# NATIVECODEC: ENCODING EVERY VIDEO FRAME WITH VARIABLE-LENGTH CAUSAL VISUAL TOKENS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reasoning about motion and object identity often depends on brief interactions that sparse frame sampling can miss. Increasing temporal resolution preserves more of this evidence, but encoding each frame independently introduces substantial visual redundancy into multimodal large language models. The central challenge is to retain temporal detail without repeatedly allocating tokens to content already represented by earlier frames. Because the amount of new information varies across frames, a compact video representation should adapt its capacity to each frame's contribution relative to its history. **We introduce NativeCodec**, a causal visual encoder for video inputs at up to 16 fps. Inspired by predictive coding, it combines complete anchor frames with variable-length incremental representations of subsequent frames. Pixel and feature reconstruction at multiple truncation lengths encourages the encoder to concentrate information complementary to its history into retained token suffixes. An offline reconstruction-based teacher supervises a per-frame length predictor, and cumulative suffix lengths determine anchor placement. **We also introduce BallBench**, which tests identity tracking through collisions between visually identical objects. With a 4B base model and a fixed suffix ratio, NativeCodec reaches 56.4 mIoU on TimeLens-Bench, 25.0 mIoU on MomentSeeker, and 26.1% accuracy on BallBench, and with predicted suffix lengths it reaches 48.7% accuracy on Video-Holmes. Relative to its Qwen3.5-4B base model, it **gains 13.9 points on BallBench, 7.6 on Video-Holmes, 6.5 on MomentSeeker, and 5.7 mIoU on TimeLens-Bench**. NativeCodec is ahead of the native pathway of its base model at every budget, and the margin widens as the budget tightens, reaching 14.2 mIoU at 4k tokens. At a suffix ratio of 1/16, NativeCodec obtains 54.4 mIoU from 16k visual tokens, while the native pathway of its base model obtains 50.9 from 64k tokens, four times the budget. The code and model will be released.

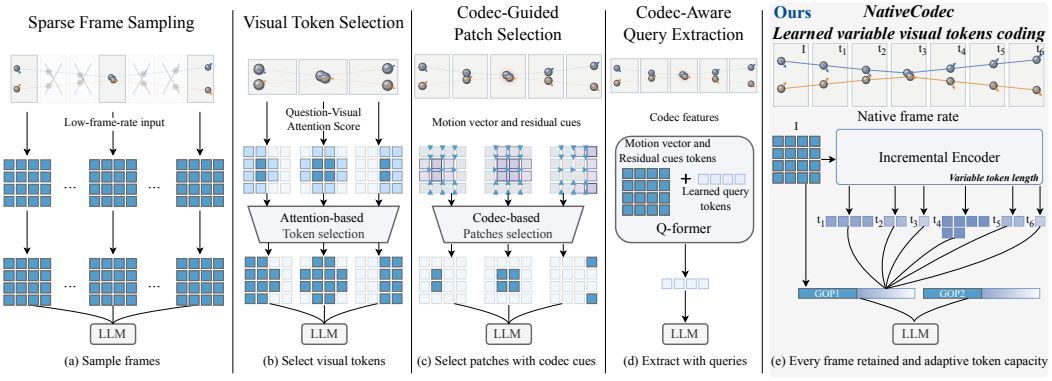


Figure 1: Visual-token interfaces for video MLLMs. Existing approaches sample frames, select visual tokens or patches, or extract compact codec features with learnable queries. NativeCodec retains the temporal positions reaching the codec with a causal I/P organization, where P-frame capacity varies with the visual increment and GOP boundaries follow the cumulative token budget of the stream.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

# 1 INTRODUCTION

**Human visual perception faces a continuously changing world** where neighboring observations share predictable content and temporal changes carry new evidence, a process that predictive coding theory explains through internal predictions and prediction errors (Rao & Ballard, 1999). H.264-style video codecs implement a corresponding organization, where I frames provide self-contained references and P frames encode updates conditioned on previously decoded frames (Wiegand et al., 2003; Sullivan et al., 2012; Bross et al., 2021). MLLMs face a direct sequence-length constraint in the visual token space, since a conventional visual encoder produces a complete grid of tokens for every sampled frame and the input sequence grows linearly with the temporal resolution (Zhang et al., 2023; Li et al., 2025; Song et al., 2024; Li et al., 2024; Xu et al., 2024). Long-video systems, therefore, use temporal subsampling, keyframe selection, spatial pooling, or token reduction to control computation (Chen et al., 2024b; Huang et al., 2025; Zhang et al., 2024b; Shang et al., 2025; Shao et al., 2025), but temporal positions removed before visual encoding cannot provide evidence to the final prediction, which is critical for reasoning that depends on brief intermediate states.

Recent codec-inspired methods use codec primitives, predictive visual codes, or codec-native motion and residual cues for video MLLMs (Sarkar et al., 2026; Hou et al., 2026; Yang et al., 2026; An et al., 2026). These studies establish the value of predictive structure, while their allocation rules remain human-designed for rate-constrained pixel reconstruction, even though the information required by a predictive frame depends on the content and the task. **The Bitter Lesson motivates replacing such manual rules with rules learned from data at scale** (Sutton, 2019). We retain I/P frames and the group-of-pictures (GOP) as a prior, while learning from reconstruction how much information each P frame should carry and when the current GOP should be refreshed with a new I frame.

We introduce NativeCodec, presenting a video to the language model as a causal, variable-length visual-token stream. Each group of pictures begins with an I-frame anchor, every subsequent admitted frame contributes a suffix whose length is selected from a small set of discrete ratios, and a new anchor opens whenever the accumulated suffix length would exceed the cost of a complete frame. We learn the representation with pixel and feature reconstruction across multiple ratios, and derive the length policy from a sequential teacher constrained by reconstruction quality. Figure 1 contrasts NativeCodec with existing interfaces. To test whether native-frame-rate input is necessary for transient interactions, we introduce BallBench, a benchmark for high-frame-rate identity reasoning, where visually identical objects move and collide, and recovering their correspondence to their initial identities requires following their trajectories through the collision, since appearance cues cannot distinguish the objects.

Our contributions are summarized as follows:

- **We introduce NativeCodec**, a causal visual encoder that combines full I-frame anchors with compact P-frame suffixes conditioned on earlier frames. This representation retains the temporal position of each admitted frame while reducing the visual tokens passed to the language model.
- **We develop a two-stage training procedure** that learns nested suffix representations through pixel and feature reconstruction and trains a length predictor from reconstruction-constrained sequential oracle targets. The predictor selects each P-frame suffix length based on the current frame and its causal context, while a fixed cumulative-token rule determines GOP boundaries.
- **We introduce BallBench**, a synthetic benchmark for short-term dynamic preservation and long-term identity maintenance through collisions between visually identical balls. Controlled frame-rate evaluations measure how temporal resolution affects identity correspondence accuracy.

# 2 METHOD

To derive the representation and training procedure, we first specify the visual-token interface a MLLM receives at inference time. To illustrate the connection, Figure 2 presents both, with the Incremental Encoder on the left and the two training stages followed by inference-time length prediction on the right. To keep dense temporal evidence accessible to the language model without exceeding its context limit, the interface represents a video as a causal sequence of tokens that exhibits three properties. **First**, because transient states must remain addressable by the downstream decoder, every admitted frame maintains its temporal position, which preserves the transient events that a uniform sampling grid removes. **Second**, because the information content varies across

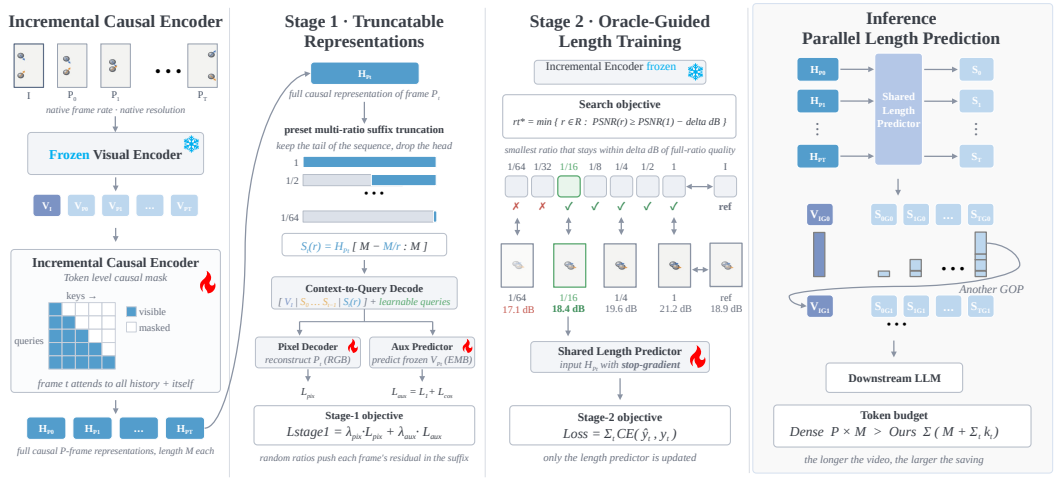


Figure 2: Overview of NativeCodec. A frozen visual encoder maps every admitted frame to a token grid, and the Incremental Encoder turns the grids into full-length causal representations of the predictive frames (left). Stage 1 truncates each representation at preset ratios and trains pixel and auxiliary decoders, so that every suffix remains decodable. Stage 2 searches for the smallest ratio whose reconstruction stays within $\delta$ dB of the teacher reconstruction and distills this search into a shared length predictor. At inference, the predictor assigns a suffix length to every frame in parallel, the predicted lengths are packed with periodic I-frame anchors into GOPs, and the resulting variable-length causal stream is passed to the downstream LLM (right).

temporal positions, the number of tokens allocated to a frame is content-adaptive and follows from the increment it adds to its causal history, so a frame changing little costs little. **Third**, because long inputs must remain compatible with a bounded sequence length, the aggregate visual token count stays under a global budget, which keeps dense temporal input compatible with a bounded context.

