Déjà Vu: Efficient Video-Language Query Engine with Learning-based Inter-Frame Computation Reuse

Jinwoo Hwang
KAIST
jwhwang@casys.kaist.ac.kr

Daeun Kim
KAIST
dekim@casys.kaist.ac.kr

Sangyeop Lee
KAIST
sangyeop-lee@casys.kaist.ac.kr

Yoonsung Kim
KAIST
yskim@casys.kaist.ac.kr

Guseul Heo
KAIST
gsheo@casys.kaist.ac.kr

Hojoon Kim
KAIST
hojoonkim@casys.kaist.ac.kr

Yunseok Jeong
KAIST
ysjeong@casys.kaist.ac.kr

Tadiwos Meaza
KAIST
tadiwos@casys.kaist.ac.kr

Eunhyeok Park
POSTECH
eh.park@postech.ac.kr

Jeongseob Ahn
Korea University
jsahn@korea.ac.kr

Jongse Park
KAIST
jspark@casys.kaist.ac.kr

ABSTRACT

Recently, Video-Language Models (VideoLMs) have demonstrated remarkable capabilities, offering significant potential for flexible and powerful video query systems. These models typically rely on Vision Transformers (ViTs), which process video frames individually to extract visual embeddings. However, generating embeddings for large-scale videos requires ViT inferencing across numerous frames, posing a major hurdle to real-world deployment and necessitating solutions for integration into scalable video data management systems. This paper introduces Déjà Vu, a video-language query engine that accelerates ViT-based VideoLMs by reusing computations across consecutive frames. At its core is ReuseViT, a modified ViT model specifically designed for VideoLM tasks, which learns to detect inter-frame reuse opportunities, striking an effective balance between accuracy and reuse. Although ReuseViT significantly reduces computation, these savings do not directly translate into performance gains on GPUs. To overcome this, Déjà Vu integrates memory-compute joint compaction techniques that convert the FLOP savings into tangible performance gains. Evaluations on three VideoLM tasks show that Déjà Vu accelerates embedding generation by up to a 2.64× within a 2% error bound, dramatically enhancing the practicality of VideoLMs for large-scale video analytics.

PVLDB Reference Format:
Jinwoo Hwang, Daeun Kim, Sangyeop Lee, Yoonsung Kim, Guseul Heo, Hojoon Kim, Yunseok Jeong, Tadiwos Meaza, Eunhyeok Park, Jeongseob Ahn, and Jongse Park. Déjà Vu: Efficient Video-Language Query Engine with Learning-based Inter-Frame Computation Reuse. PVLDB, 18(10): 3284 - 3298, 2025.
doi:10.14778/3748191.3748195

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/casys-kaist/DejaVu.

1 INTRODUCTION

Video data is pervasive in today's world, playing a critical role in various applications such as autonomous machines [23, 29, 32], education [28], healthcare [82], security and surveillance [51, 92], e-commerce [118, 120], and many others [56, 67]. Recently, Video-Language Models (VideoLMs) have demonstrated remarkable capabilities, offering significant potential for flexible and powerful video analytics tasks [19, 31, 39, 110, 111, 122]. While these powerful algorithmic advances promise enormous potential, the massive computational demand poses a significant hurdle for the widespread adoption of VideoLMs for video query systems.

The excessive computational demand arises from two primary factors. First, modern VideoLMs heavily rely on Vision Transformers (ViTs) [25] as their core engine for visual feature extraction. ViTs contain hundreds of millions of parameters and require tens of billions of FLOPs for each inference. Second, ViT inference must be performed iteratively across numerous video frames, which can be exceptionally large in number (e.g., a 1-hour video sampled at 2 FPS contains 7,200 frames). Consequently, utilizing VideoLMs for large-scale video analytics becomes nearly infeasible, even with multi-GPU datacenter resources.

Figure 1 provides an overview of video-language query systems, where ViTs process video frames to extract visual embeddings, which are then fed into query-specific operations such as video retrieval, video question answering, and video question grounding. The critical role of ViTs in this process emphasizes the importance of improving ViT inference efficiency to unleash the potential of VideoLMs for building video-language query systems.

doi:10.14778/3748191.3748195

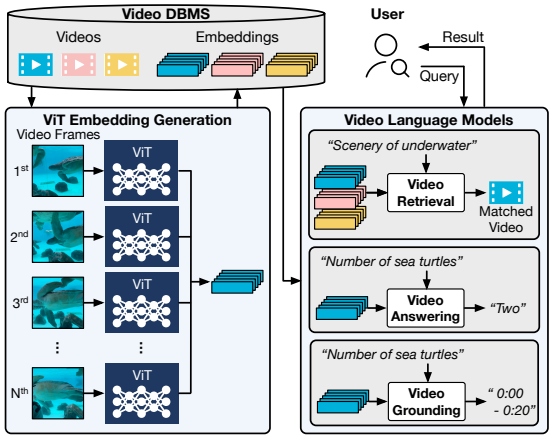

Figure 1: Overview of video language query systems supporting three example VideoLMs. Modern VideoLMs employ vision transformer (ViT) as their core engine.

Predictably, the vital need for efficient ViT inferencing has led to a large body of acceleration work in recent years [8, 12, 14, 24, 27, 47, 60, 62, 65, 76, 78, 94]. Most of these works focus on *intra*-frame computation reuse, targeting inter-token feature similarities since they aim to speed up a single ViT inference run processing one image input. While effective for single images, these methods overlook additional opportunities present in video data that exhibit *inter*-frame similarities across multiple consecutive frames.

Recently, several works [17, 26, 86] have pioneered the exploitation of *inter*-frame computation reuse for ViT inferencing on video applications. However, these works have the following two limitations, which leave them in a position of questionable utility for most practical use cases:

- **Limitation 1:** These methods manually identify reuse opportunities for given models and fixedly enable computation reuse, making it considerably difficult to locate the sweet spot on the accuracy-reuse tradeoff space.
- **Limitation 2:** Although these techniques enable significant FLOP reduction, they do not yield corresponding performance gains, as the remaining computations are sparse and, therefore, inefficient on GPUs.

To address these limitations, this work proposes Déjà Vu, an algorithm-system co-designed query engine that leverages a learning-based approach to automatically identify the computation reuse opportunities and find the sweet spot in the tradeoff space between accuracy and reuse.

The key challenges are (1) to maximize the reuse opportunities without causing significant accuracy loss in VideoLM applications and (2) to effectively translate the FLOPs savings into performance gains. Déjà Vu tackles these challenges by making the following two key contributions, each corresponding to the two limitations:

(1) ***ReuseViT: Reuse-enabling ViT model.*** We develop a modified ViT model, referred to as ReuseViT, that reuses precomputed values for similar tokens between frames. To maximize reuse opportunities, we devise a frame referencing strategy that determines inter-frame reference chains for computation reuse. ReuseViT contains *Decision Layers* that dynamically determine when to reuse computations by processing multiple inputs such as cosine similarity, attention scores, codec metadata, and reference types, allowing it to learn the importance and interactions of these hints automatically. Additionally, it employs *Restoration Layers* that learn to calibrate the reused computation values based on the changes that occurred in the current frame.

To train ReuseViT, we use reparameterization with soft gating via the Gumbel-Softmax mechanism, enabling backpropagation through discrete decisions. This allows the model to mimic hard gating, allowing gradients to flow. Further, we model error propagation by training with grouped frames to mitigate error accumulation from reusing computations across multiple frames.

(2) ***Memory and compute compactions for fast and efficient ReuseViT inferencing.*** To achieve significant performance gains from these algorithmic innovations, we propose three key system-level techniques. (1) We introduce *layer-wise computation scheduling*, which processes computations for multiple frames in a layer-by-layer manner. (2) This approach enables *cached memory compaction*, a memory management technique that clears caches after each layer is completed, allowing for optimized memory usage. Consequently, this increases the batch size during inference, leading to better hardware utilization. (3) Additionally, we implement *sparse computation compaction*, which restructures irregular data patterns caused by computation reuse into dense computations suitable for efficient GPU execution. By consolidating tokens from multiple frames, we create more regular matrices, improving the efficiency of matrix multiplication operations on GPUs, particularly when reuse rates are high.

To demonstrate the effectiveness of Déjà Vu, we perform evaluations using three different VideoLMs – (1) Video Retrieval, (2) Video Question Answering, and (3) Video Question Grounding – on their respective datasets. We observe that Déjà Vu achieves throughput improvement of 1.81×, 2.64×, and 2.54× for the three tasks, respectively, within 2% error bound. Under the same accuracy constraints, the state-of-the-art systems using inter-frame computation reuse offer only 1.32×, 2.08×, 2.20× speedup for the respective tasks.

These significant performance gains, coupled with minimal accuracy loss, underscore the solution's potential to bridge the computational gap, paving the way for VideoLMs to unlock new capabilities.

2 BACKGROUND AND MOTIVATION

2.1 Video Query Processing

A rich body of research explores accelerating large-scale video queries using AI models. We categorize them into three groups.

Task-specific CNN pipelines. Early approaches [11, 41, 42, 48] accelerate queries by training small, query-specific models as approximations for computationally expensive deep learning pipelines. Systems such as NoScope [42] and BlazeIt [41] train lightweight classifiers or regression models tailored to each query, significantly reducing inference costs for known object classes. However, these methods are inherently inflexible, as they require model training or selection for every new query, making them unsuitable for quickly adapting to new or changing requirements.

Task-agnostic proxy embeddings. The second category [4, 43] precomputes a single embedding for video frames to allow fast

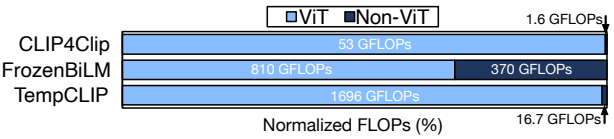

Figure 2: FLOPs breakdown across three VideoLM tasks: video retrieval (CLIP4Clip), video question answering (FrozenBiLM), and video question grounding (TempCLIP).

similarity-based retrieval without query-specific training. Methods such as TASTI [43] cluster frames in a learned embedding space, allowing queries to be answered using indexed embeddings rather than full model inference. While these approaches eliminate the per-query model burden, they may struggle with open-ended queries (i.e., broad or natural-language requests) if novel or uncommon concepts are poorly represented in precomputed embeddings.

Vision-Language Pretrained (VLP) embeddings. More recent approaches [66, 80] provide a more general solution by leveraging VLP models like CLIP [77], which map image and text queries into a shared semantic space. These models support open-ended, natural-language-based video queries without requiring predefined object classes or embedding indexes.

