Technical Appendices and Supplementary Material

In the supplementary materials, we provide the following additional details:

- **Sec. A** The comprehensive **hyper-parameters and training costs** for AdapTok.
- **Sec. B** The detailed implementation for the attention mask, quantization method and several other **adaptive inference strategies**, such as Fixed, BiThr and BiDelta token allocation strategies.
- **Sec. C** More **ablation experiments** on the token scoring metrics, mini-batch size, block number, and average token count.
- **Sec. D Human evaluation** for reconstruction and generation.
- **Sec. E** More **qualitative visualizations**, including
  - **Sec. E.1** More adaptive reconstruction results (Fig. C-D) as well as visualizations highlighting AdapTok's ability to perform content-aware (Fig. E) and temporally dynamic (Fig. F) token allocation;
  - **Sec. E.2** Video generation results on UCF-101 class-conditional generation (Fig. G) and Kinetics-600 frame prediction (Fig. H);
  - **Sec. E.3** More attention maps for latent tokens (Fig. I).
  - **Sec. E.4** The visualization of scorer predictions and corresponding reconstruction results under varying token counts (Fig. J).

A. Implementation Details

The detailed training hyper-parameter settings for the AdapTok, Scorer, AdapTok-AR (class-conditional generation on UCF-101) and AdapTok-FP (frame prediciton on Kinetics-600) are reported in Table A. The architectural configurations of the different AdapTok variants are listed in Table B.

B. More Technical Details

B.1. Attention Mask

As shown in Fig. A, the dropout mask m' is applied to the attention patterns of the encoder, decoder, and scorer. It masks out tokens at the block level, ensuring that only retained tokens are used in subsequent processing and ILP-based token allocation.

B.2. Quantization Method

Following [2], a stochastic vector quantization (SVQ) is adopted to the quantizer $\mathcal{Q}$. Similar to VQ, a codebook $C \in \mathbb{R}^{c \times d'}$ containing c codes is maintained. Given a visual feature z, the cosine similarity with all code vectors in C is computed, followed by a softmax to obtain a probability distribution. An index x_{Ind} is then sampled from this distribution via a categorical distribution:

$$x_{\text{Ind}} \sim \text{Categorical}\left(\text{softmax}\left(\left\{\frac{v \cdot C_i}{\|v\|\|C_i\|}\right\}_{i=1}^{c}\right)\right). \tag{1}$$

Given the sampled index, the quantized feature z_q is retrieved from the codebook, i.e., $z_q = C_{x_{\text{Ind}}}$. To enable differentiable training, the straight-through estimator [1] is applied.

B.3. Adaptive Inference

In addition to the Integer Linear Programming (ILP) method detailed in the main paper, we provide further details for the other three token allocation methods, including Fixed, BiThr and BiDelta.

Fixed token allocation uses the same number of tokens across all video samples and blocks. The latent mask is given by:

$$m' = [m_1 \oplus m_2 \oplus \cdots \oplus m_K\,],\ m_i = [\mathbb{1}_{j \le N_b}]_{j=1}^{M}. \tag{2}$$

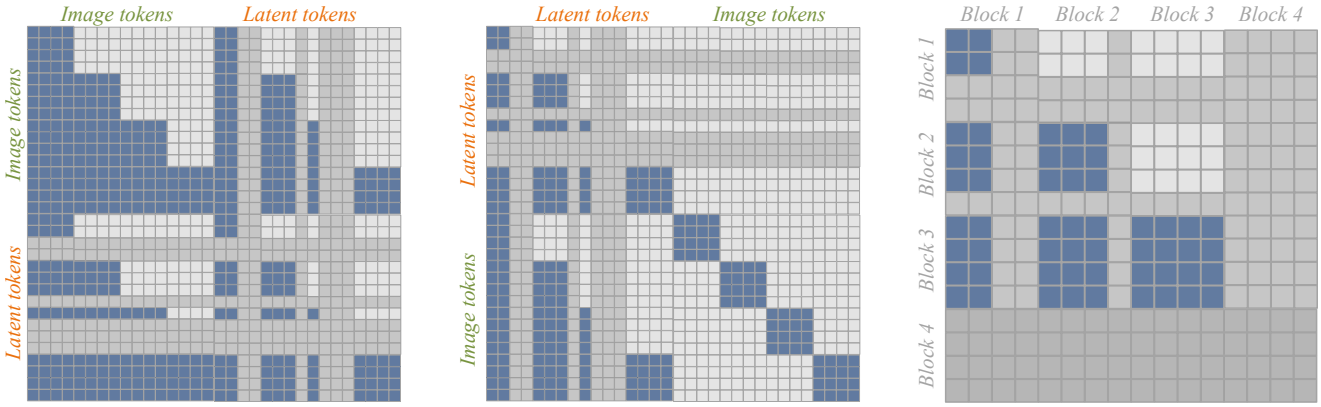
where N_b is the fixed number of tokens allocated per block.

Score-threshold binary search (BiThr) binary searches a global score threshold which assigns the minimum token counts that satisfy a desired video quality. Specifically, given a score threshold s_i, the token counts are assigned by selecting the first position where the score drops below s_i. The threshold is iteratively updated via binary search until the average token count matches the target value, as detailed in Algorithm 1.

