figure 1

Each training instance contains three aligned signals

1. Sampled keyframes from the source video
2. A generated thought video that visualizes question-relevant motion and state changes
3. The target answer sequence

At inference time, STORM does not generate thought videos, revisit frames, or call external visual tools.

Georgia Tech

Related Works: STORM

- **Latent visual and implicit reasoning:** latent visual tokens and continuous thought (e.g. Coconut); mostly designed for text reasoning, static images or spatial imagination.
- **Video-language models:** e.g. Qwen2.5-VL, LLaVA-Video, InternVL3; they typically still reason over sampled or compressed visual tokens at inference time.
- **Test-time visual reasoning:** temporal CoT, frame selection, zooming and tools make the visual evidence explicit during inference.

What is new: STORM uses generated thought videos as training-time supervision for bounded latent trajectories, avoiding repeated frame selection and external tool calls at inference.

Georgia Tech

Approach: latent formulation

The answer is conditioned on latent states that the model computes itself

Standard video QA

$$P_\theta(\mathcal{Y} \mid \mathcal{V}, \mathcal{Q}) = \prod_{t=1}^{N} P_\theta(y_t \mid y_{<t}, \mathcal{V}, \mathcal{Q}). \quad (1)$$

STORM

$$P_\theta(\mathcal{Y} \mid \mathcal{V}, \mathcal{Q}, \mathcal{Z}) = \prod_{t=1}^{N} P_\theta(y_t \mid y_{<t}, \mathcal{V}, \mathcal{Q}, \mathcal{Z}), \quad \mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_K\}. \quad (2)$$

K = 8

V video frames ***Q*** question and options ***Y*** answer tokens ***Z*** latent states z_1 … z_K

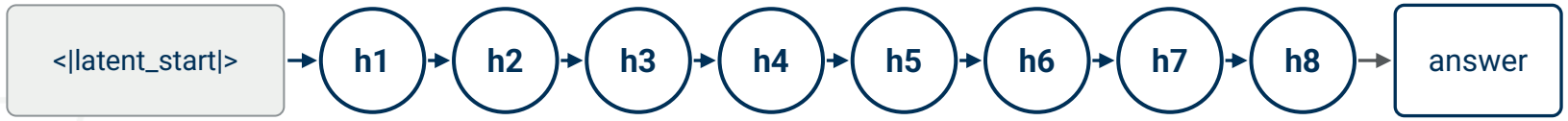
- **Latent simulation**: STORM introduces a short segment of K latent slots, where decoder hidden states serve as simulation states rather than decoded text.
- **Answer generation**: The answer is conditioned on both the original video-question context and the latent simulation states. Z acts as an internal “thought video” summary, storing dynamic visual evidence in compact latent form.
- **Bounded computation**: The model performs at most eight latent steps before returning to normal text decoding, without generating thought videos, revisiting frames, or calling external visual tools.

Georgia Tech

Approach: latent inference

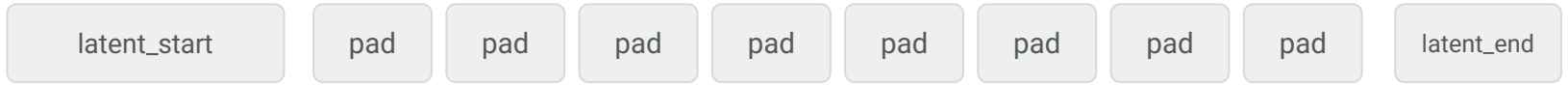
At test time, hidden states are fed back as inputs for at most 8 steps

What the model computes



hidden state goes back in as the next input

What the token sequence records



- After the question, the model emits <|latent_start|>
- A token is still predicted each step, but the sequence stores <|latent_pad|>
- Stops at <|latent_end|> or forced stop after 8 steps
- Then the answer is decoded normally

Why skip the word?

Unlike discrete text, latent representations support superposed and distributed features, so multiple reasoning factors can coexist in the same hidden states.