However, recent works [66, 101] illustrate that relying solely on models has limitations in practical video retrieval scenarios. While VLP models exhibit strong generalization capabilities, VLP models often struggle with uncommon or domain-specific queries where training data is sparse. To address these concerns, Déjà Vu leverages VideoLMs, advanced models built on top of VLP architectures that specialize in understanding video context.

2.2 Video-Language Models (VideoLMs)

VideoLMs extend VLP architectures to support complex tasks including video retrieval, video question answering, and video question grounding. For instance, video retrieval involves matching a textual description (e.g., "Show me a clip of scenery of underwater") to the most relevant video in a large corpus. Video question answering asks natural-language questions about the visual content, such as “How many sea turtles appear?”, requiring the model to understand both objects and context.

Computational challenges. Despite their impressive capabilities, the high computational complexity of VideoLMs remains a critical hurdle for potential applications. A major contributor to VideoLM overhead is the embedding generation process using large-scale VLP models [37, 45, 50, 52, 53, 77, 83, 85, 88, 114], which commonly rely on Vision Transformer (ViT) [25] architectures. ViTs often comprise hundreds of millions of parameters and require billions of FLOPs per inference which need to run on numerous video frames. Figure 2 illustrates the computational breakdown for three VideoLMs. Notably, embedding generation via ViT dominates the overall FLOPs, underscoring the need for efficient ViT acceleration.

2.3 Vision Transformer (ViT) Acceleration

Architecture of ViT. Figure 3 illustrates ViT architecture, which adapts the transformer architecture from natural language processing to the vision domain. In ViTs, an input image or video frame is

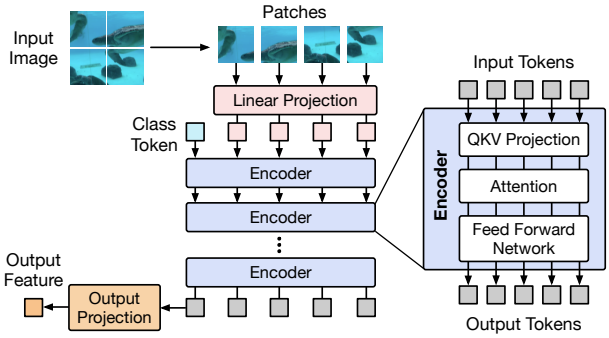

Figure 3: ViT model architecture.

partitioned into fixed-size patches, which are linearly embedded to form a sequence of tokens. A class token (CLS) is appended to this sequence to aggregate global information. The tokens pass through Transformer encoder layers, performing query-key-value projections, multi-head self-attention, and feed-forward network.

Existing acceleration methods. To address the computational demands of ViTs, various methods exploit redundancies in the data. Acceleration techniques focusing on inter-token redundancies reduce computation by pruning less important tokens or merging similar ones within a single image [8, 12, 14, 24, 27, 47, 60, 62, 65, 76, 78, 94]. While these pruning and merging methods effectively skip unnecessary computations within a single image or frame, they ignore the redundancy across frames. In large-scale video analytics, small frame-to-frame changes can be exploited to a much greater extent. Hence, single-image token pruning might still repeat most computations for each of these frames, making it less effective for long videos in which large swaths of patches remain nearly identical over short time intervals.

More recently, other methods have attempted to leverage inter-frame computation reuse in video data [17, 26, 86], offering greater computational savings. Among them, CMC [86] and Eventful Transformer [26] are most relevant to Déjà Vu, as they explicitly reuse partial computations across frames. In contrast, vid-TLDR [17] merges tokens temporally, reducing redundancy through token aggregation rather than direct computation reuse. We provide detailed comparisons against CMC and Eventful Transformer in Section 7.1.

2.4 Limitations of Video ViT Acceleration

While promising, video-targeted acceleration methods exhibit limitations that hinder their practical utility.

Challenges in balancing reuse and accuracy. Existing methods rely on manually designed strategies for computation reuse, which can make it substantially difficult to locate the optimal balance between accuracy and computational savings. For instance, Eventful Transformer [26] requires fixing the number of tokens that reuse computation at each layer. Due to the large search space created by the tens of encoder layers and the variability in video content, it is challenging to predict the consequences of increasing reuse in certain layers. As a result, these methods may lead to a suboptimal tradeoff between computation reuse and accuracy.

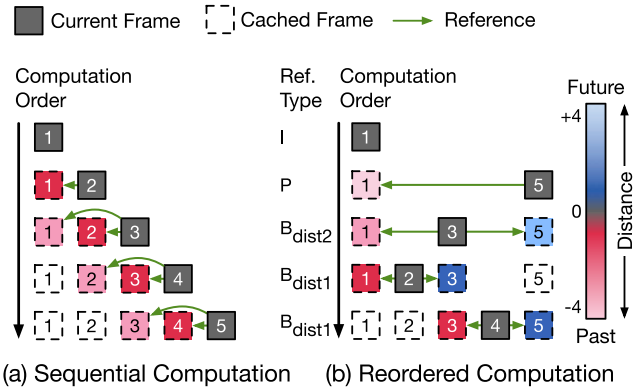

Figure 4: (a) A basic approach where each frame references preceding frames sequentially, (b) Our proposed reordering where frames reference both past and future frames.

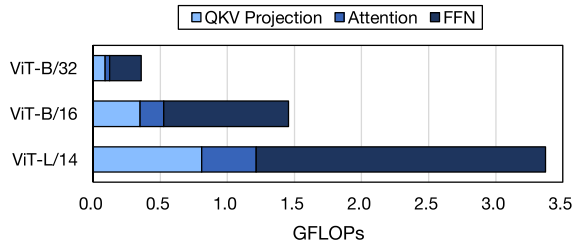

Figure 5: FLOPs breakdown of core computations within a single encoder layer of vision transformers at different scales.

Challenges in realizing FLOP savings as speedups. While reducing computational complexity (FLOPs) is crucial, achieving actual speedups requires addressing runtime factors. Mixing computed and reused tokens leads to sparse computations, causing inefficiencies on GPUs optimized for dense workloads. Our empirical analysis shows that existing video-targeted ViT acceleration techniques [26, 86] deliver limited speedups due to overheads in memory usage, data movement, and hardware utilization (see Section 7). Furthermore, some prior acceleration works [86] rely on specialized hardware accelerators, which are not readily available in standard commodity systems, limiting the accessibility.

These limitations motivate us to design a customized ViT model, dubbed ReuseViT, which can automatically identify computation reuse opportunities in video data, while carefully balancing the accuracy-reuse tradeoff. Additionally, we introduce memory-compute joint compaction techniques to effectively convert ReuseViT's FLOP savings into tangible performance gains.

3 MODEL ARCHITECTURE OF REUSEVIT

To address computational challenges in ViT-based VLMs, we propose ReuseViT, a model that automatically identifies safe (accuracy-preserving) computation reuse opportunities to accelerate inference. ReuseViT is designed to maximally reduce redundant computations while maintaining high accuracy by leveraging inter-frame similarities in video data.

3.1 Frame Selection for Computation Reuse

Determining which frames to reference is critical to maximizing computation reuse opportunities. In video data, especially at low frame rates common in VideoLM tasks, frame contents can drift significantly over time. Hence, ReuseViT reorders frames to allow referencing both the past and future frames, potentially boosting the likelihood of finding similar content.

Sequential frame computation. Figure 4(a) illustrates a basic strategy where each new frame references one or more preceding frames. Because nearer frames generally exhibit higher temporal similarity, referencing them provides reuse benefits. However, including frames further back tends to yield diminishing returns due to overlapping patches that offer little additional information.

Reordered frame computation. To harness both past and future frames, ReuseViT processes frames in an out-of-order fashion, as shown in Figure 4(b). We categorize frames into four types, I-frames (computed independently), P-frames (referencing a previous frame), B_{dist2}-frames (referencing frames two steps away), and B_{dist1}-frames (referencing immediate neighbors), reflecting terminology akin to video codecs. Following a pattern of $I \rightarrow (P \rightarrow B_{dist2} \rightarrow B_{dist1} \rightarrow B_{dist1}) \rightarrow \ldots$ helps capture bidirectional temporal redundancies that purely sequential schemes may overlook. By referencing both directions, we increase the potential for reuse and reduce the frequency of full computations. When choosing references, ReuseViT evaluates the quality and temporal distance of candidate frames to preserve accuracy without incurring excessive overhead. Subsequent sections detail this decision-making process.

3.2 Layer Selection for Reuse

Deciding which layers within the ViT architecture are suitable for computation reuse is crucial for maximizing efficiency gains without compromising performance.

FLOPs breakdown of ViT layers. Figure 5 presents the FLOPs breakdown analysis results for a transformer encoder layer of three different ViT variants. The results suggest that the query-key-value (QKV) projection and the feed-forward network (FFN) are the primary consumers of computational resources in ViTs. Unlike large language models (LLMs), where the self-attention layer is the main bottleneck due to longer sequence lengths, ViTs process shorter sequences (around 256 patches). Thus, in ViTs, the self-attention layer contributes much less to the overall computational cost relative to the QKV projection and FFN layers.

Computation patterns of layers. Besides computational cost, we consider computation patterns of each layer. The QKV projection and FFN operations are applied independently to each token, without inter-token dependencies, suiting them for computation reuse. In contrast, the self-attention operation involves interactions among all tokens, making reuse more challenging and potentially affecting model output. Given these considerations, we focus on reusing computations in the QKV projection and FFN layers, which are the most computationally intensive and token-independent.

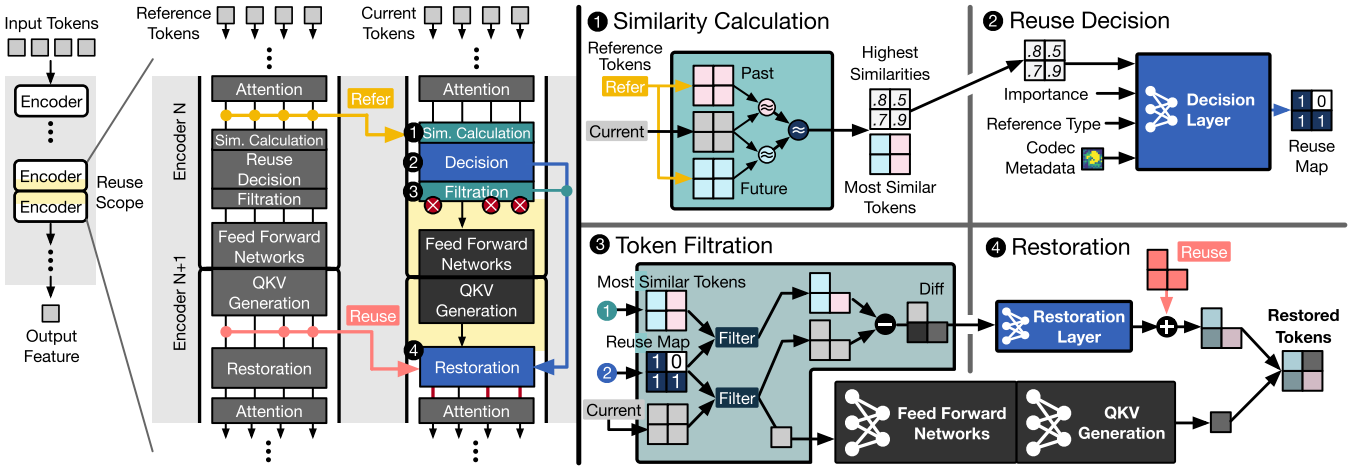

Figure 6: Overview of ReuseViT model operations.

