Table of Contents:

A Additional Methodological Details about VaporTok **2**

A.1 Computation of video spatio&temporal complexity 2

A.2 AR prior in VaporTok 2

A.3 Attention mask in reconstruction&generation pipeline 2

A.4 The complete loss function of VaporTok 3

B Supplementary Experiments on Vapor Reward **4**

B.1 Penalty reward 4

B.2 Effect of reconstruction&generation reward under different weight 4

C Analysis about VaporTok **4**

C.1 Mitigating the three core challenges of autoregressive generation 4

C.2 The alignment between FVD and reward 5

C.3 Taildrop training strategy for semantic-to-detail representation 5

C.4 Difference between naive taildrop and probabilistic taildrop 6

C.5 Different visual tokenizer paradigm 7

D Supplementary Related Work about Fixed-length Visual Tokenizer **8**

E Supplementary Implementation Details. **9**

F Visualization **9**

G Broader Impacts **9**

H Safeguards **11**

A Additional Methodological Details about VaporTok

A.1 Computation of video spatio&temporal complexity

Given a video tensor $V \in \mathbb{R}^{T\times H\times W\times 3}$, each frame is converted to grayscale via

$$g_t(x,y) = 0.299\, V_{t,x,y,1} + 0.587\, V_{t,x,y,2} + 0.114\, V_{t,x,y,3}, \tag{1}$$

,where $t = 1,\dots,T$ and $(x,y) \in \{1,\dots,H\} \times \{1,\dots,W\}$.

Spatial Complexity. Define the empirical pixel-value distribution of frame t as

$$p_t(v) = \frac{1}{HW}\big|\{(x,y) \mid g_t(x,y) = v\}\big|, \quad v = 0,1,\dots,255. \tag{2}$$

The Shannon entropy of frame t is

$$H_t = -\sum_{v=0}^{255} p_t(v)\log_2 p_t(v), \tag{3}$$

and the spatial complexity is the average frame entropy:

$$\mathrm{SC} = \frac{1}{T}\sum_{t=1}^{T} H_t. \tag{4}$$

Temporal Complexity. The temporal complexity is defined as the mean absolute difference between consecutive frames:

$$\mathrm{TC} = \frac{1}{(T-1)\,H\,W}\sum_{t=1}^{T-1}\sum_{x=1}^{H}\sum_{y=1}^{W}\big|g_{t+1}(x,y) - g_t(x,y)\big|. \tag{5}$$

A.2 AR prior in VaporTok

Inspired by the AR prior model introduced in LARP [16], we integrate a similar lightweight autoregressive model into VaporTok as shown in Figure 1. This AR model is designed to make latent tokens more compatible with downstream AR generation tasks, and thus its implementation and associated evaluation metrics can serve as a proxy for downstream generation performance.

Similar to LARP, our lightweight AR model is trained jointly with the VaporTok tokenizer in an end-to-end manner. Specifically, the model takes the quantized latent token embeddings as input, and uses the corresponding codebook IDs as labels. To address the instability caused by the training-inference discrepancy inherent to AR models, Scheduled Sampling Mixing as proposed in [1, 9] is employed.

The key difference is that the prior model used in VaporTok is trained **only on the retained latent tokens after truncation** rather than the whole latent space. Moreover, to enable efficient batchwise training, we adopt the attention masking scheme described in Section A.3 within the transformer blocks of the AR prior model.

A.3 Attention mask in reconstruction&generation pipeline

Attention mask for the VaporTok decoder. During the training of VaporTok, the spatiotemporal complexity of each video sample varies, which results in different taildrop probabilities. Additionally, since sampling is performed over the entire probability distribution during training. These two factors lead to varying truncation positions across samples. Consequently, it becomes infeasible to reconstruct all samples within a batch using a shared decoder input length.To address this issue, we design an adaptive attention mask for the VaporTok decoder to accommodate the variable token lengths caused by the probabilistic taildrop. Specifically, as illustrated in Figure 2(a): For each sample, we construct an individual attention mask: all tokens from decoder queries M are granted **full visibility**, while for latent tokens, positions **beyond the truncation point** are masked out to ensure that dropped tokens do not participate in attention computation. This prevents non-informative tokens from interfering with the training process and allows batchwise training of VaporTok.