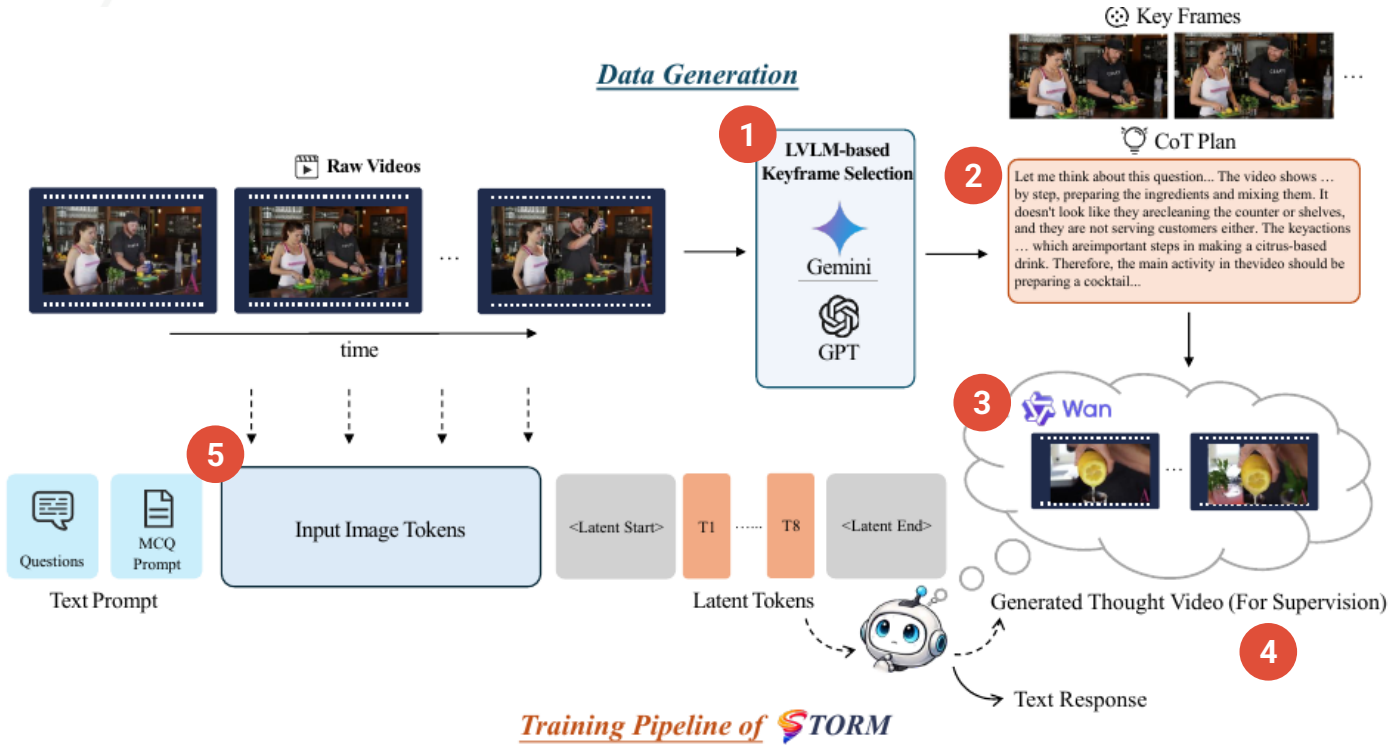
Extra cost

At most 8 steps plus two markers, instead of a text rationale of hundreds of tokens. No video generation and no tool calls.

Georgia Tech

Approach: data pipeline

Each training question gets a generated “thought video”



1. **Keyframes:** GPT-4o selects question-relevant keyframes from the source video. During this selection, it receives the question, answer choices, and correct answer.
2. **Plan:** a teacher model writes a chain-of-thought plan for answering the question.
3. **Thought video:** Wan 2.1 animates the first keyframe following the plan.
4. **Latent targets:** the vision encoder turns the thought video into features, average-pooled into 8 target vectors.
5. **Training:** in Stage I the 8 targets supervise the 8 latent states and the correct answer supervises the text; Stage II uses the answer only.

Each sample starts from an existing video QA example in Video-R1 (video, question and correct answer); about 5,000 thought videos are generated.

Georgia Tech

Approach: keyframe prompt (B.1)

B.1: GPT-4o picks the frames that matter, and it is given the correct answer

Prompt Template for Keyframe Selection

```
You are given a video represented by candidate frames.
Task: pick EXACTLY {need_n} frames from the candidates below that help answer the question.
Return ONLY JSON with this schema: {"selected":[i0,i1,...]}
Rules:
- Indices must come from the provided candidate list; they must be unique integers.
- JSON only. No extra text.

Question: {question}

Choices:
- A. {choice_0}
- B. {choice_1}
- ...

Correct answer: {answer}

frame_index: {idx}, t={t:.2f}s
[image_url: candidate frame]
```

1. **Task:** pick exactly n frames that help answer the question, and return their indices as JSON.
2. **The answer is given:** it sees the choices and the correct answer, so the chosen frames support that answer.
3. **Timestamps:** every candidate frame comes with its index and time.

Note: the model never sees this information at test time. That is fine for training, but it could teach shortcuts.

Georgia Tech

Approach: thought-video prompt (B.2)

B.2: Wan animates the keyframe following the teacher's plan

Prompt Template for Thought Video Generation

```
You are given a keyframe from a source video and a reasoning plan for answering a video
question. Generate a short thought video that visualizes the temporal evolution needed to
answer the question.

Requirements:
- Preserve the scene layout, object identity, appearance, and camera viewpoint from the
  keyframe as much as possible.
- Animate only the question-relevant motion, interaction, or state change described by the
  reasoning plan.
- Keep the temporal order physically plausible and visually coherent.
- Do not add irrelevant objects, subtitles, watermarks, text overlays, or extra scenes.
- The video should focus on the evidence needed for the final answer, rather than producing
  a generic continuation.

Question: {question}

Reasoning plan:
{teacher_cot}

Initial keyframe:
[image_url: selected keyframe]

Output:
A short coherent thought video conditioned on the keyframe and reasoning plan.
```