3.3 Criteria for Computations Reuse

An essential aspect of ReuseViT is determining when to reuse computations for specific tokens. We introduce two key components: (1) a *Decision Layer* that decides whether to reuse computations, and (2) a *Restoration Layer* that calibrates reused computations to align with the current frame's context. Figure 6 depicts the overall architecture of ReuseViT.

Decision layer. The decision layer identifies tokens suitable for computation reuse by assessing multiple informative cues for each token. By integrating various inputs, the layer makes nuanced decisions without relying on hand-crafted rules. The decision layer takes as input a concatenation of the following features:

- **Similarity Measure (s):** We compute the cosine similarity between the current token and corresponding tokens in the reference frames. A higher similarity suggests that the token content has remained unchanged, making it a candidate for reuse:

$$s_i = \max\left(\cos\left(T_i^{\text{cur}}, T_i^{\text{past}}\right),\ \cos\left(T_i^{\text{cur}}, T_i^{\text{future}}\right)\right) \tag{1}$$

  Here, $T_i^{\text{cur}} \in \mathbb{R}^D$ represents the i-th token of the current frame, while T_i^{past} and T_i^{future} are the corresponding tokens from the previous and future reference frames. This strategy follows prior work on token merging using similarity metrics [8, 14, 24].
- **Token Importance (t):** We use the attention weights from the class token to estimate each token's importance. Tokens with higher attention are more critical and may require fresh computation. This method is consistent with existing approaches that prune tokens based on class-token attention [14, 27, 62, 108].
- **Reference Type (r):** Reference frame type (e.g., I-frame, P-frame, B_{dist1}, B_{dist2}) offers insight into the reference's temporal proximity, aiding assessment of reused computation reliability
- **Codec Metadata (c):** Metadata from video codecs offers block-wise hints about spatio-temporal redundancies, providing insights into areas where video content undergoes motion or structural changes. These metadata signals, such as motion vectors and residuals, have been leveraged in prior research [36, 115, 124] to guide computational optimizations.

By combining these inputs, the decision layer, implemented as a simple two-layer MLP, makes informed decisions.

$$v_i = \text{concat}(s_i, t_i, r_i, c_i), \quad i = 1, \ldots, N \tag{2}$$

$$d_i = \text{MLP}_{\text{decision}}(v_i) \tag{3}$$

$$\mathcal{M}_i = \begin{cases} 1, & \text{if } d_i > 0 \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \ldots, N \tag{4}$$

where $\mathcal{M}_i \in 0, 1$ indicates whether to reuse (1) or recompute (0) the computations for token i. By combining similarity scores with token importance and codec metadata, the decision layer captures both visual and semantic cues, striking a balance between accuracy and reuse. This data-driven approach enables the model to adapt effectively to diverse video scenarios without manual tuning of thresholds or heuristics.

Token filtration. Based on the reuse map $\mathcal{M}$, tokens are partitioned into recompute tokens C and reuse tokens R.

$$C = T_i^{\text{cur}} \mid \mathcal{M}_i = 0, \quad \text{(recompute tokens)} \tag{5}$$

$$R = T_i^{\text{cur}} \mid \mathcal{M}_i = 1, \quad \text{(reuse tokens)} \tag{6}$$

Recompute tokens C undergo the standard feed forward network and query-key-value projection.

$$\tilde{C}^{\text{cur}} = \text{QKV}\left(\text{FFN}\left(C^{\text{cur}}\right)\right) \tag{7}$$

Restoration layer. To adjust for discrepancies between reused computations and the current frame, we introduce a restoration layer that calibrates the reused tokens. For each reuse token i, we compute the difference between the tokens.

$$\Delta R_i = R_i^{\text{cur}} - R_i^{\text{ref}} \tag{8}$$

This difference captures changes between the current and reference tokens. The restoration layer processes ΔR_i using a small two-layer MLP with a hidden size of 128, which is significantly smaller than the hidden size of 1,024 used in the FFN layers. This design choice reduces computational overhead while efficiently obtaining a calibration value for the change in the input.

$$\hat{R}_i^{\text{cur}} = \tilde{R}_i^{\text{ref}} + \text{MLP}_{\text{restoration}}(\Delta R_i) \tag{9}$$

where $\tilde{R}_i^{\text{ref}}$ is the reused computation from the reference frame, and $\hat{R}_i^{\text{cur}}$ is the calibrated token for the current frame. This calibration effectively improves accuracy with minimal computational overhead (4% compared to standard QKV and FFN computation).

Token Reconstruction We reconstruct the full set of tokens $\hat{T}^{\text{cur}}$ by combining the recomputed tokens $\tilde{C}^{\text{cur}}$ and the reused tokens $\hat{R}^{\text{cur}}$ according to the reuse map $\mathcal{M}$.

$$\hat{T}_i^{\text{cur}} = \begin{cases} \tilde{C}_j^{\text{cur}}, & \text{if } \mathcal{M}_i = 0 \\ \hat{R}_k^{\text{cur}}, & \text{if } \mathcal{M}_i = 1 \end{cases} \tag{10}$$

where i indexes the original token sequence, and j and k index the recomputed and calibrated reused tokens, respectively. This process maintains the original order of tokens, preserving positional relationships within the model.

The low-complexity design of the decision and restoration modules (e.g., small MLPs) ensures modest overhead relative to full inference, yielding significant net savings in large-scale deployments despite the small cost of restoring reused tokens.

4 TRAINING REUSEVIT

ReuseViT's efficiency requires training its decision and restoration layers for a given ViT model and dataset, keeping the pre-trained ViT model frozen. This section outlines these training techniques, focusing on handling discrete reuse decisions and modeling error accumulation through grouped frame training.

4.1 Handling Discrete Reuse Decisions

A key training challenge for ReuseViT is handling discrete reuse decisions, which obstruct gradient flow through the gating mechanism during backpropagation. Specifically, the binary decisions to reuse or recompute computations for tokens hinder gradient-based optimization essential for training.

To enable gradient flow through the gating mechanism, we approximate the hard binary decisions with continuous values during training. This soft gating mechanism allows the decision layer to be trained end-to-end using backpropagation without the need for specialized techniques. By replacing the hard decisions in Equation 4 with continuous approximations, we ensure that gradients can propagate through the layer, facilitating effective learning.

$$\mathcal{M}_{\text{soft}} = \text{GumbelSoftmax}\left(\text{MLP}_{\text{decision}}(v)\right) \approx [0,1]^N \tag{11}$$

$$\hat{T}_{\text{soft}}^{\text{cur}} = \mathcal{M}_{\text{soft}} \odot \hat{T}^{\text{cur}} + (1 - \mathcal{M}_{\text{soft}}) \odot \tilde{T}^{\text{cur}} \tag{12}$$

Here, $\odot$ denotes element-wise multiplication, $\mathcal{M}_{\text{soft}}$ is the soft mask for all N tokens, $\hat{T}^{\text{ref}}$ represents the reused computations from reference frames, and $\tilde{T}^{\text{cur}}$ represents the recomputed tokens of the current frame. We found that gradually lowering the Gumbel-Softmax temperature over the course of training helps the model transition smoothly from a fully soft gating regime toward more selective token reuse. This annealing avoids sudden spikes in the reuse mask while preserving stable gradient flow.

At inference time, we apply the hard decisions to realize the computational savings achieved by ReuseViT.

4.2 Loss Function for Accuracy and Efficiency

To train ReuseViT effectively, we formulate a loss function balancing model accuracy and computational efficiency, aiming for performance gains without degradation in prediction quality.

Training objective. Our objective is to ensure that the approximated features produced by ReuseViT remain close to the original features generated by the unmodified ViT model. By preserving the original embeddings, we maintain the model's predictive performance across different tasks without the need for task-specific fine-tuning. This self-supervised approach allows the model to generalize effectively in practical systems.

Similarity loss. We use the cosine similarity metric to encourage the approximated features to remain close to the original features of the VLP model.

$$\mathcal{L}_{\text{sim}} = 1 - \cos\left(Z^{\text{cur}}, \hat{Z}^{\text{cur}}\right) \tag{13}$$

Here, Z^{cur} is the original final feature vector, and $\hat{Z}^{\text{cur}}$ is the corresponding feature vector after computation reuse and calibration. However, using only the similarity loss may discourage computation reuse, as the model can minimize loss by recomputing all tokens, negating efficiency gains.

Reuse loss. To incentivize reuse, we introduce a reuse loss.

$$\mathcal{L}_{\text{reuse}} = \frac{1}{LN} \sum_{l=1}^{L} \sum_{i=1}^{N} \mathcal{M}_{l,i} \tag{14}$$

This loss term represents the average reuse rate across all tokens and layers. By maximizing $\mathcal{L}_{\text{reuse}}$, we encourage the model to reuse more computations, promoting computational efficiency.

Combined loss function. The overall loss function combines the similarity loss and the reuse loss:

$$\mathcal{L} = \mathcal{L}_{\text{sim}} + \alpha \cdot \max\left(0, R_{\text{target}} - \mathcal{L}_{\text{reuse}}\right) \tag{15}$$

Here, α is a weighting hyperparameter that balances the trade-off between accuracy (controlled by $\mathcal{L}_{\text{sim}}$) and computational efficiency (encouraged by $\mathcal{L}_{\text{reuse}}$), and R_{target} is the target reuse rate. The max ensures that the penalty is applied only when the reuse rate falls below the target, thus incentivizing the model to meet R_{target}. The combined loss function effectively prevents the model from trivially minimizing error through complete recomputation.

4.3 Grouped Frame Training for Robustness

Errors accumulate over time when computations are reused across multiple frames. To ensure robustness, we model error accumulation during training by adopting a grouped frame training strategy.

