Less Is More: Vision Representation Compression for Efficient Video Generation with Large Language Models

Yucheng Zhou[1*] Jihai Zhang[2*] Guanjie Chen[3] Jianbing Shen[1†] Yu Cheng[2†]

[1]SKL-IOTSC, CIS, University of Macau
[2]The Chinese University of Hong Kong [3]Shanghai Jiao Tong University
yucheng.zhou@connect.um.edu.mo

Abstract

Video generation using Large Language Models (LLMs) has shown promising potential, effectively leveraging the extensive LLM infrastructure to provide a unified framework for multimodal understanding and content generation. However, these methods face critical challenges, i.e., token redundancy and inefficiencies arising from long sequences, which constrain their performance and efficiency compared to diffusion-based approaches. In this study, we investigate the impact of token redundancy in LLM-based video generation and propose ***Vision Representation Compression*** *(*VRC*), a novel framework designed to achieve* ***More*** *in both performance and efficiency with* ***Less*** *video token representations.* VRC *introduces learnable representation compressor and decompressor to compress video token representations, enabling autoregressive next-sequence prediction in a compact latent space. The proposed approach eliminates redundancy, reduces token sequence length, and enhances the model's ability to capture underlying video structures. Our experiments demonstrate that* VRC *reduces token sequence lengths by a factor of* ***4****, achieving more than* ***9****× acceleration in inference while maintaining performance comparable to state-of-the-art video generation models. In addition,* VRC *not only accelerates the inference process but also significantly reduces memory requirements during both model training and inference.*

1. Introduction

Diffusion-based video generation models [1, 2, 5, 9] have showcased significant success in producing realistic and high-quality videos. Recently, the powerful multimodal understanding capabilities of Large Language Models (LLMs) have prompted some studies [8, 29] to explore their video generation capability. This emerging paradigm of integrating visual generation capabilities into LLMs [23, 35] lays a solid foundation for developing unified multimodal models capable of understanding and generating visual content.

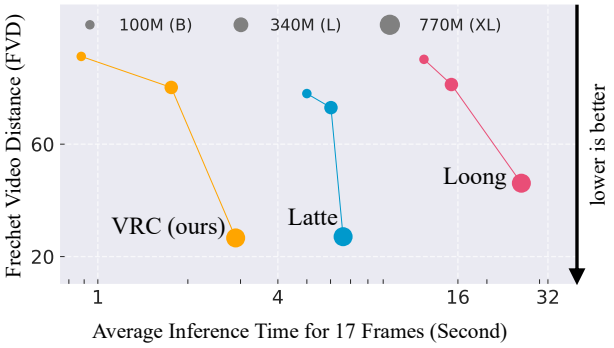

Figure 1. Comparison of FVD and inference time (for 17 frames) across models with different sizes on a single A100 GPU on FaceForensics [11]. VRC (ours) achieves better FVD and faster inference than Latte [9] and Loong [24] (our reproduction).

Recent studies [24] have explored the challenges faced by autoregressive LLMs in video generation. These challenges stem from the fundamental differences between text and visual data. While text is inherently information-dense, enabling LLMs to excel by converting text into discrete tokens and predicting the next token autoregressively, videos generally contain significant redundancy due to spatial and temporal correlations. Although video tokenizers [8, 22] are used to compress video frames into token sequences, these tokenizers are unable to eliminate redundancy between adjacent video tokens. For instance, a 17-frame video clip with 256×256 resolution can produce 5120 tokens by OmniTokenizer [22]. This redundancy can result in information leakage during autoregressive training, where the model optimizes the prediction loss by simply copying from preceding previous tokens instead of learning the underlying structures or relationships within video frames. In addition, such

*Equal Contribution.
†Corresponding Authors.

redundant information leads to excessively long token sequences from video tokenizers, increasing inference time for LLMs and causing cumulative errors by next-token prediction during inference [24].

Given these challenges, we explore a critical question: *"Is it necessary to utilize such a large number of tokens to represent a video clip for autoregressive LLM-based video generation?"* For example, modern video compression standards like H.264 [25] achieve temporal compression by recording only the differences between frames rather than storing complete frames, leading to compression rates of 200 to 500 times. Inspired by this, we propose integrating a learnable video representation compressor and a representation decompressor into the LLM-based framework. Our method compresses the visual representation after tokenization, performs autoregressive predictions in the compressed representation space, and then decodes these predictions back into the token embedding space by decompressor. This approach not only reduces redundancy between adjacent video tokens, mitigating potential shortcuts during training but also acts as an information bottleneck, encouraging the LLM to capture the latent structure underlying the video tokens. To further address information leakage during autoregressive training and error accumulation of next-token predictions during inference, we employ sequence-level prediction for all frames beyond the first. Specifically, the model predicts a sequence of tokens for a subsequent frame simultaneously, which prevents error accumulation in frame sequences and compels the model to learn the global relationships among tokens across frames, rather than relying solely on unidirectional dependencies from preceding tokens.

Our experimental results demonstrate that representation compression significantly improves the performance of autoregressive LLM-based video generation models by effectively reducing redundancy between adjacent video tokens. Moreover, this compression reduces the sequence length by 4 times, yielding substantial computational benefits, including up to $9\times$ faster inference and significantly reduced memory requirements. As shown in Figure 1, VRC not only achieves better results than Latte [9], a diffusion-based model, but also exhibits faster inference speeds compared to Latte and significantly outperforms commonly used LLM-based models of the same size, such as Loong [24]. This work unlocks the potential of LLM-based video generation models, providing a promising direction for improving their efficiency and effectiveness through representation compression. Our main contributions are summarized as follows:

- We demonstrate that compressing video representations after tokenization effectively eliminates redundancy between adjacent tokens, improving the efficiency and performance of LLM for autoregressive video generation.
- We propose **Vision Representation Compression** (VRC) for LLM-based video generation that employs a pair of learnable video representation compressor and representation decompressor to reduce token redundancy.
- To mitigate error accumulation, we employ sequence-level prediction, where the model simultaneously predicts tokens for the next sequence. This prevents reliance on unidirectional dependencies and helps the model learn global relationships across frames.
- Experimental results show that VRC outperforms the LLM baseline under the same size in inference time and performance. Through compression, we reduce token sequence length by 4 times, achieve over $9\times$ acceleration in inference, and significantly lower memory requirements in training and inference.