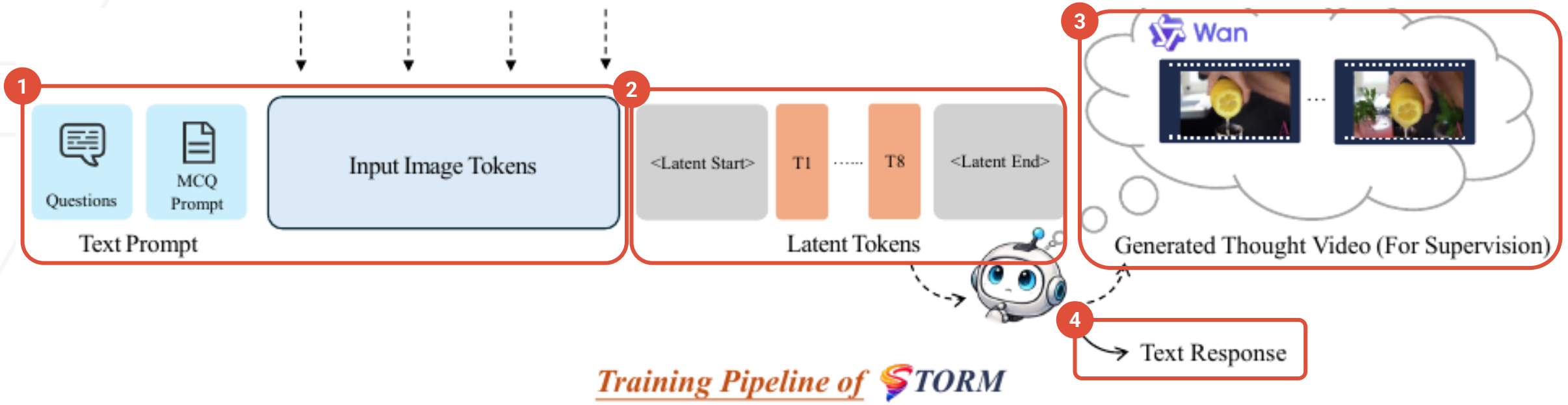
1. **Keep the scene:** preserve the scene layout, object identity, appearance and camera viewpoint.
2. **Animate what matters:** only the question-relevant motion, interaction or state change.
3. **Evidence, not a continuation:** no irrelevant objects, subtitles or extra scenes.
4. **Inputs:** the selected keyframe, the question and the teacher's reasoning plan.

So the thought video adds motion, not new appearance. Limitation noted in the paper: noisy keyframes or inaccurate generated motion may lead to imperfect latent targets.

Georgia Tech

Approach: Stage I training sequence

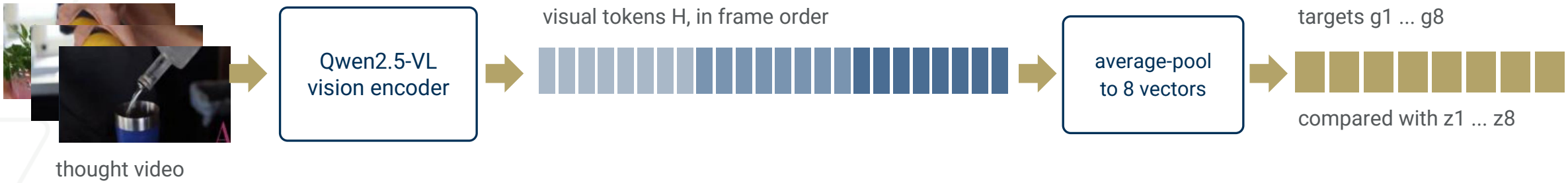
Stage I: one sequence holds the prompt, 8 latent slots and the answer



1. **Prompt:** keyframe tokens, question and options (at test time: 32 uniform frames). The model only reads these; no loss.
2. **Latent slots:** 8 hidden states, each fed back as input. Each slot is trained toward its own target.
3. **Thought video:** provides the 8 targets. Used only in Stage I, never as input.
4. **Answer:** trained with the usual cross-entropy loss; it can look at the latent slots.

Approach: Stage I targets

The 8 targets come from the thought video's own visual tokens



How the targets are made

1. Encode the thought video with Qwen2.5-VL's own vision encoder.
2. Put all visual tokens in one sequence, in frame order (M tokens; M differs per clip).
3. Average-pool the sequence into 8 nearly equal, consecutive groups.
4. The 8 averages are the targets g_1 … g_8, one per latent slot.

What this means

- Order is kept: g_1 covers the start of the thought video, g_8 the end.
- Pooling is over tokens, not fixed time intervals: a segment can span part of a frame or several frames.
- Targets have the hidden-state size, so each slot can act like a compressed visual token.
- Each g_i gives compact temporal-visual supervision for its latent slot z_i.