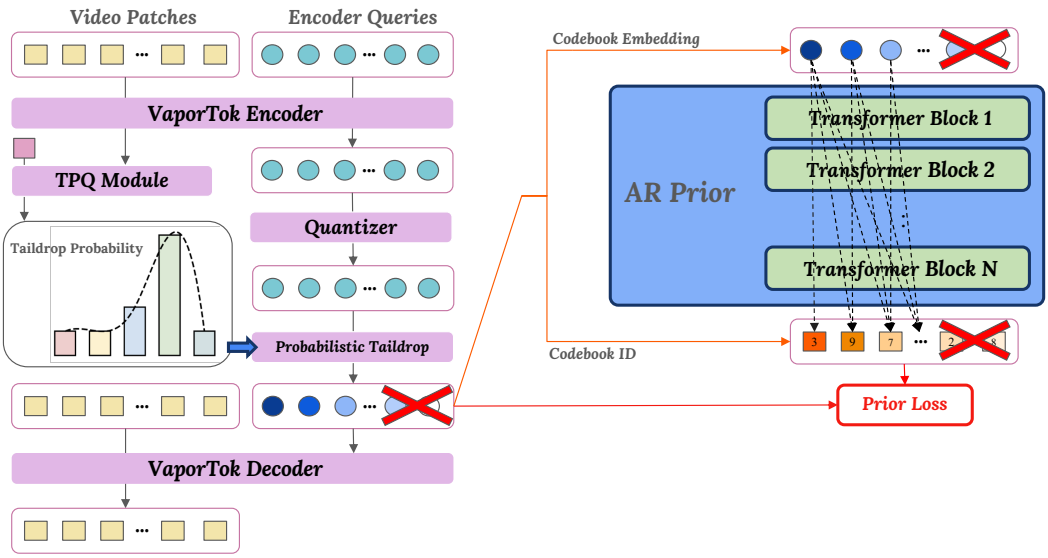


Figure 1: AR prior model in VaporTok.

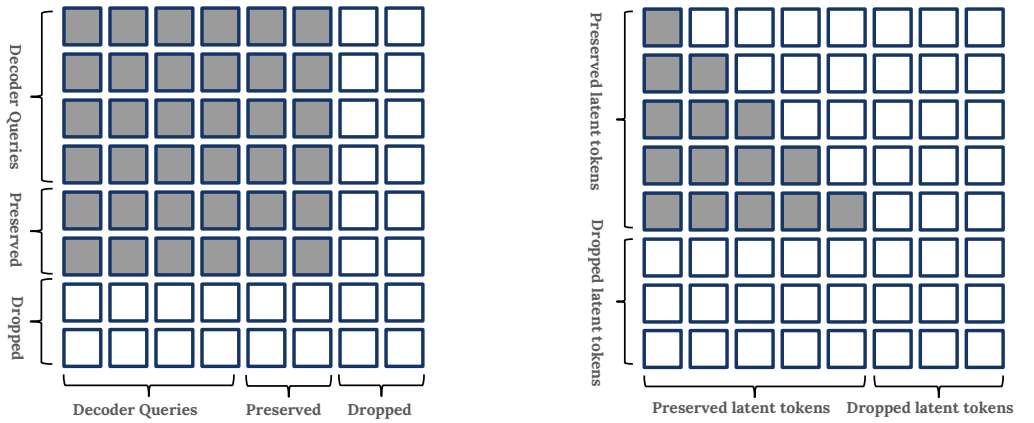


Figure 2: Attention mask for different parts.

Attention mask for the AR prior model. The AR prior model originally adopts a causal attention mask, which ensures that later tokens do not affect the prediction of earlier tokens. However, if we apply a standard causal mask without modification, tokens before the truncation point can still attend to and influence those after the truncation, which is undesirable. Due to the introduction of taildrop, tokens after the truncation point should not be supervised and influenced by prior tokens during training. To resolve this, we propose a modified attention mask as shown in Figure 2(b).

Attention mask for the downstream AR generative model. Since the AR prior model serves as a compact abstraction of the downstream AR generative model, the attention mask used in the downstream AR model is identical to that of the AR prior model, which is also illustrated as Figure 2(b).

A.4 The complete loss function of VaporTok

$$\mathcal{L}_{\text{rec}} = \lambda_{\text{L1}} \cdot \mathcal{L}_{\text{L1}} + \lambda_{\text{perc}} \cdot \mathcal{L}_{\text{perc}} + \lambda_{\text{GAN}} \cdot \mathcal{L}_{\text{GAN}} + \lambda_{\text{commit}} \cdot \mathcal{L}_{\text{commit}} \tag{6}$$

$$\mathcal{L}_{\text{representation prior}} = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i \mid x_i) \tag{7}$$

$$\mathcal{L}_{\text{probability prior}} = \text{KL}(P \,||\, GaussianPrior) \tag{8}$$

$\mathcal{L}_{\text{rec}}$ is comprised of L1 loss, perceive loss, GAN loss, and commitment loss as traditional VQ tokenizer. In Equation 7, x denotes codebook embedding, y denotes codebookid. In Equation 8, P denotes taildrop probability and $GaussianPrior$ denotes prior probability. The complete loss of VaporTok is:

$$\mathcal{L}_{\text{complete}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{rep}} \cdot \mathcal{L}_{\text{rep\_prior}} + \lambda_{\text{prob}} \cdot \mathcal{L}_{\text{prob\_prior}} \tag{9}$$

B Supplementary Experiments on Vapor Reward

B.1 Penalty reward

In our experiment of argmax sampling of probabilistic taildrop, the truncation index always becomes too small to be enough to reconstruct&generate videos, so a reward to punish such truncation is introduced in such senario. Specifically, for the i-th path, the penalty reward is defined as:

$$R^{(i)}_{\text{penalty}} = -\sum_{j=1}^{L} \text{Penalty}(I_{i,j}), \quad \text{if } I_{i,j} < N_{\text{threshold}} \tag{10}$$

where $K_{threshold}$ is a manually defined threshold specifying the minimum number of tokens tolerable for reconstructing the video clip and $\text{Penalty}(t_i)$ denotes a penalty function that imposes more punishment when the truncation index becomes smaller. Then we design a five way ablation study in argmax sampling strategy of probabilistic taildrop as shown in Table 1:

Table 1: Ablation of each rewards. Each row "w/o R" indicates the model trained without reward R. Lower is better for rFVD, gFVD and MSE; higher is better for PSNR and prior top-5 accuracy.

