Long Video Generation using the VAE-GAN method

Jingbo Yang

PhD

University of York

Computer Science

September 2024

Abstract

Video generation has emerged as a critical area in machine learning, with applications spanning entertainment, virtual reality, and surveillance. However, generating realistic and temporally coherent videos, especially for long-term sequences, remains challenging. This thesis addresses these challenges through novel hybrid models and transformer-based architectures, improving video quality, efficiency, and duration.

The thesis first analyses limitations of existing generative models. While GANs produce sharp videos but suffer from computational expense and mode collapse, VAEs are more efficient but yield blurry outputs. We propose hybrid VAE-GAN models that combine their strengths by combining the inference ability of the VAE with the generative properties of the GAN, using a VAE encoder with GAN generators to enhance video consistency and continuity.

Focusing on temporal modelling, we address the critical challenge of long-duration video generation under computational constraints. Emphasizing GPU memory efficiency, we develop a novel recall mechanism that decomposes videos into temporally coherent sub-sequences with Markovian dependencies. This enables efficient long-term modelling with fixed memory requirements. Further refinement through auto-regressive modelling enhances temporal consistency, while our introduction of the Generative Pre-trained Transformer (GPT) architecture provides global temporal perspective through latent space sequence modelling.

The thesis provides several key contributions: (1) Encoding GAN3 (EncGAN3), integrating VAE and GAN for high-quality short-term videos; (2) Recall Encoding GAN3 (REncGAN) with a recall mechanism for efficient long-duration generation, developed through iterative architectural improvements; (3) Auto-Regressive R2 (AR2) with auto-regressive recall; and (4) GPT R2 (R3) leveraging transformer architectures. These models achieve minimal, fixed GPU memory increments regardless of video length.

Experimental results demonstrate significant improvements in both short- and long-term video generation across multiple benchmarks, showing superior performance in quality, coherence, and computational efficiency. This thesis advances video generation techniques, particularly for applications requiring high-quality extended sequences.

Contents

List of Figures

List of Tables

List of Acronyms

1. Literature Method

GAN	Adversarial Generative Networks
AE	AutoEncoders
VAE	Variational Autoencoder
AAE	Adversarial Autoencoder
VQ-VAE	Vector Quantised-Variational AutoEncoder
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GPT	Generative Pre-trained Transformer
VDM	Video Diffusion Model
LDM	Latent Diffusion Models
VGAN	Generative Adversarial Network for Video
TGAN	Temporal Generator
MoCoGAN	Motion and Content decomposed Generative Adversarial Network
DVD-GAN	Dual Video Discriminator GAN
DIGAN	Dynamics-aware Implicit Generative Adversarial Network
TATS	Time-Agnostic VQ-GAN and Time-Sensitive Transformer

2. Our Method

EncGAN3	Encoding GAN3
LEncGAN	LSTM EncGAN3
REncGAN	Recall EncGAN3
R2	REncGAN
Improved R2	An improved version of R2
AR2	Auto-Regressive REncGAN
R3	GPT Recall Encoding GAN3

3. Operation

R_3VD	3-stream Video Discriminator operation in the Recall mechanism
VDrej	Video Discriminator Rejection operation

4. Module

F-SA	Factorized Self-Attention
TSA	Convolutional Temporal-wise Self-Attention
SSA	Convolutional Spatial-wise Self-Attention
Enc	Encoder
CoEnc	Content Encoder
MoEnc	Motion Encoder
Moenc	the part consist of convolutional layers in MoEnc
Mofc	the part consist of fully-connected layers in MoEnc
CoLSTM	Content LSTM
MoLSTM	Motion LSTM
G	Generator
VD	Video Discriminator
ID	Image Discriminator

5. Layer

Conv/conv	Convolutional (Layer)
1D	1-Dimensional
Conv2D/conv2d	2-Dimensional Convolutional Layer
fc	fully connected
FC	fc Layers
BN/bn	Batch Normalization (Layer)
maxpool2d	2D Max Pooling (Layer)
ReLU	Rectified Linear Unit (Activation Function)
DeConv/deconv	Transposed Convolutional (Layers)

6. Loss

KLD	Kullback-Leibler Divergence
recon	Reconstruction Error
avg	Average
concat	Concatenation

7. Metric

FID	Fréchet Inception Distance
IS	Inception Score, contains two components: Intra-Entropy H(y|x) and Inter-Entropy H(y)
video FID	Fréchet Inception Distance for videos (denote as FID in Table)
video IS	Inception Score for videos (denote as IS in Table)
FVD	Fréchet Video Distance

8. Dataset

FPS	Frames Per Second
UvA	UvA-NEMO
Weiz	Weizmann
Taichi	Tai-Chi-HD

Sky	Sky-Timelapse

List of Symbols

1. EncGAN3

N	the number of videos used for training, $i = 1, \ldots, N$.
T_i	the frame index of video i, $j = 0, \ldots, T_i$. In EncGAN3, $T_i = 15$ while $T_i > 100$ in the long-term video generation task.
T	simplification of T_i.
$\mathbf{x}_{ij}$	the j-th frame from the real i-th video.
$\mathbf{x}_j$	simplification of $\mathbf{x}_{ij}$.
$\mathbf{x}_{0:T}$	a clip of frames from index 0 to index T.
$\widehat{\mathbf{x}}_{ij}$	the j-th frame from the generated i-th video.
$\mathbf{v}_{ij}$	a different map calculated by subtracting adjacent frames, such as $\mathbf{x}_{i,j-1}$ and $\mathbf{x}_{i,j}$. Here, $j = 1, \ldots, T_i$.
$\widehat{\mathbf{x}}_n$	a frame sampled from the video generated by latent codes.
$\tilde{\mathbf{x}}_n$	a frame sampled from the video generated by Gaussian noises.
idx	index.
θ_x	parameters of the Content Encoder.
θ_v	parameters of the Motion Encoder.
$\mathbf{z}_{co}$	the content latent space.
$\mathbf{z}_{mo}$	the motion latent space.
$\mathbf{z}_\mathbf{x}$	the sampled content latent code.
$\mathbf{z}_\mathbf{v}$	the sampled motion latent code.
$\tilde{\mathbf{z}}_\mathbf{x}$	the sampled content noise.
$\tilde{\mathbf{z}}_\mathbf{v}$	the sampled motion noise.
$\mathcal{N}(0, \mathbf{I})$	standard Gaussian distribution, used to sample random noises.
$p(\mathbf{z}_\mathbf{x})$	a prior given distribution of $\mathbf{z}_\mathbf{x}$, generally given as $\mathcal{N}(0, \mathbf{I})$.
$p(\mathbf{z}_\mathbf{v})$	a prior given distribution of $\mathbf{z}_\mathbf{v}$, generally given as $\mathcal{N}(0, \mathbf{I})$.
$q_{\theta_\mathbf{x}}(\mathbf{z}_\mathbf{x}\|\mathbf{x})$	an approximate posterior distribution of $\mathbf{z}_\mathbf{x}$ given $\mathbf{x}$.
$q_{\theta_\mathbf{v}}(\mathbf{z}_\mathbf{v}\|\mathbf{v})$	an approximate posterior distribution of $\mathbf{z}_\mathbf{v}$ given $\mathbf{v}$.

2. LEncGAN, REncGAN, AR2 and R3

N_L	the number of long videos used for training, $i = 1, \ldots, N_L$. Same as N.
N_C	the number of clips within a long video, $j_c = 1, \ldots, N_C$. N_C is varied for long videos.

T_c	the frame index of a clip. Here, $T_c = 15$ for index starts from 0.
T_o	the number of overlapped frames.
r	index of the reference frame, $r = T_c - T_o$, denoting the boundary of the overlapping region. If $T_o = T_c//2 + 1$, $r = T_c//2 + 1 = T_o$.
$\mathbf{x}_r$	the binding frame, also called as the reference frame.
X^1	former real clip.
X^2	latter real clip.
$\widehat{X}^3$	stitched generated clip.
$\widehat{\mathbf{x}}_{stitch}$	$\widehat{X}^3$.
$\oplus$	pixel-wise addition
$\ominus$	pixel-wise subtraction
E_c	the Content Encoder.
E_m	the Motion Encoder.
$\mathbf{z}_c$	Simplification of $\mathbf{z}_{co}$.
$\mathbf{z}_m$	Simplification of $\mathbf{z}_{mo}$.
c_t	cell state at time step t.
h_t	hidden state at time step t.
$c + 1$	the next state of LSTM.
D_V	Video Discriminator
D_I	Image Discriminator

3. Training objectives

D_{KL}	Kullback-Leibler Divergence.
L_{Enc}	loss function of two-stream Encoder in EncGAN3.
L_G	training objective function of the Generator in EncGAN3.
L_{D_I}	training objective of the image-stream Discriminator in EncGAN3.
L_{D_V}	training objective of video-stream Discriminator in EncGAN3.
$L_{Enc,\mathbf{v}}$	replacing the reconstruction term of videos to be the reconstruction of the first frame and frame difference maps in L_{Enc}.
$L_{G,\mathbf{v}}$	replacing the reconstruction term of videos to be the reconstruction of the first frame and frame difference maps in L_G.
L_{EncG}	loss function for training the Encoder and Generator jointly in REncGAN.
$L_{D_I,R}$	the loss function of the image-stream Discriminator in REncGAN, after removing the random generator components of L_{D_I}.
$L_{D_V,R}$	the loss function of the video-stream Discriminator, after removing the random generator components of L_{D_V}.
$L_{D_V,R2}$	the video-stream Discriminator loss function in REncGAN, after applying the R_3VD operation to $L_{D_V,R}$.
$L_{EncG,AR}$	the loss function for jointly optimizing the Encoder and Generator in AR2.
$L_{D_V,AR}$	the training objective of the video-stream Discriminator in AR2.
L_{GPT}	the training objective for GPT in R3.

To my parents, whose unwavering support and belief in my abilities have been a constant source of inspiration throughout this journey.

To my mentor, Dr. Adrian G. Bors, whose guidance and encouragement were crucial in helping me navigate the challenges of this research.

And to my friends and colleagues, whose camaraderie and constructive feedback made this work not only possible but enjoyable.

Your support has made all the difference, and I am deeply grateful for your presence in my life.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to everyone who has supported me during my four years of PhD research.

I am especially grateful to my supervisor, Dr. Adrian G. Bors. From the beginning of my PhD journey to its completion, through the remote research period during the COVID-19 pandemic, the transition back to on-campus life, the challenges of overcoming research bottlenecks, and finally the intense writing phase, he has always been there to support me. His advice, guidance, and support were crucial to the progress of my research, and I am sincerely thankful.

I would also like to thank my internal assessor, Professor Nick Pears. His insightful questions and feedback on my research opened my eyes to his unique academic perspective and deep research insights. These have greatly inspired me and had a profound impact on my subsequent work. His dedication to research and focus on its purpose left a lasting impression on me.

Special thanks go to my friends: Fei Ye, Zechao Hu, Guoxi Huang, Qiran Lai, and Nat. Their support during the remote phase when I started my PhD was invaluable, helping me quickly acclimate to the PhD journey and greatly easing the sense of isolation that often accompanies research. Our discussions on various research topics and creative ideas not only provided me with valuable inspiration but also expanded and deepened my understanding of other fields.

I am also grateful to the University of York for providing the resources and environment necessary for this research.

Lastly, I would like to thank my parents, Yuepeng Yang and Meifang Ma, and my dear sister, Chengyue Yang. Throughout my PhD studies, their unwavering support and care, both physically and emotionally, have been invaluable to me, and I am deeply grateful.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Parts of the research described in this thesis have previously been published in:

- Yang, Jingbo and Bors, Adrian. "Encoder enabled GAN-based video generators." In *2022 IEEE International Conference on Image Processing (ICIP)* (pp. 1841–1845). IEEE. [1]
- Yang, Jingbo and Bors, Adrian G. "Enabling the Encoder-Empowered GAN-based Video Generators for Long Video Generation." In *2023 IEEE International Conference on Image Processing (ICIP)* (pp. 1425–1429). IEEE. [2]

Portions of this thesis incorporate material from the aforementioned publications, which have been cited appropriately throughout the text.

Chapter 1

Introduction

1.1 Introduction to Video Generation

Video generation is a dynamic and rapidly evolving field focused on creating video content from various inputs, often without explicit guidance on frame-by-frame details. It plays a critical role in several cutting-edge applications, such as entertainment, virtual reality, surveillance, and crime scene reconstruction. The field has garnered attention due to its potential to revolutionize how content is produced and visualized, particularly as AI techniques become more advanced and widely available.

However, generating high-quality, realistic video poses significant challenges, primarily in maintaining spatial and temporal coherence and dynamics across frames. Videos differ from images in that they represent movement through changes in consecutive frames, that depend on the 3D scene structure and lighting, as well as on how these would change their properties over time, complicating the generation process. Models should balance generating individual high-resolution frames with ensuring smooth transitions and dynamic, coherent motion across sequence. As the video length increases, these issues are magnified, especially when attempting to generate long-term, detailed temporal frame sequences.

Recent advancements have led to two primary categories of video generation: short-term and long-term. Short-term generation typically involves a few frames, focusing on detailed visuals and coherent, dynamic short-range motion, while long-term generation aims at producing extended sequences, requiring sophisticated techniques to maintain visual consistency, natural motion dynamics and avoid visual degradation over time.

The applications for video generation are wide-ranging. In entertainment, it offers opportunities for fully automated content creation and special effects. In virtual reality, synthetic video can create immersive environments, while it is also used in surveillance simulations and crime scene reconstruction, AI-generated videos can assist in simulating events for analysis. However, the field remains technically demanding due to its computational complexity, with ongoing research dedicated to improving efficiency, the diversity of generated content, as well as with the ability to create longer, higher-resolution videos without loss of quality.