Georgia Tech

Approach: Stage I loss

Match the targets and predict the answer; the matching term has weight 0.1

$$\mathcal{L}_{\text{latent}} = \frac{1}{K}\sum_{i=1}^{K}\|\mathbf{z}_i - \mathbf{g}_i\|_2^2, \qquad \mathcal{L}_{\text{ans}} = -\sum_{t=1}^{N}\log P_\theta(y_t \mid y_{<t}, \mathcal{V}, \mathcal{Q}, \mathcal{Z}). \tag{3}$$

latent loss (mean squared error) *answer loss (cross-entropy)*

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{ans}} + \lambda\mathcal{L}_{\text{latent}}, \tag{4}$$ **λ = 0.1**

- **Latent loss:** the average squared distance between each latent state and its target. It pulls every slot toward what the thought video shows.
- **Answer loss:** the negative log-likelihood of the answer given the video, the question and all latent states. It makes the slots useful for answering.
- **Weight 0.1:** λ controls the latent alignment strength; the paper reports no other values.
- **Why both:** with only the latent loss, the slots could copy features the answer never uses; with only the answer loss, nothing ties them to the video (the “Stage II only” ablation).

No pixels are generated by the model; the loss works on features. Stage I is trained for one epoch.

Georgia Tech

Approach: Stage I vs Stage II

Same input and latent loop in both stages; Stage II removes the targets and the latent loss

Stage I *latent-grounded supervised training*

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{ans}} + \lambda \mathcal{L}_{\text{latent}}, \quad \lambda = 0.1$$

Keyframe tokens | Question + options | <|latent_start|> | z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 | <|latent_end|> | **Answer**

read only, no loss



thought video

$$\mathcal{L}_{\text{latent}} = \frac{1}{K} \sum_{i=1}^{K} \|\mathbf{z}_i - \mathbf{g}_i\|_2^2, \qquad \mathcal{L}_{\text{ans}} = -\sum_{t=1}^{N} \log P_\theta(y_t \mid y_{<t}, \mathcal{V}, \mathcal{Q}, \mathcal{Z}).$$

targets g_i: pooled thought-video features

Stage II *answer-only consolidation*

Loss: only L_{ans}

Keyframe tokens | Question + options | <|latent_start|> | z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 | <|latent_end|> | **Answer**

read only, no loss

no thought video

no targets, no latent loss

$$\mathcal{L}_{\text{ans}} = -\sum_{t=1}^{N} \log P_\theta(y_t \mid y_{<t}, \mathcal{V}, \mathcal{Q}, \mathcal{Z}).$$

Both stages: 1 epoch, learning rate 1e-5; Stage II starts from the Stage I model. At test time the visual input is 32 uniform frames instead of keyframes.

Georgia Tech

Approach: Stage II answer-only consolidation

Remove direct latent supervision and optimize only the answer loss

- Latent slots are still present in the input pattern, but the model receives **no explicit loss on latent states**.
- This forces latent reasoning to become **self-organized and robust**, while preserving the **answer-focused objective used at inference**.

What this means

1. **Self-organized:** without a target for each slot, the model decides what the slots hold and keeps whatever helps the answer.
2. **Robust:** the slots are shaped by the answer, not by copying generated videos that may contain mistakes.
3. **Answer-focused, as at inference:** at test time there is no thought video, only the question and the answer. Stage II trains exactly that.

A Coconut-like setting: the latent slots remain in the sequence, but supervision is applied only to the final answer. It is still supervised fine-tuning, not reinforcement learning.

Georgia Tech

Approach: what each stage adds

Table 6: the two training stages are complementary

	Setting	VideoMME	TempCompass	MVBench
1	Stage I only	58.8	72.0	56.0
2	Stage II only	56.6	72.6	56.8
3	**Full STORM (Stage I + Stage II)**	**61.0**	**74.3**	**61.1**

Scores: accuracy (%) when training with Stage I only, Stage II only, or both.

For reference, plain SFT without latent slots: 55.4, 69.9 and 60.5 (Table 1).

The best performance requires both thought-video-grounded latent supervision in Stage I and answer-focused consolidation in Stage II.

1. **Stage I only:** learns grounded latent dynamics from thought-video supervision but lacks large-scale answer alignment; 2.2 to 5.1 points below the full model.

2. **Stage II only:** benefits from answer-token supervision but lacks latent-space grounding; 1.7 to 4.4 points below the full model, and below SFT on MVBench.

3. **Both stages:** the full pipeline consistently outperforms either stage alone, by 1.7 to 5.1 points.

Georgia Tech

Experiments: setup

Six multiple-choice video benchmarks; the key comparison is SFT on the same data

- **General video QA:** VideoMME (900 videos, 11 s to 1 hour), MVBench (about 20 temporal tasks), TempCompass (speed, direction, order, change)
- **Video reasoning:** VideoEspresso (CoT-based VideoQA, 14 task types), Video-Holmes (270 suspense short films), MMVU (expert-level, multi-discipline)
- **Score:** accuracy (%), the share of questions answered correctly; 32 uniform frames (LongVT and Video-Thinker: up to 32 per round)
- **Key baseline, SFT:** Qwen2.5-VL-7B-SFT, trained on the same dataset as STORM but without thought-video supervision
- **Other baselines:** text reasoning (Qwen chain-of-thought, Video-R1), frames or tools (LongVT, Video-Thinker), GPT-4o and Gemini-1.5-Pro
- **Training:** Qwen2.5-VL-7B-Instruct; video part of the Video-R1 data with 5K thought videos; $K = 8$, $\lambda = 0.1$; one epoch per stage, learning rate 1e-5, batch size 1; four NVIDIA GB200 GPUs