Missing Reward	**#Tokens**	**rFVD↓**	**PSNR↑**	**gFVD↓**	**MSE↓**	**ACC↑**
VaporTok-GRPO	299	72.5	24.1	118	4.85×10^{-3}	3.24%
w/o efficiency reward	577	59.4	25.6	88.46	3.89×10^{-3}	3.88%
w/o diversity reward	305	75.4	24.7	105.85	4.80×10^{-3}	3.24%
w/o reconstruction reward	272	91.6	23.3	138.58	5.83×10^{-3}	3.13%
w/o generation reward	286	83.6	23.8	122.21	5.33×10^{-3}	3.17%
w/o penalty reward	58	3267	10.7	3255.59	9.45×10^{-2}	2.73%

B.2 Effect of reconstruction&generation reward under different weight

We adopt a baseline weighting of 1:1:1:1:1 for the efficiency, penalty, diversity, reconstruction, and generation rewards respectively and then increased the proportion of each of reconstruction&generation rewards. As shown in Figure 3, as the weights for these two rewards grow, their numerical values also increase, which demonstrates that the higher the combined weight on these two performance rewards, the more faithfully the original reconstruction and generation quality is preserved. It is worth noting that if both weights are set too high, performance gains come at the expense of efficiency reward, contradicting our original intent that making tokenizer to be more efficient.

C Analysis about VaporTok

C.1 Mitigating the three core challenges of autoregressive generation

To mitigate the three key limitations of autoregressive (AR) visual generation, we propose a unified solution within the VaporTok framework:

Quadratic complexity with long sequences. We introduce a *sparse but sufficient* token representation by leveraging visual priors to reduce the number of tokens required for generation. Furthermore,

(a) Generation Reward

(b) Reconstruction Reward

(c) Efficiency Reward

(d) Diversity Reward

Figure 3: (a) generation reward (b) reconstruction reward (c) efficiency reward (d) diversity reward of differenct reward weight where the order is efficiency, penalty, diversity, reconstruction, generation.

we apply GRPO to adaptively compress the token sequence while preserving downstream task performance as much as possible.

Error accumulation during AR inference. We propose a *probabilistic taildrop* training strategy that pushes important tokens toward the beginning of the visual representation. As a result, during inference, the model generates the most critical tokens when the accumulation of prediction errors is still minimal, thereby mitigating the impact of error accumulation.

Training gap between the tokenizer and the AR generator. To close this gap, we adopt two complementary strategies. First, we refine the latent space following the approach of LARP [16] to make it more suitable for AR generation. Second, we leverage the same AR-aligned information to guide the training of the taildrop query module via GRPO. This enables our adaptive tokenizer to incorporate downstream task constraints into both the adaptivity mechanism and the latent token representation, thus reducing the mismatch between the tokenizer and the generator.

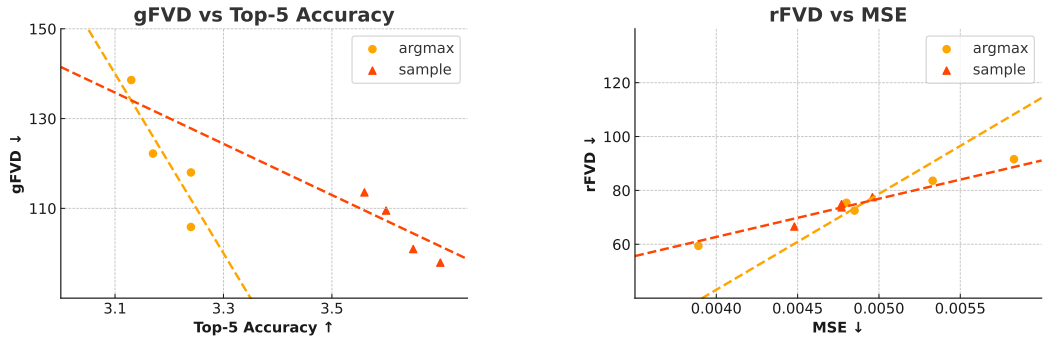
C.2 The alignment between FVD and reward

We visualize the reconstruction and generation results under both sampling and argmax settings, as shown in the Figure 4. It can be observed that rFVD strongly correlates with the reconstruction reward, while gFVD shows a similarly strong correlation with the generation reward. This further supports the validity of our reward design: **the reconstruction MSE serves as a reliable proxy for reconstruction quality, and the top-5 accuracy of the prior model effectively reflects generation quality.**

C.3 Taildrop training strategy for semantic-to-detail representation

The taildrop training strategy is designed to encourage a *semantic-to-detail* organization in the latent token representations. This is primarily achieved through the following mechanism: *for the same video sample, train the model using different numbers of tokens for reconstruction*. As a result, tokens at the beginning of the sequence are exposed to the reconstruction objective more frequently, while later tokens appear less often during training. Furthermore, even when only a small number of tokens are used, the model is still tasked with reconstructing the entire video. This encourages early tokens to encode more global, semantic information that is sufficient for reconstruction.