To formulate these properties, let $\boldsymbol{V}_t \in \mathbb{R}^{M\times d}$ denote the normalized visual-token grid of frame $t$ obtained as described in Section 2.1, which represents a self-contained reference when used as an anchor. To preserve the self-contained reference at the start of a sequence, the interface represents the first frame of a group of pictures by its complete grid $\boldsymbol{V}_1$. To represent subsequent frames under the content-adaptive property, each remaining frame $t$ within the GOP is represented as an incremental suffix of a causal representation. To generate the incremental representations, the Incremental Encoder maps the sequence of grids to a full-length representation for predictive frame,

$$[\boldsymbol{H}_2, \ldots, \boldsymbol{H}_F] = \mathrm{Enc}([\boldsymbol{V}_1, \ldots, \boldsymbol{V}_F]), \qquad \boldsymbol{H}_t \in \mathbb{R}^{M\times d}, \tag{1}$$

and to control the transmitted capacity, a per-frame suffix ratio $r_t$ selects the final $k(r_t)$ rows of that representation to form the transmitted suffix,

$$\boldsymbol{S}_t(r_t) = \mathrm{tail}_{k(r_t)}(\boldsymbol{H}_t) \in \mathbb{R}^{k(r_t)\times d}. \tag{2}$$

Because the rows correspond to the spatial positions of the frame, the truncation preserves its spatial coordinates. To enforce the global budget property over long sequences, the GOP boundary is determined dynamically by the accumulated suffix lengths. To allocate the budget dynamically, a group accumulates suffixes until adding the next one would cost more than transmitting a complete grid, at which point the next frame becomes the complete anchor of a new group,

$$\sum_{t\in\mathcal{G}} k_t + k_{t+1} > M \implies \text{frame } t+1 \text{ opens a new GOP.} \tag{3}$$

Because the codec produces per-frame suffixes, the GOP belongs to the interface, deciding where a fresh anchor becomes cheaper than continued prediction to reset the context and start a new group. This threshold is particularly useful for rapidly changing frames: when a frame would require the full ratio $r = 1$, transmitting a fresh I-frame avoids forcing its information through the Incremental Encoder and preserves a complete reference.

To realize this interface, we identify three methodological requirements. First, because the representation must take an incremental form conditioned on history, a frame representation must depend only

on earlier frames and earlier positions within itself, and Section 2.2 builds the Incremental Encoder that enforces this dependency. Second, because the per-frame suffix ratio $r_t$ varies with the content, a single representation must remain informative at every truncation length, which the model reaches by learning to push the content of a frame toward its later positions, so that the retained suffix carries the increment while the truncated positions carry the rest of its content; Section 2.4 obtains this behavior through a multi-ratio reconstruction objective. Third, because the interface is consumed as a token sequence in a single forward pass, the per-frame length has to follow from a feed-forward decision, and Section 2.4 amortizes an offline reconstruction-constrained search into such a policy. Section 2.5 describes the staged training procedure that instantiates this interface and trains each component.

### 2.1 Frame-level tokenization

A frozen ViT $\phi$ of the MLLM maps each frame to a grid of tokens that the codec consumes as input,

$$\boldsymbol{V}_t = \mathrm{RMSNorm}\big(\mathrm{Merge}(\phi(\boldsymbol{x}_t))\big), \qquad \boldsymbol{V}_t \in \mathbb{R}^{M\times d}. \tag{4}$$

Here, $\mathrm{Merge}$ groups each $2\times 2$ block of patch embeddings into one token, and the RMS normalization layer is trainable. If the patch-aligned input has spatial size $H \times W$, the resulting merge grid has size $(H/s) \times (W/s)$ and $M = (H/s)(W/s)$ tokens, where $s$ denotes the total spatial stride of the frozen vision tower after patchification and merging, so the number of tokens per frame follows the input resolution and aspect ratio. The vision transformer stays frozen during codec training, which keeps the learned representation in the feature space that the downstream MLLM expects.

### 2.2 Incremental Encoder

The Incremental Encoder $\mathrm{Enc}$ of Equation 1 flattens the spatial token grids in frame-major order and applies a four-layer causal Transformer network to the resulting long sequence of $F \cdot M$ positions, from which the rows belonging to frame $t$ form $\boldsymbol{H}_t$. The causal mask lets each token attend to tokens from earlier frames and to earlier spatial positions within its own frame, so a position late in frame $t$ has access to the whole history available at that frame. The Incremental Encoder uses standard attention with three-dimensional rotary position embeddings over frame, row, and column coordinates. Because the frame grid, the token count, and the rotary coordinates follow from the current input, the encoder requires neither a fixed spatial resolution nor a resolution-specific parameter set, and processes the patch-aligned resolutions used during training.

### 2.3 Reconstructive decoding

For a target P frame $t$, the decoder context consists of the complete I-frame anchor together with the retained suffixes of all preceding P frames and of the current P frame,

$$\mathbf{C}_t = \big[\, \boldsymbol{V}_1 \;\; \boldsymbol{S}_2(r_2) \;\; \cdots \;\; \boldsymbol{S}_t(r_t) \;\; \underbrace{\boldsymbol{q}\cdots\boldsymbol{q}}_{M} \,\big]. \tag{5}$$

Here, $\boldsymbol{q}$ is a learned query token replicated across the $M$ spatial positions. The query canvas follows the target frame's spatial grid, so the decoder reconstructs inputs with different resolutions using the same parameters. A causal mask lets each query attend to the complete I-frame anchor and the available P-frame suffixes up to its target frame, while preventing access to future frames. This causal decoding design follows the same prefix-conditioned dependency as the downstream autoregressive language model, where the I frame stays in the normalized feature space of the frozen vision tower described in Section 2.1 and the P suffixes encode temporal increments conditioned on earlier frames. The shared pixel decoder $\mathrm{Dec}_{\mathrm{pix}}$ maps this context to an RGB reconstruction, its upsampling head expands the active merge grid by the fixed token stride so that the output recovers the corresponding input size, and a second decoder with the same structure predicts the target feature grid,

$$\hat{\boldsymbol{x}}_t = \mathrm{Dec}_{\mathrm{pix}}(\mathbf{C}_t), \qquad \hat{\boldsymbol{V}}_t = \mathrm{Dec}_{\mathrm{aux}}(\mathbf{C}_t). \tag{6}$$

Both decoders share their parameters across the supported input resolutions, and their query canvas is resized to the target grid. We retain a suffix because later positions in the frame-major causal order have access to preceding positions and earlier frames, and reconstruction over multiple suffix lengths supplies the learning signal for this aggregation. For the I-frame reconstruction loss, the normalized feature grid of a frame is supplied as a complete anchor without any P-frame suffix, which provides the full-frame reference used in training and in the construction of the length targets. The packed query construction and its attention mask are specified in Appendix A.3.

### 2.4 VARIABLE-LENGTH TOKEN ALLOCATION

Each P frame introduces a different amount of visual information relative to its causal context, so a fixed-length representation would spend excess capacity on predictable frames while leaving too little for frames that change substantially. The second requirement of Section 2 therefore asks for a single causal representation whose truncations remain useful at different capacities, which lets the token length vary across P frames without training a separate encoder for each length. We obtain this property by adapting the nested representation principle of Matryoshka Representation Learning (Kusupati et al., 2022) to causal visual tokens, where shorter suffixes encode compact incremental information and longer suffixes preserve additional details. We instantiate the nested representation with the following seven candidate ratios for the suffix,

$$\mathcal{R} = \big\{1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \tfrac{1}{16}, \tfrac{1}{32}, \tfrac{1}{64}\big\}, \qquad k(r) = \max\big(1, \min(M, \mathrm{round}(rM))\big). \tag{7}$$

The candidate suffix lengths scale with the per-frame token count $M$ of the current input, so the same ratios instantiate proportional lengths across supported resolutions. Because the causal mask gives every suffix position access to all preceding positions of the frame and all earlier frames, a suffix token carries aggregated incremental information rather than the content of a single patch. During codec learning, the reconstruction objective uses a ratio shared by the selected P-frame targets in the standard branch, and uses independently sampled ratios for each P frame in the variable-ratio branch. Both branches expose the same Incremental Encoder to both homogeneous and heterogeneous suffix lengths. After the representation is learned, the length predictor $\pi$ reads the full P-frame representation $\boldsymbol{H}_t$ under a stop-gradient and maps its last spatial token to seven logits,

$$p_t = \mathrm{softmax}\big(\pi(\boldsymbol{H}_t[-1])\big), \qquad r_t = \arg\max_{r \in \mathcal{R}} p_t(r), \qquad k_t = k(r_t). \tag{8}$$

The predicted ratio determines the suffix length used for that P frame, and the predictor is trained only after the causal representation and the reconstruction decoders have been frozen. The concrete temporal-window and ratio-assignment schedule is given in Appendix A.1.

### 2.5 TRAINING OBJECTIVES AND STAGED PROCEDURE

Each training stage of NativeCodec follows from one of the requirements identified in Section 2. The first stage addresses the requirement that one representation remain informative at several truncation lengths, and it learns a causal representation whose suffixes stay useful across the candidate ratios. The second stage addresses the requirement that the length decision be feed-forward, and it converts the reconstruction criterion into a per-frame policy. At inference, the predicted lengths instantiate the interface of Section 2 under a bounded visual-token budget. Codec learning and downstream packaging use different anchor rules, and the distinction matters for reading the objectives below. During codec learning, a training window of $F$ consecutive frames takes its own first frame as the anchor and treats every remaining frame as a P frame, which isolates the learning of a reusable predictive representation from any budget decision. During downstream training and inference, the anchor positions instead follow Equation 3, and the Incremental Encoder is applied independently within each group so that a new anchor resets the context.

**Stage 1: truncatable representations.** We keep the pretrained feature extractor fixed and jointly learn the Incremental Encoder and reconstruction decoders. Let $\mathcal{T}$ denote a set of target P frames and $\mathcal{A}$ a collection of ratio assignments, where each assignment $\mathbf{r} = (r_t)_{t \in \mathcal{T}}$ specifies the suffix ratio of each target frame and may share one ratio across all target frames or use different ratios for different frames. For a given assignment, the pixel and feature reconstruction losses are defined as follows

$$\mathcal{L}_{\mathrm{pix}}(\mathbf{r}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \|\hat{\boldsymbol{x}}_t^{(\mathbf{r})} - \boldsymbol{x}_t\|_1, \;\; \mathcal{L}_{\mathrm{aux}}(\mathbf{r}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \Big( \|\hat{\boldsymbol{V}}_t^{(\mathbf{r})} - \boldsymbol{V}_t\|_1 + 1 - \cos(\hat{\boldsymbol{V}}_t^{(\mathbf{r})}, \boldsymbol{V}_t) \Big). \tag{9}$$

The representation objective averages these losses across all of the assignments,

$$\mathcal{L}_1 = \lambda_{\mathrm{pix}} \frac{1}{|\mathcal{A}|} \sum_{\mathbf{r} \in \mathcal{A}} \mathcal{L}_{\mathrm{pix}}(\mathbf{r}) + \lambda_{\mathrm{aux}} \frac{1}{|\mathcal{A}|} \sum_{\mathbf{r} \in \mathcal{A}} \mathcal{L}_{\mathrm{aux}}(\mathbf{r}) + \lambda_{\mathrm{anchor}} \mathcal{L}_{\mathrm{anchor}}. \tag{10}$$

Here, $\mathcal{L}_{\mathrm{anchor}}$ reconstructs frames designated as complete anchors from their full normalized feature grids, which preserves the full-frame reference needed to initialize predictive groups and to define the teacher reconstruction quality in the next stage. The multi-assignment objective makes one causal representation usable at multiple suffix lengths, while the oracle-guided targets keep the retained

suffixes compact without an explicit rate penalty. The held-out comparison of the composite and auxiliary-only objectives is reported in Appendix B.