Georgia Tech

Results: general video QA

With the same training data, STORM beats both SFT and text reasoning

Model	VideoMME	MVBench	TempCompass	Frames
Proprietary Models				
GPT-4o [19]	71.9	-	73.75	32
Gemini-1.5-Pro [34]	75.0	-	67.1	32
Open-Source Models				
VideoChat2 [23]	39.5	51.1	48.8	32
LLaMA-VID [24]	-	41.9	45.6	32
VideoLLaVA [26]	-	43.0	49.77	32
ST-LLM [28]	37.9	54.85	-	32
ShareGPT4Video [6]	39.9	51.2	-	32
LLaVA-Next-Video [53]	60.2	-	66.74	32
PLLaVA [41]	-	46.6	-	32
LongVA [51]	52.6	-	56.9	32
VideoLLaMA2 [8]	47.9	54.6	-	32
Video-CCAM [11]	-	**64.6**	-	32
Long-LLaVA [38]	52.9	-	-	32
Kangaroo [27]	56.0	61.1	62.5	32
LongVT-RFT (LongVT, think with tools) [47]	59.7	53.9	68.3	32
Video-Thinker-7B (think with video) [36]	60.0	62.4	67.8	32
Qwen2.5-VL-7B-Instruct(CoT) [2]	60.8	42.4	72.8	32
Qwen2.5-VL-7B-SFT	55.4	60.5	69.9	32
Video-R1-7B-wo-image [12]	53.8	60.9	69.8	32
Video-R1-7B [12]	57.4	62.7	72.6	32
STORM	**61.0**	61.1	**74.3**	32

Scores: accuracy (%), higher is better; "-" = not reported; Frames = input frames per video.

1. **Same data, better results:** +5.6 on VideoMME, +4.4 on TempCompass and +0.6 on MVBench over SFT. The gain comes from the training method, not from more data.

2. **SFT hurts VideoMME, STORM does not:** plain SFT is 5.4 points below the untuned model, while STORM is at 61.0 vs 60.8. The latent supervision may help keep general skills (our reading).

3. **Best on TempCompass:** 74.3 on speed, direction and order questions, the top open model here and on par with GPT-4o (73.75).

Caveat: STORM is not best everywhere: Video-CCAM, Video-R1 and Video-Thinker score higher on MVBench, and GPT-4o and Gemini lead on VideoMME.

Results: reasoning benchmarks

Gains are larger on reasoning benchmarks, without any test-time tools

Model	VideoEspresso	Video-Holmes	MMVU	Frames
Proprietary Models				
GPT-4o [19]	-	42.0	75.4	32
Gemini-1.5-Pro [34]	44.2	45.7	-	32
Open-Source Models				
LongVT-RFT (LongVT, think with tools) [47]	52.7	35.5	61.4	32
Video-Thinker-7B (think with video) [36]	56.4	43.1	63.4	32
Qwen2.5-VL-7B-SFT	49.6	31.4	62.1	32
Qwen2.5-VL-7B-Instruct(CoT) [2]	45	27.8	60.0	32
Video-R1-7B [12]	-	36.5	64.2	32
STORM	**58.7**	**37.8**	**65.9**	32

Scores: accuracy (%), higher is better; "-" = not reported.

Average gain over SFT: +6.4 on the reasoning benchmarks, compared with +3.5 on the general ones.

Across Tables 1 and 2, STORM has the best open-model score on 4 of the 6 benchmarks.

1. **Largest gains:** +9.1 on VideoEspresso, +6.4 on Video-Holmes and +3.8 on MMVU over SFT.