2. Related Work

2.1. Video Generation

Video generation has been extensively studied in recent years, with various approaches proposed to tackle this challenging task. The research in this area can be broadly categorized into three main paradigms: GAN-based, diffusion-based, and LLM-based methods. Early methods predominantly relied on Generative Adversarial Networks (GANs), which generate realistic video sequences by training a generator and discriminator in an adversarial framework [12, 14, 16, 19, 32]. While GAN-based methods successfully capture fine details in video frames, they often struggle with mode collapse and maintaining temporal consistency in longer video sequences. Diffusion-based video generation methods have recently emerged as a powerful alternative, leveraging stochastic processes to iteratively refine video predictions [2–4, 6, 9, 21, 33, 34, 36]. These approaches have demonstrated remarkable performance in generating high-quality and temporally consistent videos. However, diffusion-based methods are computationally intensive due to the need for multiple iterations of the diffusion process, which inherently lacks parallelizability, making them resource-demanding during training and inference.

2.2. LLM-based Video Generation

LLM-based approaches, inspired by advances in LLMs, frame video generation as an autoregressive next-token prediction task. These methods discretize video frames into tokens and leverage transformer architectures for sequence modeling [8, 24, 29]. While LLM-based models benefit from the scalability of transformers, they face significant challenges related to token redundancy and the inefficiency of handling long sequence lengths, which limit their competitiveness compared to diffusion-based methods. LLM-based video generation has attracted attention due to its compatibility with LLM infrastructures and its

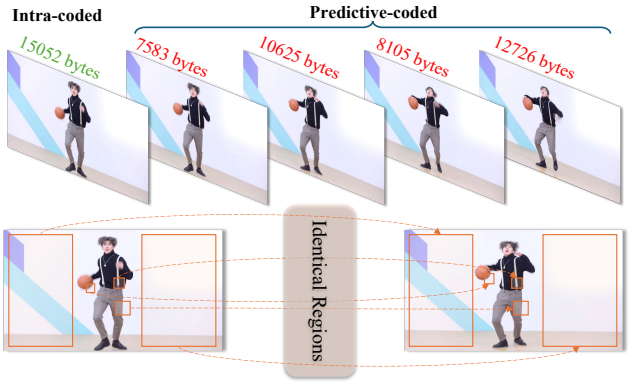

Figure 2. Redundancy between video frames. (**Top**) Comparison of intra-coded and predictive-coded frames in terms of byte size, calculated using the H.264 codec. (**Bottom**) The orange boxes mark identical regions across frames, which are used in predictive coding to reduce byte size.

potential to unify multimodal understanding and generation [23, 35]. A key component of these models is the video tokenizer, which discretizes continuous video frames into discrete tokens to enable transformer-based next-token prediction. Prominent tokenizers, such as ViT-VQGAN [30] and OmniTokenizer [22], aim to compress video data into a manageable discrete token space while preserving semantic information. Despite advancements in tokenization, redundancy in video tokens remains a critical challenge for LLM-based models. Adjacent frames often contain overlapping information, leading to inefficiencies in sequence modeling and shortcuts during training [24]. Recent efforts have attempted to mitigate these issues through techniques such as introducing causal 3D CNN in the video tokenizer [31] and using multistage training processes [24]. However, redundancy not only hampers the learning process by encouraging the model to memorize rather than generalize but also results in long sequences that significantly increase computational costs. Addressing this redundancy remains essential for improving the efficiency and effectiveness of LLM-based video generation models.

3. Rethinking LLM-based Video Generation

3.1. Redundant Information Blocks Autoregressive Model Learning

Different from text and images, videos contain substantial redundancy between adjacent frames, due to many identical regions. Modern video encoding methods such as H.264 [25] can achieve a high compression ratio by calculating the differences between frames and encoding only these differences. As shown in Figure 2, using H.264 to encode these five frames, with the first frame set as an intra-coded frame (encoded independently) and subsequent frames as predictive-coded frames (encoding only the differences relative to the first frame), the amount of information needed to record each subsequent frame is significantly less than that for the first frame.

In contrast, autoregressive video generation models predict video token sequences through next-token prediction. This paradigm allows the model to rely on previous tokens to generate subsequent redundant ones by simply copying information, which impedes the learning of deeper structures and semantics underlying the video. Recently, Wang et al. [24] observed an imbalanced loss during the training of autoregressive video generation models, where the loss for earlier frames is much higher than for later frames. This suggests that even with VQ tokenizers incorporating spatiotemporal compression, such as OmniTokenizer [22], the encoded video token sequences still exhibit significant redundancy. Moreover, the model, which learns to generate subsequent tokens by simply copying information from previous tokens, does not effectively support learning early frame predictions. To address this issue, it is crucial to further compress the information and eliminate redundancy in autoregressive video generation.

3.2. Necessity of Information Bottleneck

The information bottleneck (IB) principle plays a crucial role in representation learning by compelling models to learn compressed and high-level representations. According to this principle [17], introducing a bottleneck forces the model to focus on capturing only the most relevant features essential for predicting the target variable. This effectively eliminates redundant or irrelevant information, leading to more robust and efficient representations.

Theoretically, the information bottleneck aims to find a compressed representation T of the input X that retains as much information as possible about the target Y, while discarding unnecessary details from X. The optimization problem can be formulated as:

$$\min_{p(T|X)} \mathcal{L}_{\text{IB}} = I(X;T) - \beta I(T;Y), \tag{1}$$

where $I(X;T)$ represents the mutual information between the input X and the learned representation T, enforcing compression, and $I(T;Y)$ denotes the mutual information between the representation T and the target Y, ensuring predictive accuracy; β is a positive trade-off parameter that controls the balance between compression and relevance. The core idea of the IB principle is to minimize $I(X;T)$ while maximizing $I(T;Y)$. This balance ensures that the learned representation T is both concise and informative for downstream tasks.