Delta-score binary search (BiDelta) follows the same binary search procedure as BiThr, but operates on delta scores instead of raw scores. These delta scores, computed

Table A. **Hyper-parameters for AdapTok models.**

	AdapTok	Scorer	AdapTok-AR	AdapTok-FP
Model parameters				
Parameters	195M	89M	633M	633M
Frame Resolution	$16 \times 128 \times 128$	$16 \times 128 \times 128$	$16 \times 128 \times 128$	$16 \times 128 \times 128$
Patch Size	$4 \times 8 \times 8$	$4 \times 8 \times 8$	$4 \times 8 \times 8$	$4 \times 8 \times 8$
Hidden Size	768	768	1280	1280
Transformer Layers	12	12	30	30
Training				
Optimizer	Adam	Adam	AdamW	AdamW
Learning rate	$1e^{-4}$	$1e^{-4}$	$6e^{-4}$	$6e^{-4}$
Beta1	0.5	0.5	0.9	0.9
Beta2	0.9	0.9	0.95	0.95
Weight decay	0	0	0.05	0.05
Scheduler type	cosine	cosine	cosine	cosine
Warmup epochs	8	8	4	1
Batch size	128	128	64	64
Epochs	250	20	3000	75
GPUs	32	32	8	16
Training Time	112h	9h	59h	55h



(i) Encoder Attention Mask (ii) Decoder Attention Mask (iii) Scorer Attention Mask

Figure A. **Attention masks in AdapTok.**

Table B. **Model configurations of AdapTok variants.**

Model	Hidden Size	Depth	Heads	Parameters
AdapTok-S	512	6	8	59M
AdapTok-L	768	12	12	259M
AdapTok-XL	1024	24	16	913M

using a difference function (see Algorithm 1), quantify the marginal gain in perceptual quality from adding each token. The search aims to find a threshold over these deltas such that the resulting token allocation meets the desired token count.

C. Additional Experiments

Ablation on token allocation scoring metrics. Fig. B presents detailed comparisons of token allocation under varying token lengths for different scoring metrics. Each metric achieves the best performance on its corresponding evaluation metric, while perceptual loss consistently yields strong results across multiple metrics, demonstrating its ef-

Table C. **Ablation on mini-batch size with an average token of 512.**

Batch size	FVD ↓	PSNR ↑	LPIPS ↓	Time (ms/video) Total	Time (ms/video) IPAL	IPAL Time Proportion (%)
8	60.39	24.01	0.146	66.3	28.0	42.2
16	60.14	24.04	0.145	53.7	16.7	31.2
64	59.96	24.06	0.144	45.2	8.6	19.1
128	60.61	24.07	0.144	44.1	7.0	15.8
256	61.48	24.08	0.144	44.0	6.7	15.2
512	61.44	24.08	0.144	43.0	6.3	14.8
1024	61.37	24.09	0.144	44.1	6.8	15.5

Table D. **Ablation on mini-batch size with an average token of 1024.**

Batch size	FVD ↓	PSNR ↑	LPIPS ↓	Time (ms/video) Total	Time (ms/video) IPAL	IPAL Time Proportion (%)
8	36.55	25.68	0.115	69.0	28.1	40.7
16	36.80	25.71	0.114	56.8	16.9	29.8
64	36.36	25.72	0.114	48.1	8.5	17.8
128	36.52	25.73	0.114	46.7	7.2	15.4
256	37.02	25.74	0.114	47.4	6.9	14.6
512	36.93	25.74	0.113	46.9	7.0	14.8
1024	37.01	25.75	0.113	47.3	7.5	15.8

Table E. **Runtime ablation of IPAL on block and token counts.**

(a) Number of blocks.

Blocks	Tokens	IPAL Time (ms/video)
1	1024	7.9
2	1024	7.1
4	1024	8.0
8	1024	10.6

(b) Average tokens counts.

Blocks	Tokens	IPAL Time (ms/video)
4	32	7.8
4	128	8.5
4	512	8.3
4	1024	8.0

fectiveness for guiding adaptive token allocation.

Algorithm 1: Binary Search

Inputs: Video x, Block idx i;
Hyperparameters: Average Token N_b, tokens per block M, max iterations K;
$z = \mathcal{E}(x)$, $z_q = \mathcal{Q}(z)$;
$$s = \begin{cases} \mathcal{S}_\phi(z, z_q)_{iM:(i+1)M}, & \text{(BiThr)} \\ \mathcal{S}_\phi(z, z_q)_{iM:iM+M-1} \\ \quad -\mathcal{S}_\phi(z, z_q)_{iM+1:iM+M}, & \text{(BiDelta)} \end{cases}$$
$s_{max}, s_{min} = \max(s), \min(s)$;
for $k = 1, \cdots, K$ **do**
  $s_{mid} = (s_{max} + s_{min})/2$;
  $n_k = \text{argmax}(s < s_{min})$;
  if $\text{mean}(n_k) > N_b$ **then**
    $s_{min} = s_{mid}$;
  else
    $s_{max} = s_{mid}$;
Return: assigned tokens n_k;

Further analysis on computational costs. Although IPAL is based on ILP, its actual runtime overhead is not significant. Table C-E shows the runtime scalability of IPAL *w.r.t.* mini-batch size, the number of blocks and average token numbers. Specifically, across batch sizes ranging from 8 to 1024, FVD remains stable between 36 and 37, while IPAL accounts for only about 15% of total inference time. The allocation time also remains stable with varying average token numbers, while it increases moderately as the number of blocks grows. These results suggest that our method is efficient, scalable, and practical for real-world deployment scenarios.