Modeling error accumulation. During training, we adjust our loss functions to account for grouped frames. Specifically, we compute the losses over groups of frames by averaging the similarity loss $\mathcal{L}_{\text{sim}}$ and the reuse loss $\mathcal{L}_{\text{reuse}}$ across all frames in the group. By considering the average losses over the group, the model learns to handle error accumulation effectively over sequences of frames, rather than optimizing for individual frames in isolation. This approach ensures that the model's performance and reuse behavior are optimized not just for single frames, but for sequences where errors may propagate due to computation reuse.

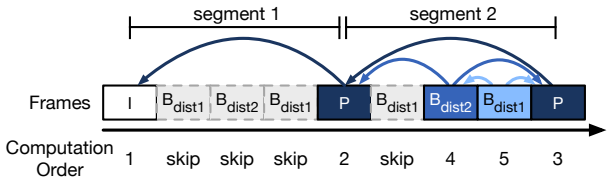

Figure 7: Illustration of our frame-grouping strategy during training. We treat each "I" or "P" frame as a segment boundary and form groups allowing the model to learn error accumulation.

Efficient grouping strategy. To balance modeling accuracy and training efficiency, we design training sequences that model error accumulation over a manageable number of frames. We use I-frames and P-frames as segment boundaries, dividing frames into segments (e.g., Segment 1 and Segment 2 in Figure 7). Frames within a segment can influence each other through computation reuse, while intermediate frames across segments remain independent, allowing for efficient modeling of longer sequences.

By constructing sequences like I-P-P, corresponding to frames 1, 5, and 9, we simulate error propagation over longer intervals without including intermediate frames (2–4 and 6–8). Within the last segment, we also include frames with remaining B_{dist2} and B_{dist1} reference types from the previous segment, so that ReuseViT is able to learn all reference types.

Empirical results indicate that grouping six frames in the pattern 1-5-9-13-11-12 strikes an effective balance between accuracy and training efficiency, allowing the model to account for error accumulation over significant temporal spans while managing computational costs. During training, we also experimented with varying group sizes and found that 6 frame groupings gave us the best empirical tradeoff between training time and error-accumulation robustness. Larger groupings occasionally made optimization less stable, while smaller ones did not fully capture longer-range drifts.

5 COMPACTION TECHNIQUES IN DÉJÀ VU

In this section, we present the compaction methods used to translate the FLOPs reduction achieved by ReuseViT into tangible performance gains. These methods address practical challenges in implementing ReuseViT efficiently on GPU hardware, where parallelism is crucial to achieving high throughput.

5.1 Layer-Wise Scheduling

A key enabler of our compaction methods is layer-wise scheduling. While the target problem differs, our scheduling scheme is inspired by FlexGen [84], a prior work that aims to deploy a large language model (LLM) serving on a single GPU. FlexGen proposes grouping multiple batches for LLM inference and iterates the batches in a round-robin to execute their computations layer-by-layer. That way, FlexGen can amortize the I/O costs across multiple batches, required for model parameters and KV cache.

On the other hand, Déjà Vu has different objectives since it aims to accelerate ViT encoders, which have smaller model sizes than

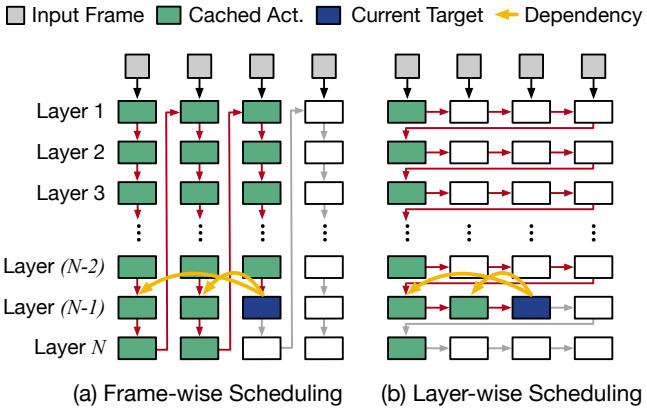

Figure 8: Comparison of (a) frame-wise scheduling and (b) layer-wise scheduling for caching intermediate activations.

LLMs and do not need KV cache. Déjà Vu employs the layer-by-layer computation scheme to mitigate the memory bloating and GPU unfriendliness problems caused by ReuseViT's computation reuse. Instead of interleaving multiple batches as in FlexGen, Déjà Vu batches multiple frames together and processes the same layer across these frames in a layer-by-layer manner within the same segment. Building upon this layer-wise scheduling, Déjà Vu employs Cached Memory Compaction and Sparse Computation Compaction to more efficiently utilize GPU memory and compute resources.

5.2 Cached Memory Compaction

A major overhead in computation reuse arises from storing cached activations, or intermediate outputs computed for reference frames, which subsequent frames rely on.

Layer-wise memory compaction. By exploiting our layer-wise scheduling and the frame-referencing structure of our model (with P-frames acting as boundaries), we compact memory usage at each layer within a segment. Frames other than P-frames do not affect future segments, so their intermediate activations can be discarded once their current segment is processed.

Figure 8 illustrates the key difference between the conventional frame-wise approach. As illustrated in Figure 8(a), in the conventional frame-wise approach, each frame runs through all layers sequentially. Intermediate outputs must remain in memory until all computations for that frame are complete. In contrast, in our layer-wise scheduling (Figure 8(b)), the same layer is applied to all frames in the batch, and once a layer is complete for each frame, any unneeded activations are immediately freed. This staggered approach substantially lowers memory overhead, enables larger batch sizes with high GPU utilization, and facilitates deployment of large-scale VideoLMs under limited memory budgets.

5.3 Sparse Computation Compaction

ReuseViT's dynamic and often sparse computation patterns pose difficulties for GPUs, which are optimized for dense, regular workloads. To address this, we use stream compaction [7], gathering active (i.e., non-reused) tokens into contiguous memory regions

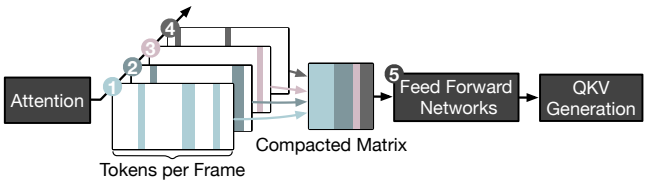

Figure 9: Sparse Computation Compaction.

so sparse computations can be converted into dense forms better suited to GPU acceleration.

Layer-wise stream compaction. We implement GPU kernels for stream compaction across different frames within the same segment, as shown in Figure 9. By accumulating active tokens from multiple frames before moving on to the next feed forward network, we form well-shaped matrices amenable to GPU acceleration. Once the batched computation is complete, we scatter the results back into their original positions in each frame's token sequence, preserving correctness for the next layer. This gather-scatter strategy ensures that even when some frames have only a few active tokens, they can still benefit from dense GPU operations.

To minimize overhead further, we prioritize CUDA-based implementations for the gather-compute-scatter procedure, avoiding frequent CPU-GPU synchronization. Such an approach not only speeds up the compaction process itself but also reduces latency spikes that might occur from excessive kernel launches or data transfers between CPU and GPU.

6 WORKFLOW OF DÉJÀ VU

6.1 Overview of Query Processing

On a query, Déjà Vu returns cached embeddings if available; otherwise it generates them with ReuseViT and stores the result.

Embedding storage overhead. Storing frame-level VLP embeddings introduces minimal storage overhead. For example, extracting embeddings at 2 FPS from a Full-HD H.264 video (~625KB/s) results in approximately 4KB/s of data (assuming 1024-dimensional FP16 embeddings at ~2KB each). This constitutes merely 0.64% of the compressed video size, with further footprint reductions achievable via advanced floating-point compression techniques [57, 61, 73].

6.2 Offline Preparation

Before handling real queries, Déjà Vu trains the decision and restoration modules for a given ViT backbone. During training, we use the self-supervised objective, which combines a similarity loss (to match the original ViT outputs) and a reuse-based loss (to encourage high reuse rates). The primary user-adjustable hyperparameter is the target reuse rate (R_{target}), defined in Equation 15. Training with different values of R_{target} allows users to navigate various points on the accuracy-speed tradeoff curve. Alternatively, one can specify a target cosine similarity, whereby the model learns to maximize the reuse rate while conforming to the accuracy threshold defined by that similarity value.

We also employ grouped-frame training to model how errors might accumulate. This improves robustness, as the model learns to balance computational savings with accuracy preservation across different video types. During training, Only the two lightweight modules are trained, while the pre-trained ViT backbone remains frozen and convergence typically occurs within an hour. Such design significantly simplifies deployment by removing the need to store, transfer, or manage multiple versions of large model weights.

6.3 Online Inference

Once deployed, Déjà Vu employs layerwise scheduling and memory compaction to translate the FLOP savings from ReuseViT into practical GPU throughput. While ReuseViT reduces the token-level computational load, effective scheduling and compaction routines ensure these savings materialize in actual GPU execution time, which is a key requirement for large-scale query systems.

To maximize GPU utilization, the system processes multiple videos in a single batch. If only one long video is available, it is split into multiple, uniformly sized segments. For each segment, four consecutive frames are collected via a small queue and fed into ReuseViT, enabling each inference call to process four times the usual batch size in frames. Batching multiple segments is crucial for efficiently utilizing GPU resources, especially when high reuse rates substantially reduce computation per video.

Frame reordering. Frame reordering is handled within the model's forward pass, which reconstructs results in the correct sequence. Consequently, the outer framework does not need to manage any frame shuffling or realignment. Although this staggered approach works well for offline analytics, where latency is less critical, it can also be adapted to streaming scenarios. In a streaming setting, Déjà Vu buffers four frames before computation, introducing a queuing latency of about two seconds at 2 FPS. For real-time applications with strict latency requirements, frame reordering can be disabled to minimize latency, albeit at the cost of lower reuse rates.

Breaking error propagation. Although grouping frames largely confines errors, prolonged reuse can still amplify minor inaccuracies. To counter this, Déjà Vu periodically inserts a fully recomputed I frame. For instance, if the system resets every twentieth frame, it prevents indefinite error buildup without incurring excessive recomputation costs. This method keeps the overhead below 5% while maintaining accuracy over long sequences.