1.2 Motivation and Research Problem

The task of generating realistic and coherent videos presents substantial challenges due to the complex interplay between spatial quality and temporal consistency. Existing video generation models, particularly those based on Generative Adversarial Networks (GANs) [3] and Variational Autoencoders (VAEs) [4], have made significant strides but still face critical limitations. These shortcomings underline the need for more advanced approaches, particularly hybrid models that combine the strengths of different architectures.

Limitations of GANs in Video Generation: GANs have been pivotal in advancing video generation, yet they encounter several major issues. The primary challenge lies in their high computational cost. Video generation is inherently more complex than image generation because it requires the model to capture both spatial coherence (within each frame) and temporal relationships (across consecutive frames). This added complexity dramatically increases the demand for computational resources, particularly in unconditional video generation, where sequences are generated without any guiding input. Furthermore, GANs suffer from a common issue known as mode collapse, where the generator produces repetitive or homogeneous outputs, leading to videos that lack variation despite using different random inputs. Solutions like G^3AN [5] attempt to address this by disentangling motion and content, but the fundamental challenges remain unresolved in many models.

Limitations of VAEs in Video Generation: In contrast to GANs, VAEs are typically less computationally demanding due to their structured latent space, which aids in efficient learning. However, their biggest drawback is the fact that their outputs generate blurred images and videos. VAEs use reconstruction loss to measure the similarity between the generated and original images, but this often results in smoothed-out, less detailed visuals. This problem is further exacerbated in the video generation, where maintaining high visual fidelity across several frames is crucial for producing temporal coherence in the resulting video. The trade-off between computational efficiency and image sharpness makes VAEs less suitable for producing the high-quality visuals that GAN-based methods can achieve.

The Need for Hybrid Models: Given these challenges, hybrid models that combine the strengths of both GANs and VAEs offer a promising solution. Hybrid models, such as VAE-GAN, leverage the encoder structure of VAEs to maintain efficient learning while incorporating adversarial training from GANs to generate sharper, more realistic outputs. This combination aims to address the shortcomings of each approach individually, using the latent space optimization of VAE for more controlled generation and the adversarial training of GAN for improved visual quality. These hybrid models have been shown to have excellent potential in producing more diverse and high-quality video sequences while managing computational costs more effectively.

Research Gap: Temporal and Spatial Coherence and Dynamic in Long-Duration Video Generation. While substantial progress has been made in short-term video generation (generating tens of frames with high visual quality), long-term video generation (generating hundreds+ frames with high visual quality) remains a significant challenge. As sequences extend to hundreds or even thousands of frames, maintaining both spatial and temporal coherence becomes increasingly difficult. Many models, such as MoCoGAN-HD [6] and DIGAN [7], have demonstrated the ability to generate high-resolution videos over

short periods. However, when tasked with generating longer sequences, these models often struggle to retain motion consistency and visual quality [8]. The underlying causes of these failures reveal deeper challenges in temporal dynamics modelling. These limitations stem from fundamental differences in motion generation dynamics across temporal scales. While the density representation of complex video data has facilitated short-term video generation, the inherent data complexity grows exponentially with longer temporal scales, making it increasingly difficult to fully model and generate long-duration videos. As a result, some approaches attempt to infer future frames by extracting implicit world rules embedded in the video data, which shifts the challenge from direct sequence modelling to an even more complex problem.

The inherent challenges of motion generation exhibit distinct characteristics in short-term versus long-term video generation. Notably, the difficulty of modelling different motion types (e.g., high-frequency movements like rapid insect wing flapping vs. low-frequency movements like slow vehicle motion) stems from optimization challenges in handling their fundamentally different temporal patterns. In short-term generation (tens of frames), methods typically succeed with regular motions (e.g., human actions like waving or boxing) due to their predictable periodicity, while struggling with irregular natural phenomena (e.g., fluid dynamics or cloud movement). Recent advances show that enforcing spatio-temporal pixel coherence can mitigate these issues for irregular motions [8]. Conversely, long-term generation (hundreds+ frames) presents an inverse difficulty pattern: regular motions become increasingly challenging due to (1) accumulating errors in each prediction step, (whereas irregular motions benefit from temporal coherence mechanisms that scale better with duration), and (2) the lack of modelling for extended durations. Without understanding complete motion cycles beyond training sequence lengths, models tend to either repeat observed patterns or generate physically implausible extrapolations, even when frame quality remains acceptable. This dichotomy manifests clearly in output quality degradation - generated sequences frequently exhibit repetitive patterns (particularly in regular motions), unrealistic rigid content (when focusing solely on pixel-wise coherence) and motion discontinuities (if prior to ensure the frame quality) [9, 10].

The technical bottleneck exacerbating these challenges lies in temporal dimension representation efficiency. General video generation approaches attempt to increase generation length by simply extending training video duration, but face prohibitive GPU memory constraints (see Table 3.7). Our analysis reveals that increasing temporal length incurs much greater memory overhead than spatial resolution increases, highlighting fundamental limitations in existing temporal representations. This observation motivates our investigation into novel video representation paradigms that mitigate the strong correlation between memory requirements and generation duration. Moreover, the computational complexity of long-term video generation increases as models must capture intricate temporal dependencies, such as object motion, lighting changes, and scene transitions, across a large number of frames. This complexity often results in degraded video quality as sequences lengthen, with models often trained on short-term sequences, failing to scale effectively [9]. Recent advancements have pushed the boundaries to generate sequences of up to 1024 frames, but challenges in preserving spatial detail and temporal coherence, especially in complex human actions, remain unresolved [9, 11].

In summary, there is a clear need for more sophisticated approaches to address the limita-

tions of current models, particularly in the realm of dynamic long-duration video generation. Hybrid methods that integrate the strengths of different architectures represent a promising direction, but further research is needed to fully realize their potential. Specifically, ensuring consistent motion and visual clarity over extended time periods remains a critical and unsolved research problem.

1.3 Overview of Key Techniques

This section discusses the foundational techniques that underpin the models discussed in this thesis. We start with core generative models, such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), and then discuss their hybrid forms [12, 13, 14]. We also cover more recent advancements, particularly Transformer-based models [15, 16, 17, 18, 19] relevant for long video generation [20, 9, 21].

VAE, GAN, and Hybrid Models. Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are fundamental to generative modelling. VAEs work by encoding data into a latent space and reconstructing it, introducing probabilistic latent variables to ensure smooth transitions between generated data samples. However, VAEs often produce image or video outputs that lack sharpness, a limitation addressed by GANs. GANs consist of a generator and a discriminator that compete with each other, resulting in highly realistic outputs but often suffering from unstable training.

Hybrid models that combine VAEs and GANs leverage the strengths of both approaches. VAEs are effective at modelling latent space, while GANs excel at generating realistic outputs. For instance, the VAE-GAN model proposed by [13] integrates the VAE decoder with the GAN generator, enhancing both latent space optimization and image quality. Similarly, the Adversarial Autoencoder (AAE) [12] utilizes adversarial learning to refine latent space representations, crucial for high-quality video generation. These hybrid models offer a balance between generated sample diversity and quality factors such as the realism of the representation.

In our first proposed method Encoding GAN3 (EncGAN3), we use a combination of VAE and GAN to model video data effectively, generating frames that maintain both temporal coherence and visual fidelity [1]. Models such as TwoStreamVAN [22] and G^3AN [5] have also contributed by incorporating multi-stream architectures to address both spatial and temporal dimensions in video [23, 24, 25, 26, 27, 28].

Transformer-Based Models for Video Generation. The advent of Transformer architectures has greatly enhanced the capabilities of generative models, especially for sequence-based tasks like video generation [15]. Unlike traditional sequence models such as Recurrent Neural Networks (RNNs) [29] or Long Short-Term Memory (LSTM) networks [30, 31, 32], Transformers use a self-attention mechanism to capture dependencies across an entire sequence simultaneously. This makes them particularly effective in modelling long-term dependencies, which is crucial for generating coherent long-duration videos.

Generative Pre-trained Transformers (GPTs), originally developed for text generation, have also proven effective for video generation [18, 19]. Their ability to handle sequential data with

long-term context allows models like GPT to auto-regressive generate future video frames based on previously generated ones, ensuring temporal consistency and mitigating issues like frame drift. This approach outperforms older models that struggle to maintain visual and temporal coherence over extended sequences.

In this work, Transformer-based architectures are central to the models designed for long video generation. For example, the proposed GPT Recall Encoding GAN3 (R3) model adapts GPT, a decoder-only Transformer to video generation tasks, using self-attention to model latent space sequences and maintain continuity over extended time intervals for the generated videos. By focusing on continuous latent variables, this approach avoids the video repetition issues common in models relying on discrete latent spaces.

Hybrid and Transformer-Based Methods in Long Video Generation. For long video generation, traditional methods such as VAEs and GANs, even hybrid forms, face challenges when scaling to handle hundreds or thousands of frames [8, 9, 10]. The integration of Transformers addresses these limitations by leveraging self-attention to preserve temporal coherence across longer sequences [9]. Additionally, models like R3 extend the capabilities of earlier hybrid models by incorporating Transformer-based latent space modelling, enabling the generation of long, dynamic video sequences with fewer artifacts and modelling more variation in motion and content.

Newer Developments and Trends. While this thesis primarily explores VAE-GAN hybrids and Transformer-based methods, it is important to note emerging developments, particularly the integration of Diffusion models [33, 34, 35, 36, 37] with Transformer architectures [21, 38]. Diffusion models, which have shown impressive results in image and video generation [34, 35], represent a distinct framework from VAEs and GANs. These models generate high-quality, temporally consistent video sequences by denoising data over time, a technique that complements the sequence-learning capabilities of Transformers. Although these approaches are not covered in this thesis, they offer promising directions for future research.

1.4 Research Objectives

The main objective of this research is to advance video generation by addressing key challenges related to the computational complexity, output quality, and temporal coherence, particularly for long-duration videos.

Enhancing Video Generation Models: This thesis introduces and evaluates several models designed to improve video generation. EncGAN3 combines VAE and GAN technologies to enhance short-term video generation by incorporating a dual-stream encoder, which aids in producing videos with improved spatial and temporal coherence [1]. This model aims to achieve high visual quality and diversity in the generated videos, as demonstrated by performance metrics.

Improving Long-Duration Video Generation: To tackle the limitations of existing models in generating long-duration videos, this thesis introduces Recall EncGAN3 (REncGAN), which incorporates a recall mechanism to model temporal relationships between video

clips [2]. This approach significantly reduces memory requirements during computation while improving coherence over extended video sequences. The goal is to generate long videos efficiently while maintaining high visual quality and spatial-temporal consistency.

Advancing Temporal Modelling: Building on the advancements of REncGAN (R2), Auto-Regressive REncGAN (AR2) and R3 introduce new strategies for modelling long-term temporal relationships. AR2 adapts the recall mechanism to an auto-regressive setting, while R3 integrates a GPT module to handle long-duration sequences more effectively. These models aim to reduce the reliance on prior information and improve the overall quality and consistency of generated videos.

Overall, the research seeks to balance computational demands with the generated video output quality, ultimately contributing to the development of robust and efficient video generation models capable of producing high-quality, coherent long-duration videos.

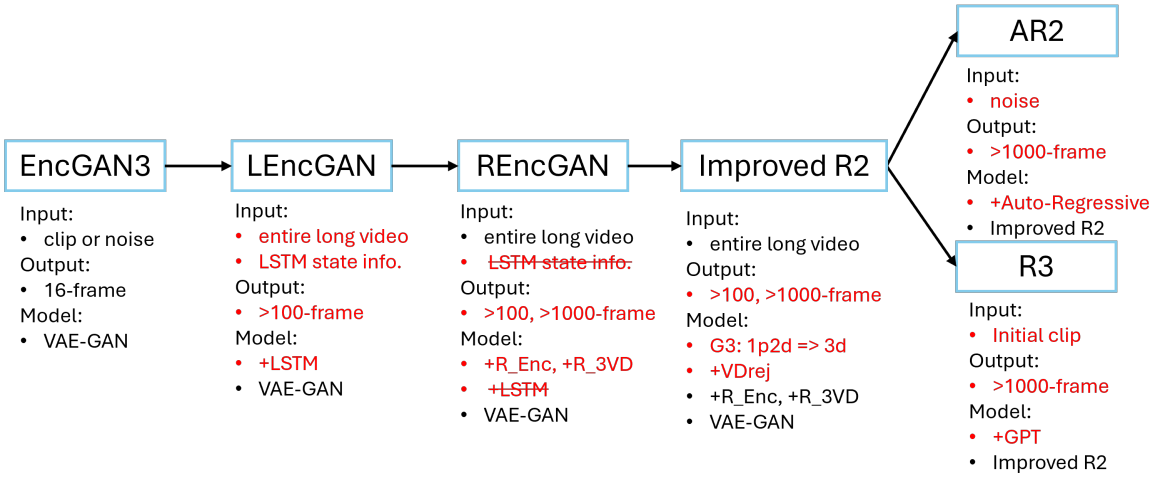


Figure 1.1: Developing digraph of our models.

1.5 Contributions of the Research

This thesis presents several advancements in video generation, focusing on improving both the quality and length of generated video sequences through innovative model architectures and mechanisms. The diagram in Figure 1.1 illustrates the models proposed in this thesis and their relationships. The key contributions of this research are detailed below:

1. **EncGAN3: Enhancing Generative Performance with Encoder-enabled Architecture**
   EncGAN3 represents a significant advancement in video generation by integrating a multi-stream architecture that leverages the strengths of both Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). Key contributions include:

   - **VAE-GAN Hybrid Model:** EncGAN3 combines the generative capabilities of GANs with the inference strength of VAEs. This hybrid approach allows the

model to generate high-quality videos by incorporating a VAE-based encoder that provides a representative latent space for the GAN generator, thus improving training stability and video fidelity.