D. Human Evaluation

We conduct a human study where evaluators rate videos from 1 to 10 on frame texture and motion coherence for both tasks, and semantic relevance for generation only. Ta-

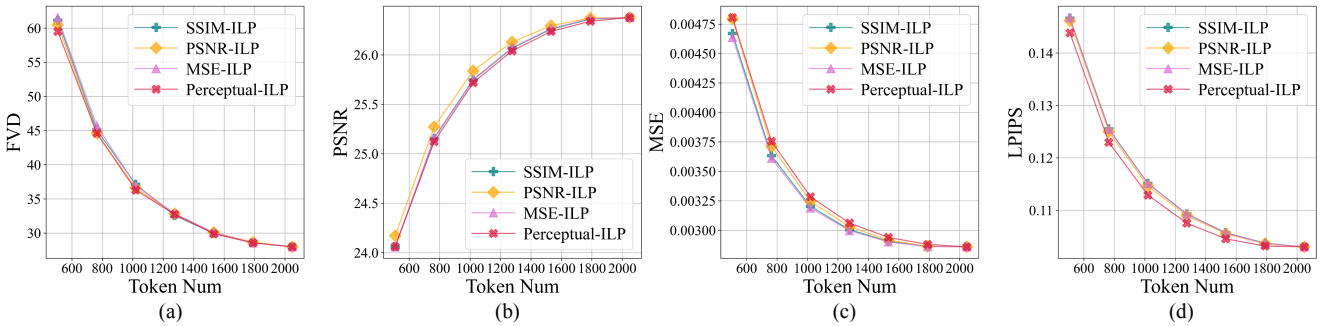


Figure B. **Comparison of scoring metrics.** SSIM, PSNR, MSE, and perceptual loss are evaluated on (a) FVD, (b) PSNR, (c) MSE, and (d) LPIPS. Perceptual loss achieves better overall performance, especially on FVD and LPIPS.

ble F indicates that AdapTok outperforms baselines on both tasks.

Table F. **Human evaluation results.**

$\mathcal{S}_{Recon}$	OmniTokenizer (8.2)	ElasticTok (6.7)	CausalTok (8.7)	AdapTok (**8.8**)
$\mathcal{S}_{Gen}$	OmniTokenizer (5.6)	TATS (4.0)	CausalTok-AR (7.1)	AdapTok-AR (**7.9**)

E. More Visualizations

E.1. Video Reconstruction

1D latent token space matters. In Fig. C, we visualize reconstruction results under different token budgets. Compared to ElasticTok, which relies on local 2D spatial tokens and tends to produce block-like artifacts, AdapTok leverages 1D latent space that enables a coarse-to-fine reconstruction process: early tokens capture global structure, while later tokens refine local details.

Video reconstruction comparison. Fig. D shows comparisons between the original video and the reconstructed video.

Content-aware allocation. As shown in Fig. E, AdapTok adaptively allocates tokens based on the visual complexity of each scene. Static or low-motion segments receive fewer tokens, while dynamic or visually complex regions are assigned more, enabling efficient token usage without compromising important content.

Temporal dynamics. Fig. F illustrates how AdapTok adaptively allocates tokens over time. When scene changes, more tokens are assigned to capture the transition, while fewer tokens are used in stable or redundant segments.

E.2. Video Generation

We present class conditional generation results on UCF-101 in Fig. G and frame prediction results on Kinetics-600 in Fig. H.

E.3. Token Contribution Analysis

Fig. I provides additional attention maps of latent tokens from various samples. These further demonstrate that early tokens tend to attend broadly to capture global context, while mid-to-late tokens focus more on local details.