For video generation, the presence of redundant information between consecutive frames often leads to shortcut learning, where models simply copy information from pre-

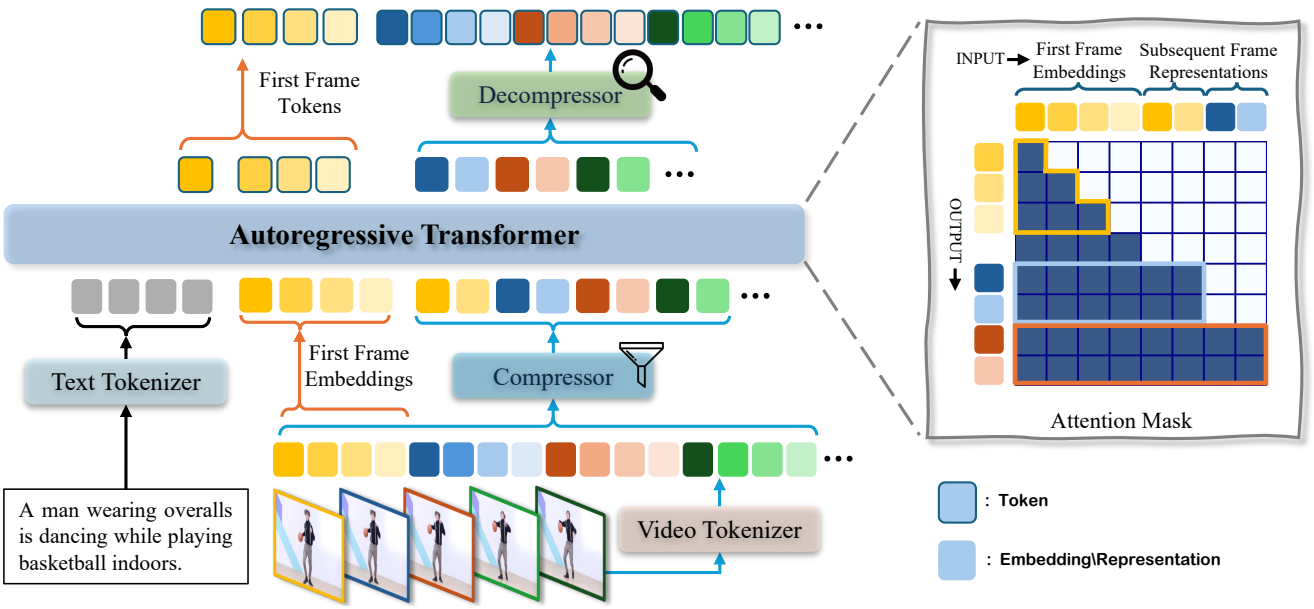

Figure 3. Overview of the Vision Representation Compression (VRC) Framework. The input video embeddings are further compressed through a learnable representation compressor to eliminate redundant information. The generation of the first frame tokens is conditioned on previous first frame tokens and text tokens. The generation of subsequent frame representations is conditioned on the first frame tokens, text tokens, and previous frame representations. VRC generates a sequence of frame representations simultaneously, implemented by an attention mask shown on the right, combining both causal and bidirectional attention. We omit the embedding layer for clarity and use colored blocks with/without borders to indicate tokens and embeddings/representations. We use different colors to indicate tokens/embeddings/representations from different frames

vious frames rather than learning meaningful latent structures. Introducing IB offers two significant advantages: (1) By compressing the representation space, it reduces redundancy between adjacent video tokens, which helps prevent the model from overfitting to trivial patterns, thereby encouraging the learning of more substantial temporal dependencies. (2) The bottleneck enforces the extraction of high-level and abstract features, thus enhancing the model's ability to capture the underlying dynamics of video sequences. Theoretically, this approach aligns with the principle that reducing the mutual information between the previous frames and subsequent representations ($I(X;T)$) compels the model to focus on features different from previous frames.

4. Vision Representation Compression for Video Generation with LLMs

4.1. LLM-based Video Generation

The LLM utilized for video generation is a transformer model designed for next-token prediction. To process both text and video, a text tokenizer and a video tokenizer are employed to convert inputs into discrete tokens.

Given a video sequence consisting of $1+T$ frames with resolution $H \times W$, the video tokenizer compresses the frames into a sequence of $(1+\frac{T}{t}) \times N$ tokens, where

$$N = \frac{H}{p} \times \frac{W}{p}, \tag{2}$$

where $p \times p$ represents the downsampling rate in the video tokenizer. The video tokenizer utilizes a causal 3D convolutional neural network (CNN) with a temporal stride of t to downsample the input sequence along the time dimension, resulting in discrete tokens.

The LLM-based video generation is framed as an autoregressive sequence modeling problem. The input to the LLM is a sequence of tokens that includes both the text prompt tokens and the discrete video tokens. The model learns to predict the next video token in the sequence conditioned on all previous tokens:

$$\mathcal{L}_{\text{LLM}} = \mathbb{E}_{x_i}\left[-\log P_\theta(x_i \mid x_{<i}; x_{\text{text}})\right] \tag{3}$$

where x_i denotes the i-th token in the video sequence.

4.2. Vision Representation Compression

In our approach, the first frame tokens x_0 are independent of subsequent frames due to the causal 3D CNN design employed in the video tokenizer (e.g., OmniTokenizer). Since the generation process is conditioned solely on the text input, we retain the first frame embeddings separately from the compression step.