**Stage 2: oracle-guided length training.** After the truncatable representation has been learned, we freeze the Incremental Encoder and reconstruction decoders and train only the length predictor $\pi$. We construct its supervision with an offline sequential teacher. For each P frame, a complete-grid reconstruction of that frame provides a reference quality

$$q_t^{\text{teach}} = \text{PSNR}(\boldsymbol{x}_t^{\text{teach}}, \boldsymbol{x}_t). \tag{11}$$

For each P frame $t$, we take the complete normalized feature grid $\boldsymbol{V}_t$ and feed it to the pixel decoder as a standalone I-frame anchor. The resulting RGB reconstruction $\boldsymbol{x}_t^{\text{teach}}$ provides the reference quality for that frame, since the teacher reconstruction uses the full grid rather than a P-frame suffix. The teacher then processes P frames in temporal order. Given the suffix lengths selected for preceding frames, let $\hat{\boldsymbol{x}}_t^{(r|\mathbf{k}_{<t})}$ denote the reconstruction obtained when the current frame uses candidate ratio $r$. The teacher selects the smallest ratio that satisfies the reconstruction constraint,

$$r_t^* = \min\left\{ r \in \mathcal{R} : \text{PSNR}\left(\hat{\boldsymbol{x}}_t^{(r|\mathbf{k}_{<t})}, \boldsymbol{x}_t\right) \geq q_t^{\text{teach}} - \delta \right\}. \tag{12}$$

If no candidate satisfies the constraint, the full ratio is retained. The selected ratio is added to the causal reconstruction prefix for the next frame, so each target is produced in temporal order and conditioned on the ratios chosen for the preceding frames. The predictor is trained with cross-entropy,

$$\mathcal{L}_2 = -\frac{1}{F-1}\sum_{t=2}^{F} \log p_t(r_t^*), \tag{13}$$

where $p_t$ is predicted from the P-frame representation. This amortizes offline reconstruction-based search into a per-frame length policy in one forward pass without inference-time search. Appendix E reports the sensitivity of the length policy to the reconstruction tolerance $\delta$.

Downstream alignment keeps the codec frozen and trains the language model on the stream it produces. Within each group, the I-frame anchor and the suffixes of the predictive frames are concatenated in temporal order, and every retained token enters the language model at its temporal and spatial coordinates. The language model is trained with a standard autoregressive objective on answer tokens only, so the visual stream serves as context while response tokens carry the loss. The textual timestamp convention of this interface is specified in Appendix A.2.

**Inference: parallel length prediction.** After codec learning, the Incremental Encoder and reconstruction decoders are frozen. Because the length predictor is feed-forward, it assigns a suffix length to every P frame in parallel, and the frames are organized into GOPs by the boundary condition of Equation 3. Each GOP begins with an I-frame anchor and continues with P frames represented by selected suffixes until the accumulated suffix length would exceed $M$, at which point the next frame becomes a new I frame and resets the causal context. Retained visual tokens preserve temporal and spatial coordinates when inserted into the MLLM input. This realizes the interface through which the MLLM consumes dense video evidence as a causal, variable-length visual-token sequence. Appendix D provides a frame-level analysis of the resulting length policy.

### 2.6 BallBench: a benchmark for high frame-rate dynamic video understanding

To evaluate whether models can exploit high temporal resolution, we introduce BallBench, a benchmark for high frame-rate dynamic video understanding. Each video contains between four and ten visually identical balls recorded at thirty frames per second, with a total duration of eight to twelve seconds divided into three phases: an initial numeric label on each stationary ball, then frictionless perfectly elastic motion and collisions after all labels are removed, and finally a new set of numeric labels assigned as a derangement so that no ball keeps its initial label. The task is identity correspondence: for a given initial label, the model must determine which final label corresponds to the same physical ball in the scene. BallBench is designed to isolate two capabilities. Short-term dynamic preservation asks whether the model can use the transient events that low frame-rate sampling would skip, such as ball contacts, rebounds, and velocity exchanges, to link trajectories across a collision. Long-term identity maintenance asks whether the model can track a physical identity across visually identical motion without color, texture, or trail cues. Because the same video can be rendered at different frame rates, BallBench directly measures the value of temporal resolution for a given model. Its simulator, split construction, and sample format are detailed in Appendix F and Appendix G.

Table 1: Performance of open-source video MLLMs on BallBench, TimeLens-Bench, MomentSeeker (MS), Video-Holmes (VH), and Video-MME-v2 (MME-v2). TimeLens-Bench reports mIoU over its three official subsets (AN = ActivityNet, CH = Charades, QV = QVHighlights), MS reports mIoU, all other columns report accuracy, and all values are percentages. All evaluated models share a visual-token budget of 64k: NativeCodec receives frames at 16 fps and the baselines at their default frame rates, and any video whose sequence exceeds the budget has its frames subsampled uniformly until it fits. NativeCodec is reported with a constant suffix ratio of $1/16$, and † marks the same interface with the predicted suffix lengths of Section 2.5. **Bold** marks the best open-source score, and underline marks the second. We train Qwen3.5-4B on identical data, yielding its $^{sft}$ version.

| **Benchmark** | **BallBench** | **TimeLens-Bench** | | | | **MS** | **VH** | **MME-v2** |
|---|---|---|---|---|---|---|---|---|
| **Model / metric** | **ACC** | **mIoU$^{AN}$** | **mIoU$^{CH}$** | **mIoU$^{QV}$** | **mIoU$^{All}$** | **mIoU** | **ACC** | **ACC** |
| GPT-4o (OpenAI & etc., 2024) | – | 40.4 | 41.8 | 52.1 | 42.8 | – | 42.0 | – |
| GPT-5 (OpenAI & etc., 2026) | – | 42.9 | 40.5 | 56.8 | 44.3 | – | – | 44.7 |
| Gemini-2.0-Flash (Team et al., 2023) | – | 49.3 | 46.7 | 60.8 | 50.3 | – | 30.6 | – |
| Gemini-2.5-Flash (Comanici et al., 2025) | 15.8 | 52.5 | 48.6 | 64.3 | 53.0 | – | – | – |
| Gemini-2.5-Pro (Comanici et al., 2025) | 25.5 | 58.1 | 52.8 | 70.4 | 58.2 | – | 45.0 | – |
| InternVL2.5-8B (Chen et al., 2024c) | 15.3 | 4.1 | 9.3 | 3.0 | 5.8 | 2.8 | 23.6 | – |
| InternVL3-8B (Zhu et al., 2025) | 13.9 | 11.9 | 23.8 | 9.0 | 15.7 | 4.6 | 32.3 | – |
| InternVL3.5-4B (Wang et al., 2025a) | 12.9 | 11.2 | 20.3 | 10.4 | 14.3 | 3.2 | 37.4 | 24.2 |
| Qwen2.5-VL-7B (Bai et al., 2025b) | – | 31.4 | 39.3 | 31.6 | 34.3 | – | 27.8 | 25.3 |
| Qwen3-VL-4B (Bai et al., 2025a) | 13.7 | 48.8 | 49.3 | 59.7 | 50.8 | 15.3 | 39.4 | 24.2 |
| Qwen3-VL-8B (Bai et al., 2025a) | 14.3 | 48.0 | 47.2 | 59.4 | 49.6 | 9.8 | 42.5 | **27.9** |
| Qwen3.5-9B (Team, 2026) | 14.7 | 49.9 | 45.6 | 64.0 | 50.7 | 12.8 | 46.3 | 26.0 |
| LLaVA-OV-2-8B (An et al., 2026) | 17.1 | 53.8 | 53.5 | 66.4 | 55.8 | 19.6 | 32.5 | 19.9 |
| Mage-VL-4B (Yang et al., 2026) | 16.3 | 45.4 | 50.7 | 57.4 | 49.3 | 14.4 | 39.3 | 24.3 |
| Qwen3.5-4B (Team, 2026) | 12.2 | 49.4 | 46.0 | 64.6 | 50.7 | 18.5 | 39.5 | 24.1 |
| Qwen3.5-4B$^{sft}$ (Team, 2026) | 15.3 | 51.7 | 49.5 | 66.6 | 53.2 | 18.8 | 44.0 | 23.3 |
| **NativeCodec-4B** | 26.1 | **54.5** | **53.8** | **67.7** | **56.4** | **25.0** | 47.1 | 24.6 |
| *Δ (vs. Qwen3.5-4B)* | +13.9 | +5.1 | +7.8 | +3.1 | +5.7 | +6.5 | +7.6 | +0.5 |
| **NativeCodec-4B†** | **28.2** | 52.7 | 51.4 | 66.2 | 54.4 | 24.7 | **48.7** | 23.7 |

## 3 EXPERIMENTS

We evaluate NativeCodec through downstream accuracy on the ActivityNet (Yu et al., 2019), Charades (Sigurdsson et al., 2016), and QVHighlights (Lei et al., 2021) splits of TimeLens-Bench (Zhang et al., 2026) for temporal grounding, on Video-Holmes (Cheng et al., 2025) for complex video reasoning, and on MomentSeeker (Yuan et al., 2026a) for long-video moment retrieval, and on Video-MME-v2 (Fu et al., 2026) for general video understanding tasks, together with BallBench, the benchmark introduced in Section 2.6. All training data, model architecture instantiation, and specific hyperparameter choices follow the procedure in Section 2.5 and are described in detail in Appendix A; overall training cost and module sizes are reported in Appendix A.4.

### 3.1 COMPARISON WITH OPEN-SOURCE VIDEO MLLMS

We evaluate widely adopted open-source video MLLMs covering the interfaces in Section 4: Qwen3.5-4B and Qwen3.5-9B, Qwen3-VL-4B and Qwen3-VL-8B (Bai et al., 2025a), Qwen2.5-VL-7B (Bai et al., 2025b), InternVL2.5-8B (Chen et al., 2024c), InternVL3-8B (Zhu et al., 2025), and InternVL-3.5-4B (Wang et al., 2025a), LLaVA-OV-2 (An et al., 2026), and Mage-VL (Yang et al., 2026). Table 1 reports their performance on BallBench, on TimeLens-Bench (9,404 questions over three subsets), on Video-Holmes (1,837 questions over seven reasoning categories), on Video-MME-v2 (Fu et al., 2026) for general video understanding, and on the text-only query subset of MomentSeeker (1,000 queries), with TimeLens-Bench decomposed into its three subsets of temporal grounding.