E.4. Scorer Predictions

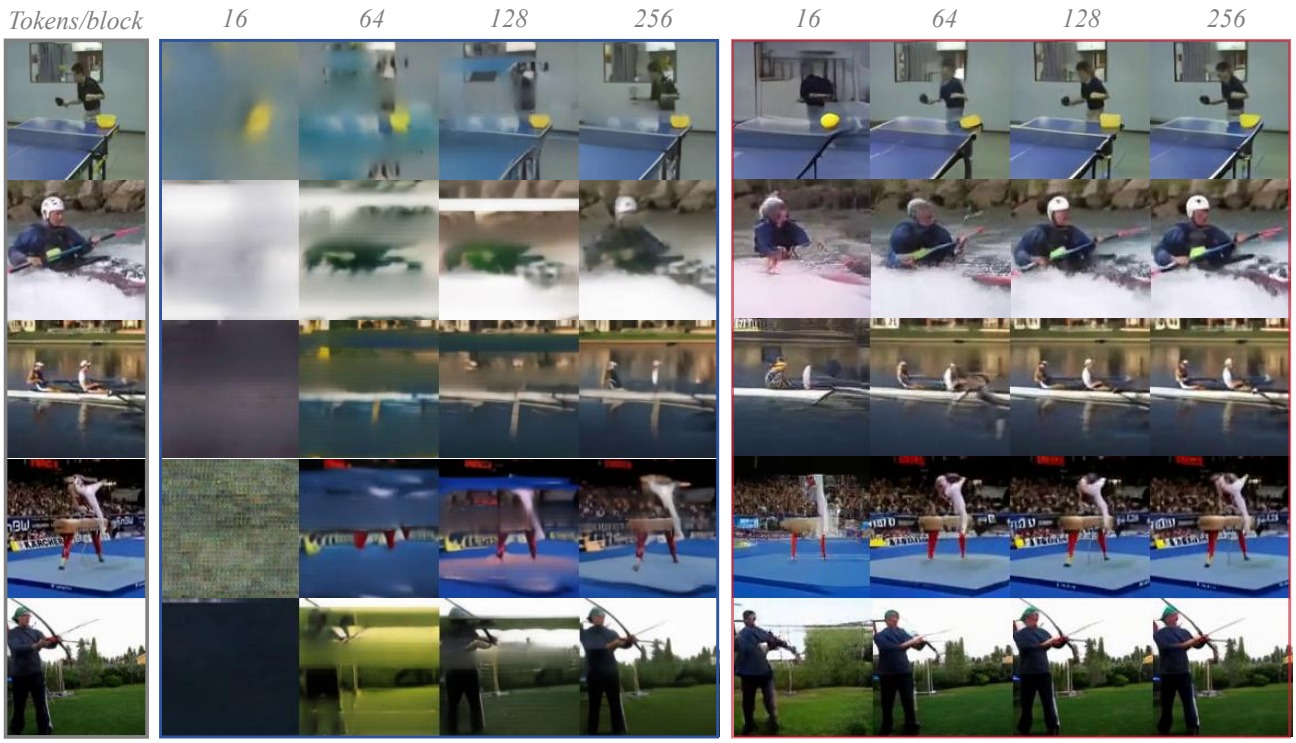
Fig. J illustrates the scorer's prediction quality and the effect of increasing token counts on reconstruction. As shown in Fig. Ja, the predicted scores closely match the ground truth scores (perceptual loss) across all blocks. As more tokens are allocated within each block (moving along the x-axis), the score decreases, indicating improved reconstruction quality. This trend is visually confirmed in Fig. Jb, where reconstructions become progressively more accurate as token counts increase.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 1

[2] Hanyu Wang, Saksham Suri, Yixuan Ren, Hao Chen, and Abhinav Shrivastava. Larp: Tokenizing videos with a learned autoregressive generative prior. *arXiv preprint arXiv:2410.21264*, 2024. 1

[3] Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video. *arXiv preprint arXiv:2410.08368*, 2024. 5



(a) GT (b) ElasticTok (c) AdapTok

Figure C. **Reconstruction comparison with ElasticTok [3].** AdapTok reconstructs videos in a coarse-to-fine manner, where early tokens capture global structure and later ones refine local details, while ElasticTok suffers from blocky artifacts.