In addition, it is worth noting the distinction between the two evaluation metrics used in visual reconstruction and generation: rFVD and gFVD. By design, gFVD is consistently worse (higher) than rFVD. This is because rFVD evaluates the reconstruction quality using ground-truth codebook ID directly obtained from the encoder, whereas gFVD evaluates the generation quality using codebook ID predicted by the autoregressive model. *Briefly, the only difference between rFVD and gFVD is that*

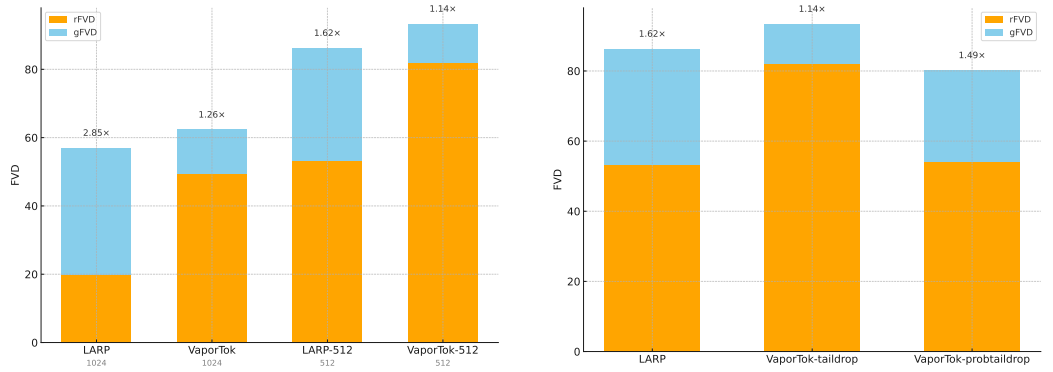


(a) Alignment between gFVD and ACC

(b) Alignment between rFVD and MSE

Figure 4: It can be observed that rFVD and MSE exhibit a strong positive correlation, while gFVD and ACC show a clear negative correlation, indicating the effectiveness and rationality of the proposed reward design.

reconstruction relies on ground-truth ID, while generation depends on autoregressively predicted ID, making the gap between gFVD and rFVD a direct indicator of the degree of AR error accumulation.



(a) Comparison between LARP and VaporTok with naive taildrop under 1024 and 512 token budgets

(b) Comparison between no technique, naive taildrop, and probabilistic taildrop under 500 token budgets

Figure 5: (a) Naive taildrop reduces the gap between rFVD and gFVD, but it leads to a drop in reconstruction performance, which in turn results in degraded generation performance. (b) Probabilistic taildrop, by incorporating a visual prior, avoids the reconstruction performance degradation caused by taildrop, while preserving its original ability to reduce the gap between rFVD and gFVD.

To quantify this effect, we report the **gFVD/rFVD ratio** as shown in Table 2 of the main paper and Figure 5a to measure the discrepancy between reconstruction and generation. We conduct experiments under both 1024-token and 512-token settings. The results show that with taildrop enabled, the gap between gFVD and rFVD becomes smaller, demonstrating the effectiveness of taildrop in mitigating AR error accumulation; specifically, the gFVD/rFVD ratio decreases from 2.85 to 1.26 under the 1024-token setting, and from 1.62 to 1.14 under the 512-token setting.

C.4 Difference between naive taildrop and probabilistic taildrop

The distinction between naive and probabilistic taildrop primarily manifests in training and inference:

Training: While naive taildrop uses uniform sampling for truncation during training, probabilistic taildrop samples from a learned, prior-informed distribution. (We also investigate three specific sampling strategies under the probabilistic framework.)

- *Truncation by argmax of taildrop probability.* The sequence is truncated at the index corresponding to the maximum value in the taildrop probability distribution. While simple,

this approach always uses the same number of tokens for a given input, limiting the semantic-to-detail effect.

- *Sampling from the full taildrop probability.* The truncation index is sampled from the entire taildrop probability distribution. This allows different truncation lengths for the same input and leads to strong reconstruction performance.
- *Sampling indices before the argmax index.* We sample a truncation index from the region before the argmax position, based on the taildrop probability. This also enables varying token counts across samples, though typically results in fewer tokens and slightly degraded reconstruction quality.

Inference: During inference, naive taildrop uses the full set of tokens for decoding (which is the default inference mode used for all reported results in this paper. Alternatively, one may adopt a threshold-based token selection strategy, as in [21]). In contrast, probabilistic taildrop performs decoding using all tokens preceding the argmax index of the taildrop probability distribution, achieving an adaptive number of tokens based on input complexity.

It is worth noting that, among the three sampling strategies for probabilistic taildrop during training (from argmax to sample and then to pre-sample), the degree of *semantic-to-detail* structure in the tokens used during inference gradually increases because of the frequency of sampling index before argmax index is gradually frequent. In other words, these strategies increasingly mitigate AR error accumulation. This trend is also validated by the experimental results reported in Table 2 of the main text as the gFVD/rFVD ratio becomes smaller.

Furthermore, while naive taildrop helps narrow the gap between reconstruction and generation (rFVD vs. gFVD), its uninformed dropping during training—without accounting for visual priors—results in compromised reconstruction quality. In contrast, our probabilistic taildrop incorporates a learned prior distribution, maintaining competitive reconstruction performance (rFVD) and achieving superior generation quality by alleviating AR error accumulation.

C.5 Different visual tokenizer paradigm

Visual tokenizers can be classified according to various criteria, and one particularly informative distinction is how they map visual patches to latent tokens, yielding two families: Patchwise-Token and Holistic-Token.