NativeCodec-4B leads the open-source models in Table 1 on the benchmarks that demand dense temporal evidence. Its fixed-ratio variant reaches 26.1 on BallBench, while its predicted-length variant reaches the best BallBench accuracy in the table at 28.2; the best MomentSeeker mIoU at 25.0, against 19.6 for the second interface; the best TimeLens-Bench mIoU among open-source

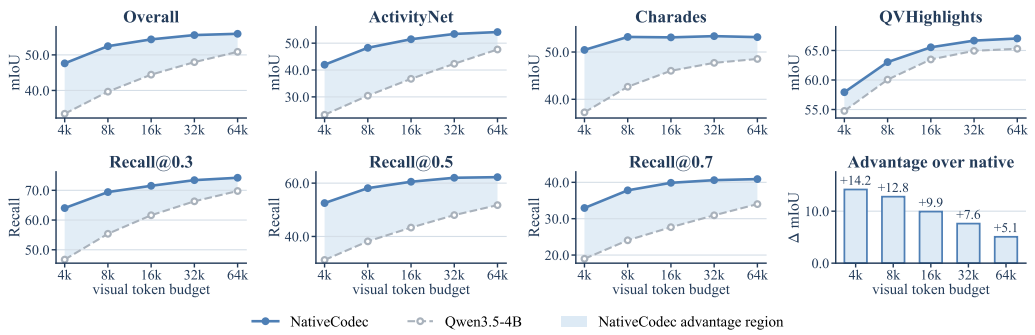


Figure 3: Visual token budget. Both systems are evaluated at the same visual-token budgets and the same frame rate. NativeCodec keeps the spatial size of each frame fixed to the input video and uses a constant suffix ratio of $1/16$, so the budget determines how many frames enter the codec stream, while the Qwen3.5-4B native pathway samples frames uniformly under the same budget. The shaded area marks the NativeCodec advantage, and the last panel reports its margin on the overall mIoU.

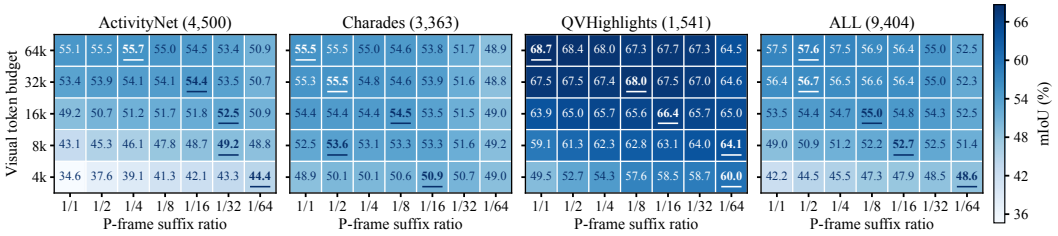


Figure 4: Joint sweep of visual-token budget and suffix ratio on TimeLens-Bench, for the full benchmark and each subset. Each row fixes a visual-token budget and each column a suffix ratio, and the best ratio of each row is in bold with an underline. Colours share a common scale across panels.

models at 56.4, with the best open-source score on each of the subsets; and, with predicted suffix lengths, the best Video-Holmes accuracy at 48.7, against 46.3 for the second. Relative to its Qwen3.5-4B base it gains 13.9 points on BallBench, 7.6 on Video-Holmes, 6.5 on MomentSeeker, and 5.7 on TimeLens-Bench overall, using a constant suffix ratio of $1/16$. On BallBench, every open-source baseline stays within four points of the random level of 13.9 and the best of them reaches only 17.1.

### 3.2 Visual token budget

Figure 3 compares NativeCodec with the native visual pathway of its Qwen3.5-4B base model at the same visual-token budgets. NativeCodec keeps every frame's spatial size fixed to the input video and uses a constant suffix ratio of $1/16$, so the budget determines how many frames enter the codec stream, while the native pathway samples frames uniformly under the same budget and frame rate. NativeCodec is ahead at every budget, and the margin widens as the budget tightens: $+14.2$ mIoU at 4k, $+12.8$ at 8k, $+9.9$ at 16k, $+7.6$ at 32k, and $+5.1$ at 64k. At 16k tokens NativeCodec reaches 54.4 mIoU, which is above the 50.9 the native visual pathway reaches with four times the budget.

Figure 4 sweeps the two decisions jointly. Each row fixes a visual-token budget and each column fixes a suffix ratio, respectively, and the corresponding best ratio of each row is consistently set in bold with an underline. At the largest budgets the choice is nearly free, since at 64k every ratio down to $1/16$ stays within 1.2 mIoU of the best setting. The ordering reverses as the budget tightens: at 4k the smallest ratio wins, with $1/64$ reaching 48.6 mIoU against 42.2 at $1/1$, so a tight budget is better spent on more frames than on more tokens per frame. The complete held-out ratio sweep and reconstructions are provided in Appendix C. A budget stops mattering once it stops binding. Evaluation applies the budget by subsampling the frames of any video whose sequence exceeds the limit, uniformly across the clip, until the sequence fits, so a video that already fits is passed to the model unchanged, and the gap between the 32k and 64k columns shrinks as the ratio shrinks, from $+1.1$ mIoU at ratio 1 to $+0.05$ at $1/16$, because a smaller ratio brings fewer videos over the 32k limit. The flattening at the top of the budget axis therefore reflects the length distribution of the benchmark, and it affects the native pathway in the same way, which keeps the fixed-budget comparison fair.

### 3.3 FRAME RATE

Table 2 reports BallBench accuracy versus frame rate. Every open-source baseline stays within four points of the 13.9 random baseline at all frame rates, with varying trends: three gain from additional frames, two peak at the lowest rate, and the two vary without trend. Only NativeCodec stays far above the random baseline at all rates, improving from 22.9 at 2 fps to 26.1 at 16 fps. This separates the two properties BallBench tests: admitting native-frame-rate input and using it. Some baselines can nevertheless score higher at 2 fps than at 16 fps because their native video training and inference protocols are centered around low-rate inputs, making the 2-fps setting closer to their training distribution; this reflects protocol alignment rather than greater stability.

Table 2: BallBench accuracy versus input frame rate, evaluated on the full set of 490 questions. NativeCodec runs with a constant suffix ratio of $1/16$.

| Model | 16 fps | 8 fps | 4 fps | 2 fps |
|---|---|---|---|---|
| Random | 13.9 | 13.9 | 13.9 | 13.9 |
| InternVL-3.5-4B | 12.9 | 12.7 | 15.1 | 15.5 |
| Qwen3-VL-4B | 13.7 | 11.2 | 9.0 | 8.0 |
| Qwen3-VL-8B | 14.3 | 10.7 | 9.6 | 6.3 |
| Qwen3.5-4B | 12.2 | 13.2 | 8.6 | 10.4 |
| Qwen3.5-9B | 14.7 | 11.9 | 9.9 | 7.4 |
| LLaVA-OV-2-8B | <u>17.1</u> | <u>17.8</u> | 16.3 | <u>17.8</u> |
| Mage-VL-4B | 16.3 | 15.9 | <u>17.1</u> | 14.1 |
| **NativeCodec-4B** | **26.1** | **24.3** | **23.5** | **22.9** |

## 4 RELATED WORK

### 4.1 MULTIMODAL LARGE LANGUAGE MODELS AND TOKEN COMPRESSION

Video MLLMs combine visual representations with temporal modeling (Wang et al., 2024b; Lin et al., 2024; Cheng et al., 2024; Song et al., 2024; Li et al., 2024; Ren et al., 2024; Xu et al., 2024), with further work on memory, temporal grounding, and frame selection for long videos (Jin et al., 2024; Huang et al., 2024; Chen et al., 2024a; Song et al., 2025; Wang et al., 2025b). Token pruning, merging, and selection reduce visual computation (Chen et al., 2024b; Huang et al., 2025; Shang et al., 2025; Zhang et al., 2024b; Shao et al., 2025). High-resolution MLLMs address spatial token costs through visual search, hierarchical processing, distillation, and evidence routing (Wu & Xie, 2024; Shen et al., 2025; Liu et al., 2026b; Wei et al., 2026; Yuan et al., 2026b; Mao et al., 2026). Video-MME, LongVideoBench, MLVU, and CG-Bench evaluate general and long-video understanding (Fu et al., 2025; Wu et al., 2024; Zhou et al., 2025; Chen et al., 2025), and E-VAds targets e-commerce short videos (Liu et al., 2026a). BallBench focuses on identity correspondence through motion and collisions between visually identical objects.

### 4.2 CODEC-INSPIRED VISUAL REPRESENTATIONS

Neural video codecs learn predictive representations under rate-distortion objectives (Li et al., 2021; Jia et al., 2025). Codec-inspired MLLMs use codec primitives, predictive codes, or motion and residual cues for video understanding (Zhang et al., 2024a; Sarkar et al., 2026; Hou et al., 2026; Yashima et al., 2026; An et al., 2026; Yang et al., 2026). Learned visual tokenizers produce compact, decodable representations (Van Den Oord et al., 2017; Esser et al., 2021; Yu et al., 2023; Wang et al., 2024a; Ge et al., 2024), while video representation learning uses masked reconstruction, feature prediction, or contrastive objectives (Tong et al., 2022; Li et al., 2023; Xu et al., 2021; Wang et al., 2024c). NativeCodec learns causal P-frame suffixes through multi-ratio pixel and feature reconstruction. A reconstruction-supervised predictor selects each P frame's suffix length, and a fixed cumulative-token rule determines GOP boundaries. This yields a compact visual-token sequence that retains the temporal position of each admitted frame.

## 5 CONCLUSION AND LIMITATIONS

NativeCodec represents video as a causal sequence of full-frame anchors and compact predictive suffixes, preserving each admitted frame's temporal position with fewer visual tokens. Multi-ratio pixel and feature reconstruction supports variable-length representations, and a reconstruction-supervised predictor selects each predictive frame's suffix length from its causal context. A cumulative-token rule determines when to introduce a new anchor. Experiments on temporal grounding, video reasoning, and identity tracking demonstrate improved accuracy under constrained visual-token budgets. BallBench further evaluates whether models can follow visually identical objects through motion and collisions. These results support causal, variable-length visual tokens as an effective interface for temporal video understanding. Due to limited computational resources, we have not yet extended training to larger models or explored reinforcement learning.

## AI USE STATEMENT

In this work, we used generative AI tools for language polishing and formatting assistance during the preparation of this manuscript. We have not used generative AI tools for generating research ideas, designing the method, conducting experiments, or analyzing results. All experimental code and model training were implemented and verified by the authors. We take full responsibility for the final content of this work.