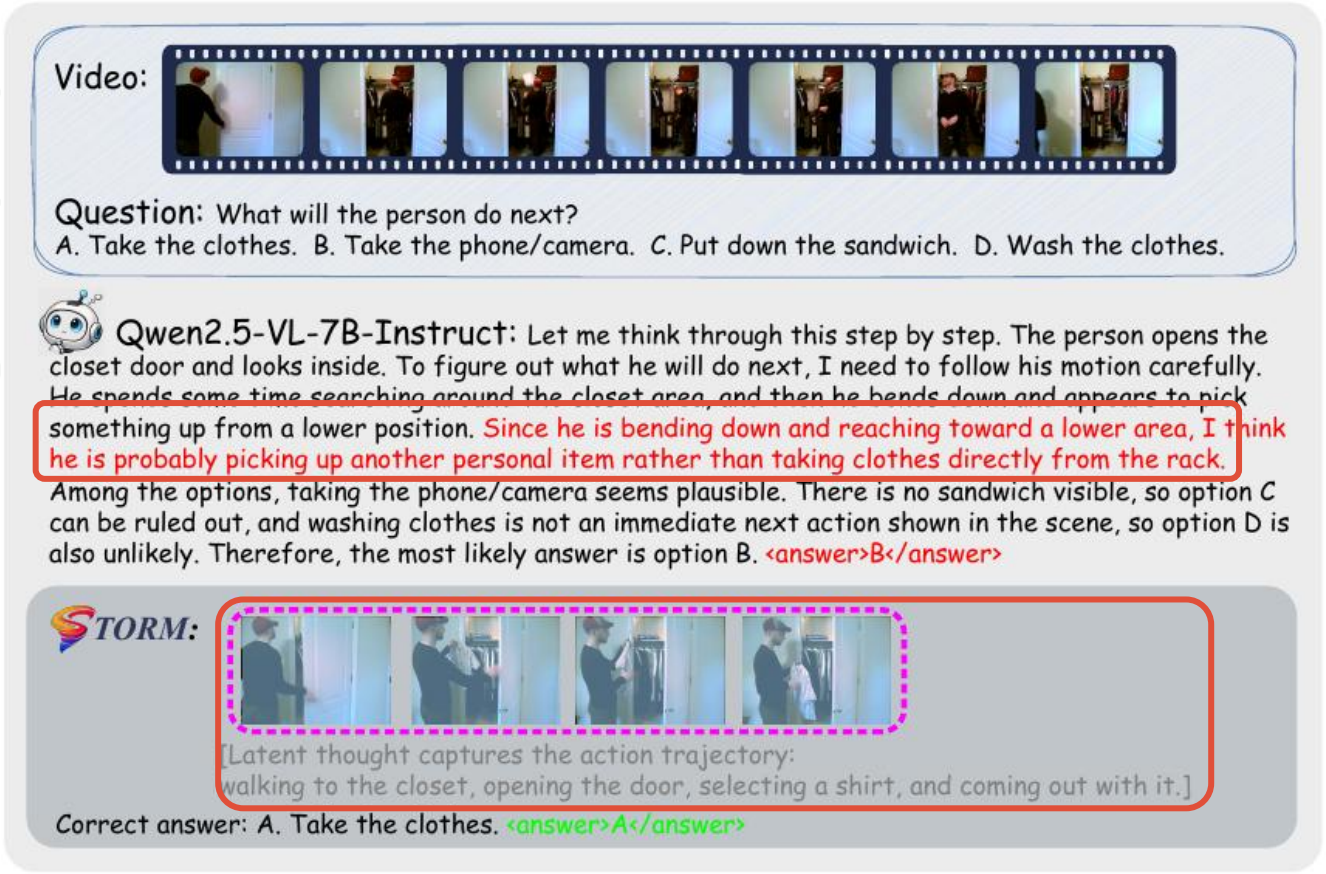
2. **No tools needed:** STORM beats LongVT (tools) on all three and Video-Thinker (video) on two, without looking at the frames again. It is also above Gemini-1.5-Pro on VideoEspresso.

3. **Where Video-Thinker wins:** on Video-Holmes (43.1 vs 37.8), where clues are spread across long suspense films. Looking again can find evidence that 32 frames missed (our reading).

Georgia Tech

Results: case study

Back to the example: text reasoning gets it wrong, STORM gets it right



1. **Text reasoning:** Qwen notices the right details (“opens the closet door”, “bends down”) but then guesses he is picking up “another personal item” and answers B.

2. **STORM:** answers A, take the clothes. Its latent thought captures the action trajectory: walking to the closet, opening the door, selecting a shirt and coming out with it.

The mistake is in connecting the motion over time, not in seeing a single frame.

Caveat: one chosen example, compared with the untuned model rather than SFT. The frame strip is the authors’ illustration; the latents are not decoded into images.

Georgia Tech

Results: model size and cost

It works at 3B and runs about 35× faster than tool-based methods

Smaller model (Table 3)

Model	VideoMME	TempCompass	MMVU
Qwen2.5-VL-3B-Instruct	50.9	55.0	42.0
Qwen2.5-VL-3B + STORM	**55.4**	**67.5**	**58.5**

Scores: accuracy (%), higher is better.

- +4.5 on VideoMME, +12.5 on TempCompass and +16.5 on MMVU with Qwen2.5-VL-3B.
- The method is not tied to one model size.

Caveat: the 3B baseline is the untuned model, not SFT on the same data, so part of this gain is ordinary fine-tuning.

Inference cost on MMVU (Table 4)

Model	Throughput	Latency (s/item)
LongVT	0.065 it/s	15.44
Video-Thinker	0.058 it/s	17.20
STORM	2.14 it/s	**0.47**

Throughput: questions per second.

Latency: seconds per question; lower is better.

- 0.47 s per question, against 15.44 s for LongVT and 17.20 s for Video-Thinker: about 33 to 37 times faster.
- It is also more accurate on MMVU: 65.9 vs 61.4 and 63.4.
- Reason: reasoning runs through a compact set of latent tokens, so the extra cost stays bounded and predictable.

Caveat: 0.47 s is converted from case-study throughput (2.14 it/s); the other two use provided runtime statistics. Making the thought videos adds training-time cost.

Georgia Tech

Ablation: number of latent slots

More latent slots do not help: 8 works best

Latent size	VideoMME	TempCompass	MVBench
4	58.9	71.1	55.0
8	**61.0**	**74.3**	**61.1**
16	52.2	67.3	55.5
256 (no average)	52.9	68.7	54.7

Scores: accuracy (%). Latent size = number of latent slots; 256 (no average) = 256 slots without average pooling.