In the Patchwise-Token paradigm as depicted in Figure 6(a) , each learned token corresponds one-to-one with a visual patch, thereby preserving the spatial structure imposed. Briefly, given a video input $V \in \mathbb{R}^{T\times H\times W\times 3}$, the encoder outputs a downsampled feature map

$$Z = \mathrm{Enc}(V) \ \in \ \mathbb{R}^{\frac{T}{f_T}\times\frac{H}{f_H}\times\frac{W}{f_W}\times D}, \tag{11}$$

where f_T, f_H, f_W are the temporal and spatial downsampling factors. The reconstructed video is then obtained as

$$\widehat{V} = \mathrm{Dec}\big(Z\big) \in \mathbb{R}^{T\times H\times W\times 3}. \tag{12}$$

In the Holistic-Token paradigm as depicted in Figure 6(b), each latent token may assume different semantic roles depending on the training strategy, resulting in a more flexible representational scope. Different from Patchwise-Token paradigm mainly depends on 3D CNN as the backbone, Holistic-Token paradigm usually employs Transformer blocks as the backbone, so a simple patch embedding layer is needed to be conducted to $V \in \mathbb{R}^{T\times H\times W\times 3}$

$$P = \mathrm{Patchify}(V) \in \mathbb{R}^{(\frac{T}{f_T}\times\frac{H}{f_H}\times\frac{W}{f_W})\times D}, \tag{13}$$

and then P will be concated with K learnable query tokens $Q \in \mathbb{R}^{K\times D}$ and the combined sequence will be passed into the encoder:

$$Z_P \oplus Z_Q = \mathrm{Enc}\big(P \oplus Q\big) \in \mathbb{R}^{(\frac{T}{f_T}\times\frac{H}{f_H}\times\frac{W}{f_W}+K)\times D}, \tag{14}$$

where $\oplus$ denotes concatenation and Z_P,Z_Q denotes the represatation of P, Q after encoder respectively. During detokenization, $M \in \mathbb{R}^{(\frac{T}{f_T}\times\frac{H}{f_H}\times\frac{W}{f_W})\times D}$, the decoder query of the same shape as the video patches P, will be concatenated with Z_Q and passed to the decoder to renconstruct the input video.

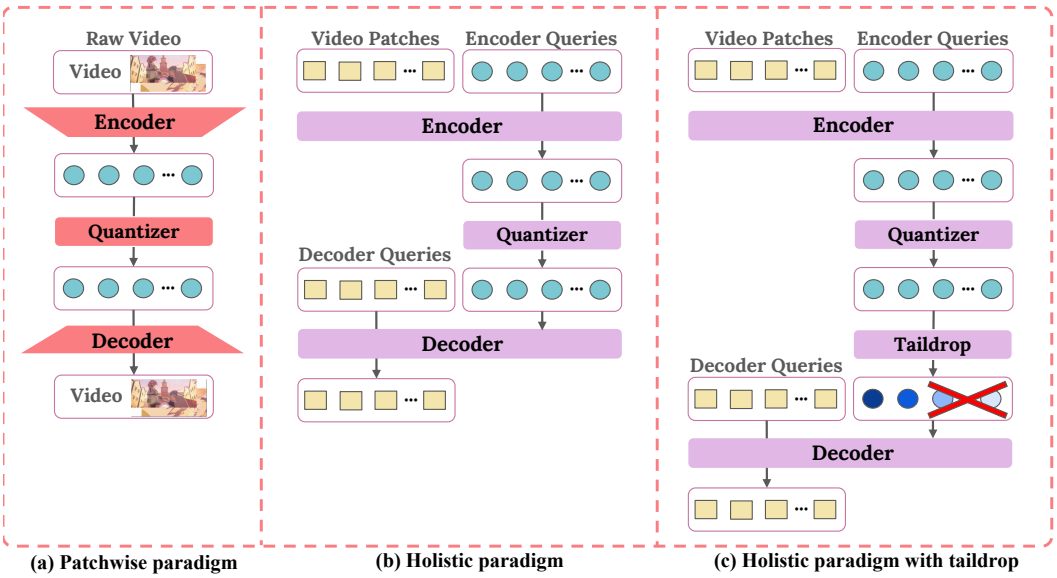


Figure 6: Different paradigm of visual tokenizer: (a) The patchwise-token paradigm typically represents each token as encoding information from a specific spatial region and usually adopts a CNN as the backbone.(b) The holistic-token paradigm is not constrained by fixed spatial positions and can flexibly adjust the information each token represents based on the training strategy. (c) Taildrop is a training technique commonly used in the holistic-token paradigm, enabling the token sequence to exhibit a semantic-to-detail property.

$$\widehat{V} = \mathrm{Dec}\big(M \oplus Z_Q\big) \in \mathbb{R}^{T\times H\times W\times 3}. \tag{15}$$

It is worth noting that *the taildrop training strategy is typically employed in holistic-token tokenizers*. This is primarily because the query representation is not constrained by fixed spatial regions, allowing it to flexibly adapt to different training objectives and strategies. Moreover, such flexibility enables the model to achieve effective representation learning with less training data.