- **Quantitative and Qualitative Improvements:** Extensive experiments on benchmark datasets demonstrate that EncGAN3 achieves competitive results in terms of video Fréchet Inception Distance (FID) [39, 40] and video Inception Score (IS) [9, 41, 42] metrics. These results highlight the model's ability to generate coherent and high-quality video sequences, setting a new standard for generative performance in video generation.

2. **REncGAN (R2): Enabling Long-Duration Video Generation with the Recall Mechanism.**
REncGAN builds upon the foundation laid by EncGAN3, advancing the ability to generate long-duration videos efficiently. This development stems from our initially developed model LSTM EncGAN3 (LEncGAN), which is our earlier attempts that introduced LSTM for long video generation. While LEncGAN marked an important step forward, the reliance on LSTM made it less efficient and often led to discontinuities in video generation due to the inherent limitations of its memory mechanism when handling extended temporal dependencies.

   REncGAN addresses these limitations by removing the LSTM module and introducing the more efficient Recall mechanism. This mechanism enables the model to handle long-duration video generation by focusing on the temporal relationships between successive short clips rather than modelling each frame individually. The core contributions of REncGAN can be summarized as follows:

   - **Temporal modelling via Recall Mechanism:** By leveraging the Recall mechanism, REncGAN models temporal dependencies between consecutive short clips, reducing the memory footprint significantly while still capturing the long-term dynamics required for generating extended video sequences. This allows for the generation of long videos without the computational overhead of frame-by-frame processing.
   - **Markov Chain-Based Inter-Clip Relationships:** Unlike LEncGAN, which depended on LSTM, REncGAN employs a Markov chain approach to model inter-clip relationships, sharing weights across chain states. It models Markovian dependencies of sub-sequences, that is clips. This strategy ensures the continuity and consistency in the generated videos, improving efficiency without sacrificing the quality or coherence of long video sequences.
   - **Advanced Loss Function and Evaluation:** The new loss function developed for REncGAN ensures better synchronization between the encoding and generation processes, leading to superior performance in the long-duration video generation task. The experimental results demonstrate that REncGAN excels in generating videos with higher visual quality and stronger spatial-temporal consistency than the previous approaches.

   By replacing the LSTM with the Recall mechanism, REncGAN not only it improves upon the efficiency and performance of LEncGAN but also offers a more robust solution for generating coherent and continuous long-duration videos.

3. **AR2 and R3: Reducing Reliance on Prior Information while Maintaining Quality.**
   The introduction of AR2 and R3 further extends the capabilities of the REncGAN framework by addressing the reliance on prior information and enhancing long-duration video generation. Key contributions include:

   - **AR2 - Auto-Regressive Recall Mechanism:** AR2 adapts the recall mechanism to an auto-regressive setting, reducing the dependency on prior inputs while maintaining proper-quality long video generation. This approach improves the model's ability to handle long video sequences without relying heavily on prior video clips.
   - **R3 - Long-Duration Modelling with GPT:** R3 introduces a Generative Pre-trained Transformer (GPT) module to model long-duration sequences within the latent space. This global perspective mechanism ensures the generation of more dynamic and diverse video content, reducing repetitive patterns and enhancing the overall quality of long videos.
   - **Comprehensive Temporal Modelling:** This research development provides three key temporal modelling approaches: intra-clip relationship modelling (EncGAN3), Markov-chain-based inter-clip modelling (R2), and long-duration modelling within the latent space (R3). This comprehensive strategy provides a framework for generating longer and more temporally complex video sequences while maintaining efficient training requirements.

Together, these contributions advance the field of video generation by improving the quality, coherence, and efficiency of generated videos across varying lengths and complexities.

1.6 Structure of the Thesis

The thesis is structured as follows:

1. **Chapter 1:** Introduces the topic, outlining the key research objectives and contributions.
2. **Chapter 2:** Provides the necessary background by reviewing related work, including generative models like GANs and VAEs, hybrid approaches, and transformer-based generative models.
3. **Chapter 3:** Focuses on EncGAN3, presenting the design of the model and its contributions to video generation.
4. **Chapter 4:** Introduces REncGAN (R2), highlighting its recall mechanism and improvements over earlier models in generating long, coherent video sequences.
5. **Chapter 5:** Covers the AR2 and R3 models, detailing their advancements in latent space modelling and the integration of GPT-based architectures for enhanced performance.

6. **Chapter 6:** Concludes the thesis by summarizing the key findings, contributions, and potential future research directions.

Chapter 2

Background

This chapter reviews the literature relevant to our research, focusing on generative models and their applications in video generation. Section 2.1.1 introduces Generative Adversarial Networks (GAN), covering its foundational principles, common uses, and specific limitations in video generation. In Section 2.1.2, we discuss Variational Autoencoders (VAE), highlighting their distinctions from GAN and potential advantages. Section 2.1.3 examines hybrid models that combine VAE and GAN, aiming to harness the strengths of both frameworks. Furthermore, Section 2.1.4 explores the use of Transformer-based models, emphasizing their utility in sequential data processing, particularly for video generation.

The review then transitions to video generation techniques in Section 2.2, outlining the field's progression from conditional to unconditional video generation methods. We explore how early methods leveraged prior information, such as initial frames or semantic maps, to simplify video generation. As models evolved, unconditional video generation emerged, generating videos from minimal input like random noise, offering advantages in flexibility and representation. Building on these developments, we examine innovations in long video generation, focusing on maintaining temporal consistency and video quality over extended sequences.

Finally, Section 2.3 reviews the specific models considered for the research undertaken in this thesis. We start with model G^3AN [5] and TwoStreamVAN [22] frameworks, which are presented together with other video generative models relying on VAE and GAN architectures. Then we introduce the transformer, which represents the basis for the GPT.

This section highlights the advancements and unique contributions of these models in the context of generating coherent and high-quality video sequences.

2.1 Generative Frameworks

In this section, we review the basic generative deep learning architectures.

2.1.1 Generative Adversarial Networks(GAN)

The Generative Adversarial Network (GAN) framework, introduced by Goodfellow et al. in 2014 [3], revolutionized the field of image generation by providing a novel adversarial structure based on the game theory for training generative deep learning networks. GANs consist of two key components: a Generator (G) and a Discriminator (D), where G creates new images, and D attempts to distinguish between real images from a dataset and those generated by G. This adversarial process drives the Generator to create increasingly realistic images over time.

The success of GANs in image generation led to their application in video generation tasks, where the goal is to generate a sequence of coherent images with temporal consistency. Early video GAN models, such as VGAN [43], introduced a two-stream architecture that separated content and motion to generate realistic videos, effectively imposing temporal restrictions on the output.

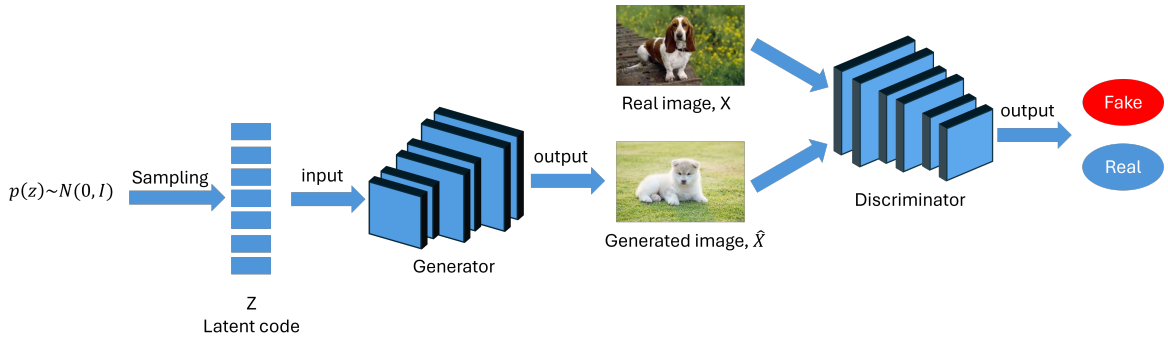


Figure 2.1: Data flow of GAN.

Background. As shown in Figure 2.1, the core idea behind GANs is the adversarial relationship between the Generator and Discriminator. The Generator learns to produce data (images or videos) that mimic real data, while the Discriminator attempts to correctly classify the input as either real or fake. This relationship is formalized through the adversarial loss function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data(x)}}[\log D(x)] + \mathbb{E}_{z\sim P_{noise(z)}}[\log(1 - D(G(z))] \tag{2.1}$$

where the minimization of weights in G affects the loss component $[\log(1 - D(G(z))]$, which forces the weights in G optimized for the direction on generating more realistic images so that could confuse D. While the maximization of weights in D affects the values in loss components $[\log D(x)]$ and $[\log(1 - D(G(z))]$ that directs D learns to identify generated images from real images. The training of D considers the real images x from the real data distribution $P_{data(x)}$ and the images generated from G while the training of G considers solely the random noises z come from a given distribution $P_{noise(z)}$, which generally given as a standard Gaussian distribution ($\mathcal{N}(0, \mathbf{I})$). The training on random noises is to ensure that the Generator G does not simply memorize and reproduce a single image but instead generates a variety of plausible outputs.

Limitations of using GANs in video generation. Despite its initial success, applying

GANs to video generation introduces new challenges. The first major limitation is the high computational cost. Unlike image generation, video generation requires learning not only spatial coherence but also temporal relationships between frames. This additional complexity demands significantly more computational resources, particularly for unconditional video generation, where the model generates entire video sequences from random noise without any guiding input.

Another common issue is the mode collapse, where the Generator fails to produce diverse outputs, even with a random noise input. In video generation, this leads to a lack of variation in the generated video sequences. Solutions such as G^{3}AN [5] address this by using disentangled representations of content and motion, effectively mitigating mode collapse and enhancing control over the generated videos.

2.1.2 Variational Autoencoder(VAE)

The Variational Autoencoder (VAE) was introduced in 2014 by Kingma and Welling [4] for image generation tasks. Like GANs, the VAE framework has been applied in the video generation domain by treating videos as sequences of images, where each frame is generated using the preceding frame and motion information. This enables the sequential generation of video frames, thus forming a video clip. For instance, models like VideoVAE [42] fall under the conditional video generation category, where each subsequent frame is conditioned on previous ones. Unlike GANs, which rely heavily on adversarial learning, VAEs focus on learning latent representations of data, making them a popular alternative for generative tasks.

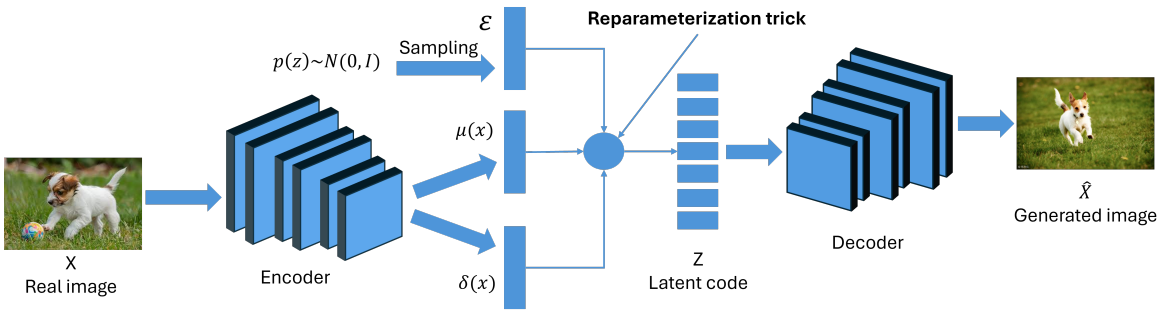


Figure 2.2: VAE architecture.

Background. As illustrated in Figure 2.2, the VAE consists of two key components: the encoder and the decoder, transferring a set of latent variables between them. The encoder compresses the input data into a latent distribution, typically a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, while preserving the most critical information from the input. The decoder then samples from this latent space and reconstructs the input data. The goal of the encoder is to map the input into a latent representation, from which the decoder can generate a realistic reconstruction.

To make backpropagation possible in deep learning, VAEs employ the reparameterization trick [4], which enables gradients to flow through the latent space. Instead of directly sampling from the distribution, the model first samples from a standard normal distribution

$\mathcal{N}(0, \mathbf{I})$, then scales this sample by the learned variance $\mathbf{\Sigma}$ and shifts it by the mean $\boldsymbol{\mu}$. This operation ensures that the latent space remains differentiable, allowing gradients to be propagated through both the encoder and decoder.

The loss function of VAE consists of two components: the reconstruction error and a regularization term (Kullback–Leibler divergence or KLD), defined as:

$$\mathcal{L} = -\frac{1}{L}\sum_{l} \mathbb{E}_{\sim q_\theta(z|x_i)}[\log p(x_i|z^{(i,j)})] - \sum_{j=1}^{J} \frac{1}{2}[1 + \log(\mathbf{\Sigma}) - \mathbf{\Sigma} - \boldsymbol{\mu}^2] \tag{2.2}$$

where the first term represents the reconstruction loss, which encourages the latent space distribution to accurately capture the input, and the second term is the Kullback–Leibler divergence (KLD), acting as a regularization term. KLD pushes the latent space distribution toward a standard normal distribution $\mathcal{N}(0, \mathbf{I})$, ensuring that the latent space is smooth and is characterized by sufficient variation. This regularization helps maintain the variation in the latent space and avoids the collapse of the latent distribution, which can occur in simpler AutoEncoders (AE), where the model tends to produce similar outputs for the same or even different inputs.