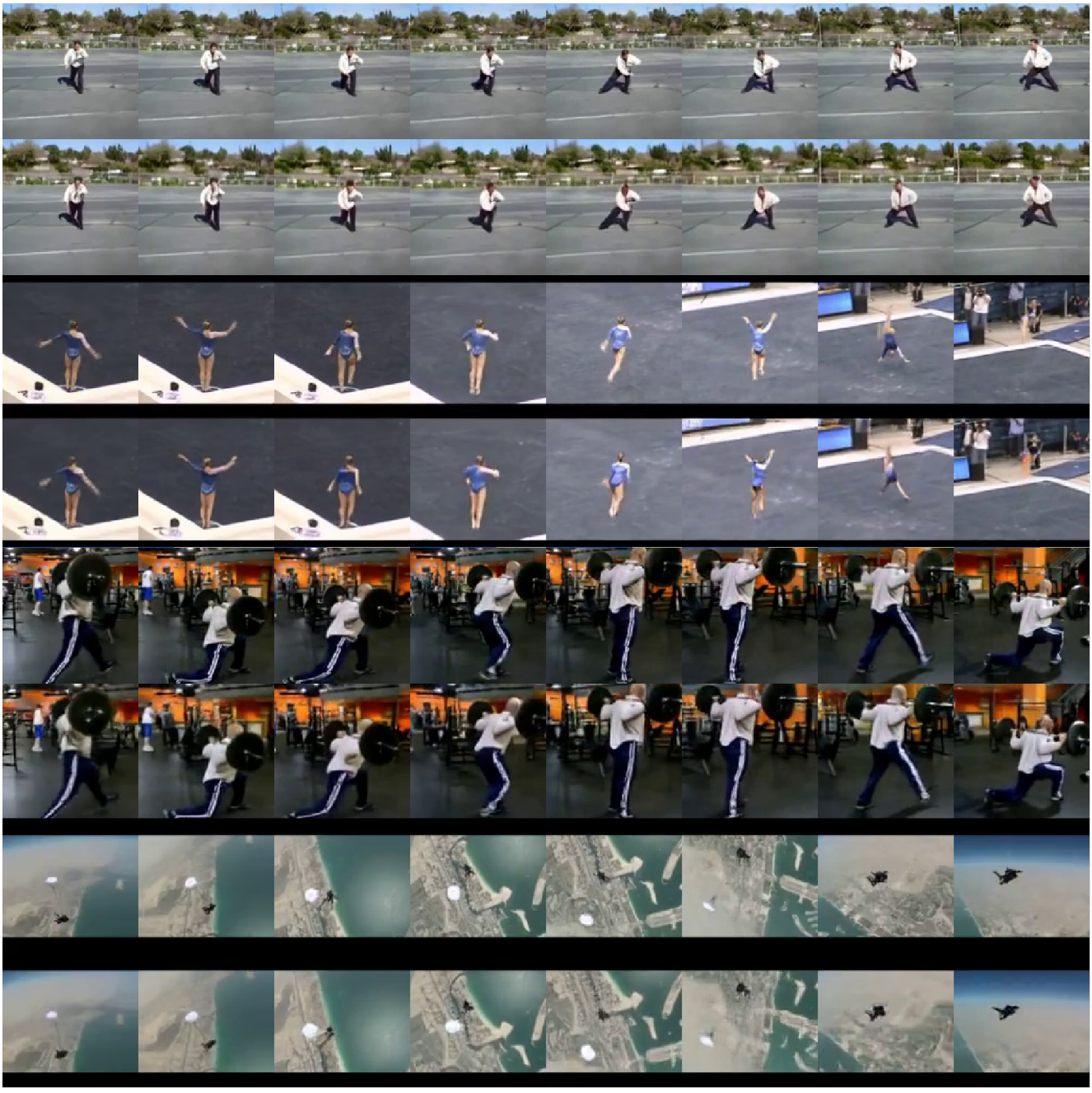


Figure D. **Comparison between ground-truth videos and reconstructed videos.** The top row shows the ground-truth videos, and the bottom row shows the reconstructions.

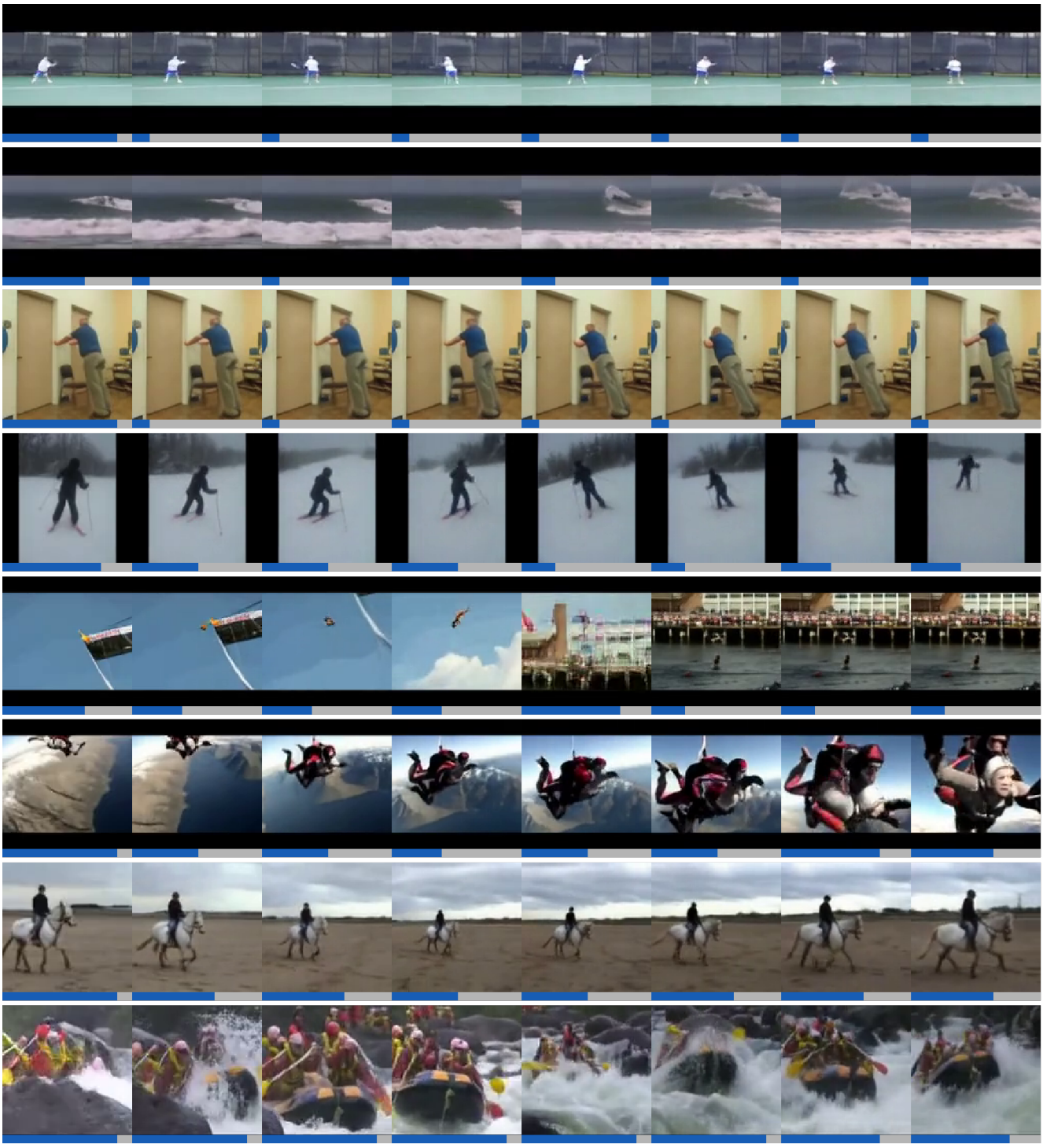


Figure E. **AdapTok performs adaptive content-aware tokenization.** From the top to the bottom, AdapTok allocates more tokens as scene complexity and motion increase. Blue bars represent the token counts used per block.