## ETHICS STATEMENT

This work involves no human subjects, no sensitive data, and no potentially harmful applications. BallBench is built from synthetic simulations and contains no personally identifiable information. We identify no direct negative societal impacts of this research.

## REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of our results. The source code for NativeCodec, including the causal visual encoder, the length prediction module, and the BallBench simulator, will be released publicly upon acceptance. Detailed training configurations, hyperparameters, and dataset preparation scripts will be provided in the supplementary materials.

## REFERENCES

Xiang An, Yin Xie, Feilong Tang, Yunyao Yan, Huajie Tan, Didi Zhu, Changrui Chen, Xiuwei Zhao, Bin Qin, Kaicheng Yang, et al. Llava-onevision-2: Towards next-generation perceptual intelligence. *arXiv preprint arXiv:2605.25979*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021.

Guo Chen, Yicheng Liu, Yifei Huang, Baoqi Pei, Jilan Xu, Yuping He, Tong Lu, Yali Wang, and Limin Wang. Cg-bench: Clue-grounded question answering benchmark for long video understanding. In *International Conference on Learning Representations*, volume 2025, pp. 45647–45682, 2025.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18407–18418. IEEE, 2024a.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024b.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024c.

Junhao Cheng, Yuying Ge, Teng Wang, Yixiao Ge, Jing Liao, and Ying Shan. Video-holmes: Can mllm think like holmes for complex video reasoning? *arXiv preprint arXiv:2505.21374*, 2025.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, et al. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 2024.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flexattention: A programming model for generating fused attention variants. *Proceedings of Machine Learning and Systems*, 7, 2025.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Kaituo Feng, Manyuan Zhang, Hongyu Li, Kaixuan Fan, Shuang Chen, Yilei Jiang, Dian Zheng, Peiwen Sun, Yiyuan Zhang, Haoze Sun, et al. Onethinker: All-in-one reasoning model for image and video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5432–5443, 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118. IEEE, 2025.

Chaoyou Fu, Haozhi Yuan, Yuhao Dong, Yi-Fan Zhang, Yunhang Shen, Xiaoxing Hu, Xueying Li, Jinsen Su, Chengwu Long, Xiaoyao Xie, et al. Video-mme-v2: Towards the next stage in benchmarks for comprehensive video understanding. *arXiv preprint arXiv:2604.05015*, 2026.

Yuying Ge, Sijie Zhao, Ziyun Zeng, Yixiao Ge, Chen Li, Xintao Wang, and Ying Shan. Making llama see and draw with seed tokenizer. In *International Conference on Learning Representations*, volume 2024, pp. 35206–35231, 2024.

Haowen Hou, Zhen Huang, Zheming Liang, Qingyi Si, Chenglin Li, Shuai Dong, Kele Shao, Ruilin Li, Dianyi Wang, Nan Duan, et al. Adacodec: A predictive visual code for video mllms. *arXiv preprint arXiv:2606.02569*, 2026.

Bin Huang, Xin Wang, Hong Chen, Zihan Song, and Wenwu Zhu. Vtimellm: Empower llm to grasp video moments. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14271–14280. IEEE, 2024.

Xiaohu Huang, Hao Zhou, and Kai Han. Prunevid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19959–19973, 2025.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12543–12552. IEEE, 2025.

Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13700–13710. IEEE, 2024.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. *Advances in Neural Information Processing Systems*, 34:11846–11858, 2021.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021.

Kunchang Li, Yali Wang, Yizhuo Li, Yi Wang, Yinan He, Limin Wang, and Yu Qiao. Unmasked teacher: Towards training-efficient video foundation models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19891–19903. IEEE, 2023.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *Science China Information Sciences*, 68(10):200102, 2025.

Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *European conference on computer vision*, pp. 323–340. Springer, 2024.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5971–5984, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.342. URL https://aclanthology.org/2024.emnlp-main.342/.

Xianjie Liu, Yiman Hu, Liang Wu, Ping Hu, Yixiong Zou, Jian Xu, and Bo Zheng. E-VAds: An e-commerce short videos understanding benchmark for MLLMs. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=IhhgzNZNGW.

Xianjie Liu, Yiman Hu, Yixiong Zou, Liang Wu, Jian Xu, and Bo Zheng. Hide: Rethinking the zoom-IN method in high resolution MLLMs via hierarchical decoupling. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=v6Tpe2LRrW.

Zhongkuan Mao, Xianjie Liu, Tianyu Meng, Yidong Wang, Wenzhuo Zhao, Ronghao Xian, Yao Jiang, Fei Shen, Junfeng Fang, Yong Dai, et al. Thinking once is enough: Intermediate-layer evidence routing for high-resolution vqa. *arXiv preprint arXiv:2607.27830*, 2026.

OpenAI and etc. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

OpenAI and etc. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. Timechat: A time-sensitive multimodal large language model for long video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14313–14323. IEEE, 2024.

Sayan Deb Sarkar, Rémi Pautrat, Ondrej Miksik, Marc Pollefeys, Iro Armeni, Mahdi Rad, and Mihai Dusmanu. Cope-videolm: Leveraging codec primitives for efficient video language modeling. *arXiv preprint arXiv:2602.13191*, 2026.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22857–22867. IEEE, 2025.

Kele Shao, Keda Tao, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. A survey of token compression for efficient multimodal large language models. *arXiv preprint arXiv:2507.20198*, 2025.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. Zoomeye: Enhancing multimodal llms with human-like zooming capabilities through tree-based image exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6613–6629, 2025.

Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European conference on computer vision*, pp. 510–526. Springer, 2016.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18221–18232. IEEE, 2024.

Zhende Song, Chenchen Wang, Jiamu Sheng, Chi Zhang, Shengji Tang, Jiayuan Fan, and Tao Chen. Dreamframe: Enhancing video understanding via automatically generated qa and style-consistent keyframes. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 13030–13037, 2025.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Junke Wang, Yi Jiang, Zehuan Yuan, Binyue Peng, Zuxuan Wu, and Yu-Gang Jiang. Omnitokenizer: A joint image-video tokenizer for visual generation. *Advances in Neural Information Processing Systems*, 37:28281–28295, 2024a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. Internvideo2: Scaling foundation models for multimodal video understanding. In *European conference on computer vision*, pp. 396–416. Springer, 2024c.

Yuxuan Wang, Yiqi Song, Cihang Xie, Yang Liu, and Zilong Zheng. Videollamb: Long streaming video understanding with recurrent memory bridges. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24170–24181. IEEE, 2025b.

Lai Wei, Liangbo He, jun lan, Lingzhong Dong, Yutong Cai, Siyuan Li, Huijia Zhu, Weiqiang Wang, Linghe Kong, Yue Wang, Zhuosheng Zhang, and Weiran Huang. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bVXx0K0j4c.

Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7): 560–576, 2003.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094. IEEE, 2024.

Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6787–6800, 2021.

Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. Slowfast-llava: A strong training-free baseline for video large language models. *arXiv preprint arXiv:2407.15841*, 2024.

Senqiao Yang, Kaichen Zhang, Zhaoyang Jia, Jinghao Guo, Yifei Shen, Xinjie Zhang, Xiaoyi Zhang, Haoqing Wang, Xiao Li, Peng Zhang, et al. Mage-vl: An efficient codec-native streaming multimodal foundation model. *arXiv preprint arXiv:2607.24904*, 2026.

Daichi Yashima, Shuhei Kurita, Yusuke Oda, and Komei Sugiura. Remora: Multimodal large language model based on refined motion representation for long-video understanding. *arXiv preprint arXiv:2602.16412*, 2026.

Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10459–10469. IEEE, 2023.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

Huaying Yuan, Jian Ni, Zheng Liu, Yueze Wang, Junjie Zhou, Zhengyang Liang, Bo Zhao, Zhao Cao, Ji-Rong Wen, and Zhicheng Dou. Momentseeker: A task-oriented benchmark for long-video moment retrieval. *Advances in Neural Information Processing Systems*, 38, 2026a.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-opd: Learning to see fine details for multimodal llms via on-policy self-distillation. *arXiv preprint arXiv:2605.18740*, 2026b.

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations*, pp. 543–553, 2023.

Jun Zhang, Teng Wang, Yuying Ge, Yixiao Ge, Xinhao Li, and Limin Wang. Timelens: Rethinking video temporal grounding with multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10419–10429, 2026.

Pingping Zhang, Jinlong Li, Kecheng Chen, Meng Wang, Long Xu, Haoliang Li, Nicu Sebe, Sam Kwong, and Shiqi Wang. When video coding meets multimodal large language models: A unified paradigm for video coding. *arXiv preprint arXiv:2408.08093*, 2024a.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024b.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, et al. Mlvu: Benchmarking multi-task long video understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13691–13701. IEEE, 2025.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

# Appendix: NativeCodec: Encoding Every Video Frame with Variable-Length Causal Visual Tokens

## Contents


**Appendix A** **Training Setup** Page 17
Temporal windows, ratio assignments, staged optimization, and downstream alignment settings.

**Appendix B** **Pixel and auxiliary losses** Page 20
Held-out reconstruction comparison of the composite and auxiliary-only objectives.

**Appendix C** **Suffix ratio ablation** Page 21
Reconstruction quality and visual examples across the seven candidate suffix ratios.

**Appendix D** **Length policy analysis** Page 22
Per-frame token allocation, token retention, and controlled frame-rate analysis.

**Appendix E** **Reconstruction tolerance sensitivity** Page 24
Retention, compression, and PSNR behavior under different Stage 2 tolerances.

**Appendix F** **BallBench Details** Page 25
Simulator, motion phases, data splits, task construction, and reproducibility details.

**Appendix G** **BallBench training sample** Page 26
Complete prompt, tracking rationale, and answer format for one episode.

# A TRAINING SETUP

We follow the procedure in Section 2.5. The training set consists of the OneThinker-SFT-video subset (Feng et al., 2026) and the BallBench training split, with every video resampled to sixteen frames per second, and five percent of the training videos are held out as an internal set for reconstruction evaluation. Stage 1 samples a bounded set of predictive-frame targets for the pixel and feature decoders rather than reconstructing every predictive frame in every window. The downstream MLLM uses a frozen vision tower with sixteen-pixel patches and $2 \times 2$ merging, so the total spatial stride is $s = 32$ and a $512 \times 512$ input yields a $16 \times 16$ grid with $M = 256$ tokens of dimension $d = 2560$. The codec is trained with longest-edge settings of $384$, $512$, and $768$ pixels using shared encoder and decoder parameters, while the merge grid and query canvas are updated for each active input shape. Its variable-ratio branch is activated with probability $0.5$ and averages seven independently sampled per-frame ratio assignments. Stage 2 freezes the representation and trains the length policy from teacher-based oracle labels with $\delta = 0.8$ dB.