7 EVALUATION

7.1 Methodology

End tasks. To evaluate Déjà Vu's effectiveness, we select three distinct Video-Language Models (VideoLMs), each for a different task. All models use CLIP [77] (ViT) embeddings as their visual backbone. We focus on:

- **Video retrieval.** We use CLIP4Clip [64] on the MSR-VTT [106] dataset, which contains 10,000 text-annotated video clips of 10 to 30 seconds in length, and report top-5 recall.
- **Video question answering.** For video QA, we use FrozenBiLM [111] on How2QA [55], which has 44,007 QA pairs on 60-second clips, and report multiple-choice accuracy.
- **Video Question Grounding.** For video question grounding, which requires localizing a temporal segment for an answer, we

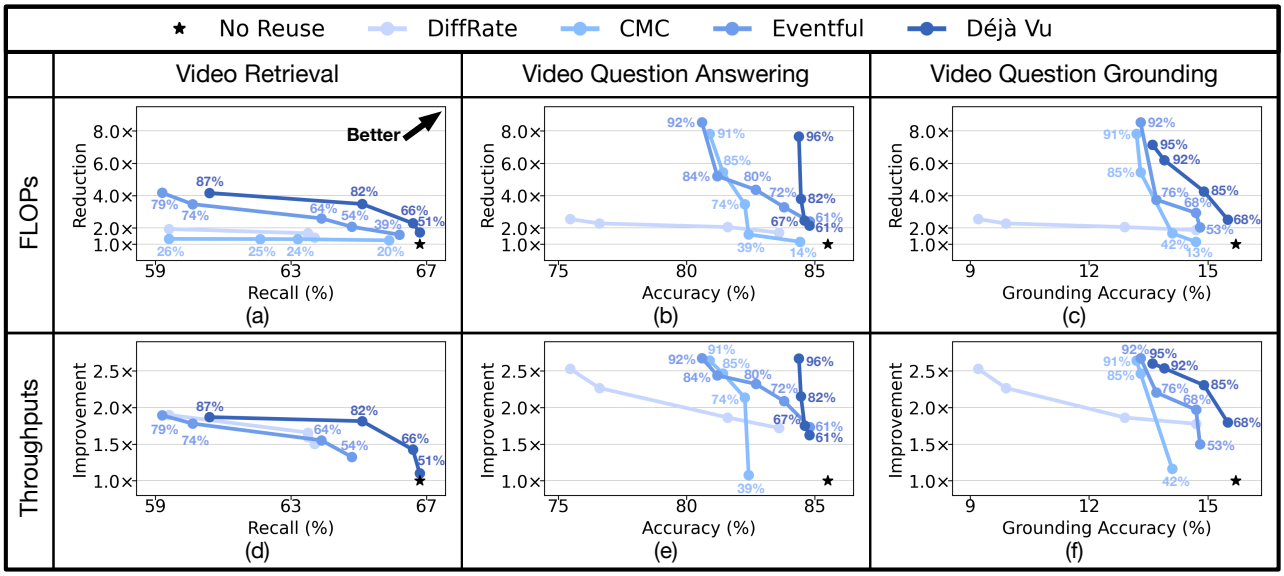

Figure 10: Tradeoff space of baselines and Déjà Vu. Y-axis for FLOPs reduction and throughput are normalized to original model with no reuse.

use TempCLIP [104] on NExT-GQA [104], with videos averaging 45 seconds. We report GQA@Accuracy, measuring correct answers with correct temporal overlap.

For a fair comparison, all methods sample frames at 2 FPS, aligning with common practice [3, 10, 64, 74, 99, 104, 111, 117]. We use standard dataset splits for training and evaluation for all tasks.

Baselines. As no prior methods directly accelerate VideoLMs, our baselines comprise conceptually similar temporal ViT acceleration approaches and a representative image-based method for contrast. We evaluate three acceleration methods:

- **CMC.** CMC [86] identifies tokens for reuse via a fixed mean squared error (MSE) threshold. Originally designed for action recognition, it can leverage specialized video codec hardware for its similarity computations.
- **Eventful Transformer.** Eventful Transformer [26] employs a simple but rigid static policy, recomputing a fixed number of tokens per layer regardless of video content. While simple and efficient, this non-adaptive strategy can cause error accumulation in dynamic scenes.
- **DiffRate.** DiffRate [14] is an image-based baseline, adapted for VLP models, that combines token pruning and merging. It automatically determines the accuracy-efficiency tradeoff through backpropagation. Though originally evaluated on image classification, we adapted the policy for VLP models.

CMC and Eventful Transformer propose kernels to reuse computations in the attention layer. In our experiments, these kernels did not yield speedups because the attention layers have relatively small computational costs. Therefore, for both methods, we focus on their core reuse strategies without attention-layer optimizations.

Implementation details. We run all experiments on a server with two 16-core Intel Xeon Gold 6226R CPUs (2.9 GHz), 192 GB of RAM, and an NVIDIA RTX 3090 GPU (24 GB of GDDR6 memory). The software environment includes Ubuntu 24.04 as the operating system, CUDA 12.1, cuDNN 8.9.0, and PyTorch 2.1.0.

7.2 Tradeoff between FLOPs and Accuracy

Figure 10 (a)–(c) shows tradeoffs between FLOPs reduction for embedding generation (y-axis) and accuracy (x-axis) for three tasks: video retrieval, video QA, and video question grounding. The x-axis shows the accuracy metric, and the star-shaped marker near the bottom right of each graph indicates the unapproximated model's accuracy. The y-axis shows the achieved FLOPs reduction for embedding generation. A point higher on the y-axis indicates greater FLOPs reduction at the same accuracy level, while a point further to the right indicates better accuracy. Thus, methods near the top-right corner represent more favorable efficiency-accuracy tradeoffs. Next to each marker, we show the computation reuse rate, except for DiffRate, which applies token pruning and merging instead.

Notably, techniques that exploit temporal redundancy (CMC, Eventful Transformer, and Déjà Vu) consistently achieve higher FLOPs reduction than the image-focused DiffRate, whose pruning/merging targets only spatial redundancy within individual frames. Among these temporal-based baselines, Déjà Vu attains the best tradeoff, typically matching or exceeding each method's accuracy at a lower computational cost.

7.3 Tradeoff between Throughput and Accuracy

Figures 10 (d)-(f) show throughput improvement for the embedding generation on the y-axis. While Déjà Vu and DiffRate are measured on a real GPU system, CMC and Eventful Transformer lack GPU-optimized implementations, so we interpolate their throughput assuming our compaction techniques applied. Here, we normalize throughput to that of the original model, and only configurations

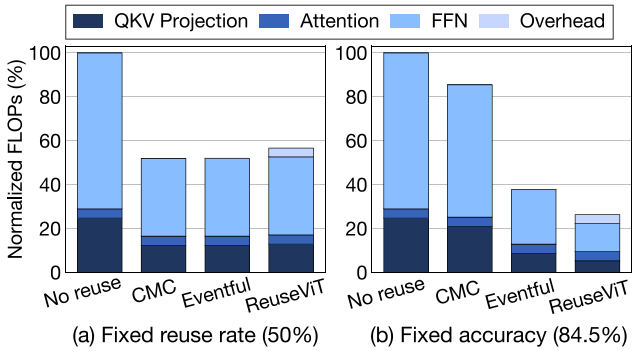

Figure 11: Breakdown of FLOPs on video retrieval. (a) FLOPs breakdown at fixed reuse rate. (b) Reuse rates comparison at equal accuracy.

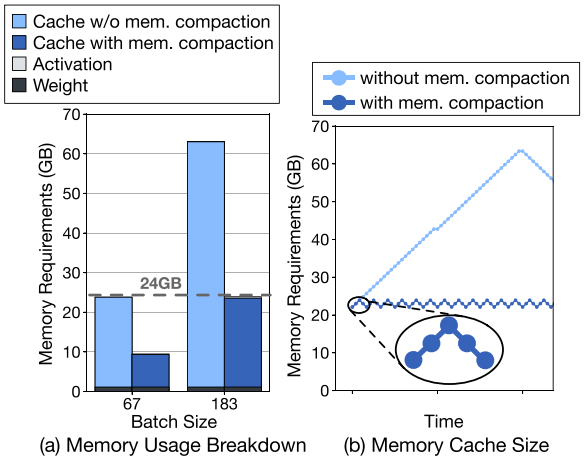

Figure 12: Comparison of peak memory usage with and without cached memory compaction. The dotted line shows memory capacity of RTX 3090 GPU.

that yield an actual improvement are shown, and each marker can be matched to those in the FLOPs plots via the reuse rates.

Consistent with previous FLOPs-based results, Déjà Vu achieves the highest throughput-accuracy tradeoff. For instance, Déjà Vu reaches speedups of up to 1.81× for video retrieval, 2.64× for video question answering, and 2.54× for video question grounding. By contrast, Eventful Transformer, the next-best baseline, achieves speedups of 1.32×, 2.08×, and 2.20×, respectively.

While baselines exploiting inter-frame redundancies remain competitive for VideoLM tasks, Déjà Vu's tailored optimizations achieve superior throughput-accuracy tradeoffs. Even incremental improvements are crucial, for instance, a 1–2% accuracy gain in leading video QA benchmarks typically requires several months of intensive research [110, 111], and modest speedups can significantly reduce costs and query latency in large-scale deployments.

Interestingly, DiffRate configurations become more competitive for throughput, as the high complexity of computation reuse can hinder its realization. However, Déjà Vu's careful optimizations still ensure higher overall speedups.

7.4 Breakdown Analysis

At the same reuse rate, Déjà Vu has slightly lower performance than CMC or Eventful Transformer. For instance, in Figure 10 (e) when reuse is at 61%, Déjà Vu shows a marginally lower throughput. Figure 11 (a) shows a FLOPs breakdown of these methods at an identical reuse rate on the video retrieval task. In this scenario, the reuse rate is intentionally fixed across all methods to isolate and highlight algorithmic overhead. The main additional overhead in ReuseViT arises from the decision layers and restoration layers, which add about 4% to normalized FLOPs.

However, Figure 11 (b) shows that at the same accuracy on video question answering, Déjà Vu achieves a significantly higher reuse rate. Thus, at equivalent accuracy levels, Déjà Vu effectively compensates for the additional overhead by adaptively calibrating reused computations, enabling it to reuse more tokens without sacrificing accuracy. Hence, while Déjà Vu incurs some overhead at the same reuse rate, its adaptive mechanisms enable higher reuse where accuracy must be maintained, which ultimately yields stronger end-to-end speedups when matching target accuracy levels.