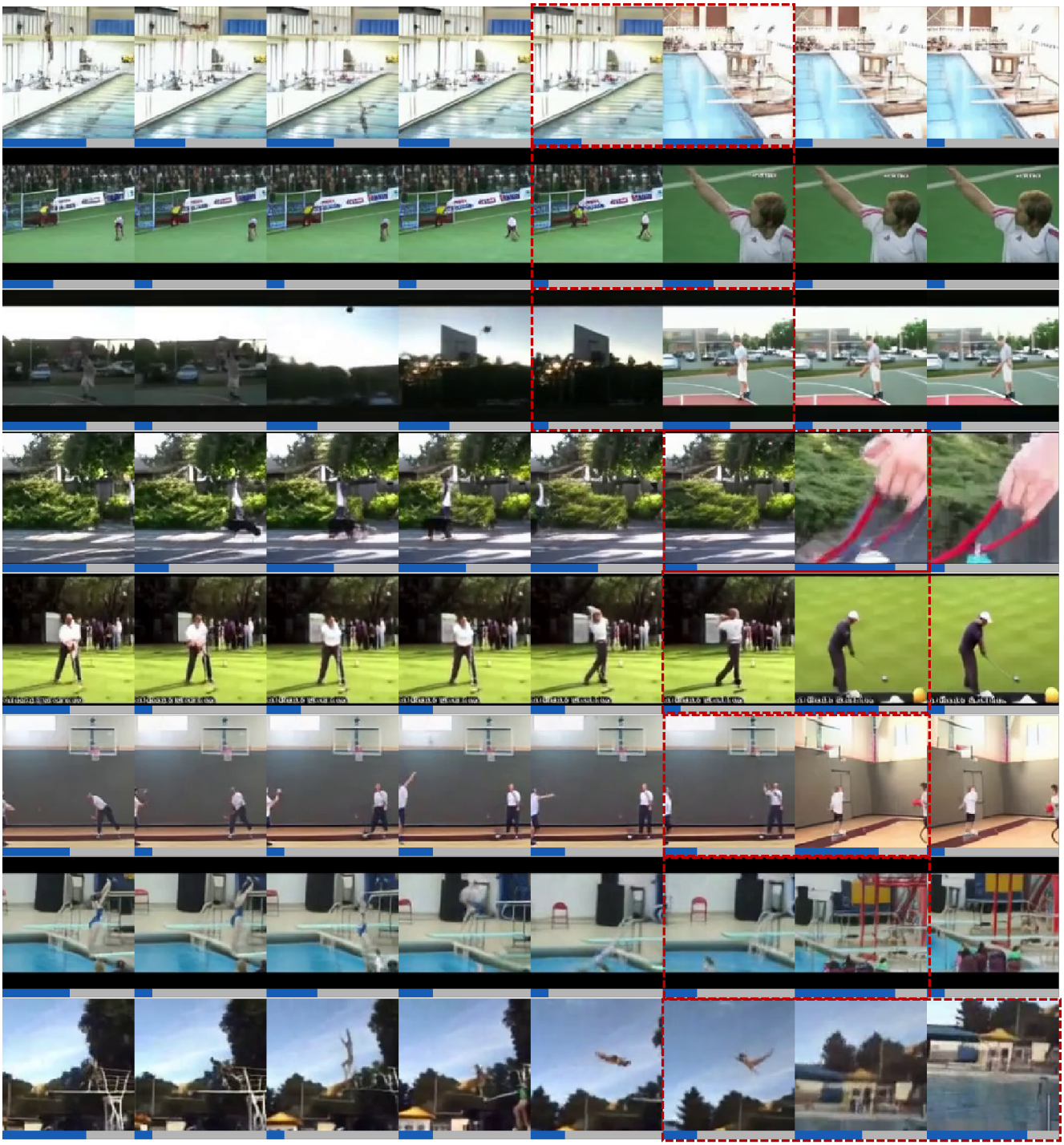


Figure F. **AdapTok adaptively allocates tokens based on temporal dynamics.** Red boxes highlight the scene transitions, where AdapTok allocates more tokens to capture important temporal changes.