## A.1 TEMPORAL WINDOWS AND RATIO ASSIGNMENTS

The codec is trained on contiguous temporal windows sampled at $16$ fps. We use windows of $1$, $2$, and $4$ seconds, containing respectively $F = 16$, $32$, and $64$ frames. The first frame of each window is treated as an I-frame anchor and the remaining $F - 1$ frames are P frames. For a window of duration $w$, the target frame times are uniformly spaced at $w/F$ over the half-open interval $[t_0, t_0 + w)$, where $t_0$ is the sampled start time. When a video does not provide enough decoded frames at the end of a window, the last decoded frame is repeated to preserve the fixed window length. Samples in one local batch are drawn from the same video and use separated temporal windows when the video duration permits.

The candidate suffix ratios are

$$\mathcal{R} = \left\{1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \tfrac{1}{16}, \tfrac{1}{32}, \tfrac{1}{64}\right\}.$$

For a grid with $M$ tokens, ratio $r$ retains $k(r) = \max(1, \mathrm{round}(rM))$ tokens, clipped to $M$. In the shared-ratio branch, all P frames in a forward pass use the same ratio, and all seven candidates are evaluated with equal weight. With probability $0.5$, training instead uses the variable-ratio branch: each P frame independently samples one ratio from $\mathcal{R}$, and seven independently sampled per-frame schedules are evaluated. The variable branch therefore exposes the decoder to heterogeneous prefix lengths while preserving the same nested causal representation.

For Phase 1, each resolution and temporal-window configuration uses a batch of two per device. The $1$-second window reconstructs all $15$ P frames, whereas the $2$- and $4$-second windows randomly select $15$ P frames for the pixel and auxiliary losses in each forward pass. The effective global batch is $8$ GPUs $\times 2$ samples $\times 8$ gradient-accumulation steps. Phase 1 runs for $150{,}000$ micro-steps. In Phase 2, the codec representation and reconstruction modules are frozen. The length predictor is trained for $20{,}000$ steps from sequential oracle labels: for each P frame, all seven ratios are tested under the already selected history, and the smallest ratio whose PSNR is within $\delta$ dB of the complete-grid teacher reconstruction is used as the target.

## A.2 ANCHOR-ONLY TEMPORAL METADATA

The codec and the downstream MLLM use two complementary forms of temporal information. Every admitted frame retains a frame-aware temporal coordinate in the visual stream, while explicit textual timestamp markers are attached only to I-frame anchors. Let $g_i$ denote the frame index of the $i$-th I frame, and let $\tau(g_i)$ denote its textual timestamp marker. The downstream multimodal prefix therefore has the schematic form

$$[\tau(g_1),\ \boldsymbol{V}_{g_1},\ \boldsymbol{S}_{g_1+1},\ \ldots,\ \boldsymbol{S}_{g_2-1},\ \tau(g_2),\ \boldsymbol{V}_{g_2},\ \boldsymbol{S}_{g_2+1},\ \ldots].$$

Here, $\tau(g_i)$ is inserted before the complete visual grid of the corresponding I frame. P-frame suffixes are appended in temporal order and do not receive independent textual timestamp markers.

This convention follows the anchor-conditioned causal structure learned in Stage 1. During Stage 1, the reconstruction decoder receives a complete I-frame grid followed by the retained P-frame suffixes in a causal prefix, and reconstructs each target P frame from that prefix. The decoder itself does not

consume textual timestamp markers, so the correspondence to the downstream interface is structural: the I frame provides the complete temporal anchor, and each P suffix contributes an incremental update conditioned on the anchor and its preceding history. The downstream MLLM therefore receives the same causal units while retaining explicit time metadata at GOP boundaries.

Attaching one textual timestamp marker per GOP also keeps temporal metadata proportional to the number of anchors. If one textual timestamp marker occupies $c_\tau$ language-model tokens, per-frame textual timestamping would add approximately $N_{\text{frame}} c_\tau$ tokens, whereas anchor-only textual timestamping adds $N_{\text{I}} c_\tau$ tokens, where $N_{\text{I}}$ is the number of I frames. The saving is especially relevant at small suffix ratios. With $M = 256$, ratio $1/64$ retains only four visual tokens for a P frame, so repeated textual timestamp markers can become comparable to or larger than the visual payload after tokenization. Anchor-only metadata preserves the temporal reference channel of the Qwen3.5-based MLLM while preventing textual timestamp overhead from offsetting the benefit of variable-length visual compression.

P frames retain temporal information through their frame-aware visual coordinates, their order in the causal stream, and the sampling interval. Under uniform sampling with interval $\Delta t$, if a P frame is $m$ admitted frames after the I-frame anchor at time $\tau(g_i)$, its temporal location is determined by

$$t_{g_i+m} = \tau(g_i) + m\Delta t.$$

Thus, the absence of an independent textual timestamp marker for a P frame does not remove its temporal position from the visual representation. The textual timestamp marker identifies the absolute start of a GOP, while the visual coordinates and sampling grid specify the relative location of each subsequent P frame.

### A.3 Parallel query decoding and attention mask

The reconstruction decoder is implemented as a packed self-attention decoder so that multiple target P frames can be reconstructed in one forward pass. For every target, a shared learnable vector is expanded to the target merge grid to form an $M$-token query canvas. The canvas receives the target frame's temporal and spatial rotary coordinates, while the visual prefix consists of the complete I-frame anchor followed by the retained P-frame suffixes. The packed sequence therefore has the form

$$[\text{I anchor},\ \text{P suffix prefix},\ \text{query canvas}_0,\ \text{query canvas}_1, \ldots].$$

The decoder predicts RGB pixels from the query canvas; the auxiliary decoder uses the same query construction to predict the target feature grid.

We use Qwen-style grouped-query self-attention with partial rotary position embeddings. The current 4B configuration has $16$ query heads, $4$ key–value heads, head dimension $256$, and two decoder layers. The attention mask is a block-causal mask rather than a dense square mask. Prefix tokens attend causally within the visual prefix. The query canvas for target P frame $t$ can attend to the complete I-frame anchor, the suffixes of P frames up to and including $t$, and its own preceding query tokens. It cannot attend to suffixes from future P frames or to query tokens belonging to another target canvas. Thus, multiple P-frame canvases are evaluated in parallel without allowing information to leak between targets; the causal history remains shared, but each target has an isolated query block.

For variable-length prefixes, the mask is constructed separately for each batch element and masks both right padding and unavailable history. During Phase 2 oracle search, the seven ratio candidates for the current P frame are packed as separate branches. Each candidate branch can read the shared history and its own candidate suffix, but it cannot read the suffix or query canvas of another ratio branch. We implement these block masks with PyTorch FlexAttention (Dong et al., 2025) and its `BlockMask` representation, avoiding materialization of a dense attention matrix.

### A.4 Training cost and module size

Phase 1 was trained for ten days on eight NVIDIA H800 GPUs, and Phase 2 was trained for four days on the same eight-H800 setup. The downstream MLLM alignment stage then took eight days on eight H800 GPUs. The three stages therefore took twenty-two days of wall-clock time in total, or approximately $4{,}224$ H800 GPU-hours.

The four-layer Incremental Encoder and the two-layer reconstruction modules deliberately use asymmetric depth. Each Qwen-style block follows the Qwen3.5-4B configuration with hidden size $d = 2560$, 16 query heads, 4 key–value heads, head dimension 256, and a SwiGLU intermediate size of 9,216. The Incremental Encoder allocates more capacity to causal aggregation of information across frames, while the shallower decoders focus on mapping the retained prefix and learned query canvas back to pixels or features. Counting all trainable weights in the current shared 4B implementation, including the upsampling head, the pixel decoder has approximately 242.85 million parameters and the auxiliary feature decoder has approximately 214.97 million parameters. Each is a single parameter-sharing module reused across the 384, 512, and 768 longest-edge configurations; changing the grid changes the query canvas but does not create a new parameter copy.

The length predictor is a lightweight MLP head analogous to the feed-forward sublayer of a Transformer block, rather than an additional attention block. For each P frame, it takes the last token of the incremental representation as input. Because this token has causally attended to all preceding tokens in the frame and to the available history, it summarizes the information needed for the length decision. The predictor maps the resulting 2560-dimensional vector through a hidden layer of width 128 and outputs seven ratio logits. Its parameter count is 328,711 (0.329M). At inference it requires only one MLP evaluation per P frame, with complexity $O(dh + hC)$ for hidden width $h = 128$ and $C = 7$ ratio classes. It adds no self-attention layer and performs no reconstruction for the length decision, so its overhead is negligible relative to the Incremental Encoder and reconstruction decoder; the seven-ratio reconstruction sweep is used only to build Phase 2 oracle targets. Because the predictor evaluates all P frames in parallel, it introduces no significant inference delay; its 0.329M parameters are approximately $0.008\%$ of the Qwen3.5-4B backbone.

## B PIXEL AND AUXILIARY LOSSES

Stage 1 combines a pixel reconstruction loss with an auxiliary feature loss. To isolate their contributions, the same encoder is trained twice, once with both losses and once with the auxiliary loss only, and the auxiliary loss is evaluated on held-out frames at each retained ratio.

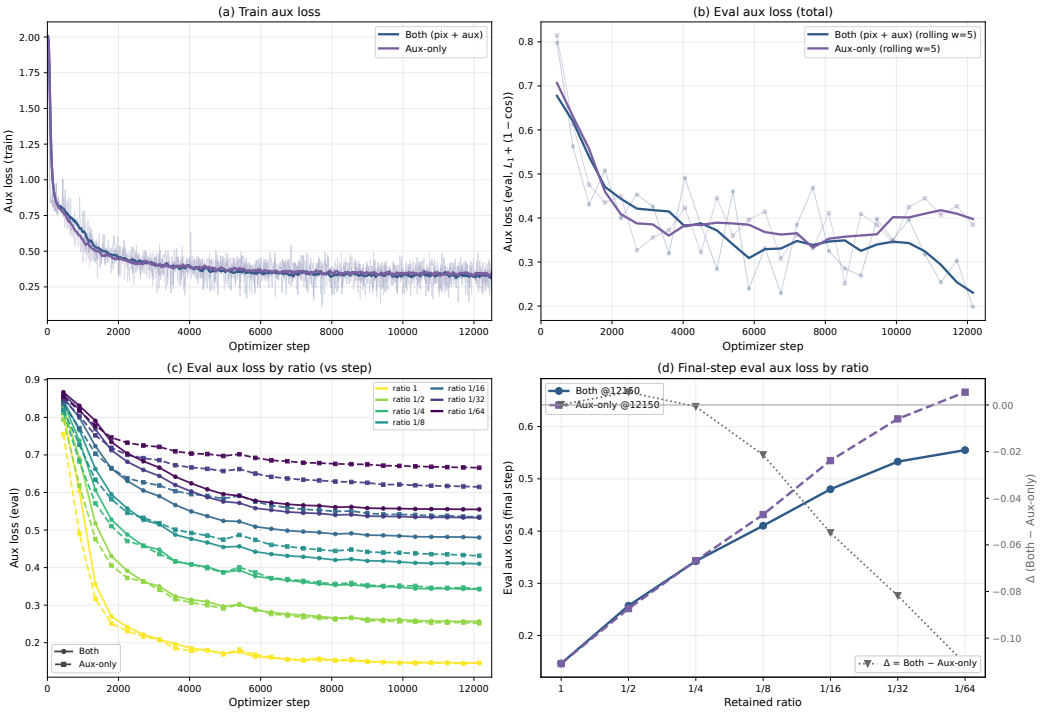


Figure 5: Pixel and auxiliary reconstruction losses under two training configurations. Panel (a) shows the auxiliary loss on the training set, panel (b) the auxiliary loss on held-out frames smoothed over five evaluations, panel (c) the held-out auxiliary loss for each retained ratio, and panel (d) the held-out auxiliary loss at the final step for each ratio together with the difference between the two configurations.