D Supplementary Related Work about Fixed-length Visual Tokenizer

Visual tokenizers for understanding tasks typically rely on contrastive learning, for example: CLIP[11] trains paired image and text encoders with an InfoNCE contrastive loss over matched image–caption pairs, enabling strong zero-shot transfer across diverse vision task; SigLIP[25] replaces CLIP's softmax-based InfoNCE loss with an independent pairwise sigmoid loss, removing the need for global normalization and scaling more efficiently to very large or small batch sizes. TULIP[14] augments CLIP-style pretraining with generative data augmentation and unified image–image, text–text, and image–text contrastive objectives plus reconstruction regularization to learn fine-grained visual features without sacrificing semantic alignment.

Whereas visual tokenizers designed for generation usually employ VAE-based architectures: VQ-VAE[15] first introduce vector quantization into VAE, transforming data from continuous spaces into discrete tokens to simplify modeling and circumvent issues of "posterior collapse" in VAE framework; VQ-GAN[3] improves image reconstruction quality by introducing adversarial loss and using a Transformer for autoregressive visual generation; FSQ[8] projects representations into a lower dimensional space for quantization into fixed values, while its variant LFQ[22, 6] further simplifies the process by using binary quantized representations. This new kind of quantization method effectively enhances the AR generation paradigm by dramatically enlarging the vocab size and improving the encoding efficiency. Beyond these patch-to-token VAEs, there are also VAEs that learn holistic tokens, such as TiTok[23] and LARP[16], by compressing visual information into a holistic query, they eliminate the patch-to-token correspondence constraint, yielding a more flexible architecture with inherent compression potential.

Recently, along with the rapid development of the unified model[18, 20, 19, 27], there are also several visual tokenizers designed for both generation and understanding, such as TokenFlow[10], SemHiTok[2] and UniTok[7]. These works focus on unifying visual generation and understanding within a single visual tokenizer by employing discrete representations and hierarchical or multi-codebook strategies, addressing the differing requirements in token granularity and semantic level between generation and understanding tasks.

While the above methods have shown impressive results for static images, extending visual tokenizers to video requires capturing both temporal continuity and spatial detail, presenting new challenges for tokenizer architectures and training paradigms. Recent work addresses this by integrating temporal modeling[17], diffusion-guided reconstruction[4], hierarchical codebooks[26], and coordinate-based patch schemes[5] to efficiently compress and faithfully reconstruct long video sequences.

E Supplementary Implementation Details.

VaporTok first patchifies the input video into a sequence of tokens. In all experiments, we set the patch sizes to $f_T = 4$, $f_H = 8$, $f_W = 8$, so that a $16 \times 128 \times 128$ video clip is split into $4 \times 16 \times 16 = 1024$ patches. The number of encoder query tokens is set to $k = 1024$. The quantizer and prior model is set as same as [16], where the factorized codebook is employed of size 8192 with embedding dimension $d_{codebook} = 8$ and prior model is adapted from a small GPT-2 backbone[12]. For our taildrop probability query module, we set the number of transformer blocks as $I = 2$, and the softmax temperature is set to 1.8. Due to the high computational cost of training, we trained for 30 epochs on the UCF101 and K600 datasets using the pretrained model provided by LARP[16], which required 90 hours on 8 A100 GPUs.

For our parallel sample GRPO, we set the group size $G = 8$, the KL penalty weight $\beta = 0.1$, the number of inner iterations $\mu = 2$, and the clipping bounds to $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$ as in [24]. The default reward weights for efficiency, penalty, diversity, reconstruction, and generation are set to 1:1:1:1:1. The GRPO training process uses the UCF101 dataset for a single epoch, which takes 3 hours on a single A100 GPU.

For our AR generative model, we adopt a LLaMA-style transformer [13]. In the class-conditional generation task on UCF-101 we prepend a `[cls]` token to represent the category, and a `[stop]` token to cease the generation process when encountering it. The generation task is trained on the UCF101 dataset for 3000 epochs, which takes 40 hours on 8 A100 GPUs.

F Visualization

We present visualizations of VaporTok's reconstruction and final class-based generation results, as shown in the Figure 7 and Figure 8.

G Broader Impacts

Our Adaptive Video Tokenizer is designed exclusively for autoregressive video generation tasks, and is not intended for video understanding or classification; by allocating fewer tokens to low-complexity videos and more to high-detail videos, our method reduces computational and bandwidth costs for real-time generative applications (e.g., interactive video editing, virtual content creation); it enables fast, adaptive video synthesis for artistic tools and educational simulators, lowering barriers for non-expert users to generate high-fidelity video content; it facilitates deployment of generative video models on edge devices (e.g., AR/VR headsets, mobile phones) by reducing token sequence length and inference latency; however, improved efficiency in video generation could be misused to produce highly realistic deepfake videos, exacerbating misinformation campaigns; although not designed for recognition, the underlying tokenizer could be adapted to generate misleading synthetic footage for surveillance evasion or identity spoofing; training on unbalanced datasets may lead the tokenizer to allocate token budgets unevenly, causing generative artifacts that disproportionately affect certain demographics; while inference is more efficient, training the dual-branch model remains GPU-intensive, contributing to carbon emissions.