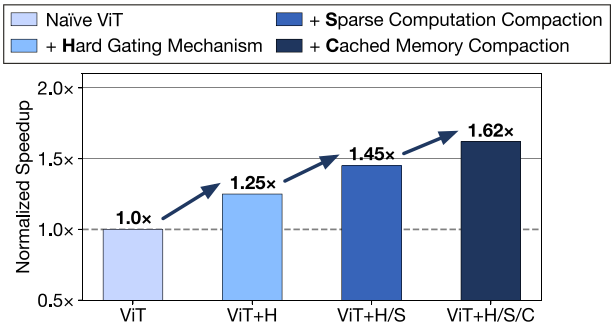

Figure 13: Ablation study of Déjà Vu's inference optimization techniques on the video QA task.

7.5 Memory Overhead

Figure 12(a) compares the peak GPU memory usage of ReuseViT with and without cached memory compaction, across different batch sizes. The blue bars show memory usage when intermediate computations are cached without compaction, while the green bars include compaction. Without compaction, the model processes only 67 batches before running out of memory, but compaction raises this limit to 183, improving GPU utilization and throughput. By keeping memory usage within GPU limits, practitioners can deploy larger batch sizes or process multiple video streams concurrently, which in turn boosts overall system throughput. This is particularly valuable in server-based or cloud environments where memory is a critical and often expensive resource.

Figure 12(b) illustrates how the cache size changes during inference. Without compaction, the cache grows linearly until all layers are processed. With compaction, cached memory for processed frames is freed after certain layers, creating a sawtooth pattern and significantly reducing peak memory usage.

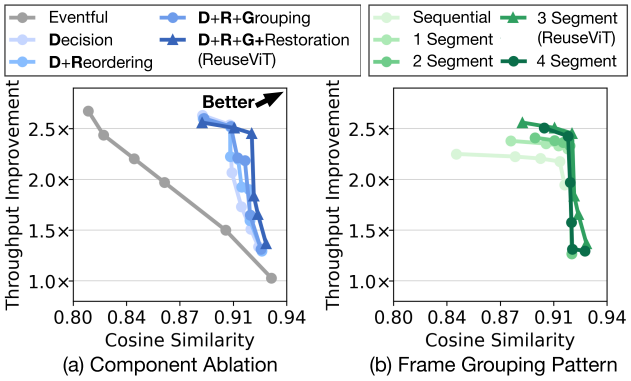

Figure 14: Ablation study of ReuseViT's design choices on the NeXT-GQA dataset. The full ReuseViT configuration is denoted by a triangular marker.

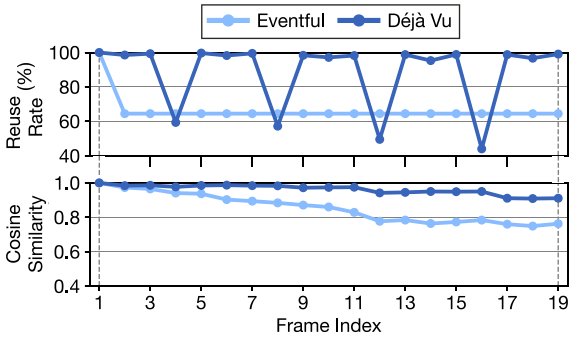

Figure 15: Comparison of Déjà Vu's adaptive reuse strategy versus Eventful Transformer's static strategy on a video segment from How2QA over time.

7.6 Ablation Study for Inference Speedup

We conduct an ablation study of inference speedup with a ReuseViT configuration achieving a 61% reuse rate on the video question answering task to show how each technique contributes to Déjà Vu's performance. Figure 13 highlights the throughput gains from each step. The first bar indicates reusing computation with hard gating alone yields a 1.25× speedup. Layer-wise sparse computation compaction then increases the speedup to 1.45× by shaping operands to better utilize the GPU kernel. Finally, layer-wise memory compaction enables larger batches, improving parallelization and raising the total speedup to 1.62×. These results confirm Déjà Vu's optimizations translate FLOPs savings into real throughput gains. Each component offers complementary benefits, gating reduces compute, sparse-compaction improves GPU workload efficiency, and memory compaction allows larger batches, unlocking significantly more performance together than individually.

7.7 Ablation Study for Design Choices

To evaluate the contributions of ReuseViT's core components, we perform an ablation study on the NeXT-GQA dataset. Figure 14 assesses visual embedding quality, measured by cosine similarity to the original embedding without reuse, and throughput improvement normalized against the original ViT. Higher values on both axes indicate better efficiency and accuracy.

Figure 14(a) presents incremental component comparisons. Starting from a configuration with only the adaptive decision layer, we sequentially incorporate frame reordering, grouped frame training, and finally, the restoration layer. Notably, even the initial decision layer alone already provides substantial improvements over the static Eventful Transformer baseline, highlighting the significant advantage of adaptive reuse decisions. Each subsequent addition further enhances the tradeoff between efficiency and accuracy. Although the restoration layer introduces a slight computational overhead reducing peak throughput, it notably improves embedding quality at higher reuse rates.

Figure 14(b) examines various frame grouping strategies. Compared to a sequential processing baseline, all reordered groupings achieve improved tradeoffs. Performance consistently increases with segment lengths up to three segments, while extending to four segments yields slightly inferior results. Thus, a three-segment grouping strategy effectively balances computational efficiency and visual embedding quality.

7.8 Adaptability to Video Content

We compare the adaptability of Eventful Transformer and Déjà Vu on a video segments from How2QA, with both methods reaching 84.5% accuracy. Figure 15 depicts the fluctuations in computation reuse rates (top) and cosine similarities (bottom) over time.

Initially, both methods fully compute the first frame, then adopt their reuse strategies. Eventful Transformer maintains a fixed reuse rate, leading to a more pronounced decline in cosine similarity. In contrast, Déjà Vu adapts when error begins to accumulate, lowering the reuse rate to protect accuracy and raising it when frames are more similar. Moreover, Déjà Vu learns to adjust reuse behavior based on reference frame types (I- or P-frames), further mitigating error accumulation. By dynamically balancing reuse and recomputation, Déjà Vu preserves higher feature similarity over time and achieves a superior tradeoff between computational efficiency and model accuracy compared to methods with fixed reuse strategies.

8 LIMITATIONS AND FUTURE WORK

While Déjà Vu largely improves computational efficiency for video-language tasks, several limitations remain, motivating future work.

Attention Layer Reuse. Déjà Vu currently targets only FFN and QKV layers. These layers dominate computational costs at standard resolutions, for instance, 257 tokens for ViT-L/14. However, as resolution and token counts rise, such as 577 tokens (CLIP ViT-L/14 at 336px), 785 tokens (DINO ViT-B/8 [13]), and 1370 tokens (DINOv2 ViT-G/14 at 518px [68]), attention layers form an increasingly substantial fraction of total FLOPs, reaching up to 23.5%. Addressing this requires exploring attention reuse strategies or leveraging sparse attention kernels optimized for GPUs, like those proposed in Eventful Transformer. Another approach involves specialized hardware accelerators, such as those introduced in CMC.

Broader task generalization. This work evaluates Déjà Vu specifically on retrieval, question answering, and grounding tasks. Adapting our learned reuse strategies to other video tasks, such as action recognition or object detection, would involve adjustments to task-specific model architectures and setups. While this adaptation is conceptually straightforward, its practical exploration remains an open and promising direction for future research.

Adaptive periodic refresh and online adaptation. Another limitation relates to the adaptive periodic refresh strategy within Déjà Vu. This strategy could benefit from additional refinement for improved handling of long-form video inputs and enhanced robustness across various video domains. Furthermore, future research might explore online fine-tuning methods designed to dynamically adjust reuse policies during inference. Such adjustments would enhance adaptability and efficiency within evolving real-world scenarios.

9 RELATED WORK

CNN-based pipelines for known classes. Early systems for known object categories often relied on trained CNNs or proxy models. For instance, NoScope [42] developed lightweight cascades for static-camera queries. Focus [35] and BlazeIt [41] built approximate indexes that were refined by heavier models. Other approaches included Tahoma's [2] use of cascades with input transformations and MIRIS [5] targeted multi-object tracking. Some systems precomputed object tracks, like OTIF [6], or used detector ensembles, as seen in FiGO [11]. More recent contributions include DoveDB [105] unifying training and tracklet ingestion, Seiden [4] leveraging high-accuracy models with sampling to avoid proxies, and InQuest [81] paring cheap proxies with oracles for streaming analytics. While cost-effective for predefined categories, these systems struggle with zero-shot or open-ended queries without retraining.

Resource management and domain-specific analytics. Another line of work focuses on scheduling or adapting analytics pipelines at scale. Scanner [75], for example, exploits domain metadata for raw video decoding and dataflow. VideoStorm [116] and Chameleon [38] schedule model cascades for concurrent streams. ODIN [89] monitors domain drift to retrain models for known classes under new conditions. Further techniques involve adaptive query reordering by Hydro [40], using geospatial metadata to skip frames as in Spatialyze [46], or integrating caching to accelerate iterative queries, demonstrated by VIVA [44].

Open-vocabulary indexing and domain-agnostic exploration. To address fixed label sets, several systems enable querying arbitrary classes without retraining. Panorama [119] learns a generic feature space. TASTI [43] reuses a learned semantic index for new labels, and Boggart [1] provides a model-agnostic index. Other systems support interactive exploration, such as EVA [109]. VOCALExplore [22] trains detectors on-the-fly via active labeling, while SeeSaw [66] leverages CLIP embeddings for zero-shot retrieval. LVS [49] approximates embeddings from cached ones, and SketchQL [101] uses sketch-based queries. Domain-agnostic storage solutions like VStore [107], TASM [21], and VSS [33], along with decoding-stage acceleration in CoVA [36] and TVM [123], also enhance query speed.

Modified architectures and acceleration for ViTs. Following ViT's [25] success, many subsequent architectures aim to reduce computation using techniques such as local attention or convolutions [18, 30, 34, 63, 70, 96, 97]. For video processing, temporal mixing layers are common additions [16, 54, 79, 91, 95, 100, 112], though no standard has emerged. Standard ViTs are accelerated via token pruning (removing less important tokens) [12, 14, 27, 47, 62, 78] or token merging (combining similar tokens) [8, 12, 14, 17, 58, 65]. Pruning techniques can be fixed-rate [78] or adaptive [27], sometimes combined with merging strategies [47, 62]. Merging approaches, exemplified by ToMe [8] and TCFormer [65], downsample tokens, with recent extensions to temporal attention [17, 58]. Some methods integrate both pruning and merging [12, 14] or propose specialized hardware [24, 60, 76, 94, 113].