Figure 5 reports the comparison. The two configurations are indistinguishable at ratio 1, where no truncation takes place, and the advantage of the pixel loss grows as the ratio shrinks, reaching about $0.11$ lower auxiliary loss at $1/64$. Adding the pixel loss therefore improves the frame representation only where the representation is truncated. At the full ratio the training objective has no effect on held-out alignment, which indicates that the pixel loss governs where the information of a frame is placed rather than improving the representation overall.

We use the encoder trained with the composite pixel-and-auxiliary objective for downstream MLLM alignment. The auxiliary-only encoder is retained as an ablation for this reconstruction analysis; training a separate downstream MLLM with that variant would require an additional long alignment run, which was outside the available training budget.

This is the behavior the variable-length interface requires. Because the retained suffix must remain decodable without the rows that truncation removes, the pixel loss supplies the signal that keeps the last positions of a frame informative, and the effect grows exactly over the range of ratios that inference can select. The same comparison also shows why the training loss alone does not separate the two configurations, since the two curves of panel (a) nearly coincide while the held-out behavior at small ratios differs.

## C SUFFIX RATIO ABLATION

The suffix ratio sets how many tokens each predictive frame contributes to the codec stream, so its effect is visible directly in the reconstruction. Table 3 reports the reconstruction quality at each of the seven preset ratios of Equation 7, together with the $I^*$ setting that encodes every frame independently as an I frame, and Figure 6 shows the same frames reconstructed at each of them. Quality decreases monotonically as the ratio shrinks, and the causal stream exceeds the independent I-frame encoding at the full ratio and stays close to it down to $1/4$, which shows that the retained suffix builds on the causal context.

Table 3: Reconstruction quality at each suffix ratio of Equation 7, measured on the held-out set. $I^*$ encodes every frame independently as an I frame, so no ratio applies to it.

| Suffix ratio $r$ | $I^*$ | 1 | $1/2$ | $1/4$ | $1/8$ | $1/16$ | $1/32$ | $1/64$ |
|---|---|---|---|---|---|---|---|---|
| PSNR (dB) | 18.46 | 18.76 | 18.57 | 18.44 | 18.27 | 18.18 | 17.80 | 17.54 |




Figure 6: Reconstruction of the same frames at each preset suffix ratio of Equation 7. The first row shows the ground-truth frames, and each remaining row shows the reconstruction when every predictive frame keeps the suffix ratio marked on the left.

# D LENGTH POLICY ANALYSIS

The length predictor answers the second requirement of Section 2: it decides how many tokens each predictive frame keeps, and it makes that decision in a single forward pass. This appendix examines the policy from two directions, following its decisions frame by frame on representative clips in Figure 7 and varying the sampling rate under a controlled setting in Figure 8.

On the six clips of Figure 7, a static clip keeps its anchor at the full budget and compresses every predictive frame to the smallest ratio, while clips with faster motion move individual frames to larger ratios and open additional anchors once adding another suffix would exceed $M$. Token retention rises from $7.6\%$ to $53.7\%$ across the six clips, so the policy concentrates tokens on the frames that change rather than distributing them uniformly over the clip.

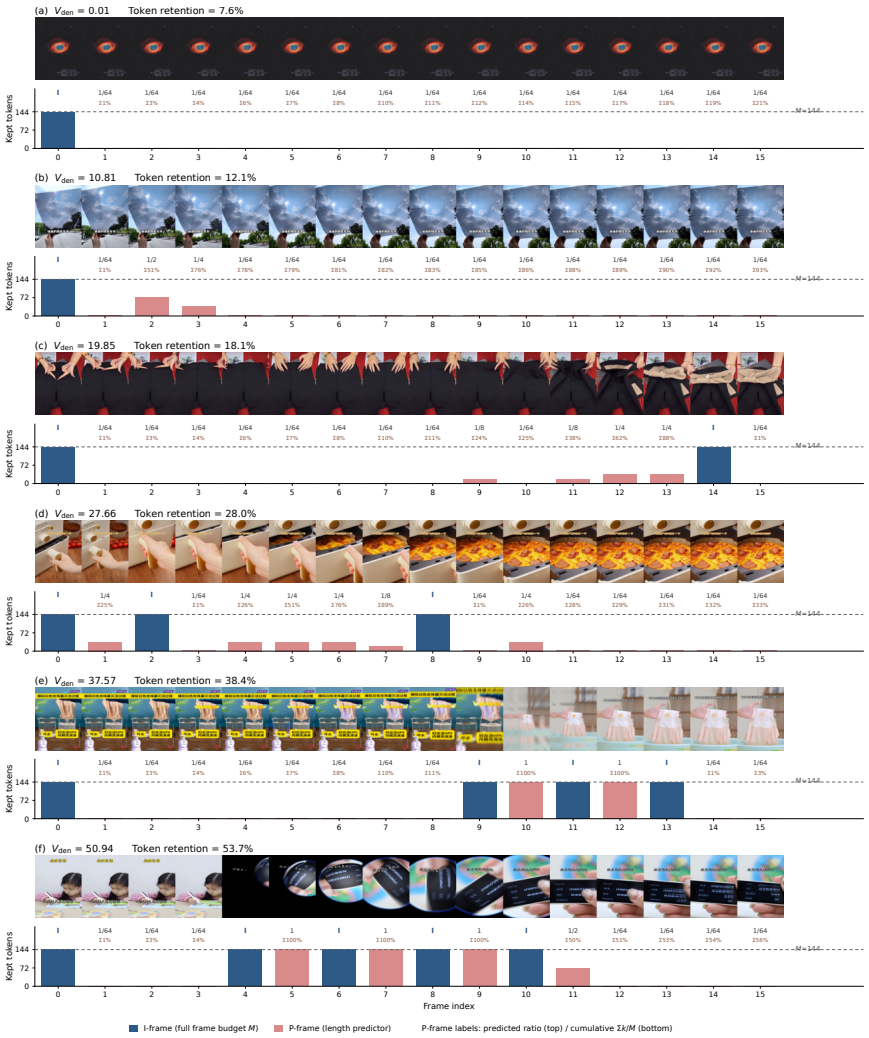


Figure 7: Per-frame token allocation on six representative clips, ordered by the measured change rate $V_{\text{den}}$. Each panel shows the frames of a clip, the tokens kept per frame, and the labels of the length policy: the anchor or predictive type of each frame, its predicted suffix ratio, and the cumulative suffix cost of the current group as a fraction of the full frame budget $M$.

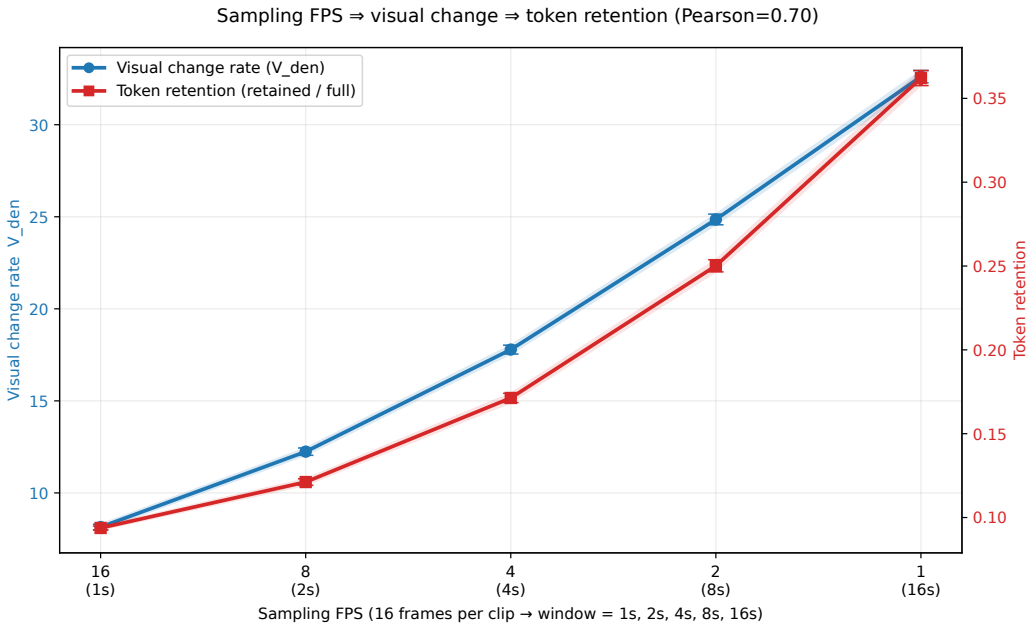


Figure 8: Sampling rate and token retention under a controlled setting. Each of $1{,}000$ clips is sampled at five rates with the frame count held fixed at sixteen, so only the temporal spacing of the frames changes. The horizontal axis gives the sampling rate and the window length it implies, the left axis the measured change rate $V_{\text{den}}$, and the right axis the fraction of tokens that the length policy retains.

Figure 8 reaches the same conclusion from the other side. Coarser sampling raises the measured change rate $V_{\text{den}}$ from $8.15$ to $32.61$ and the fraction of tokens that the policy retains from $9.4\%$ to $36.2\%$, and the two quantities reach a correlation of $r = 0.70$ over the sampled points.

## E RECONSTRUCTION TOLERANCE SENSITIVITY

We study how the Stage 2 reconstruction tolerance affects the length policy while keeping the Stage 1 representation fixed. For each setting, only the Stage 2 length predictor is retrained with the corresponding $\delta$. The predicted retention fraction, denoted by `pred_frac`, is the average fraction of full-grid visual tokens retained by the predicted policy, and the oracle retention is the corresponding fraction selected by the sequential reconstruction oracle. We report the equivalent compression factor as the reciprocal of the retention fraction. The PSNR gap is the measured magnitude of the gap between the predicted reconstruction and the full-grid teacher.