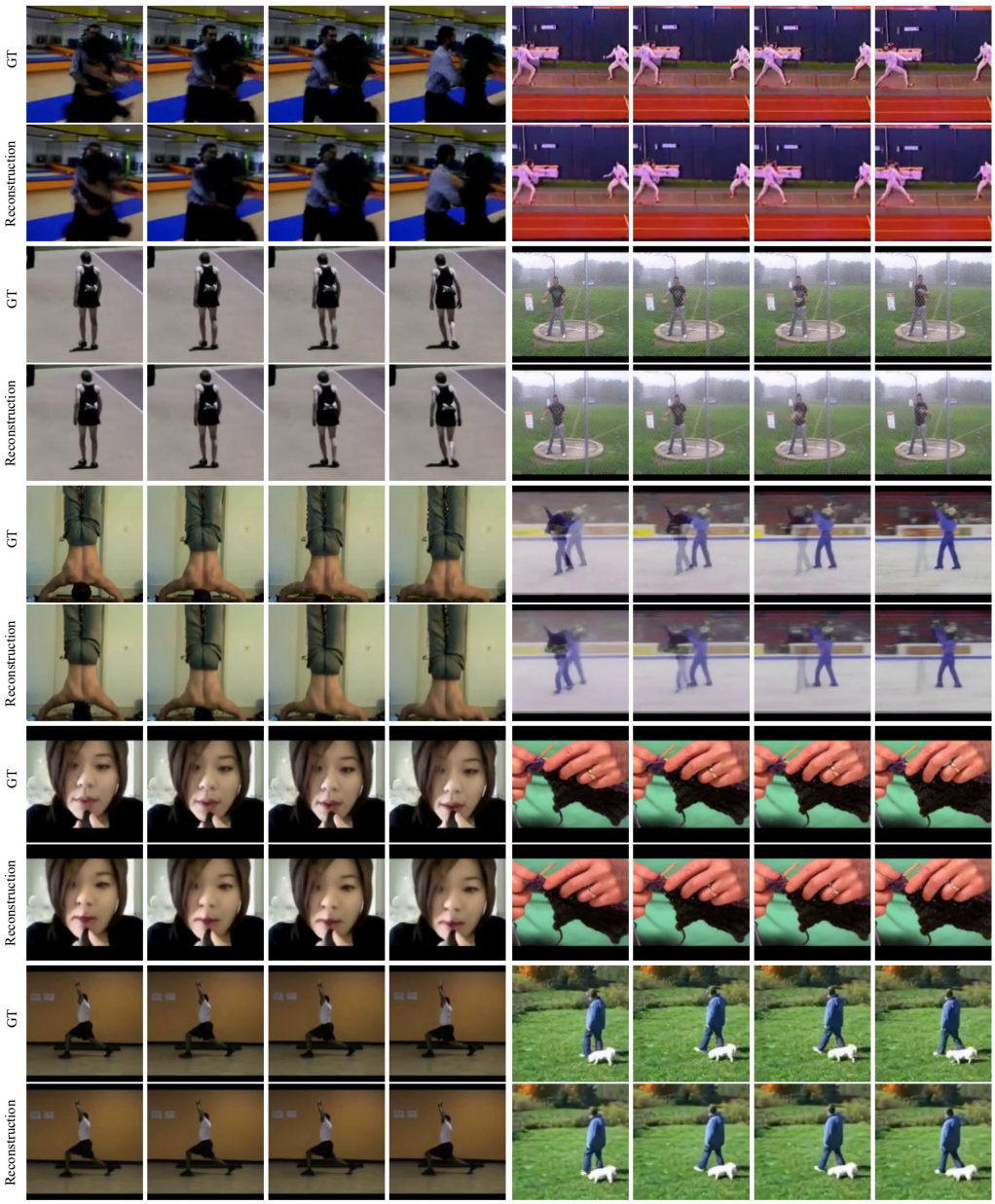


Figure 7: Video reconstruction on UCF101.