Comparing VAE with GAN. In contrast to GANs, VAEs benefit from the capability to probabilistically represent the features of learned data. This often leads to more efficient learning compared to the adversarial framework of GANs, making VAE models less computationally intensive and faster to train. However, a significant drawback of VAEs is their tendency to produce blurry images. This happens because the reconstruction loss prioritizes an averaged similarity between the generated and input images but struggles with fine-grained details, especially at pixel-level resolution.

While GANs use adversarial training to enforce sharper and more realistic outputs, VAEs rely on reconstruction, which can introduce uncertainty in the decoding process. As a result, even though VAEs are designed to handle variations in output images through latent space sampling, the generated images may not match their inputs. This limitation becomes even more evident in video generation, where temporal coherence and clarity are essential. GAN-based models typically generate sharper images compared to VAE-based models due to the adversarial loss function, which is more effective at producing visually sharper outputs.

2.1.3 Hybrid methods VAE-GAN

The hybridization of VAE and GAN can take various forms, with components of one model being incorporated into the other. For instance, VAE/GAN [13] offers a hybrid approach by merging the decoder and generator into a unified module. This model balances VAE and GAN equally, combining their strengths in latent space optimization and image generation. Another approach is represented by the Adversarial Autoencoders (AAE) [12], which integrate adversarial learning to enhance the latent space of VAE. Below, we discuss the detailed workings of VAE/GAN and AAE as two representative hybrid models.

VAE/GAN. The VAE/GAN model [13] combines the strengths of VAE and GAN into a single framework, as illustrated in Figure 2.3. The process begins with an input image $\mathbf{x}$,

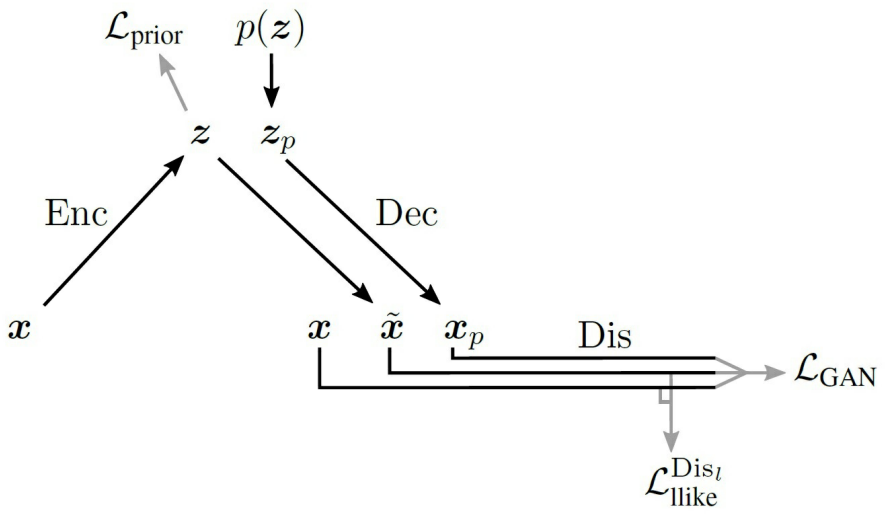


Figure 2.3: Data flow through the combined VAE/GAN model during training, from [13].

which is encoded into a latent space. From this space, latent codes $\mathbf{z}$ are sampled from this space, along with samples $\mathbf{z_p}$ from a prior distribution $p(\mathbf{z})$ (a standard normal distribution). Both $\mathbf{z}$ and $\mathbf{z_p}$ are then passed through the Decoder/Generator and compete with real images in the discriminator. This results in a three-stream data flow in the discriminator, optimizing both the latent space and the reconstructed images.

The inclusion of $\mathbf{z_p}$ provides sufficient positive samples during optimization, balancing the generation of latent space with image reconstruction. Without $\mathbf{z_p}$, the discriminator would primarily focus on image reconstruction, neglecting the quality of the latent space. By including $\mathbf{z_p}$, the VAE/GAN model ensures better regularization of the latent space.

The VAE/GAN loss function is defined as:

$$Loss = L_{prior} + L_{like}^{Dis_l} + L_{GAN}, \tag{2.3}$$

which includes three components: L_{prior} (the KLD regularization term), L_{GAN} (the standard GAN loss), and $L_{like}^{Dis_l}$. The term $L_{like}^{Dis_l}$ computes feature-level reconstruction error between real images and their generated counterparts, where the superscript Dis_l denotes features extracted from the l-th layer of the Discriminator, and the subscript $like$ indicates similarity measurement. This design allows assessing visual perception similarity beyond pixel-level comparisons. Additionally, the inclusion of $\mathbf{z_p}$ balances optimization between latent space regularization and image reconstruction. These properties have been crucial in developing models [44, 22, 1].

Adversarial Autoencoders (AAE). In contrast, AAE [12] introduces adversarial learning to improve the latent space encoding, as shown in Figure 2.4. In this model, the latent code sampled from the latent space $q(\mathbf{z})$ competes with samples drawn from a standard normal distribution $p(\mathbf{z})$ in a discriminator, encouraging the encoder to produce a latent space that closely resembles the prior distribution. This adversarial process replaces the KL-Divergence (KLD) regularization term commonly used in VAE, offering better-encoded latent spaces but at the cost of higher computational complexity.

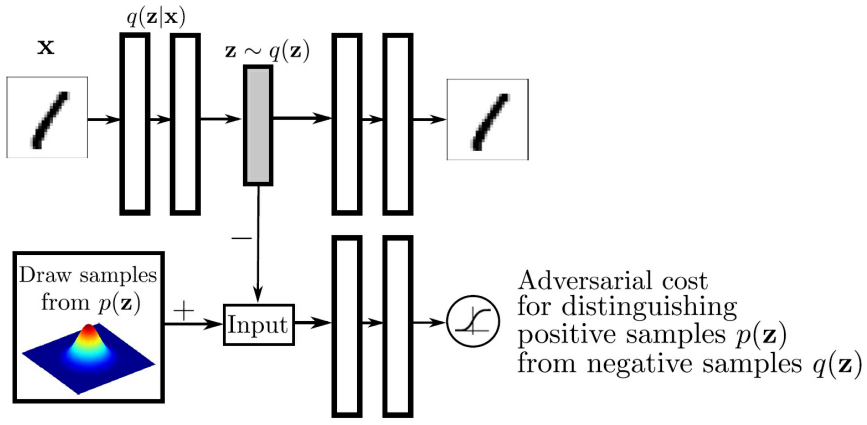


Figure 2.4: Architecture of an adversarial autoencoder(AAE) from paper [12].

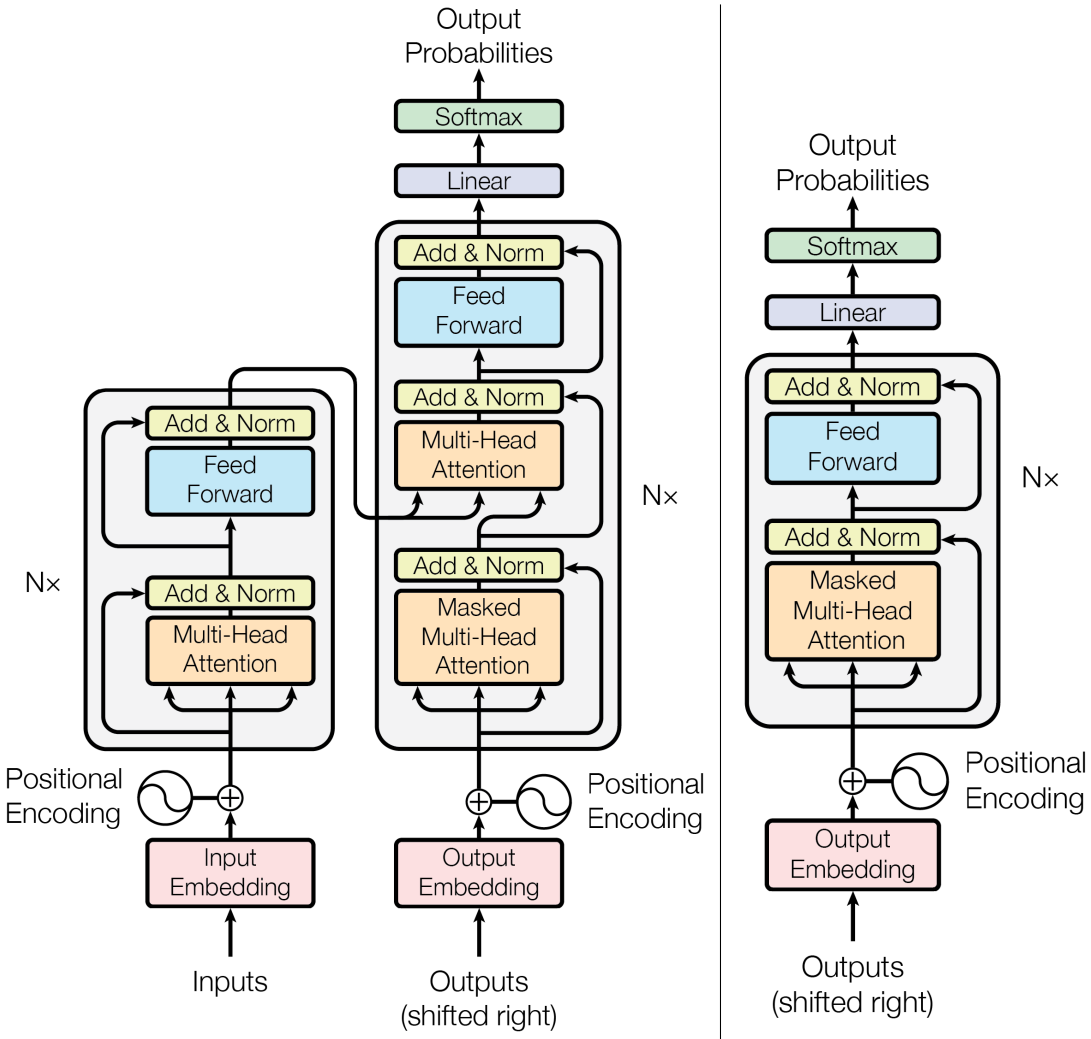
Figure 2.4 illustrates the typical VAE structure where latent codes generated by the encoder are further processed in the adversarial learning module. The latent codes $q(\mathbf{z})$ and the prior samples $p(\mathbf{z})$ are fed into a discriminator to distinguish between the two distributions. The adversarial loss effectively replaces the KLD term, resulting in superior encoding performance but requiring more resources. Such a latent processing mechanism have been shown in several models [45, 46, 47, 48].

2.1.4 Transformer Network

The Transformer model, introduced by Vaswani et al. in 2017 [15], has become a foundational architecture in sequence modelling tasks, particularly excelling in natural language processing. Unlike recurrent models such as RNNs [29] and LSTMs [30], which process sequences step by step, the Transformer leverages self-attention mechanisms to capture dependencies between all elements of the input sequence simultaneously. This enables it to model long-term dependencies effectively, which is critical for many generative tasks, while delivering outstanding performance [49, 50].

The core innovation of the Transformer is the Self-Attention mechanism. In this mechanism, each element in the sequence attends to every other element through a series of weighted connections computed via the Query, Key, and Value operations. The attention weights measure the relevance of one element to another, allowing the Transformer to capture relationships between distant elements in a highly parallelizable manner.

As illustrated in Figure 2.5 (a), the Transformer architecture consists of two primary components: an Encoder and a Decoder. The Decoder employs a Masked Self-Attention mechanism, ensuring that each element in the sequence only attends to preceding elements, thereby preventing future information leakage during sequence generation. A notable application of the Transformer Decoder is in the GPT, which utilizes only the decoder portion to model sequences and generates sequences in an auto-regressive manner [16, 17, 18]. GPT has demonstrated remarkable success in generating long, coherent text sequences by learning from vast amounts of text data [19]. Compared to traditional sequence models like LSTMs,



(a) Transformer [15] (b) GPT [17]

Figure 2.5: Architectures of the Transformer and GPT. (a) shows the original Transformer architecture including both encoder and decoder, as proposed in [15]. (b) illustrates the GPT architecture, which is based on the Transformer while using a decoder-only structure.

the Transformer-based GPT model offers significant advantages, including the ability to handle long-range dependencies and improved parallelization, both of which are crucial for generating long video sequences.

2.1.5 Diffusion-based Generative Models

Diffusion models have emerged as a powerful paradigm for generative modelling, leveraging a progressive denoising process to synthesize data through iterative refinement [33]. Unlike GANs that rely on adversarial training, diffusion models learn to reverse a predefined forward diffusion process — a Markov chain that gradually corrupts data by adding the random noise — thereby enabling high-quality and diverse sample generation [34].