Compact latent trajectories provide more effective supervision than retaining redundant visual tokens.

Open question: should the number of steps depend on the question? Here it is always 8.

1. **4 slots:** too few; MVBench drops by 6.1.
2. **16 slots, or 256 without adaptive pooling:** VideoMME drops to about 52, below plain SFT (55.4). A badly chosen size is worse than no latent reasoning.

Why: the authors suggest small budgets lack capacity, while large budgets dilute the supervision signal with redundant visual tokens. Single runs, no error bars.

Georgia Tech

Analysis: what do the latents contain?

The latents look like visual features and keep video-specific information

Fig. 5: t-SNE of video (blue), keyframe (yellow) and latent (red)

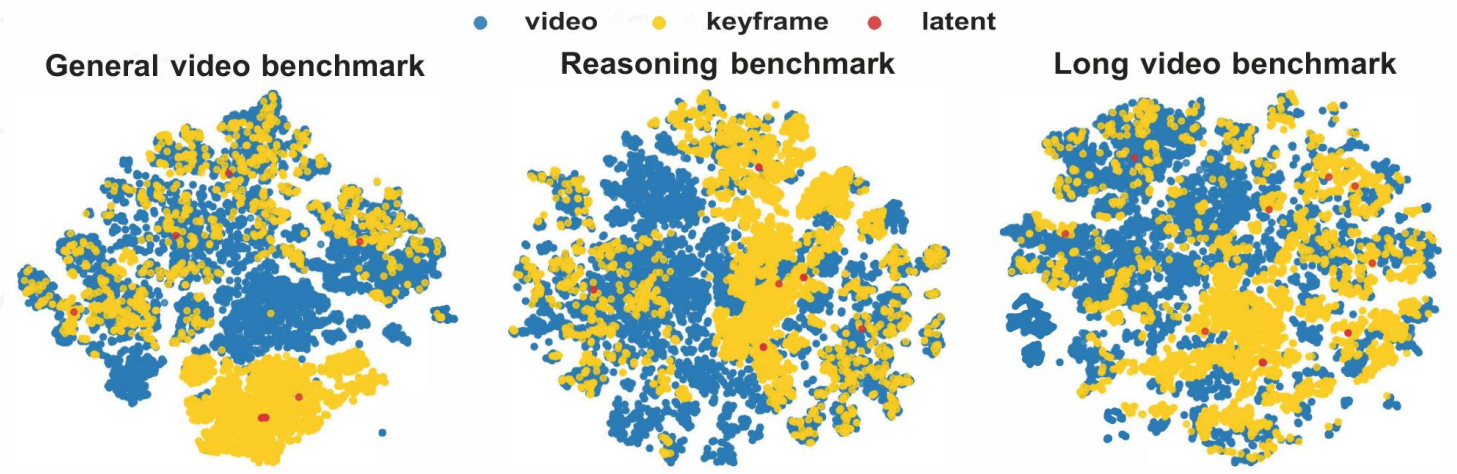
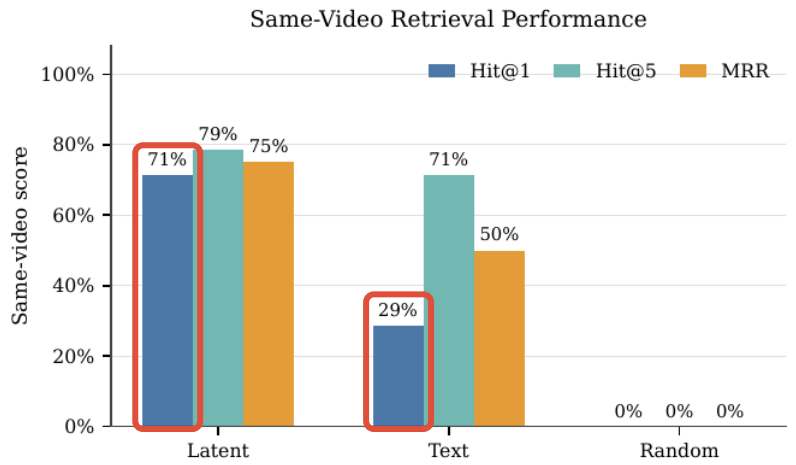


Fig. 5: keyframes form sub-clusters within the video frames, and the latent tokens fall inside or adjacent to the keyframe regions.

Appendix Fig. 6: same-video retrieval



Hit@1: right video ranked first; **Hit@5:** in the top 5; **MRR:** average of 1/rank.

Fig. 6: the latents rank their own video first 71% of the time; text reasoning, 29%.

Caveat: finding the right video shows the latents remember the video, not that they understand its motion. The t-SNE uses 20 examples.

Georgia Tech

Part 3: Comparison, limitations and discussion

Georgia Tech

What Persisted and What Changed?