Figure G. **Class conditional video generation on UCF-101.**

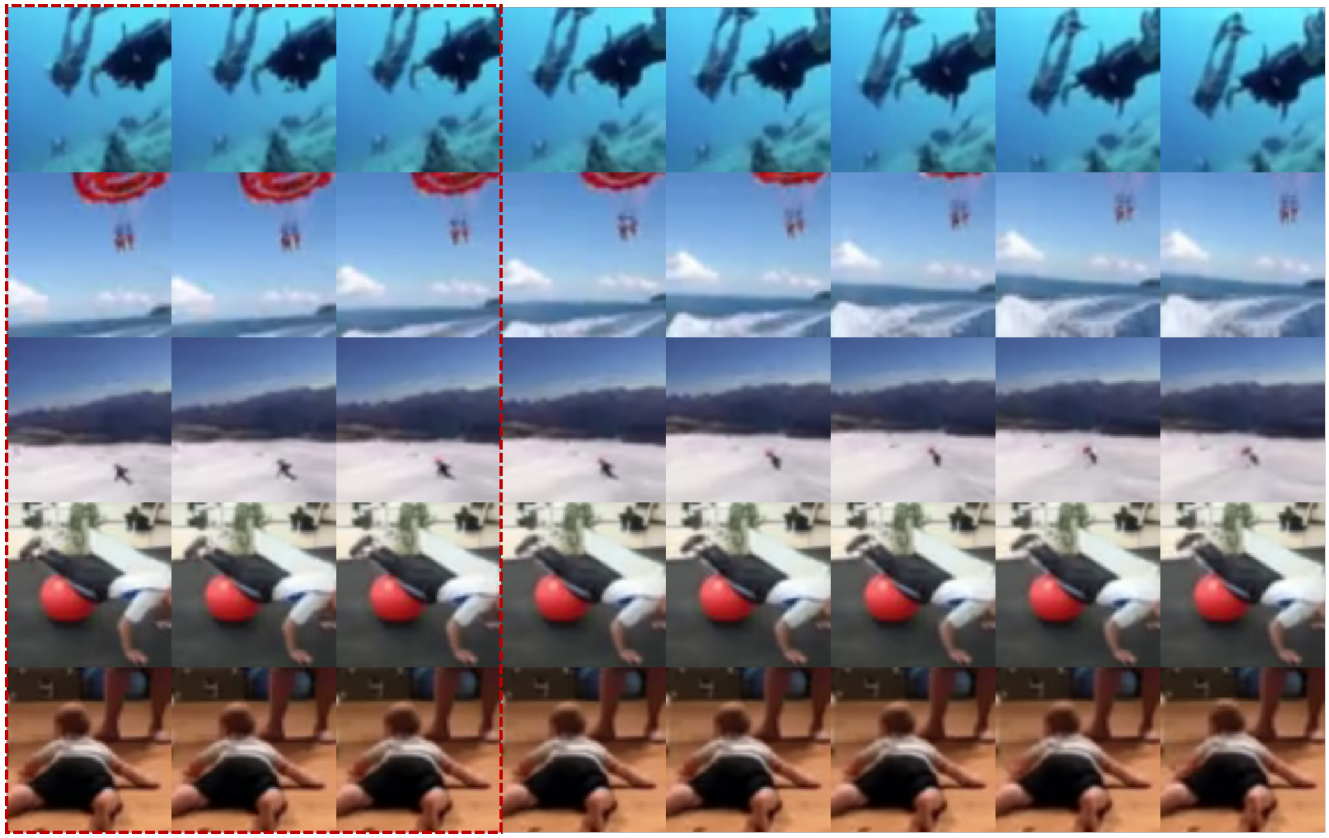


Figure H. **Frame prediction results on Kinetics-600.** Red boxes represent the condition frames.