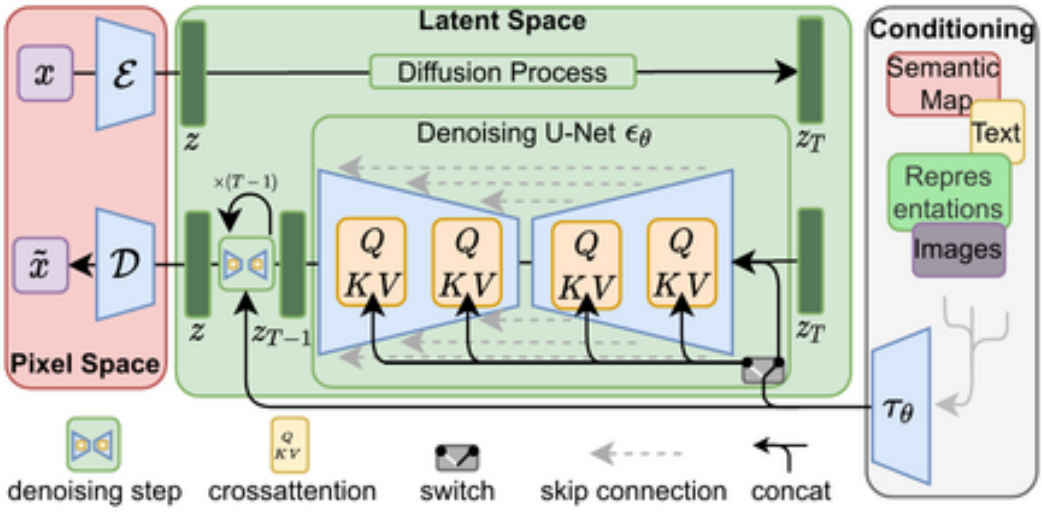


Figure 2.6: A conditional latent diffusion model [53].

Recent advances have extended this framework to video generation by incorporating temporal modelling techniques. Key innovations include frame-wise noise scheduling to maintain inter-frame consistency and spatio-temporal attention mechanisms to capture long-range dependencies across video sequences [36, 51]. For example, Video Diffusion Models (VDMs) [35] adapt the standard 2D U-Net architecture to 3D convolutions, explicitly modelling temporal dynamics by treating video as a volumetric data structure. Concurrently, CogVideo [52] combines diffusion with large-scale transformer architectures, achieving text-to-video synthesis through hierarchical generation and cross-modal alignment.

Compared to GANs, diffusion models mitigate mode collapse and produce samples with higher fidelity, though at increased computational costs and less straightforward controllability. To address efficiency, recent methods often perform denoising in a compressed latent space rather than directly on pixels, named as Latent Diffusion Models (LDMs) [54, 55]. As illustrated in Figure 2.6, an LDM comprises three core components:

1. **Variational Autoencoder (VAE) (left box)**: Maps pixel space to a lower-dimensional latent space.
2. **Diffusion Process (middle box)**: Operates in the latent space, where the forward process (upper part) adds noise via fixed equations, and the reverse process (bottom part) learns to iteratively denoise using a U-Net conditioned on control signals.
3. **Conditioning Mechanisms (right box)**: Cross-attention layers inject guidance from modalities like text or semantic maps, enabling controlled generation.

This framework significantly reduces computational demands while preserving generation quality, making it particularly suitable for high-resolution video synthesis.

2.2 Video Generation

Video generation is a rapidly evolving field that focuses on synthesizing video content from various inputs, including latent variables or prior information [56]. This section reviews the advancements in video generation methods, divided into short-term and long-term video generation.

2.2.1 Short-term Video Generation

Short-term video generation typically involves creating videos consisting of a limited number of frames, often focusing on achieving high-quality visuals and coherent motion over a brief duration. Early methods in this area utilized extensions of image generation models to handle the temporal dimension, such as 3D convolutions [57, 58, 59]. However, this approach proved computationally intensive and was not found as being practical in many applications.

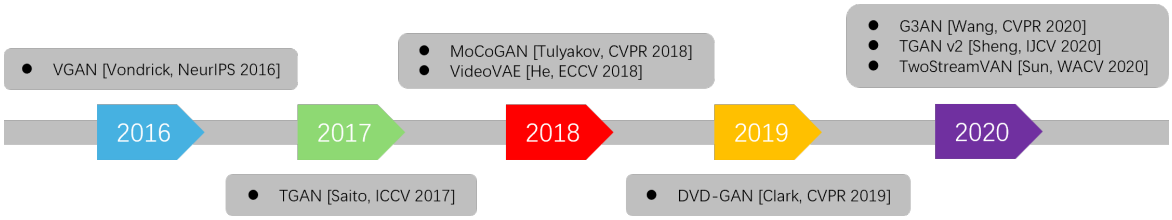


Figure 2.7: The Roadmap of Video Generation

Roadmap of Video Generation. As illustrated in Figure 2.7, one of the pioneering models in short-term video generation was VGAN (2016) [43], which introduced a two-stream architecture to handle motion and content separately. This model represented an important step forward in video dynamics modelling but was limited by the simplicity of its motion-handling mechanism.

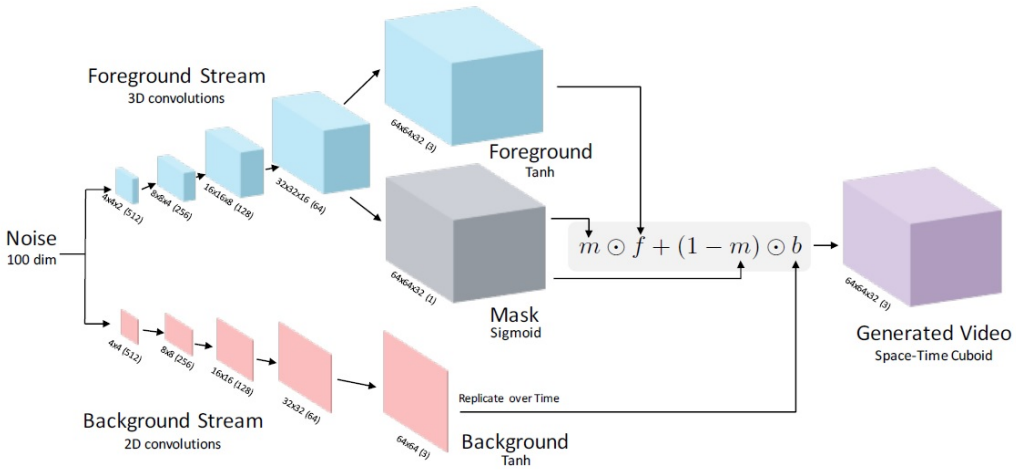


Figure 2.8: **VGAN Network Architecture [43].** This only illustrates the generator, which is the main modification part.

1. **VGAN: Two-Stream Generation (2016).** VGAN (generative adversarial network for video) introduced the use of independent foreground and background streams, where the foreground stream captured the dynamic components of the video, while the background stream focused on the static elements. These two streams were then combined using a mask generated by the motion stream to produce the final video, as illustrated in Figure 2.8. Although this model brought forth the idea of separating motion and content, it was constrained by the limited resolution (64×64) and short video sequences (32 frames) it could generate. Despite these limitations, VGAN laid the foundation for future work in video generation by introducing the concept of dual streams for video reconstruction.

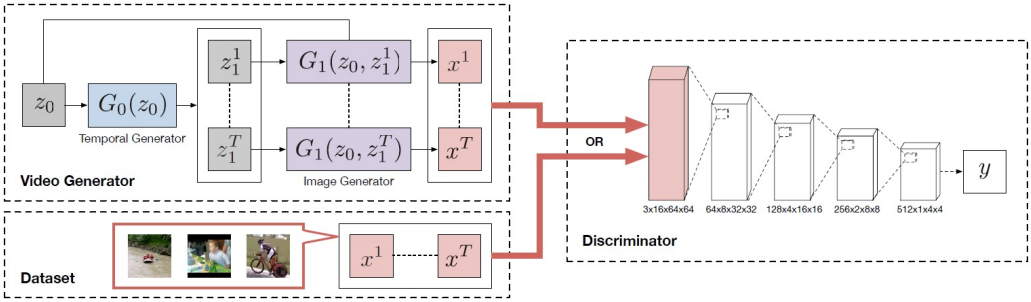


Figure 2.9: **TGAN Network Architecture [60].** The architecture is consist of three parts, the processing generated video, the real video from the dataset and the discriminator. And the Video Generator contains two generators, an Image Generator and a Temporal Generator.

2. **TGAN: Temporal Generator (2017).** Building upon the two-stream idea, Temporal Generative Adversarial Net (TGAN) [60] extended it further by introducing two separate generators, that is a temporal generator and an image generator, as seen in Figure 2.9. The temporal generator transforms an input noise into a sequence of temporal latent codes, which are then used by the image generator to reconstruct the individual frames of the video. By decoupling the generation of temporal and spatial components, TGAN improved the temporal coherence of the generated video, addressing one of the main shortcomings of VGAN. This method allowed for the generation of more complex and temporally consistent video sequences.

3. **MoCoGAN: LSTM-enhanced Temporal Modelling (2018).** Motion and Content decomposed Generative Adversarial Network (MoCoGAN) [61], introduced in 2018, marked a significant advancement by incorporating LSTM networks to enhance the temporal modelling of video sequences. This model employed a convolutional LSTM to encode the motion latent code, which, when combined with the content latent code, generated each video frame sequentially. As illustrated in Figure 2.10, this approach allowed MoCoGAN to generate videos with longer, more fluid sequences, overcoming the limitation of short video length seen in earlier models.

4. **DVD-GAN: Large-scale Dataset Training (2019).** Dual Video Discriminator GAN (DVD-GAN) [62] pushed the boundaries of video generation by training on large and complex datasets, such as Kinetics-600. This model leveraged high-capacity architectures to generate high-quality videos. However, despite its scalability, DVD-GAN

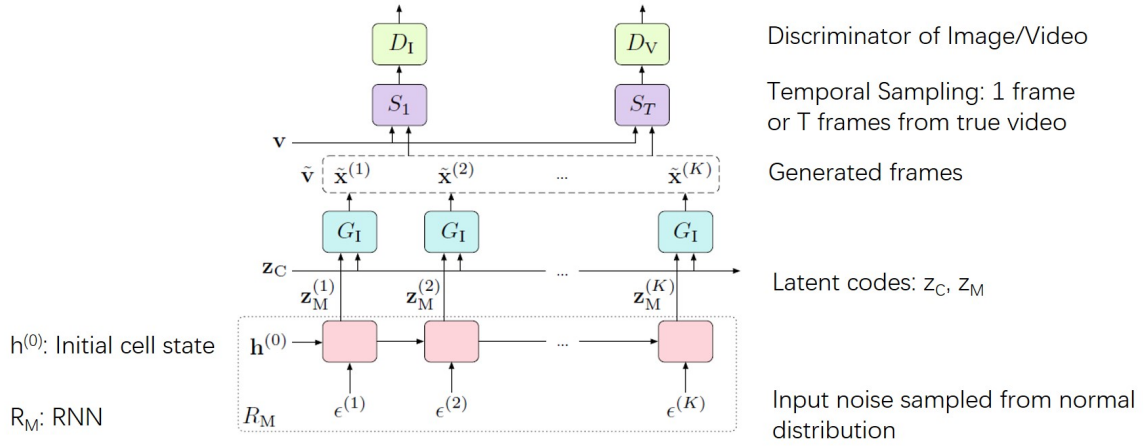


Figure 2.10: **The MoCoGAN framework [61]**. The framework is a hierarchical structure containing LSTM, Generator and Discriminator.

faced challenges in maintaining object shape consistency across frames during temporal variations, limiting its practical applicability in certain use cases.

5. **G^{3}AN: Multi-scale Motion and Content Handling (2020).** G^{3}AN [5] introduced a three-stream generator that processed motion and content at multiple scales, offering finer control over both spatial and temporal aspects of the generated video. This model improved temporal and spatial coherence by allowing the generator to fuse motion and content information at different resolutions. G^{3}AN addressed both the resolution limitations of earlier models and the challenges of maintaining temporal consistency, making it a significant contribution to the field.

6. **TGAN v2: Memory-efficient Video Generation (2020).** TGAN v2 [63] further refined the original TGAN by introducing multiple sub-generators to address GPU memory limitations, enabling the generation of higher-resolution videos. While this model improved the handling of spatial resolution, its primary focus remained on enhancing resolution rather than extending the temporal length of generated videos.

Refined Directions and Technological Developments. As video generation models matured, researchers began to explore more refined directions by integrating classic neural architectures, including LSTMs and VAEs, with GAN-based frameworks to enhance both video quality and temporal consistency.

1. **LSTM and VAE Integration in Video Generation.** Following MoCoGAN, which introduced LSTM into the video generation process, subsequent models also combine GANs with other well-established neural network modules such as LSTM, VAE and Transformer. For example, VideoVAE [42] used LSTM to generate control signals for steering the generation of video sequences. These control signals, along with the outputs of an encoder, helped create latent codes, enabling better handling of video dynamics. While these techniques are no longer considered cutting-edge, they continue to offer robust solutions for managing temporal coherence in video generation.

2. **Development of Dual-Stream Discriminators.** The dual-stream discriminator structure, initially introduced by MoCoGAN and further refined by DVD-GAN, remains a foundational approach in video generation. By separately evaluating spatial and temporal features, dual-stream discriminators ensure that generated videos maintain coherence both within frames and across time. Although the technique has become a more conventional part of the video generation toolkit, it is continually being optimized for better realism and consistency in video sequences.

From VGAN to G^3AN, the development of video generation models has progressively evolved from simple image-based extensions to more sophisticated approaches for modelling complex video dynamics. Future research continue to seek a balance between generation quality, temporal coherence, and computational efficiency to further enhance the effectiveness of video generation, particularly those GAN based methods [64, 65, 66, 67, 68].

2.2.2 Long-term Video Generation

Long-term video generation focuses on creating video sequences with extended temporal durations, often requiring sophisticated modelling techniques to ensure the continuity and consistency over many frames. The challenge in this area lies in maintaining high visual quality and coherent motion across a longer span of time, while dealing with significant computational complexity and memory constraints [69].