Same building blocks, different task / supervision / cost

	TimeSformer (2021)	STORM (2026)	Verdict
Input	frame patches, 16 x 16	frame patches, 14 x 14 and merged (Qwen2.5-VL)	**kept**
Temporal attention	in every encoder block	comes from the backbone, STORM adds none	**moved**
Frame selection	fixed-stride sampling, 3 spatial crops at test	answer-aware GPT-4o keyframes in training, 32 uniform frames at test	**changed**
Output	CLS vector -> MLP -> 1 class label	LLM writes the answer after up to 8 latent steps	**changed**
Long videos	96 frames max, average many clips	32 frames, even for 1-hour videos	**still open**
Supervision	human class labels	answer labels plus generated-video features	**changed**
Cost they focus on	training GPU-hours and FLOPs	inference latency per question	**changed**

Georgia Tech

Costs Reported

The costs are not comparable. STORM's offline cost is missing.

	TimeSformer (2021)	STORM (2026)
Training	416 V100-hours on K400	4 GB200 GPUs, 1 epoch per stage, batch size 1; GPU-hours not reported
Offline data	None	GPT-4o keyframes, teacher plans, Wan 2.1 videos for about 5K samples; cost not reported
Inference	0.59 TFLOPs (3 views); 36 min for 20K val videos on 8 V100s	At most 8 latent steps; 0.47s per MMVU item, converted, hardware not reported
Compared against	SlowFast, I3D	LongVT, Video-Thinker

STORM is faster than LongVT and Video-Thinker at test time. Since the cost moved into a data pipeline that was not priced, we cannot say the whole pipeline is cheaper.

Georgia Tech

Limitations: TimeSformer

Most of these showed up in Part 1...

Limitation	Evidence
Motion needs lots of data	On SSv2 it only beats the CNNs with 75% or more of the training data, and is still behind at full data (62.4 vs 65.2)
Training depends heavily on ImageNet pretraining	From scratch: 64.8% on Kinetics-400. With ImageNet-21K initialization: 78.0%
Long video is still processed as clips	On HowTo100M, a 96-frame clip covers 102.4s, but videos span several minutes and require 4 clips on average at test time. 32 → 96 frames improves accuracy only 61.2% → 62.6%.

Georgia Tech

Limitations: STORM

Authors' list vs. what we think is missing

Listed by the authors

- The targets depend on keyframe selection, the teacher plan and the quality of the generated video.
- Evaluation is multiple-choice QA with Qwen2.5-VL backbones only.
- GPT-4o selection and Wan 2.1 generation add offline cost.

Open questions

Stage I targets from...	Tested?
generated thought video (STORM)	yes
real keyframes	no (listed as future work)
whole real video	no
teacher's text plan	no

- All results at 32 frames, Qwen's own report (up to 768) is higher on VideoMME and MVBench
- No run-to-run variance, K = 16 drops VideoMME by 8.8
- Latency hardware not reported, latents hard to inspect

Georgia Tech

Societal Implications

Cheaper video models: good and bad

Benefits

- Lower training and inference cost makes video models usable by smaller groups.
- Useful applications: video QA for blind and low-vision users, education, sports and health analysis.

Risks

- Cheap action recognition and video reasoning make it easier to monitor people.
- Web-sourced training videos (Kinetics and HowTo100M come from YouTube) raise consent and privacy concerns.
- Latent reasoning is hard to audit in areas like driving or healthcare.
- Biases or wrong physics in the teacher and the video generator can be passed on to the model.

Georgia Tech

Conclusion

- TimeSformer: video classification can be done with attention alone. Divided attention makes long clips affordable, and ImageNet pretraining makes training cheap.
- STORM: a few latent steps trained with generated videos beat the same-data SFT baseline on all six benchmarks (by 0.6 to 9.1 points), at low inference cost.
- Still open: whether the generated video is needed, evaluations that really require temporal reasoning, and long videos.

Ways to Use Compute for Video

Different ways of spending compute to handle information across video frames.

Encoder Attention

TimeSformer

Uses all input frames throughout the video encoder. Temporal attention is applied in every block.

Latent Steps

STORM

Uses 32 inputs frames. Then gives the LLM 8 latent reasoning steps per questions.

More Frames

TimeSformer was tested with up to 96 frames. 32 to 96 frames adds only 1.4 points.

STORM's main experiments used 32 frames.

Discussion Question: With a fixed budget, should extra compute go to encoder attention, per-question latent steps, or more frames?

Georgia Tech

Do STORM's Benchmarks Need Time?

TimesSformer measured whether temporal info mattered to benchmarks. STORM didn't.

TimeSformer's control: space-only attention

-1.1

Kinetics-400
76.9 vs 78.0

-22.9

Something-Something-V2
36.6 vs 59.5

STORM's Benchmarks: VideoMME, MVBench, TempCompass, VideoEspresso, Video-Holmes, MMVU

No single-frame or shuffled-frame control. MVBench and TempCompass were designed around temporal questions, but that's just design intent.

Discussion Question: How much of STORM's performance actually depends on temporal information? Do its benchmarks need temporal information?

Discussion Questions

1. With a fixed compute budget, should it be spent on encoder attention, latent reasoning steps, or more frames?

What happens when the video is an hour long? What would work?

2. Is the generated thought video necessary?

What ablation would test it: targets from real keyframes, from the text plan, or from shuffled videos?

3. Do these benchmarks need temporal information?

How much of VideoMME, MVBench or TempCompass can be answered from single frames, and how would you check?

Georgia Tech