The subsequent video embedding sequence, excluding the first frame, is represented as $\mathbf{x}_{N:(1+\frac{T}{t})\times N}$. As shown in Figure 3, this sequence undergoes compression via a learnable representation compressor that reduces its size from $(\frac{T}{t}) \times N$ to $(\frac{T}{t}) \times M$, i.e.,

$$M = \frac{H}{p \times r} \times \frac{W}{p \times r}, \tag{4}$$

where r represents the downsampling ratio, and H and W denote the height and width of the input frames, respectively. The resulting compressed embeddings are denoted as $\mathbf{z}_{0:(1+\frac{T}{t})\times M}$. The compressed embeddings are concatenated with the text embeddings $\mathbf{x}_{\text{text}}$ and the uncompressed first frame embeddings $\mathbf{x}_{0:N}$ to form the input sequence for the autoregressive transformer. This sequence is defined as:

$$\mathbf{s} = \text{Concat}(\mathbf{x}_{\text{text}}, \mathbf{x}_{0:N}, \mathbf{z}_{0:(1+\frac{T}{t})\times M}), \tag{5}$$

where $\mathbf{s}$ denotes the combined sequence of embedding input to the transformer model. The $\text{Concat}(\cdot)$ operation refers to the concatenation operation. VRC performs both downsampling and upsampling operations exclusively along the spatial dimensions of the token embeddings. This design choice ensures that no compression is applied along the temporal dimension, thereby preventing information leakage across time steps.

4.3. Next Sequence Prediction

As shown in Figure 3, the distribution of the first frame tokens is conditioned on the text tokens: $P(\mathbf{x}_i|\mathbf{x}_{<i}; \mathbf{x}_{\text{text}})$. The prediction of subsequent frames is conditioned on both the first frames and the text input. For subsequent frames, VRC generates their continuous embeddings in the compressed representation space, effectively optimizing the use of model capacity. To reduce reliance on ground-truth tokens during training and to enhance generalization, we replace traditional next-token prediction with a next-sequence prediction strategy. Instead of predicting a single token, VRC generates a sequence of embeddings simultaneously. Specifically, the model predicts M embeddings $\mathbf{z}_{i\times M:(i+1)\times M}$ conditioned on the prior text and frame embeddings: $P_\theta(\mathbf{z}_{i\times M:(i+1)\times M}|\mathbf{x}_{\text{text}}; \mathbf{x}_{0:N}; \mathbf{z}_{0:i\times M})$. This is achieved using a tailored causal attention mask, as shown in Figure 3. The embeddings within the same predicted sequence segment, $\mathbf{z}_{i\times M:(i+1)\times M}$, can fully attend to each other bidirectionally. However, they do not access subsequent sequence segments, thereby preserving the autoregressive nature of the generation process. Moreover, the bidirectional attention within each sequence leverages the spatial coherence inherent in visual data, which aligns better with the non-sequential structure of image representations:

$$\mathbf{A}_{i,j} = \begin{cases} 1, & \text{if } j \leq i \text{ or } (i,j) \in \text{same segment} \\ 0, & \text{otherwise} \end{cases}, \tag{6}$$

Algorithm 1 Inference Procedure for VRC

Require: Text prompt $\mathbf{x}_{\text{text}}$, maximum frame count T
Ensure: Generated video tokens $\mathbf{X}_{\text{video}}$
Initialize $\mathbf{s} \leftarrow \mathbf{x}_{\text{text}}$ ▷ Concatenate text tokens
Step 1: Generate first frame tokens autoregressively
for $i = 1$ to N **do**
  Generate token $\mathbf{x}_i \sim P(\mathbf{x}_i \mid \mathbf{x}_{<i}, \mathbf{x}_{\text{text}})$ using autoregressive decoding
end for

Step 2: Compress the first frame tokens
Encode the first frame tokens $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ to obtain the sequence of embeddings: $\mathbf{E} = \text{Embed}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$
Compress the embedding sequence $\mathbf{E}$ using the representation compressor to get compressed embeddings: $\mathbf{Z} = \text{Compress}(\mathbf{E})$
Update $\mathbf{s} \leftarrow \text{Concat}(\mathbf{s}, \mathbf{Z})$ ▷ Concatenate compressed first frame with text tokens

Step 3: Autoregressively generate subsequent frames and compress them
for $i = 1$ to $\frac{T}{t}$ **do**
  Generate the compressed embeddings for the next sequence segment:
  $\mathbf{Z}_{i\times M:(i+1)\times M} \sim P(\mathbf{Z}_{i\times M:(i+1)\times M} \mid \mathbf{s})$
  Upsample the compressed embeddings to obtain the token representations: $\mathbf{X}_i \leftarrow \text{Upsample}(\mathbf{Z}_{i\times M:(i+1)\times M})$
  Downsample the token representations back to compressed embeddings: $\mathbf{Z}_{i\times M:(i+1)\times M} \leftarrow \text{Compress}(\mathbf{X}_i)$
  Update $\mathbf{s} \leftarrow \text{Concat}(\mathbf{s}, \mathbf{X}_i)$ ▷ Concatenate compressed embeddings to the sequence
end for

Step 4: Convert final embeddings to video tokens
$\mathbf{X}_{\text{video}} \leftarrow \text{Decode}(\mathbf{s})$
return $\mathbf{X}_{\text{video}}$

where $\mathbf{A}_{i,j}$ denotes the attention mask for the embeddings. This allows each embedding $\mathbf{z}_j$ within the current sequence to attend to previous embeddings and those within the same sequence but restricts access to subsequent sequences. By generating embeddings in chunks and using bidirectional attention within each chunk, VRC achieves efficient sequence modeling without information leakage, enhancing both prediction accuracy and representation learning.