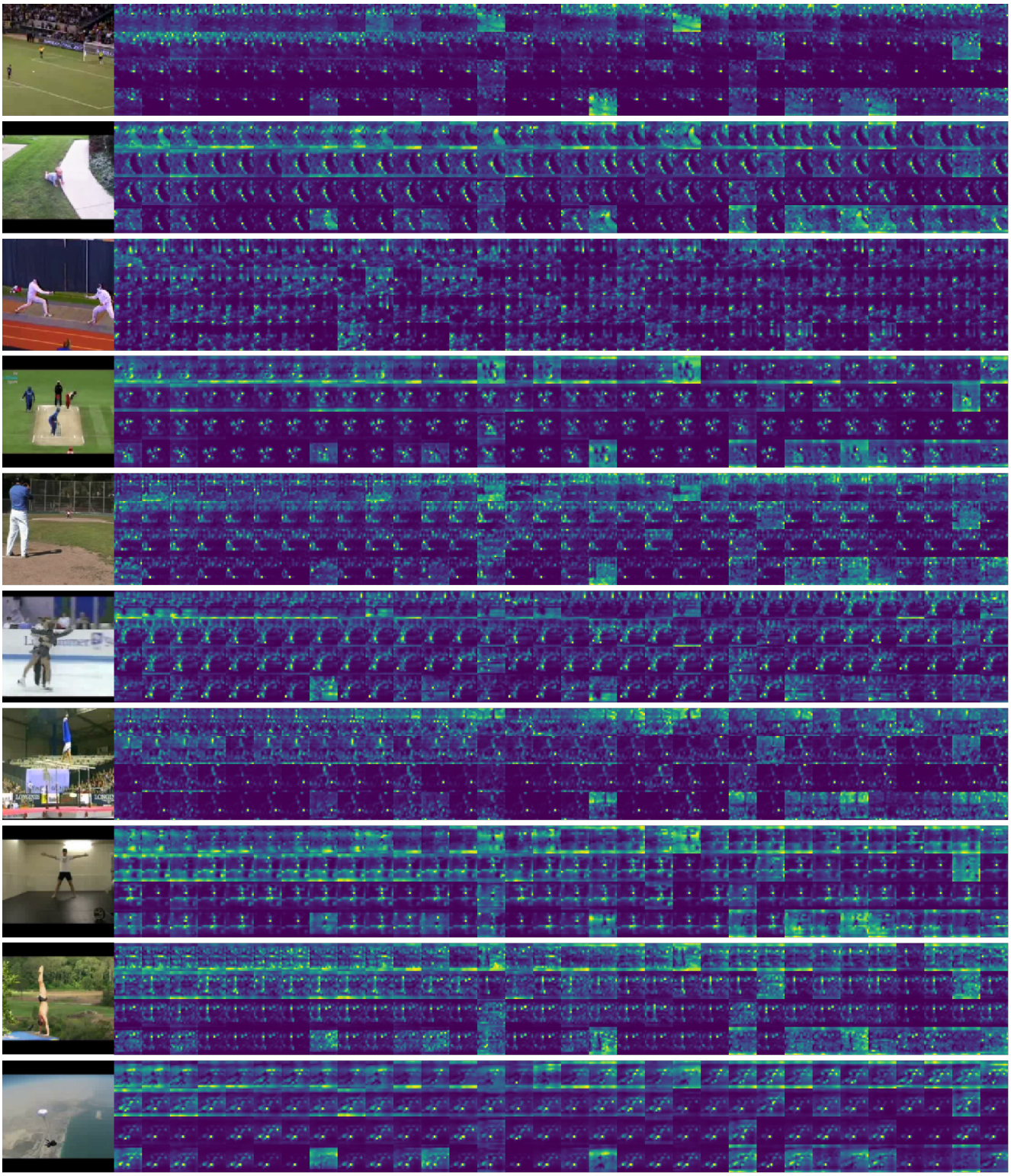


Figure I. **Additional attention maps for latent tokens.** Each map shows how a token attends to spatial patches, ordered from top-left to bottom-right. Early tokens attend broadly to capture global context, while later ones focus on local details.

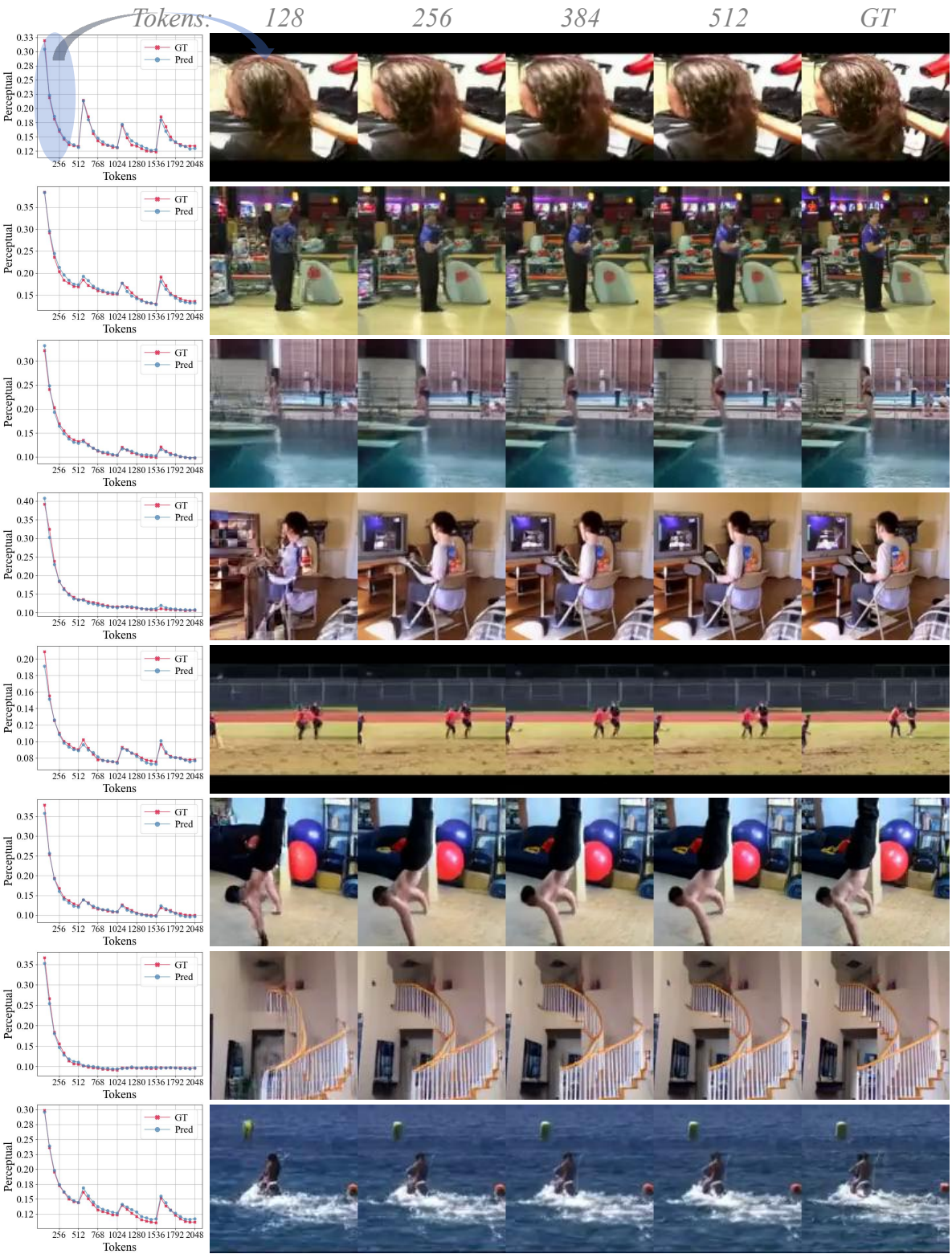


Figure J. **The visualization of scorer predictions and corresponding reconstruction results.** (a) Scorer predictions. Red curves represent the GT scores, while blue denotes the predictions. A total of 2048 tokens are divided into 4 blocks. The x-axis represents the index of the last selected token for each block. (b) Corresponding reconstruction results of frames from the first block under different token counts.