Computation reuse and decision making. Reusing computations across frames offers gains beyond intra-frame redundancy. Eventful Transformer [26] and CMC [86] use manual cross-frame reuse decisions. In contrast, Déjà Vu employs a fully learnable mechanism for improved efficiency. Prior work on temporal redundancy in CNNs includes incremental activation updates [9, 69], use of motion vectors [87], delta updates [71, 72], or discarding redundant frames and regions [20, 59, 90, 93, 103]. Several studies also learn reuse decisions end-to-end. These include methods for dynamic fusion [121] or layer skipping [15, 98, 102]. While these predominantly target CNNs, Déjà Vu adapts this concept to VideoLMs by integrating a learnable reuse mechanism in ViTs for significant speedups while preserving accuracy.

10 CONCLUSION

This paper addresses the computational challenges posed by ViT in the context of large-scale video analytics exploiting VideoLMs. We introduce Déjà Vu, a novel algorithm-system co-designed solution, recognizing the untapped potential of exploiting a learning approach to enable inter-frame computation reuse. Central to Déjà Vu is the ReuseViT model, innovatively adapted for VideoLM tasks. This model is specifically designed for enabling computation reuse opportunities in ViT inferencing by judiciously determining the reuse targets and restoring the reuse-introduced errors automatically. We also build a ReuseViT-based Déjà Vu system, which effectively utilizes GPUs for high throughput ViT inferencing for VideoLM queries. Our evaluation shows that Déjà Vu achieves a substantial performance boost while marginally compromising the accuracy of VideoLM tasks. These findings suggest that this work represents a practical initial step toward actualizing the full potential of VideoLMs for video-language query systems.

ACKNOWLEDGMENTS

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT), through the Information Technology Research Center (ITRC) program (No. IITP-2025-RS-2020-II201795), the Development of ML compiler framework for on-device AI (No. RS-2024-00459797), and the Graduate School of Artificial Intelligence Semiconductor program (No. IITP-2025-RS-2023-00256472).

REFERENCES

[1] Neil Agarwal and Ravi Netravali. 2023. Boggart: Towards General-Purpose acceleration of retrospective video analytics. In *NSDI*.

[2] Michael R Anderson, Michael Cafarella, German Ros, and Thomas F Wenisch. 2019. Physical representation-based predicate optimization for a visual analytics database. In *ICDE*. IEEE, 1466–1477.

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. Vivit: A video vision transformer. In *ICCV*. 6836–6846.

[4] Jaeho Bang, Gaurav Tarlok Kakkar, Pramod Chunduri, Subrata Mitra, and Joy Arulraj. 2023. Seiden: Revisiting query processing in video database systems. *VLDB* 16, 9 (2023).

[5] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. 2020. Miris: Fast object track queries in video. In *SIGMOD*. 1907–1921.

[6] Favyen Bastani and Samuel Madden. 2022. OTIF: Efficient tracker pre-processing over large video datasets. In *sigmod*. 2091–2104.

[7] Markus Billeter, Ola Olsson, and Ulf Assarsson. 2009. Efficient stream compaction on wide SIMD many-core architectures. In *HPG*. 159–166.

[8] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token merging: Your vit but faster. *ICLR* (2023).

[9] Mark Buckler, Philip Bedoukian, Suren Jayasuriya, and Adrian Sampson. 2018. EVA^2: Exploiting Temporal Redundancy in Live Computer Vision. In *ISCA*. 533–546.

[10] Adrian Bulat, Juan Manuel Perez Rua, Swathikiran Sudhakaran, Brais Martinez, and Georgios Tzimiropoulos. 2021. Space-time mixing attention for video transformer. *NeurIPS* 34 (2021), 19594–19607.

[11] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2022. Figo: Fine-grained query optimization in video analytics. In *SIGMOD*. 559–572.

[12] Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. PuMer: Pruning and Merging Tokens for Efficient Vision Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 12890–12903.

[13] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *ICCV*. 9650–9660.

[14] Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. 2023. Diffrate: Differentiable compression rate for efficient vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 17164–17174.

[15] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. 2020. Dynamic convolution: Attention over convolution kernels. In *CVPR*. 11030–11039.

[16] Feng Cheng, Xizi Wang, Jie Lei, David Crandall, Mohit Bansal, and Gedas Bertasius. 2023. VindLU: A Recipe for Effective Video-and-Language Pretraining. In *CVPR*.

[17] Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J Kim. 2024. vid-TLDR: Training Free Token merging for Light-weight Video Transformer. In *CVPR*. 18771–18781.

[18] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. 2021. Twins: Revisiting the design of spatial attention in vision transformers. *NIPS* 34 (2021), 9355–9366.

[19] Anthony Colas, Seokhwan Kim, Franck Dernoncourt, Siddhesh Gupte, Zhe Wang, and Doo Soon Kim. 2020. TutorialVQA: Question Answering Dataset for Tutorial Videos. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*. 5450–5455.

[20] Xiangxiang Dai, Peng Yang, Xinyu Zhang, Zhewei Dai, and Li Yu. 2022. RESPIRE: Reducing Spatial–Temporal Redundancy for Efficient Edge-Based Industrial Video Analytics. *IEEE Transactions on Industrial Informatics* 18, 12 (2022), 9324–9334.

[21] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. 2021. TASM: A Tile-Based Storage Manager for Video Analytics. In *ICDE*. 1775–1786.

[22] Maureen Daum, Enhao Zhang, Dong He, Stephen Mussmann, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. 2023. Vocalexplore: Pay-as-you-go video data exploration and model building. *VLDB* 16, 13 (2023), 4188–4201.

[23] Shuchisnigdha Deb, Christopher R Hudson, Daniel W Carruth, and Darren Frey. 2018. Pedestrians Receptivity in Autonomous Vehicles: Exploring a Video-based Assessment. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*.

[24] Peiyan Dong, Mengshu Sun, Alec Lu, Yanyue Xie, Kenneth Liu, Zhenglun Kong, Xin Meng, Zhengang Li, Xue Lin, Zhenman Fang, et al. 2023. Heatvit: Hardware-efficient adaptive token pruning for vision transformers. In *HPCA*. IEEE.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

[26] Matthew Dutson, Yin Li, and Mohit Gupta. 2023. Eventful transformers: leveraging temporal redundancy in vision transformers. In *ICCV*. 16911–16923.

[27] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. 2022. Adaptive token sampling for efficient vision transformers. In *ECCV*. Springer.

[28] Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Jerry Chun-Wei Lin. 2023. Large language models in education: Vision and opportunities. In *BigData*.

[29] Ujwalla Gawande, Kamal Hajari, and Yogesh Golhar. 2020. Pedestrian Detection and Tracking in Video Surveillance Vystem: Issues, Comprehensive Review, and Challenges. *Recent Trends in Computational Intelligence* (2020), 1–24.

[30] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. 2021. Levit: a vision transformer in convnet's clothing for faster inference. In *ICCV*.

[31] Deepak Gupta and Dina Demner-Fushman. 2022. Overview of the MedVidQA 2022 shared task on medical video question-answering. In *Proceedings of the 21st Workshop on Biomedical Language Processing*.

[32] Ramyad Hadidi, Jiashen Cao, Matthew Woodward, Michael S Ryoo, and Hyesoon Kim. 2018. Distributed Perception by Collaborative Robots. *IEEE Robotics and Automation Letters* 3, 4 (2018), 3709–3716.

[33] Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. 2021. VSS: A Storage System for Video Analytics. In *SIGMOD*.

[34] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. 2021. Rethinking spatial dimensions of vision transformers. In *ICCV*.

[35] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *OSDI*.

[36] Jinwoo Hwang, Minsu Kim, Daeun Kim, Seungho Nam, Yoonsung Kim, Dohee Kim, Hardik Sharma, and Jongse Park. 2022. CoVA: Exploiting Compressed-Domain Analysis to Accelerate Video Analytics. In *ATC*.

[37] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*.

[38] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodík, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: Scalable Adaptation of Video Analytics. In *SIGCOMM*.

[39] Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie. 2022. Prompting visual-language models for efficient video understanding. In *ECCV*. Springer.

[40] Gaurav Tarlok Kakkar, Jiashen Cao, Aubhro Sengupta, Joy Arulraj, and Hyesoon Kim. 2024. Hydro: Adaptive Query Processing of ML Queries. *arXiv preprint arXiv:2403.14902* (2024).

[41] Daniel Kang, Peter Bailis, and Matei Zaharia. PVLDB. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. In *2019*.

[42] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. In *PVLDB*.

[43] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2021. Task-agnostic Indexes for Deep Learning-based Queries over Unstructured Data. In *PVLDB*.

[44] Daniel Kang, Francisco Romero, Peter D Bailis, Christos Kozyrakis, and Matei Zaharia. 2022. VIVA: An End-to-End System for Interactive Video Analytics.. In *CIDR*.

[45] Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *ICML*.

[46] Chanwut Kittivorawong, Yongming Ge, Yousef Helal, and Alvin Cheung. 2024. Spatialyze: A Geospatial Video Analytics System with Spatial-Aware Optimizations. *VLDB* (2024).

[47] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Mengshu Sun, Wei Niu, Xuan Shen, Geng Yuan, Bin Ren, Minghai Qin, et al. 2022. Spvit: Enabling faster vision transformers via soft token pruning. *ECCV* (2022).

[48] Ziliang Lai, Chris Liu, Chenxia Han, Pengfei Zhang, Eric Lo, and Ben Kao. 2022. Everest: A top-k deep video analytics system. In *SIGMOD*. 2357–2360.

[49] Yunghee Lee and Jongse Park. 2024. LVS: A Learned Video Storage for Fast and Efficient Video Understanding. In *CVPRW*.

[50] Dongxu Li, Junnan Li, Hongdong Li, Juan Carlos Niebles, and Steven CH Hoi. 2022. Align and prompt: Video-and-language pre-training with entity prompts. In *CVPR*.

[51] Da Li, Zhang Zhang, Kai Yu, Kaiqi Huang, and Tieniu Tan. 2019. ISEE: an intelligent scene exploration and evaluation platform for large-scale visual surveillance. *TPDS* (2019).

[52] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*.

[53] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *NIPS* (2021).

[54] Kunchang Li, Yali Wang, Yizhuo Li, Yi Wang, Yinan He, Limin Wang, and Yu Qiao. 2023. Unmasked teacher: Towards training-efficient video foundation models. In *ICCV*.

[55] Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. 2020. HERO: Hierarchical Encoder for Video+ Language Omni-representation Pre-training. In *EMNLP*.

[56] Mengze Li, Tianbao Wang, Haoyu Zhang, Shengyu Zhang, Zhou Zhao, Wenqiao Zhang, Jiaxu Miao, Shiliang Pu, and Fei Wu. 2022. Hero: Hierarchical spatio-temporal reasoning with contrastive action correspondence for end-to-end video object grounding. In *MM*.