The predicted representations of subsequent video frames are processed by a learnable representation decompressor, which decompresses them and computes the logits for the corresponding video tokens. These logits, along with those from the first frame tokens, are then used to compute the overall loss. The loss function is composed of two components: one for the generation of the first frame and another for the generation of the subsequent frames, i.e.,

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{first}} + \mathcal{L}_{\text{sub}}, \\ \mathcal{L}_{\text{first}} &= \mathbb{E}_{x_i}\left[-\log P_\theta(x_i \mid x_{<i}; x_{\text{text}})\right], i < N, \\ \mathcal{L}_{\text{sub}} &= \mathbb{E}_{x_{i\times N:(i+1)\times N}}\left[-\log P_\theta\left(x_{i\times N:(i+1)\times N} \mid \right.\right. \\ &\qquad \left.\left. x_{\text{text}}; x_{0:N}; \mathbf{z}_{0:i\times M};)\right], 1 \leq i \leq \frac{T}{\tau}. \end{aligned} \tag{7}$$

4.4. Inference

During the inference phase, the VRC model follows a sequential process to generate video frames conditioned on a given textual prompt. This process can be broken down into three primary stages: (1) generating the tokens for the first frame autoregressively, (2) compressing the first frame tokens, (3) autoregressively generating and compressing subsequent frames, and (4) converting the final embeddings into video tokens. The details of the entire procedure are summarized in Algorithm 1.

5. Experiments

5.1. Experimental Settings

We evaluate our proposed VRC framework on both class-to-video and text-to-video generation. Then we analyze the effect of model size, model initialization, representation compression ratio, and memory consumption. Finally, we demonstrate samples of our generation results for both class-to-video and text-to-video generation and give qualitative analysis.

Implementation Details. All experiments were conducted on 8 NVIDIA A100 GPUs, utilizing a global batch size of 96. The model was implemented in three model sizes: Base (B), Large (L), and XLarge (XL), with parameter counts of 100M, 340M, and 770M, respectively. We employed the Adam optimizer [7] with a learning rate set to 1×10^{-4}. The video representation model compresses inputs at three levels with compression ratios of 4, 16, and 64, with the default set to 4. The architecture of the compressor and decompressor modules varies based on the compression ratio: for a ratio of 4, we utilize 2-layer convolutional neural networks (CNNs) for both the compressor and decompressor, while for ratios of 16 and 64, we employ 3-layer and 4-layer CNNs along with corresponding transposed convolutional layers. For video tokenization, we leverage the OmniTokenizer [22], which encodes video sequences consisting of 17 frames at a resolution of 256×256 pixels into discrete tokens of size $5 \times 32 \times 32$. For text-to-video generation, the parameter of our model is 1.2B. In addition, we initialize the parameter of our video model through a pretrained class- or text-to-image model.

Datasets and Evaluation Metrics. We evaluate our proposed VRC framework on class-to-video and text-to-video generation tasks. For class-to-video generation, following Latte [9], we conduct experiments on the FaceForensics [11], SkyTimelapse [27], UCF101 [15], and Taichi-HD [13] datasets. Following previous works [9, 14, 29], the evaluation metric is the Fréchet Video Distance (FVD) [20], which assesses the quality of generated videos by comparing feature distributions between generated and real videos using a pretrained Inception network. Lower FVD scores indicate higher video fidelity. We utilize the FVD implementation from StyleGAN-V [14]. For text-to-video generation, we train on 300K samples from the Vimeo dataset [10] and evaluate using both FVD and CLIP Similarity (CLIPSIM) [26]. CLIPSIM quantifies the semantic alignment between the generated videos and the input text. Zero-shot evaluations are performed on the MSR-VTT [28] dataset, where CLIP-based similarity scores are calculated between the generated and ground-truth videos. Higher CLIPSIM scores indicate better alignment with the textual prompts. Both evaluations are conducted on videos with a resolution of 256×256 in 16 frames.

Table 1. FVD comparison on class-to-video generation datasets. * indicates that we reproduced the results of Loong using training on 17 frames. "FFS" represents the FaceForensics dataset. Bold numbers indicate the best performance.

Method	FFS	SkyTimelapse	UCF101	Taichi-HD
GAN-based Video Generation Model				
MoCoGAN [18]	124.7	206.6	2886.9	-
MoCoGAN-HD [18]	111.8	164.1	1729.6	128.1
DIGAN [32]	62.5	83.11	1630.2	156.7
StyleGAN-V [14]	47.41	79.52	1431.0	-
MoStGAN-V [12]	39.70	65.30	1380.3	-
Diffusion-based Video Generation Model				
PVDM [33]	355.92	75.48	1141.9	540.2
LVDM [4]	-	95.20	372.0	99.0
Latte [9]	27.08	42.67	333.61	97.09
LLM-based Video Generation Model				
VideoGPT [29]	185.9	222.7	2880.6	-
Loong* [24]	46.11	62.71	254.47	105.53
VRC (Ours)	**26.64**	**41.95**	**250.53**	**96.39**

Table 2. Comparison of average time consumption of generating one video clip of 17 frames on a single A100 GPU, under different model sizes. Blue text indicates the acceleration factor of our proposed method (VRC) compared to the baseline models.

Method	B (100M)	L (340M)	XL (770M)
Latte [9]	5.00 s (5.68×)	6.02 s (3.42×)	6.61 s (2.29×)
Loong [24]	12.32 s (13.98×)	15.22 s (8.64×)	26.08 s (9.03×)
VRC (Ours)	**0.88 s**	**1.76 s**	**2.89 s**

5.2. Quantitative Results

Class-to-Video Generation. As shown in Table 1, our proposed framework, **Vision Representation Compression** (VRC), outperforms other methods across all datasets on FVD scores. It demonstrates the effectiveness of our vision representation compression strategy in eliminating redundancy and improving model learning. In particular, VRC achieves this superior performance with significantly faster inference speeds, thanks to the reduced sequence

Table 3. Comparison of video models on CLIPSIM and FVD metrics for zero-shot text-to-video generation on MSR-VTT. "Para." indicates the model parameter.