Extending video generation in spatial and temporal dimension. Initially, video generation models focused on relatively short sequences, typically around 16 frames, prioritizing high-resolution frame generation while being constrained by computational limits. As technology progressed, researchers developed more efficient video representations, enabling the generation of videos with increasingly higher resolutions. Contemporary methods can produce videos with resolutions ranging from 64×64 [5, 70, 35] to 128×128 [71, 63, 20, 7, 9, 36], 256×256 [8, 10, 72, 11], and even 1024×1024 [8, 6] pixels in special cases. For example, TGAN v2 [63] addresses GPU memory limitations by employing multiple sub-generators, thus efficiently managing higher resolutions.

Despite these advancements, generating long temporal video sequences remains a significant challenge. Producing longer videos requires accurate modelling of temporal dependencies while ensuring consistency in movement, lighting, and perspective changes. Models trained on shorter sequences often struggle when they are extended to longer durations, leading to rapid quality degradation and loss of temporal coherence, according to the results from [9, 8].

Recent developments have made notable progress in this area, with models now generating sequences of 64 frames [6], 128 or 256 frames [7, 11], and even up to 1024 frames [9] or longer [8, 10, 73], while previously the number of generated frames for videos have been around 10 to 32 frames [22, 5, 43, 60, 61, 74]. Although these advancements mark significant progress, challenges in preserving the quality and coherence of long video sequences remain.

Roadmap of Long Video Generation. As shown in Figure 2.11, MoCoGAN-HD [6], introduced in 2021, tackled the challenges of generating higher-resolution and longer video sequences. However, it faced difficulties in maintaining realistic structural changes in motion,

often resulting in videos that resembled ordered sequences of similar images.

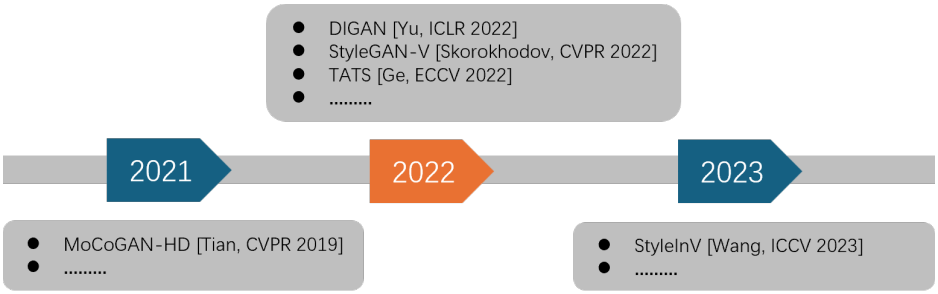


Figure 2.11: The Roadmap of Long Video Generation

Dynamics-aware implicit generative adversarial network (DIGAN) (2022) [7] represents a significant advancement in long-term video generation by incorporating spatio-temporal modelling to produce videos of up to 128 frames without notable quality degradation. DIGAN effectively handles both spatial and temporal aspects, resulting in more realistic and coherent videos of human actions.

StyleGAN-V (2022) [8] aimed for "infinite" video generation by employing a computationally cheaper video representation. While it achieved the capability to generate videos with resolutions of up to 1024×1024, it struggled with complex movements and often resulted in repetitive patterns due to the reuse of motion information.

Time-Agnostic VQ-GAN and Time-Sensitive Transformer (TATS) (2022) [9] advanced the field by generating long video sequences through capturing temporally related frame latent codes, achieving generating up to 1024 frames. Despite its success in extending generated video lengths and improving temporal smoothness, TATS encountered difficulties in generating complex human movements, often repeating initial frames.

StyleInV (2023) [11] utilizes a similar approach to MoCoGAN-HD by training a motion generator based on latent codes from a pre-trained StyleGAN2 image generator. It incorporates non-autoregressive training and supports style transfer through fine-tuning, enhancing its ability to generate high-resolution, long videos.

This progress in video generation underscores a growing focus on enhancing the temporal coherence and resolution of long video sequences. The shift from handling short-term sequences to generating long-term videos reflects the increasing sophistication in internal representations and reduced reliance on prior information.

2.2.3 Diffusion-based Video Generation

Recent advances in diffusion models have significantly impacted video generation [35, 36, 75, 37]. By extending image diffusion frameworks to the temporal domain, models like Imagen Video [76] generate high-fidelity videos through cascaded diffusion processes, while Make-A-Video [77] leverages text-image alignment for zero-shot text-to-video synthesis. A key innovation is the use of 3D U-Net architectures with spatio-temporal attention [35], enabling efficient modelling of frame dependencies. For long-term generation, latent diffusion

approaches (e.g., LVDM [75]) reduce computational costs by operating in compressed latent spaces. Despite their advantages in quality, diffusion models face challenges in real-time generation due to iterative sampling, prompting research into accelerated inference techniques while maintaining the generation quality [54].

2.3 Related models

In this section, we examine the details of several models that have laid the groundwork for the development of the models proposed in this thesis.

2.3.1 G^{3}AN for Video Generation

G^{3}AN is a GAN-based video generation model using factorized spatio-temporal convolution techniques to process video data in multiple streams [5].

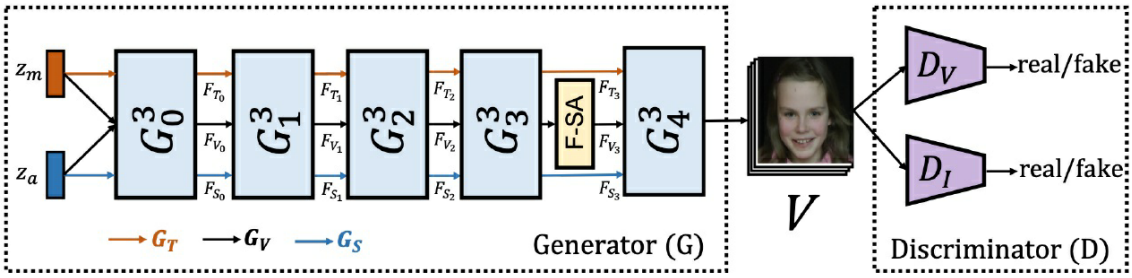


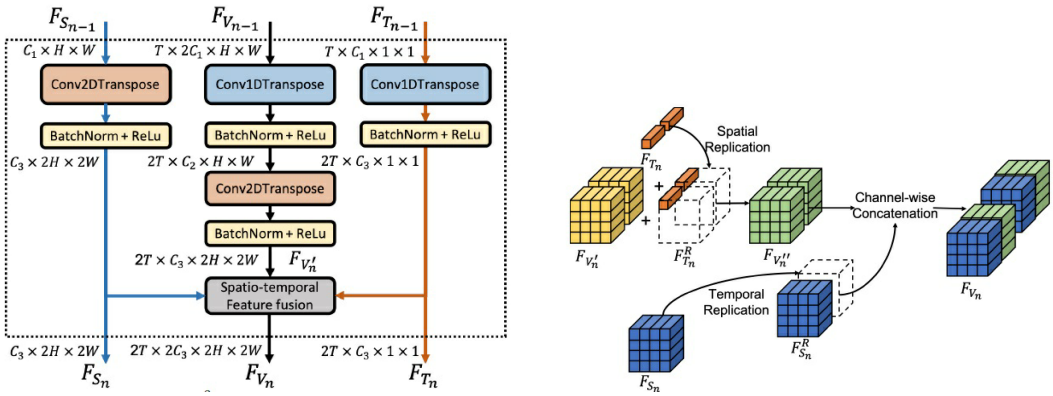
Figure 2.12: Overview of G^{3}AN architecture [5].

Architecture Overview. As shown in Figure 2.12, G^{3}AN employs a three-stream generator, where each stream processes motion, content, and video data separately. It is paired with a two-stream discriminator that evaluates both image and video data. This structure allows the model to effectively capture both spatial and temporal information, which is critical for generating coherent and realistic video sequences [6, 5, 62].

G^3 Modules. The generator in G^{3}AN uses specialized G^3 modules, which are based on factorized transposed spatio-temporal convolutions. These modules allow the model to process motion and content information separately, fusing spatial and temporal features to enhance the overall video stream. Figure 2.13 illustrates the internal structure of the G^3 modules and their spatial-temporal fusion mechanism, which enables the model to produce high-quality video sequences that maintain temporal consistency across frames.

2.3.2 LSTM for Sequence Modelling

Long Short-Term Memory (LSTM) networks are widely used for modelling long-term dependencies in sequential data, making them an important tool in video generation [30, 32].



(a) G^3 module structure [5] (b) Spatial-temporal fusion [5]

Figure 2.13: The internal structure of the G^3 modules within the generator (a) and the spatial-temporal fusion mechanism used in the G^3 module (b).

LSTM excels at capturing temporal dynamics across frames, which is essential for generating temporal coherent video sequences. The architecture includes memory cells that can retain information over extended sequences, allowing the model to learn long-range temporal dependencies.

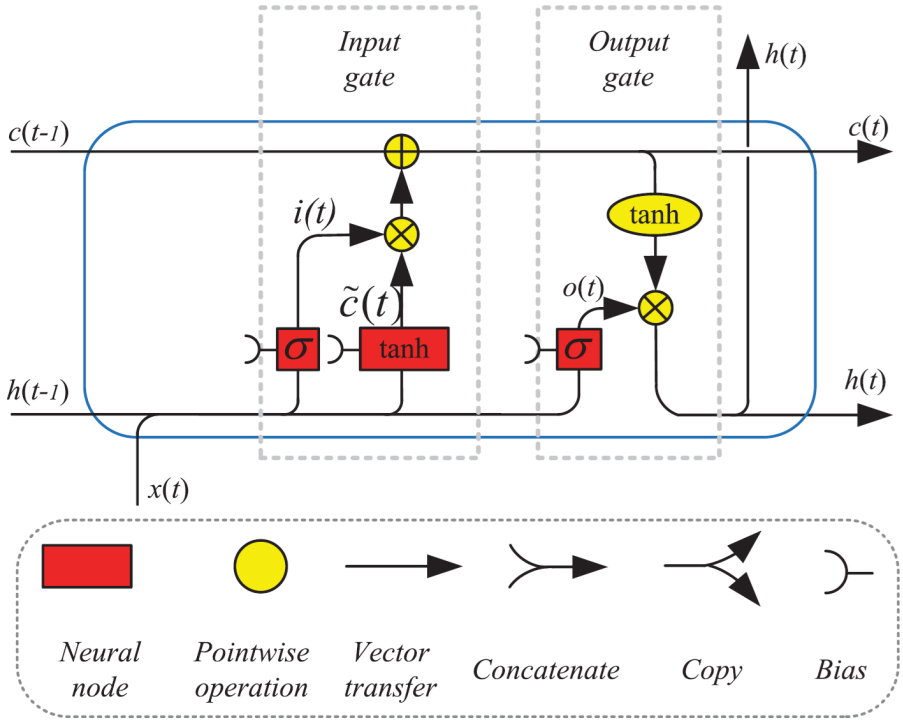


Figure 2.14: Long Short-Term Memory (LSTM) structure [30, 32].

As depicted in Figure 2.14, LSTM networks employ memory cells that contain input, output, and forget gates to regulate the flow of information. This mechanism enables the model to learn which information to remember or discard, effectively mitigating the vanishing gradient

problem that often plagues traditional recurrent neural networks (RNNs) [29].

While LSTM is effective for sequential data, it tends to face scalability issues, particularly when handling very long sequences or sequences consisting of items with complex content. To address these limitations, ConvLSTM was introduced. ConvLSTM integrates convolutional operations into the LSTM framework, allowing it to capture both spatial and temporal information simultaneously [31]. This is particularly beneficial for video data, where spatial features across frames are critical for generating coherent sequences. The use of convolutional layers enables ConvLSTM to efficiently process high-dimensional data, such as video frames, while maintaining the temporal dependencies learned by traditional LSTMs.

In video generation, several models leverage LSTM and ConvLSTM architectures to process sequential information. For instance, MoCoGAN utilizes a ConvLSTM to disentangle motion and content representations, allowing for the generation of realistic video sequences with controlled motion dynamics. Similarly, VideoVAE employs LSTM networks to model temporal dependencies in the latent space, facilitating the generation of high-quality video sequences. Another notable approach is TwoStreamVAN, which combines ConvLSTM with a two-stream architecture to effectively capture both spatial and temporal features in video data. By employing LSTM or ConvLSTM, these models can effectively model the sequential nature of video data, enabling them to generate long and coherent video sequences.

Recent advancements in LSTM and ConvLSTM architectures have significantly enhanced their ability to handle sequential data, solidifying their role as essential components in the video generation domain. Through the combination of memory retention and spatial feature extraction, these models continue to set the foundation for innovative approaches in generating realistic and coherent video content.

2.3.3 GPT for Sequence Learning

GPT have significantly impacted sequence learning, providing a robust alternative to LSTMs for capturing long-term dependencies. Unlike LSTM, which processes sequences step-by-step, the transformer-based architecture in GPT employs a self-attention mechanism, allowing the model to attend to all elements in a sequence simultaneously. This makes GPT highly effective in modelling complex temporal relationships, which is essential for generating realistic and temporally coherent video sequences over extended periods [17].

As shown in Figure 2.5 (b), GPT processes input sequences through tokenization, mapping discrete words into continuous vectors known as tokens. In the Natural Language Processing field, these tokens are passed through positional encoding to preserve the order of the sequence. When adapted for video generation, as seen in models like VideoGPT, the tokenization process is handled through a Vector Quantised-Variational AutoEncoder (VQ-VAE) [78], which learns discrete latent representations of raw video frames using 3D convolutions and axial self-attention. Instead of directly tokenizing individual video frames, VideoGPT compresses the video into downsampled discrete latent sequence, which are then modelled auto-regressively. By incorporating spatio-temporal position encodings, these latents are passed through a GPT-like architecture to generate new video frames. This approach reduces the computational cost compared to predicting each pixel by a GPT [79]

while maintaining both temporal coherence and high visual fidelity across video sequences.