Table 4: Sensitivity of the Stage 2 length policy to the reconstruction tolerance $\delta$. All settings reuse the same Stage 1 checkpoint; only the length predictor is retrained.

| $\delta$ | Predicted retention | Oracle retention | Predicted compression | Oracle compression | PSNR gap |
|---|---|---|---|---|---|
| 0.0 dB | 16.00% | 19.06% | 6.25× | 5.25× | 0.28 dB |
| 0.3 dB | 11.20% | 12.80% | 8.93× | 7.81× | 0.23 dB |
| 0.5 dB | 5.80% | 8.50% | 17.24× | 11.76× | 0.31 dB |
| 0.8 dB | 4.28% | 5.50% | 23.36× | 18.18× | 0.25 dB |

As $\delta$ increases, the oracle permits a larger reconstruction slack and the policy retains fewer visual tokens, reducing the predicted retention fraction from $16.00\%$ to $4.28\%$. This corresponds to an increase in the predicted compression factor from $6.25\times$ to $23.36\times$. The measured PSNR gap remains between $0.23$ and $0.31$ dB across the tested settings, indicating that the retrained predictor follows the oracle policy while trading reconstruction slack for a more compact visual stream.

# F BALLBENCH DETAILS

BallBench is generated by a deterministic simulator rather than recorded from real video, which lets every frame carry exact ground truth and lets one episode be rendered at any frame rate. Each episode places several balls in a square box and advances them with frictionless, perfectly elastic collisions, so the motion follows from the initial state alone. Radii, speeds, masses, and initial configurations are drawn from a seeded generator, and every simulation is checked for conservation of momentum and kinetic energy, with episodes that exceed the numerical tolerance discarded and regenerated. A split is therefore reproducible from its seed and episode index alone, and the training and evaluation splits are drawn from different seeds, so no video appears in both.

Each video has three phases: an initial labeled phase in which every ball carries a unique number, an unlabeled motion phase in which the balls are visually identical and appearance carries no identity, and a final labeled phase in which the same numbers are reassigned as a derangement, which guarantees that no ball keeps its initial number. A question names one initial number and asks for the final number of the same physical ball, so the answer can only be reached by following the ball through every collision and bounce. The phases last two, four to eight, and two seconds at thirty frames per second and $512 \times 512$ resolution, so a video runs eight to twelve seconds.

The training split contains 6,000 videos with four to six balls, which yields 30,000 questions, and the evaluation split contains 70 videos with four to ten balls, which yields 490 questions. Training questions are accompanied by a tracking rationale written in directional language. The rationale states no coordinates: it describes positions with regions of the box and directions with compass phrases, and its timestamps fall on the sampling grid, so every moment it mentions is a frame the model sees. The rationale is what makes the benchmark learnable, since training on the same questions without it stays at the level of random guessing, about $14\%$ accuracy.

Figure 9 shows one training episode at sampled frames, and Section G shows the complete sample built from it. Six balls move in the box, the labels disappear when the motion phase starts, and the derangement at the end assigns a new set of numbers; the question names the ball that carried the initial label $1$ and the ground truth is label $3$. The training input pairs that question with a rationale in directional language: the rationale anchors the ball at the moment the labels disappear, follows it segment by segment through each collision and wall rebound, and closes with the identity chain that links the two labels. The answer target states the same conclusion in one sentence and ends with the label, so the supervision covers both the intermediate chain and the final label, and the rationale of this episode runs to $4{,}479$ characters.

## G BALLBENCH TRAINING SAMPLE

Every training question is paired with the complete sample shown in Figure 10, which is built from the episode of Figure 9. The prompt states the three phases of an episode and the tracking procedure, the response renders the chain of thought inside <think> tags, and the answer follows the closing tag. The prompt is shown in black, the <think> block in blue, and the answer in red.

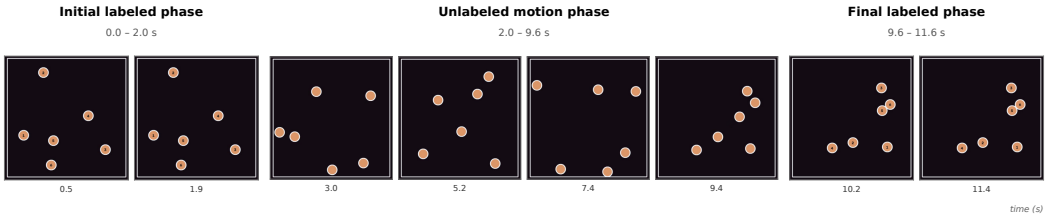


Figure 9: One BallBench sample with six balls, shown at sampled frames from the three phases of the episode.

**BallBench training sample**

Prompt · <think> block · Answer

```
The video shows several balls inside a square box, and it has three
consecutive phases:
1.  Initial labeled phase (the beginning):  each ball shows a number
printed on it.  The numbers are 1, 2, 3, ..., with one unique number
per ball - no two balls share the same number.  These numbers identify
the balls.
2.  Unlabeled motion phase (the main part):  the numbers disappear and
the balls move around, colliding with each other and bouncing off the
box walls.  During this phase all balls look exactly the same (same
size, same color), so the ONLY way to tell them apart is to follow each
ball continuously from the initial phase.
3.  Final labeled phase (the end):  the balls stop moving and each ball
shows a NEW number printed on it.  The new numbers are exactly the SAME
set of numbers reassigned among the balls:  every ball now shows a
number that is different from the one it had initially.
Rules to keep in mind:
- A number always stays attached to its own ball.
- Whenever the question says 'ball N', it means the ball that showed
number N in the initial labeled phase.
- Answer using only what you can see in the video.
Your task:  object identity tracking.  The question names one ball by
its initial number.  During the unlabeled motion phase all balls look
exactly the same, so appearance gives no clue - the only reliable
method is continuous spatial tracking, step by step:  (1) note where
the labeled ball is at the start; (2) follow it as it moves, moment by
moment, always matching it to its most recent position and direction;
(3) at collisions and wall bounces the direction of motion changes, so
keep following the same ball through the change; (4) at the final
frozen frame, read the new number printed on the ball you have tracked.
The answer is a single number.
Question:  Which final label corresponds to ball 1 from the beginning?
Please provide your final answer strictly following this format:
Answer:  \boxed{xxx}
<think>
Task:  ball 1 is the ball that showed the number 1 in the initial
labeled phase.  I must follow this one ball through the unlabeled
motion phase and report the new number it shows at the end.
Method:  during the motion phase all balls look identical, so I will
track ball 1 by position and direction only.  I anchor it at the start,
```

follow its motion segment by segment through every collision and wall bounce, and finally read its new label on the frozen final frame.
Anchor at 0.00s: ball 1 starts in the left-center area, very close to the left wall, roughly halfway between the center and the left wall. Its closest neighbors are ball 5, to its right and ball 6, up and to its right.
The labels disappear at 2.00s and the balls begin to move; from here on the balls cannot be told apart by appearance.
From 2.00s to 3.12s, ball 1 moves steadily to the left and slightly upward.
At 2.50s, it is in the left-center area, very close to the left wall, roughly halfway between the center and the left wall.
At 3.00s, it is in the left-center area, very close to the left wall, roughly halfway between the center and the left wall.
Between 3.12s and 3.19s, ball 1 reaches the left wall and rebounds; it was moving to the left and slightly upward and now moves to the right and slightly upward.
From 3.19s to 3.25s, it moves steadily to the right and slightly upward.
Between 3.25s and 3.31s, ball 1 collides with another ball that was moving to the left, in the left-center area, very close to the left wall; the other ball flies off downward and slightly to the right, while ball 1 bounces and now moves up and to the left.
Between 3.31s and 3.38s, ball 1 reaches the left wall and rebounds; it was moving up and to the left and now moves up and to the right.
From 3.38s to 5.94s, it moves steadily up and to the right.
At 3.50s, it is in the left-center area, very close to the left wall, roughly halfway between the center and the left wall, still moving up and to the right.
At 4.00s, it is in the left-center area, very close to the left wall, roughly halfway between the center and the left wall, still moving up and to the right.
At 4.50s, it has moved from the left-center area, very close to the left wall into the left-center area, roughly halfway between the center and the left wall, still moving up and to the right.
At 5.00s, it is in the left-center area, roughly halfway between the center and the left wall, still moving up and to the right.
At 5.50s, it has moved from the left-center area into the top-center area, roughly halfway between the center and the top wall, still moving up and to the right.
Between 5.94s and 6.00s, ball 1 collides with another ball that was moving to the left and slightly upward, in the top-center area; the other ball flies off up and to the left, while ball 1 bounces and now moves to the right.
From 6.00s to 9.12s, it moves steadily to the right.
At 6.50s, it is in the top-center area, roughly halfway between the center and the top wall, still moving to the right.
At 7.00s, it is in the top-center area, roughly halfway between the center and the top wall, still moving to the right.
At 7.50s, it is in the top-center area, roughly halfway between the center and the top wall, still moving to the right.
At 8.00s, it is in the top-center area, roughly halfway between the center and the top wall, still moving to the right.
At 8.50s, it has moved from the top-center area into the top-right area, roughly halfway between the center and the top wall, still moving to the right.
At 9.00s, it is in the top-right area, roughly halfway between the center and the right wall, closing in on another ball, still moving to the right.

```
Between 9.12s and 9.19s, ball 1 collides with another ball that was
moving to the left and slightly downward, in the top-right area; the
other ball flies off downward and slightly to the left, while ball 1
bounces and now moves upward and slightly to the left.
From 9.19s to 9.62s, it moves steadily upward and slightly to the left.
At 9.50s, it is in the top-right area, roughly halfway between the
center and the right wall.
At 9.62s the balls freeze and new labels appear.
The ball I have tracked comes to rest in the top-right area; its last
known direction was upward and slightly to the left, which is
consistent with this final position.
It now shows the new label 3.
Identity chain:  the ball that started as ball 1 is the same physical
ball that now shows label 3.
</think>
Putting the whole trajectory together, ball 1 ended up carrying label 3.
Answer:  \boxed{3}
```

Figure 10: The complete training sample of BallBench for the episode of Figure 9, spanning the prompt, the <think> chain of thought, and the answer.

The prompt supplies the phase structure and the procedure, while the rationale supplies the frame-by-frame chain, so the model is told what the episode contains and how to follow the ball, and is then trained to produce the chain itself. The rationales are generated by replaying the deterministic simulation and verbalizing the tracked trajectory, which keeps them consistent with the physics of the episode. At inference the prompt is unchanged, and the model answers with the same format it saw during training.