Method	Backbone	Para.	CLIPSIM↑	FVD↓
CogVideo [6]	Diffusion	5B	0.2631	1294
MagicVideo [36]	Diffusion	-	-	998
ModelScopeT2V [21]	Diffusion	1.7B	0.2930	550
Show-1 [34]	Diffusion	4.3B	0.3072	538
VideoPoet [8]	LLM	8B	**0.3049**	**213**
Loong [24]	LLM	7B	0.2903	274
VRC (Ours)	LLM	1.2B	0.2890	308

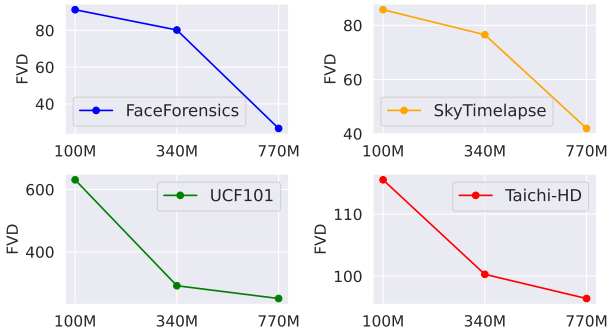

Figure 4. Performance comparison on various class-to-video datasets across different model sizes of our method.

length enabled by representation compression. Specifically, VRC shows an acceleration of over 9× compared to the typical autoregressive LLM-based method (Loong), as shown in Table 2. Moreover, VRC, as an LLM-based video generation model, even outperforms diffusion-based models such as Latte in inference speed, showcasing its efficiency.

Text-to-Video Generation. As shown in Table 3, VRC achieves results comparable to recent proposed video generation models such as Loong, VideoPoet, and Show-1. Considering that our current model size is significantly smaller than that of VideoPoet and Loong due to computational resource limitations, these results are particularly impressive. Compared to ModelScope, which has a similar model size, VRC demonstrates similar CLIPSIM scores and better FVD scores. It is worth noting that VideoPoet is not open-sourced, and other diffusion-based models exhibit similar generation speeds to Latte. As shown in Table 2, VRC achieves much faster inference speeds while maintaining comparable generation quality.

5.3. Analysis

Scaling Laws Analysis. To evaluate whether our proposed VRC framework can scale effectively, we assess the performance of class-to-video generation across four datasets using models of varying sizes, ranging from 100M to 770M parameters. The results are shown in Figure 4. Due to great performance with a compression rate of 4 in Figure 5, the scale analysis experiments are conducted with a compression rate of 4. Performance consistently improves as the model size increases, demonstrating the scalability of our proposed VRC framework.

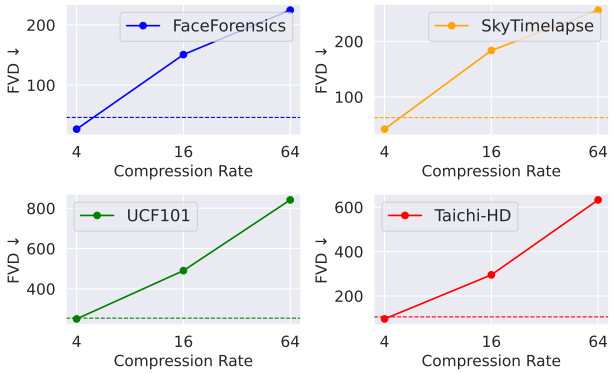

Figure 5. Performance of different compression rates on various datasets. The dotted line is the FVD score of the baseline without any representation compression (Loong⋆).

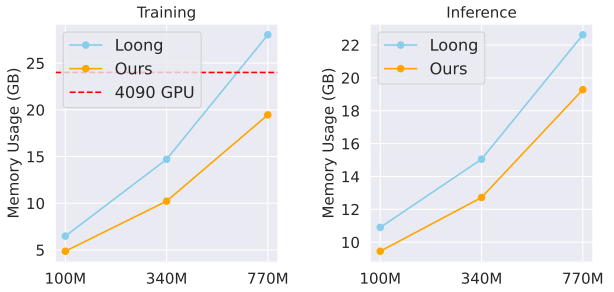

Figure 6. GPU memory usage for training (single A100, batch size 1) and inference (single A100, batch size 16) across different model sizes. The ref dotted line is the GPU memory of NVIDIA 4090 GPU (24GB).

Analysis of Compression Rate. We evaluate the effect of different representation compression rates during training, ranging from 4 (2×2), 16 (4×4), to 64 (8×8), on four class-to-video generation datasets. The results, shown in Figure 5, indicate that the 4 compression rate achieves the best performance, with further increases failing to improve results. The model performance decreases as the compression rate further increases. Excessive compression can lead to information loss, leaving insufficient information for the model to predict subsequent frames accurately. In particular, models trained with a compression rate of 4 (2×2) outperform the baseline model without compression (Loong⋆), as indicated by the dotted line. This empirical evidence demonstrates that in VRC, representation compression enhances the model's learning capability through eliminating redundant information.