[57] Ruiyuan Li, Zheng Li, Yi Wu, Chao Chen, and Yu Zheng. 2023. Elf: Erasing-based lossless floating-point compression. *VLDB* 16, 7 (2023), 1763–1776.

[58] Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. 2024. VidToMe: Video Token Merging for Zero-Shot Video Editing. *CVPR* (2024).

[59] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *SIGCOMM*.

[60] Zheng Li, Soroush Ghodrati, Amir Yazdanbakhsh, Hadi Esmaeilzadeh, and Mingu Kang. 2022. Accelerating Attention through Gradient-Based Learned Runtime Pruning. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 902–915. https://doi.org/10.1145/3470496.3527423

[61] Panagiotis Liakos, Katia Papakonstantinopoulou, and Yannis Kotidis. 2022. Chimp: efficient lossless floating point compression for time series databases. *VLDB* 15, 11 (2022), 3058–3070.

[62] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *ICLR*.

[63] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*.

[64] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. 2022. CLIP4Clip: An empirical study of CLIP for end to end video clip retrieval and captioning. *Neurocomputing* (2022).

[65] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. 2021. Token Pooling in Vision Transformers. arXiv:2110.03860 [cs.CV]

[66] Oscar Moll, Manuel Favela, Samuel Madden, Vijay Gadepally, and Michael Cafarella. 2023. SeeSaw: interactive ad-hoc search over image databases. *PACMMOD* (2023).

[67] Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. 2017. Marioqa: Answering questions by watching gameplay videos. In *ICCV*.

[68] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. 2024. DINOv2: Learning Robust Visual Features without Supervision. *TMLR* (2024).

[69] Bowen Pan, Rameswar Panda, Camilo Fosco, Chung-Ching Lin, Alex Andonian, Yue Meng, Kate Saenko, Aude Oliva, and Rogerio Feris. 2021. Video Adaptive Redundancy Reduction. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[70] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. 2022. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *ECCV*.

[71] Mathias Parger, Chengcheng Tang, Thomas Neff, Christopher D Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. 2023. MotionDeltaCNN: Sparse CNN Inference of Frame Differences in Moving Camera Videos with Spherical Buffers and Padded Convolutions. In *ICCV*.

[72] Mathias Parger, Chengcheng Tang, Christopher D Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. 2022. DeltaCNN: End-to-end CNN inference of sparse frame differences in videos. In *CVPR*.

[73] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. 2015. Gorilla: A fast, scalable, in-memory time series database. *VLDB* (2015).

[74] AJ Piergiovanni, Weicheng Kuo, and Anelia Angelova. 2023. Rethinking video vits: Sparse video tubes for joint image and video learning. In *CVPR*.

[75] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. 2018. Scanner: Efficient video analysis at scale. *TOG* (2018).

[76] Yubin Qin, Yang Wang, Dazheng Deng, Zhiren Zhao, Xiaolong Yang, Leibo Liu, Shaojun Wei, Yang Hu, and Shouyi Yin. 2023. FACT: FFN-attention Co-optimized transformer architecture with eager correlation prediction. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–14.

[77] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*. PMLR, 8748–8763.

[78] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *NIPS* 34 (2021).

[79] Shuhuai Ren, Sishuo Chen, Shicheng Li, Xu Sun, and Lu Hou. 2023. TESTA: Temporal-Spatial Token Aggregation for Long-form Video-Language Understanding. In *EMNLP*.

[80] Francisco Romero, Caleb Winston, Johann Hauswald, Matei Zaharia, and Christos Kozyrakis. 2023. Zelda: Video analytics using vision-language models. *arXiv preprint arXiv:2305.03785* (2023).

[81] Matthew Russo, Tatsunori Hashimoto, Daniel Kang, Yi Sun, and Matei Zaharia. 2023. Accelerating Aggregation Queries on Unstructured Streams of Data. *VLDB* (2023).

[82] Edward Sanderson and Bogdan J Matuszewski. 2022. FCN-transformer feature fusion for polyp segmentation. In *Annual conference on medical image understanding and analysis*. Springer.

[83] Sheng Shen, Chunyuan Li, Xiaowei Hu, Yujia Xie, Jianwei Yang, Pengchuan Zhang, Zhe Gan, Lijuan Wang, Lu Yuan, Ce Liu, et al. 2022. K-lite: Learning transferable visual models with external knowledge. *NIPS* (2022).

[84] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *ICML*.

[85] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2022. Flava: A foundational language and vision alignment model. In *CVPR*.

[86] Zhuoran Song, Chunyu Qi, Fangxin Liu, Naifeng Jing, and Xiaoyao Liang. 2024. CMC: Video Transformer Acceleration via CODEC Assisted Matrix Condensing. In *ASPLOS*.

[87] Zhuoran Song, Feiyang Wu, Xueyuan Liu, Jing Ke, Naifeng Jing, and Xiaoyao Liang. 2020. Vr-dann: Real-time video recognition via decoder-assisted neural network acceleration. In *MICRO*. IEEE, 698–710.

[88] Yuchong Sun, Hongwei Xue, Ruihua Song, Bei Liu, Huan Yang, and Jianlong Fu. 2022. Long-Form Video-Language Pre-Training with Multimodal Temporal Contrastive Learning. *NIPS* (2022).

[89] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. 2020. ODIN: automated drift detection and recovery in video analytics. *VLDB* (2020).

[90] Raúl Taranco, José-María Arnau, and Antonio González. 2023. δLTA: Decoupling Camera Sampling from Processing to Avoid Redundant Computations in the Vision Pipeline. In *MICRO*.

[91] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. 2022. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS* (2022).

[92] Andeep S Toor, Harry Wechsler, and Michele Nappi. 2019. Biometric surveillance using visual question answering. *Pattern Recognition Letters* (2019).

[93] Toshiaki Wakatsuki, Sekitoshi Kanai, and Yasuhiro Fujiwara. 2021. Accelerate Inference of CNNs for Video Analysis While Preserving Exactness Exploiting Activation Sparsity. In *Proceedings of Machine Learning and Systems 3 (MLSys)*.

[94] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *HPCA*.

[95] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. 2023. Videomae v2: Scaling video masked autoencoders with dual masking. In *CVPR*.

[96] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*.

[97] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2022. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media* 8, 3 (2022).

[98] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. 2018. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*.

[99] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. 2024. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *ICLR* (2024).

[100] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. 2024. Internvideo: General video foundation models via generative and discriminative learning. *ECCV* (2024).

[101] Renzhi Wu, Pramod Chunduri, Ali Payani, Xu Chu, Joy Arulraj, and Kexin Rong. 2024. SketchQL: Video Moment Querying with a Visual Query Interface. *Proceedings of the ACM on Management of Data* (2024).

[102] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. 2018. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*.

[103] Zuxuan Wu, Caiming Xiong, Yu-Gang Jiang, and Larry S Davis. 2019. Liteeval: A coarse-to-fine framework for resource efficient video recognition. *NeurIPS* (2019).

[104] Junbin Xiao, Angela Yao, Yicong Li, and Tat-Seng Chua. 2024. Can i trust your answer? visually grounded video question answering. In *CVPR*.

[105] Ziyang Xiao, Dongxiang Zhang, Zepeng Li, Sai Wu, Kian-Lee Tan, and Gang Chen. 2023. DoveDB: A Declarative and Low-Latency Video Database. *VLDB* (2023).

[106] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*.

[107] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. Vstore: A data store for analytics on large videos. In *Proceedings of the Fourteenth EuroSys Conference 2019*. 1–17.

[108] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. 2022. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *AAAI*.

[109] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Umakishore Ramachandran. 2022. EVA: A symbolic approach to accelerating exploratory video analytics with materialized views. In *SIGMOD*.

[110] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2021. Just ask: Learning to answer questions from millions of narrated videos. In *ICCV*.

[111] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022. Zero-Shot Video Question Answering via Frozen Bidirectional Language Models. In *NIPS*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.).

[112] Shusheng Yang, Xinggang Wang, Yu Li, Yuxin Fang, Jiemin Fang, Wenyu Liu, Xun Zhao, and Ying Shan. 2022. Temporally efficient vision transformer for video instance segmentation. In *CVPR*.

[113] Seungjae Yoo, Hangyeol Kim, and Joo-Young Kim. 2024. AdapTiV: Sign-Similarity Based Image-Adaptive Token Merging for Vision Transformer Acceleration. In *MICRO*. IEEE.

[114] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. 2021. Florence: A new foundation model for computer vision. *arXiv* (2021).

[115] Mu Yuan, Lan Zhang, Xuanke You, and Xiang-Yang Li. 2023. PacketGame: Multi-Stream Packet Gating for Concurrent Video Inference at Scale. In *SIGCOMM*.

[116] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *NSDI*.

[117] Hang Zhang, Xin Li, and Lidong Bing. 2023. Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding. In *EMNLP Demo*.

[118] Shengyu Zhang, Ziqi Tan, Jin Yu, Zhou Zhao, Kun Kuang, Jie Liu, Jingren Zhou, Hongxia Yang, and Fei Wu. 2020. Poet: Product-oriented video captioner for e-commerce. In *ACM MM*.

[119] Yuhao Zhang and Arun Kumar. 2020. Panorama: A Data System for Unbounded Vocabulary Querying over Video. In *VLDB*.

[120] Zhipeng Zhang, Xinglin Hou, Kai Niu, Zhongzhen Huang, Tiezheng Ge, Yuning Jiang, Qi Wu, and Peng Wang. 2022. Attract me to buy: Advertisement copywriting generation with multimodal multi-structured information. *arXiv preprint arXiv:2205.03534* (2022).

[121] Zhe Zhang, Chunyu Wang, Weichao Qiu, Wenhu Qin, and Wenjun Zeng. 2020. AdaFuse: Adaptive Multiview Fusion for Accurate Human Pose Estimation in the Wild. *CoRR* abs/2010.13302 (2020). arXiv:2010.13302 https://arxiv.org/abs/2010.13302

[122] Yue Zhao, Ishan Misra, Philipp Krähenbühl, and Rohit Girdhar. 2023. Learning video representations from large language models. In *CVPR*.

[123] Tianxiong Zhong, Zhiwei Zhang, Guo Lu, Ye Yuan, Yu-Ping Wang, and Guoren Wang. 2023. TVM: A Tile-based Video Management Framework. *VLDB* (2023).

[124] Andong Zhu, Sheng Zhang, Xiaohang Shi, Ke Cheng, Hesheng Sun, and Sanglu Lu. 2024. Crucio: End-to-End Coordinated Spatio-Temporal Redundancy Elimination for Fast Video Analytics. In *INFOCOM*.