Building upon this, TATS further enhances GPT-based video generation [9]. TATS addresses the challenge of generating long video sequences by modifying the padding mechanism during GPT training, using replicated initial 16 frames instead of padding with zeros. It maps pixels to discrete latent representations of video frames using a 3D (2D space and 1D time) VQ-GAN [80], and trains GPT on those representations. This hierarchical approach allows TATS to maintain both long-term temporal consistency and high-resolution frame quality over extended video durations [9]. By padding with replication, TATS efficiently models local temporal dynamics within short clips while ensuring global consistency for longer sequences, outperforming earlier models like VideoGPT in generating temporally coherent and visually appealing long videos. Compared to VideoGPT, TATS enhances the ability to generate longer video sequences, optimizing for extended temporal consistency rather than providing frames with empty content.

Recent advances in video generation leveraging GPT highlight the model's strengths in maintaining high-quality temporal consistency, particularly in long-term video prediction tasks. By refining latent spaces through auto-regressive processes, GPT-based models have achieved state-of-the-art results in producing smooth and temporally coherent video sequences [21, 81].

2.4 Conclusion

In this chapter, we have explored the foundational frameworks, techniques, and models central to the development of generative models for video generation. Beginning by exploring the core generative frameworks that underpin the models discussed in this thesis, including Generative Adversarial Networks (GAN), Variational Autoencoders (VAE), and their hybrid forms, VAE-GAN. These models offer distinct advantages in terms of generating high-quality visual content, with GAN excelling at sharp image generation and VAE providing better latent space representations. Hybrid methods aim to balance the strengths of both frameworks, achieving more robust and coherent generative results, particularly in the context of video generation.

We then transitioned to the specific challenges and advancements in video generation. Section 2.2 outlined the evolution of techniques from short-term to long-term video generation, focusing on how the field has progressed to address the issues of temporal consistency and video quality over extended sequences. Conditional methods, which leverage additional input information, have proven effective for short-term generation, while unconditional approaches offer more flexibility for longer video generation. However, generating long videos with high temporal and spatial fidelity remains a significant challenge, requiring innovative solutions.

Challenges and Future Directions. Despite significant progress in the field of video generation, there remain several key challenges. One of the primary obstacles is the generation of long, high-resolution videos that maintain realism both in content and motion in longer duration with temporal dynamics. While models have improved in producing coherent short-term sequences, long-term generation still struggles with issues such as loss of temporal consistency and degraded video quality over time. Moreover, computational demands remain high, with training times stretching over days, which limits the scalability of

these approaches. Additionally, current evaluation metrics such as Fréchet Inception Distance (FID) [39, 40, 82] and Inception Score (IS) [41, 42], while useful for assessing content quality, are not fully reliable for evaluating the motion and dynamics of generated videos.

Future research will likely focus on refining these models to improve both the quality and efficiency of video generation, while exploring new evaluation metrics. The rise of diffusion models offers promising alternatives to GANs, particularly in generating high-fidelity and temporally consistent videos, though their computational demands remain a bottleneck [54, 55]. Meanwhile, advances in controllable generation frameworks, such as CLIP [83] and ControlNet [84] provide new avenues for improving generation precision. Additionally, the integration of Transformer-based models like GPT into video generation has also shown potential in capturing long-range dependencies and producing coherent sequences over extended durations. These emerging technologies collectively hold significant potential for overcoming current limitations in video generation.

In summary, this chapter has laid the groundwork for understanding the current state of video generation and its key challenges, providing the context for my contributions. My research builds on the foundations of GAN, VAE, and Transformer models, with a focus on addressing the difficulties of long-term video generation and advancing the capabilities of generative frameworks in this evolving field.

Chapter 3

EncGAN3: Encoding GAN3 Video Generator

Recently, video processing has gained significant attention in the field of machine learning, particularly in deep learning. Video generation is a critical area in computer vision with applications ranging from entertainment to surveillance, including video synthesis for entertainment purposes, data augmentation for training other networks, movie production, video reconstruction for advertising, and crime scene reconstruction, among others. However, generating high-quality, diverse videos that maintain spatial and temporal coherence remains a challenging task. Unlike video classification, video generation involves more complexity due to the need to predict information under a high degree of uncertainty. Additionally, it poses more challenges than image generation, as it requires the estimation of spatio-temporal information rather than merely scene reconstruction. The generated videos should exhibit smoothness and coherence in motion. This chapter introduces a hybrid VAE-GAN approach for video generation, which incorporates an encoder to infer video information, thereby enhancing the generative architecture of the video GAN.

3.1 Introduction

Despite considerable progress, generating videos that maintain both spatial and temporal coherence remains a formidable challenge. Existing methods often struggle to balance these aspects, leading to videos that either lack visual fidelity or exhibit temporal inconsistencies. In this chapter, I combine the inference properties of Variational Autoencoders (VAEs) with the generation capabilities of Generative Adversarial Networks (GANs), proposing a hybrid VAE-GAN model for video generation.

The Generative Adversarial Network (GAN) [3] and the Variational Autoencoder (VAE) [4], together with diffusion networks [34], represent the main deep generative frameworks developed so far. GANs can generate images that show sharp visual results but require computationally expensive training, and sometimes their results show unexpected artifacts. Meanwhile, VAEs use inference mechanisms requiring relatively less computational cost and

offer more stable training, but tend to produce comparatively blurry results. Hybrid VAE-GAN models in image generation attempt to combine the complementary characteristics of GAN and VAE while reducing their weaknesses [12, 13, 45]. However, none of these methods can be used directly for video generation, which requires temporal synchronization between sequences of frames with moving objects and moving regions permanently changing the information from frame to frame. In this chapter, we propose to combine the complementary characteristics of VAEs and GANs and use them for video generation.

In this work, we propose EncGAN3, which represents a VAE-GAN hybrid generative network specifically designed for video generation. EncGAN3 introduces a dual-stream encoder to enhance the generation process by increasing training stability, improving generation performance, and enabling the creation of realistic videos at various resolutions, which are afterward synthesized into a coherent video stream. EncGAN3 employs an encoder to provide a representative latent space for the GAN generator, replacing the random seed used in the classical GAN model. The encoder processes content and motion through two separate streams, which are then fed into a three-stream generator to ensure spatial, temporal, and spatio-temporal consistency. A two-stream discriminator evaluates the quality of the images and videos, ensuring coherence and realism.

Many existing GAN-based models for the video generation task either consider nested generators to achieve video generation unconditionally [43, 60, 61, 62, 63], or associate additional modules that can provide prior information from ground truth as conditions to achieve video generation conditionally [44, 22]. For the former unconditional generative models, the typical nesting idea is to use a sequence generator to generate a sequence of noises, with each noise as input or part of the input to an image generator for generating sequential images as a video sequence. However, such a generation way accumulates errors twice, due to two generation step, increasing the generation difficulty. In contrast, the conditional generation using conditions from ground truth, such as images [44, 40] or class type [22], provides useful prior information. The useful information could benefit the generator instead of accumulating errors and thus, reduce the generation difficulty. But it also makes the generation rely on the condition inputs. Most conditional generative models are designed to combine the extra condition provided module with the generator, such as using skip connections between the encoder and generator [44]. Such a design makes the generator lose generation independence at testing time. Different from both generation ways, our proposed EncGAN3 addresses these limitations by avoiding dependency on conditions while still benefiting from prior information. EncGAN3 is trained with an Encoder, which provides useful prior information from ground truth to benefit the learning of the generator. With the direction of prior information, the GAN generator can learn better and more stable at training time, as shown in Table 3.5 for the improved performance and the ablation of sampling strategy for the more stable learning in Section 3.4.5. This architecture supports both conditional and unconditional video generation by optionally using the encoder at test time. The hybrid generation capability is achieved through a training objective function that considers generation from both random noise and latent codes, as detailed in Section 3.3.1. Besides, EncGAN3 trains the encoder and generator separately to enhance the generation independence of the video generator. Moreover, the conditional generation for EncGAN3 just indicates the usage of real video as input, the generation content is irrelated to the input condition. The generated videos of EncGAN3 could be visually totally different from their input real videos.

Video representation through dual-stream decomposition is initially generally employed in video-related tasks, such as video recognition [24, 26, 28], video classification [23], object detection and action recognition [85, 25, 27]. Subsequently, this representation mechanism has been adopted generally in video generation methods [43, 60, 61, 62, 5, 74]. This structure allows for more precise modelling of content and motion, resulting in higher-quality video generation. EncGAN3 implements a dual-stream architecture in both the encoder and discriminator for video information compression and classification. For the more complex task of video generation, we utilize a three-stream approach in the generator. The generator employs a primary video stream to produce video outputs and two auxiliary streams, content and motion streams, that extend from the encoder. These auxiliary streams provide crucial information to the main video generation stream by fusing at multiple scales, following the method described in [5].

Our extensive experiments on four benchmark datasets—UvA-NEMO, Weizmann, KTH, and UCF101—demonstrate that EncGAN3 achieves competitive results in terms of video FID and video IS metrics when compared to earlier models. This work contributes to the field by providing a viable model for generating high-quality videos, paving the way for future research and applications.

The contributions of this chapter are as follows:

1. Enabling a video GAN generator with an inference mechanism using a VAE-based Encoder.
2. Developing a multi-stream video generative model, EncGAN3, enabled with inference mechanisms for two different streams processing content and movement, respectively.
3. Demonstrating through quantitative and qualitative results the advantages of EncGAN3 in terms of the visual quality and diversity of generated videos.

The rest of this chapter includes the description of the EncGAN3 structure in Section 3.2. In Section 3.3.1, we present the training algorithm and loss functions used for training the video EncGAN3 model. The experimental results are provided in Section 3.4 and the conclusions of this chapter are drawn in Section 3.5.

3.2 EncGAN3 Structure

In this section, we introduce EncGAN3, a VAE-GAN hybrid video generation model that integrates an encoder-enabled Generative Adversarial Network (GAN) to enhance the quality and realism of generated videos. EncGAN3, similar to other deep learning video processing architectures, decomposes video content into separate streams of content and movement, providing a robust framework for generating high-quality video sequences. This model leverages the complementary generative capabilities for GANs and inference abilities for VAEs, allowing both conditional and unconditional video generation modes, while providing significant improvements in both the spatial and temporal consistency of the generated videos.

The EncGAN3 model architecture, shown in Figure 3.1, comprises three primary components: the Encoder, the Generator, and the Discriminator. Each component is designed to

handle specific tasks within the video generation pipeline, ensuring that both content and motion are accurately processed and synthesized. The Encoder processes input video frames to extract latent representations of content and motion. The Generator utilizes these latent representations to generate realistic video frames, ensuring consistency across both spatial and temporal dimensions. The Discriminator evaluates the realism of the generated frames, guiding the Generator through adversarial training.

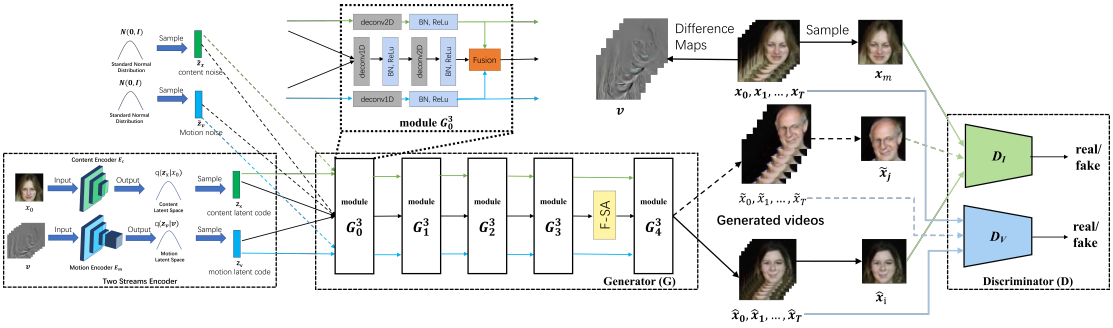


Figure 3.1: The architecture of EncGAN3: a two-stream Encoder, a three-stream Generator and a two-stream Discriminator for processing the content and movement information corresponding to the generated video.

The Encoder in EncGAN3 features a dual-stream architecture, designed to separately handle content and motion information from video frames. This separation allows for a more precise representation of video dynamics, as utilized in previous video processing methods [85, 43, 5, 22]. The content stream processes the first frame of the input video to generate a latent space (mean and variance) representing static content features. It consists of convolutional layers followed by fully connected layers to compress the spatial information. The motion stream processes the difference maps between consecutive frames to generate a latent space representing motion features. The difference maps are calculated by subtracting consecutive frames and thus, the number of difference maps is equal to the input video length minus one. Each difference map is first processed by a network structure similar to the content stream, with convolutional layers for feature extraction followed by fully connected layers for compression. The output features of each difference map would then be further compressed to produce a latent space through a block consisting of several fully connected layers.

The Generator in EncGAN3 generates video frames by combining content and motion latent codes. The generation operates through three parallel streams: content, motion and video. The content stream reconstructs the static content features from the content latent code. The motion stream reconstructs the dynamic motion features from the motion latent code. The video stream integrates outputs of the content and motion streams at the end of each G^3 module, ensuring spatial and temporal consistency across the generated frames at different scales. This integration involves adding temporal features and concatenating spatial features after size matching. Additionally, the video stream incorporates a factorized self-attention (F-SA) module to enhance the coherence of the generated video. The F-SA module consists of a temporal-wise self-attention followed by a spatial-wise self-attention, enabling the Generator to utilize cues from all spatio-temporal features while modelling relationships between distinct regions.