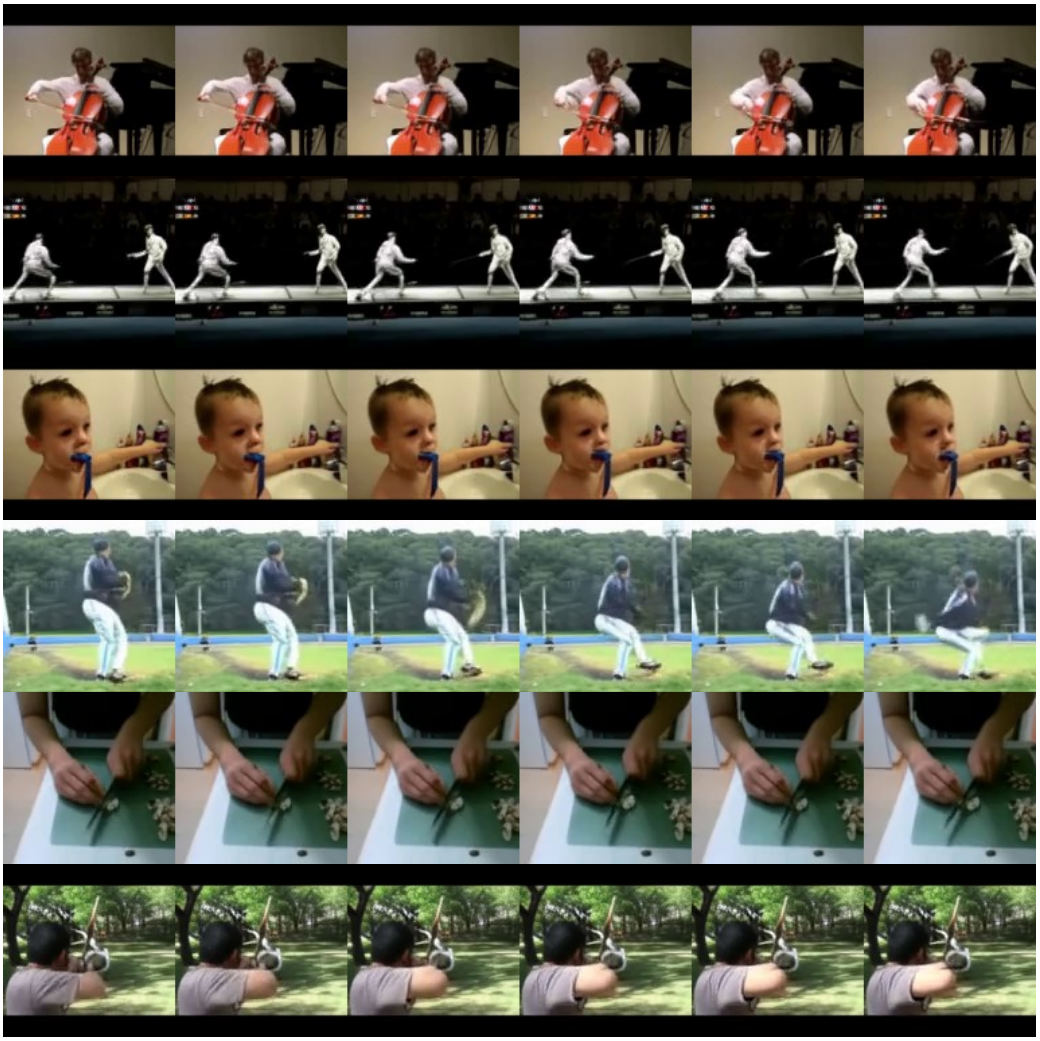


Figure 8: Class-based video generation on UCF101.

H Safeguards

To mitigate potential misuse and harms, we will release a detailed model card specifying that use for deceptive or harmful video synthesis (e.g., deepfakes) is prohibited; distribute weights under a non-commercial, no-derivatives license (e.g., CC BY-NC-ND) and/or via an API with rate limits rather than open weight download; embed imperceptible watermarks in generated videos for provenance tracking and provide a companion detection model to flag synthetic content; maintain a public issue tracker for misuse reports and regularly update the model to address discovered biases or vulnerabilities; publish training logs, compute cost estimates, and carbon-emission metrics to inform users of environmental impact.

References

[1] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[2] Zisheng Chen, Chunwei Wang, Xiuwei Chen, Hang Xu, Jianhua Han, and Xiandan Liang. Semhitok: A unified image tokenizer via semantic-guided hierarchical codebook for multimodal understanding and generation. *arXiv preprint arXiv:2503.06764*, 2025.

[3] 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*, pages 12873–12883, 2021.

[4] Yuying Ge, Yizhuo Li, Yixiao Ge, and Ying Shan. Divot: Diffusion powers video tokenizer for comprehension and generation, 2024.

[5] Huiwon Jang, Sihyun Yu, Jinwoo Shin, Pieter Abbeel, and Younggyo Seo. Efficient long video tokenization via coordinate-based patch reconstruction, 2025.

[6] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-magvit2: An open-source project toward democratizing auto-regressive visual generation, 2024.

[7] Chuofan Ma, Yi Jiang, Junfeng Wu, Jihan Yang, Xin Yu, Zehuan Yuan, Bingyue Peng, and Xiaojuan Qi. Unitok: A unified tokenizer for visual generation and understanding. *arXiv preprint arXiv:2502.20321*, 2025.

[8] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

[9] Tsvetomila Mihaylova and André FT Martins. Scheduled sampling for transformers. *arXiv preprint arXiv:1906.07651*, 2019.

[10] Liao Qu, Huichao Zhang, Yiheng Liu, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Daniel K Du, Zehuan Yuan, and Xinglong Wu. Tokenflow: Unified image tokenizer for multimodal understanding and generation. *arXiv preprint arXiv:2412.03069*, 2024.

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

[12] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[13] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[14] Zineng Tang, Long Lian, Seun Eisape, XuDong Wang, Roei Herzig, Adam Yala, Alane Suhr, Trevor Darrell, and David M. Chan. Tulip: Towards unified language-image pretraining, 2025.

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

[16] 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.

[17] Junke Wang, Yi Jiang, Zehuan Yuan, Bingyue 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, 2024.

[18] Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*, 2024.

[19] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, Song Han, and Yao Lu. Vila-u: a unified foundation model integrating visual understanding and generation, 2025.

[20] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024.

[21] Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video, 2025.

[22] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[23] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation, 2024.

[24] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.

[25] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[26] Ziqin Zhou, Yifan Yang, Yuqing Yang, Tianyu He, Houwen Peng, Kai Qiu, Qi Dai, Lili Qiu, Chong Luo, and Lingqiao Liu. Hitvideo: Hierarchical tokenizers for enhancing text-to-video generation with autoregressive large language models, 2025.

[27] Xianwei Zhuang, Yuxin Xie, Yufan Deng, Dongchao Yang, Liming Liang, Jinghan Ru, Yuguo Yin, and Yuexian Zou. Vargpt-v1.1: Improve visual autoregressive large unified model via iterative instruction tuning and reinforcement learning, 2025.