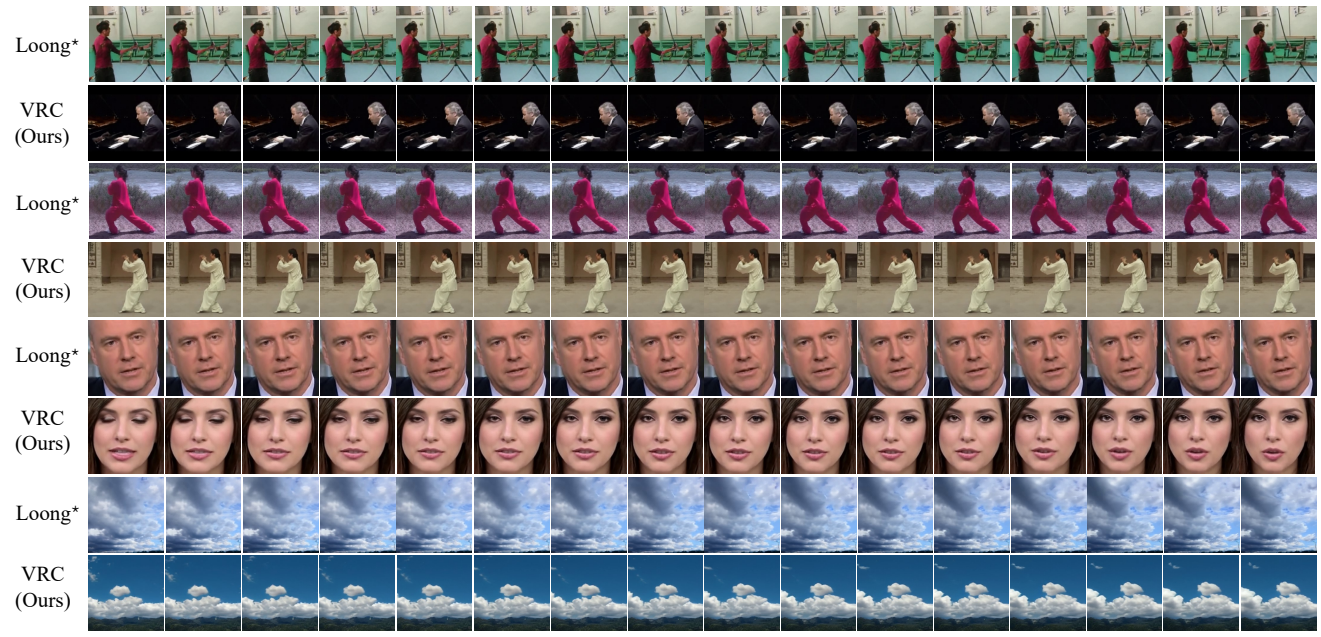

Figure 7. Videos generated by Loong* and VRC are sampled on the following class-to-video datasets: UCF101, Taichi-HD, FaceForensics, and SkyTimelapse.

Figure 8. Text-to-video results generated by VRC. Each video contains 17 frames, displayed here with an interval of 2 for clarity.

Analysis of GPU Memory. Figure 6 shows the GPU memory usage of our method compared to the baseline (Loong) across model sizes (100M, 340M, 770M). Our approach consistently requires less GPU memory, with a reduction of around 30% during training and around 20% for inference. This efficiency allows better scalability to larger models and lower latency in practical deployments. In addition, our method can be fine-tuned on a single NVIDIA 4090 GPU (24GB memory) even for the 770M model, whereas Loong exceeds this capacity.

5.4. Qualitative Analysis

We present qualitative samples of our generated results for class-to-video generation and text-to-video generation in Figure 7 and Figure 8, respectively. As shown in Figure 7, VRC produces video clips of higher quality compared to Loong* across all four datasets. For instance, in the UCF samples, Loong* (first row) shows cumulative errors in the generated case. In the Taichi-HD samples, Loong* (third row) confuses the person's chest and arm, whereas VRC (fourth row) generates a more realistic and anatomically correct human body. In FaceForensics, the face generated by Loong* (fifth row) appears almost static, while VRC (sixth row) produces dynamic facial expressions and movements. For text-to-video generation, the samples demonstrate that VRC effectively follows textual instructions to generate coherent, realistic, and smooth video clips.

6. Conclusion

In this work, we proposed **Vision Representation Compression** (VRC), a framework for autoregressive LLM-based video generation that leverages representation compression to address redundancy and inefficiency. VRC's learnable compressor and decompressor modules reduce sequence length and enhance the model's ability to capture video structures. Experiments show VRC achieves better generation results on multiple class-to-video datasets, surpassing even diffusion-based models like Latte in FVD scores and offering up to $9\times$ faster inference compared to common LLM-based methods like Loong. This work highlights the potential of representation compression as a key strategy for advancing LLM-based video generation, and paves a new way for more efficient, scalable, and effective LLM-based video generation models.

References

[1] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. *arXiv preprint arXiv:2401.12945*, 2024. 1

[2] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. 1, 2

[3] Guanjie Chen, Xinyu Zhao, Yucheng Zhou, Tianlong Chen, and Yu Cheng. Accelerating vision diffusion transformers with skip branches, 2024.

[4] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2022. 2, 6

[5] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022. 1

[6] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022. 2, 7

[7] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[8] Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023. 1, 2, 7

[9] Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024. 1, 2, 6

[10] Luca Rossetto, Heiko Schuldt, George Awad, and Asad A Butt. V3c–a research video collection. In *MultiMedia Modeling: 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8–11, 2019, Proceedings, Part I 25*, pages 349–360. Springer, 2019. 6

[11] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018. 1, 6

[12] Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Mostgan-v: Video generation with temporal motion styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5652–5661, 2023. 2, 6

[13] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019. 6

[14] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3626–3636, 2022. 2, 6

[15] K Soomro. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6

[16] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. *arXiv preprint arXiv:2104.15069*, 2021. 2

[17] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 3

[18] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1526–1535. Computer Vision Foundation / IEEE Computer Society, 2018. 6

[19] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018. 2

[20] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 6

[21] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *CoRR*, abs/2308.06571, 2023. 2, 7

[22] Junke Wang, Yi Jiang, Zehuan Yuan, Binyue Peng, Zuxuan Wu, and Yu-Gang Jiang. Omnitokenizer: A joint image-video tokenizer for visual generation. *arXiv preprint arXiv:2406.09399*, 2024. 1, 3, 6, 11

[23] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 1, 3

[24] Yuqing Wang, Tianwei Xiong, Daquan Zhou, Zhijie Lin, Yang Zhao, Bingyi Kang, Jiashi Feng, and Xihui Liu. Loong: Generating minute-level long videos with autoregressive language models. *arXiv preprint arXiv:2410.02757*, 2024. 1, 2, 3, 6, 7

[25] 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. 2, 3

[26] Chenfei Wu, Lun Huang, Qianxi Zhang, Binyang Li, Lei Ji, Fan Yang, Guillermo Sapiro, and Nan Duan. Godiva: Generating open-domain videos from natural descriptions. *arXiv preprint arXiv:2104.14806*, 2021. 6

[27] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2364–2373, 2018. 6

[28] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016. 6

[29] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021. 1, 2, 6

[30] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021. 3

[31] 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. 3

[32] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. *arXiv preprint arXiv:2202.10571*, 2022. 2, 6

[33] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18456–18466, 2023. 2, 6

[34] David Junhao Zhang, Jay Zhangjie Wu, Jia-Wei Liu, Rui Zhao, Lingmin Ran, Yuchao Gu, Difei Gao, and Mike Zheng Shou. Show-1: Marrying pixel and latent diffusion models for text-to-video generation. *International Journal of Computer Vision*, pages 1–15, 2024. 2, 7

[35] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024. 1, 3

[36] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022. 2, 7

A. Model Details

The model architecture includes several variants of GPT with different sizes, tailored to balance computational efficiency and model performance. The configurations are as follows:

- **B (100M):** 12 layers, 12 heads, hidden dimension is 768
- **L (3430M):** 24 layers, 16 heads, hidden dimension is 1024
- **XL (770M):** 36 layers, 20 heads, hidden dimension is 1280
- **1.2B:** 22 layers, 32 heads, hidden dimension is 2048

All model variants use a codebook size of 8192.

The architecture also incorporates a **compressor** and a **decompressor**, designed to achieve a compression rate of 4. These components leverage convolutional operations, activation functions, and batch normalization for effective feature transformation and spatial adjustments.

- **Compressor:** The compressor reduces the spatial resolution of the input while preserving its essential features. It consists of a convolutional layer with a stride of 2 and a kernel size of 3, enabling progressive downsampling. The final layer employs a 1×1 convolution to adjust the output channel dimensions without altering the spatial resolution.
- **Decompressor:** The decompressor restores the spatial resolution to its original size. It employs a transposed convolutional layer with a stride of 2 and a kernel size of 4 to upsample the input. Similar to the compressor, a final 1×1 convolution ensures the output matches the required channel dimensions.

Both the compressor and decompressor utilize ReLU activation functions and batch normalization to enhance nonlinearity and improve training stability.

B. Importance of Image Pretraining

The training pipeline for video generation models, including class-to-video and text-to-video tasks, is divided into two distinct stages. In the first stage, a class- / text-to-image model is trained. This involves using class or text descriptions along with 10 randomly sampled frames from each video as the training dataset. And the second stage begins, focusing on training the class- / text-to-video model. Throughout both stages, the image and video training data are encoded using the same video tokenizer, OmniTokenizer [22], ensuring consistent data representation across the training pipeline.

To validate the effectiveness of image generation pretraining for initialization in VRC, we conduct an ablation study by comparing models trained with and without image generation pretraining across four class-to-video generation datasets. As shown in Table 4, models initialized with image generation pretraining achieve significantly better FVD scores across all datasets. These results highlight the importance of image generation pretraining for providing a strong initialization, which enhances the performance of VRC in video generation tasks.

Table 4. FVD comparison with and without image generation pretraining across different datasets.

Pretraining	FaceForensics	SkyTimelapse	UCF101	Taichi-HD
✓	26.64	41.95	250.53	96.39
×	73.74	80.41	279.74	135.22

C. User Study

We conducted a user study to compare our VRC framework with the baseline model (Loong*). For this, we sampled 50 videos per model using identical text prompts. Participants were shown pairs of videos generated by the two models in a randomized order and asked to choose their preferred video based on Video Consistency and Video-Text Matching. The study was blind, with users unaware of the model identities. We collected 300 responses. As shown in Figure 9, our VRC model was preferred by users in both categories, with 70% favoring VRC for video consistency and 79% for text alignment, demonstrating superior visual coherence and semantic accuracy.

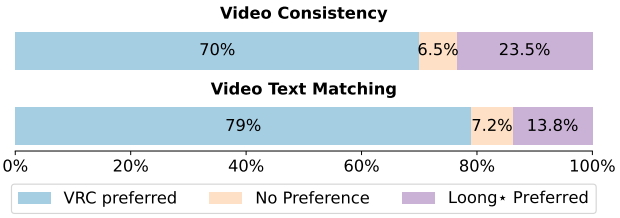

Figure 9. User study on text-to-video generation with 17 frames.

D. Data Source of Figure 2

To generate the data used in Figure 2, we first encode a sequence of five frames into a video using the H.264 encoding standard. The following FFmpeg command is used for this purpose:

```
ffmpeg -i ./frames/frame_%d.png -c:v
    libx264 -crf 23 -g 5 -keyint_min 5 -
    sc_threshold 0 output.h264
```

where the command takes the input images (‘frame_1.png‘, ‘frame_2.png‘, etc.) and compresses them into an H.264 video file (‘output.h264‘). Key parameters include:

- `-crf 23`: Specifies the compression level, with 23 being the default value that balances quality and file size.

- `-g 5`: Sets the Group of Pictures (GOP) length to 5 frames, ensuring one I-frame (intra-coded frame) is followed by four P-frames (predictive-coded frames).
- `-keyint_min 5`: Ensures a minimum interval of 5 frames between two consecutive I-frames.
- `-sc_threshold 0`: Disables scene change detection, enforcing the specified GOP structure regardless of content changes.

Next, we extract the file size of each encoded frame to analyze the amount of data used for the I-frame and subsequent P-frames. This is done using the following FFprobe command:

```
ffprobe -show_frames -select_streams v -
    show_entries frame=pkt_size -of csv=p=0
    output.h264 > frame_sizes.csv
```

where the command processes the H.264 video file and outputs a CSV file (‘frame_sizes.csv‘) where each row corresponds to the size (in bytes) of a specific frame. The ‘pkt_size‘ field provides the size of the encoded frame, enabling us to analyze the differences in data usage between the I-frame and P-frames. The resulting data highlights the redundancy in video frames, as P-frames typically require significantly less data than the I-frame.”