The Discriminator in EncGAN3 comprises two parallel streams, each focusing on different

aspects of the generated video. The Image Stream Discriminator (D_I) assesses the realism of individual frames by sampling one frame from each generated video and comparing it to frames sampled from real videos. The Video Stream Discriminator (D_V) evaluates the temporal coherence of the entire video sequence by analysing the transitions between frames. Each stream employs convolutional layers to extract relevant features and outputs a probability score indicating the authenticity of the input.

In the following, we describe the video data processing by the EncGAN3. EncGAN3 has two processing streams representing content and motion. During the training phase, EncGAN3 processes input videos through the Encoder, generating latent codes that represent both content and motion. These latent codes, together with random noise, are then fed into the Generator to produce realistic video frames. The Discriminator evaluates these generated videos, providing feedback to the Generator to improve its performance over time.

In the inference phase, EncGAN3 supports both unconditional and conditional video generation modes. For unconditional video generation, random noise is used as the input for the Generator. For conditional video generation, the Encoder first processes a video to create a latent space. From this latent space, latent codes are sampled. Each of these latent codes can then be used to generate an entire video through the Generator. Different from the reconstruction, due to the adversarial loss term, the videos generated from these latent codes are entirely different from the input videos used to create the latent space. To improve the performance, the ability of generating from random noises is further removed when only need to generate from the latent space, such as LEncGAN, REncGAN and R3. It is maintained in AR2, which generates from the random noise.

Unlike the auto-regressive generation models, the Generator in EncGAN3 produces the entire video at once. This approach leverages the comprehensive latent codes to ensure spatial and temporal coherence in the generated videos, allowing for efficient and high-quality video generation across content and motion dimensions.

3.3 EncGAN3 Running

In the following, we present how we train the EncGAN3 model. While the training has similar characteristics to VAE-GAN hybrid architectures used for processing images [13, 45], it has also specific video generation characteristics. Such characteristics refer mainly to the ability to represent movement in video and the EncGAN3 has a dual generation pipeline corresponding to the content (scene representation) and movement. Each of these is generated individually while at the end they are combined to create the video. Each of the three modules in EncGAN3, which have been described in Section 3.2, visualized in Figure 3.1, has its own objective function and is trained and optimized individually in the following order: Discriminator, Encoder and Generator. We remove the noise-based generation loss term from the objective function for LEncGAN, REncGAN and R3 models, while this is kept for EncGAN3 and AR2, similarly with GAN models.

3.3.1 Training Objective

First, the loss function of the two-stream Encoder for content and motion L_{Enc} is defined as:

$$L_{Enc} = \sum_{i=1}^{N} \sum_{j=0}^{T_i} \|\mathbf{x}_{ij} - \widehat{\mathbf{x}}_{ij}\| + D_{KL}(q_{\theta_\mathbf{x}}(\mathbf{z}_\mathbf{x}|\mathbf{x})||p(\mathbf{z}_\mathbf{x})) + D_{KL}(q_{\theta_\mathbf{v}}(\mathbf{z}_\mathbf{v}|\mathbf{v})||p(\mathbf{z}_\mathbf{v})) \tag{3.1}$$

where the first term represents the reconstructions of the video frames and the other two terms represent the Kullback-Leibler divergences (KLD) ensuring that the latent spaces of the images z_x and video z_v are consistent with their priors.

The $\{\mathbf{x}_{ij}\}$ and $\{\widehat{\mathbf{x}}_{ij}\}$ are the j-th frame from the real i-th video and its corresponding reconstruction, respectively. In this context, $j = 0, \ldots, T_i$, where T_i indicates the frame index of video i (starting from 0). Thus, the length of video i is $T_i + 1$. Additionally, $i = 1, \ldots, N$, where N represents the number of videos used for training. The reconstructions $\{\widehat{\mathbf{x}}_{ij}\}_{j=0}^{T_i}$ are made as close as possible to their corresponding original video sequences $\{\mathbf{x}_{ij}\}_{j=0}^{T_i}$ by the mean absolute error, called the reconstruction term. The reconstructions $\{\widehat{\mathbf{x}}_{ij}\}_{j=0}^{T_i}$ are generated by the inferred latent space estimated by the encoder:

$$\widehat{\mathbf{x}}_{i,0:T_i} = \mathrm{G}(\mathrm{Enc}(\mathbf{v}_{i,1:T_i}, \mathbf{x}_{i0})) \tag{3.2}$$

where $\mathbf{v}_{1:T_i} = \{\mathbf{v}_1, \ldots, \mathbf{v}_{T_i}\}$ represent difference maps that are calculated by subtracting adjacent frames, as:

$$\mathbf{v}_{ij} = \mathbf{x}_{i,j-1} \ominus \mathbf{x}_{ij},\ j = 1, \ldots, T_i \tag{3.3}$$

where $T_i + 1$ video frames $\{\mathbf{x}_{ij}\}_{j=0}^{T_i}$ could produce T_i difference maps $\{\mathbf{v}_{ij}\}_{j=1}^{T_i}$. Therefore, during the training, the input video is split into one video frame $\mathbf{x}_{i0}$, corresponding to its first frame, and T_i difference maps $\mathbf{v}_{ij}$, $j = 1, \ldots, T_i$, where $T_i = 15$ for a total of 16 frames.

The Kullback–Leibler divergence D_{KL} components in Equation (3.1) ensure that the probabilities of the latent variable associated with the content $\mathbf{z}_\mathbf{x}$ and with the motion $\mathbf{z}_\mathbf{v}$ are generated by the two encoders of parameters $\theta_\mathbf{x}$ and $\theta_\mathbf{v}$, respectively, and are consistent with their Gaussian priors. Minimizing L_{Enc} leads to better encoding of both content and motion by the two encoders, implementing the variational distributions $q_{\theta_\mathbf{x}}(\mathbf{z}_\mathbf{x}|\mathbf{x})$ and $q_{\theta_\mathbf{v}}(\mathbf{z}_\mathbf{v}|\mathbf{v})$. Both $p(\mathbf{z}_\mathbf{x})$ and $p(\mathbf{z}_\mathbf{v})$ are enforced as normal distributions in order to force the encoders to implement the variational distributions as close to the standard normal distribution. Equation (3.2) characterizes the relationship between the streams of content and motion, processed by the two Encoders. Considering the two-stream process in Encoder, we perform an ablation study in section 3.4.5 where replacing the reconstruction term from Equation (3.1) and (3.4) with a content reconstruction term of first frame $\sum_{i=1}^{N} \|\mathbf{x}_{i0} - \widehat{\mathbf{x}}_{i0}\|$ and a motion reconstruction term of all difference maps $\sum_{i=1}^{N} \sum_{j=1}^{T_i} \|\mathbf{v}_{ij} - \widehat{\mathbf{v}}_{ij}\|$, as in Equation (3.9) and (3.10).

The second Generator objective L_G is the key part linking the VAE and GAN structures, representing actually the Decoder to Encoder and Generator to Discriminator. This defines the Generator as the quintessential integrator block for the VAE and GAN structures in the EncGAN3 model. Hence, the loss function L_G contains both VAE and GAN components,

representing the most complex of the three objectives. L_G is given by:

$$\begin{aligned} L_G =& \mathbb{E}_{\widehat{\mathbf{x}}_n \sim G(\mathbf{z_x}, \mathbf{z_v})} \log[D(\widehat{\mathbf{x}}_n)] + \mathbb{E}_{\tilde{\mathbf{x}}_n \sim G(\tilde{\mathbf{z}}_\mathbf{x}, \tilde{\mathbf{z}}_\mathbf{v})} \log[D(\tilde{\mathbf{x}}_n)] \\ &+ \mathbb{E}_{\mathbf{z_x} \sim p(\mathbf{z_x}), \mathbf{z_v} \sim p(\mathbf{z_v})} \log[D(G(\mathbf{z_x}, \mathbf{z_v}))] \\ &+ \mathbb{E}_{\tilde{\mathbf{z}}_\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}), \tilde{\mathbf{z}}_\mathbf{v} \sim \mathcal{N}(0, \mathbf{I})} \log[D(G(\tilde{\mathbf{z}}_\mathbf{x}, \tilde{\mathbf{z}}_\mathbf{v}))] - \sum_{i=1}^{N} \sum_{j=0}^{T_i} \|\mathbf{x}_{ij} - \widehat{\mathbf{x}}_{ij}\| \end{aligned} \tag{3.4}$$

where the last term in the right side of Equation (3.4) is the same as the first term in Equation (3.1), representing the reconstruction error between the real data $\mathbf{x}_{ij}$ and its corresponding reconstruction $\hat{\mathbf{x}}_{ij}$. The other L_G components belong to the GAN objective, based on Binary Cross Entropy loss for the binary classification task of the Discriminator. These GAN objective terms consider the latent space of the content $\mathbf{z_x} \sim p(\mathbf{z_x})$ and of movement $\mathbf{z_v} \sim p(\mathbf{z_v})$ modelled by corresponding encoders, and considering the random noises of $\tilde{\mathbf{z}}_\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$, for the content and $\tilde{\mathbf{z}}_\mathbf{v} \sim \mathcal{N}(0, \mathbf{I})$, for the movement. The reconstruction of the video frames $\widehat{\mathbf{x}}$ is made considering the latent codes created by the encoder, $\mathbf{z_x}$ and $\mathbf{z_v}$ corresponding to the image x and the video v. The 2nd and 4th components use the normal GAN generator objective terms, where the codes $\tilde{\mathbf{z}}_\mathbf{x}$ and $\tilde{\mathbf{z}}_\mathbf{v}$ are sampled from the normal distributions for content and motion, respectively. L_G represents an optimization procedure guiding the model not only to create realistic videos but also to enforce the standard distribution for the encoding latent space used for generating data.

The third objective function of the two-stream Discriminator is an adversarial loss, similar to those used in [61, 22]. The two streams, representing the content and the movement information, each with its own Discriminator, are trained in parallel. The objective function of the image-stream Discriminator L_{D_I} is :

$$\begin{aligned} L_{D_I} =& \mathbb{E}_{\mathbf{x}_n \sim p(\mathbf{x})} \log[D(\mathbf{x}_n)] + \mathbb{E}_{\widehat{\mathbf{x}}_n \sim G(\mathbf{z_x}, \mathbf{z_v})} \log[1 - D(\widehat{\mathbf{x}}_n)] \\ &+ \mathbb{E}_{\tilde{\mathbf{x}}_n \sim G(\tilde{\mathbf{z}}_\mathbf{x}, \tilde{\mathbf{z}}_\mathbf{v})} \log[1 - D(\tilde{\mathbf{x}}_n)] \end{aligned} \tag{3.5}$$

where $\mathbf{x}_n \sim p(\mathbf{x})$ is the real image content, $\widehat{\mathbf{x}}_n$ is generated from the latent codes presenting the image content, and $\tilde{\mathbf{x}}_n$ is the image generated using the standard Gaussian random noise. Images $\widehat{\mathbf{x}}_n$ and $\tilde{\mathbf{x}}_n$ are randomly sampled from their video clips, where n indicates a sampled value from $\{0, \ldots, T\}$.

The objective function of video-stream Discriminator L_{D_V} is considered in a similar way to L_{D_I}, as :

$$\begin{aligned} L_{D_V} =& \mathbb{E}_{\mathbf{x}_{0:T} \sim p(\mathbf{x}_{0:T})} \log[D(\mathbf{x}_{0:T})] + \mathbb{E}_{\widehat{\mathbf{x}}_{0:T} \sim p(\widehat{\mathbf{x}}_{0:T})} \log[1 - D(\widehat{\mathbf{x}}_{0:T})] \\ &+ \mathbb{E}_{\tilde{\mathbf{z}}_\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}), \tilde{\mathbf{z}}_\mathbf{v} \sim \mathcal{N}(0, \mathbf{I})} \log[1 - D(G(\tilde{\mathbf{z}}_\mathbf{x}, \tilde{\mathbf{z}}_\mathbf{v})] \end{aligned} \tag{3.6}$$

where $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$ and $\widehat{\mathbf{x}}_{0:T} = \{\widehat{\mathbf{x}}_0, \ldots, \widehat{\mathbf{x}}_T\}$ represent the real videos and their reconstructions, while $p(\mathbf{x}_{0:T})$ and $p(\widehat{\mathbf{x}}_{0:T})$ are their probabilities. The second term represents the results generated by the Generator together with the latent codes provided by the Encoder, while the third term represents the results by the Generator with random noise inputs.

3.3.2 Training and Inference

During the training, first the Discriminator is updated by optimizing L_{D_I} and L_{D_V} using (3.5) and (3.6), then the Encoder using L_{Enc} as in (3.1), and eventually the Generator L_G according to (3.4). The training procedure is shown in Algorithm 1.

Algorithm 1 EncGAN3 Training Procedure

Required: Content Encoder (CoEnc), the part consist of convolutional layers in Motion Encoder (Moenc), the part consist of fully-connected layers in Motion Encoder (Mofc), Clip Generator (G), Image Discriminator (ID), Video Discriminator (VD)
Input: A short video $\mathbf{x}_{0:T}$
Output: Discrimination results of True or False T/F

Content and Motion Encoding:
$\mathbf{z}_{co} \leftarrow \text{CoEnc}(\mathbf{x}_0)$
for $t = 1 \rightarrow T$ **do**
    $\mathbf{v}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{x}_t$
    $emb_t \leftarrow \text{Moenc}(\mathbf{v}_t)$
end for
$\mathbf{z}_{mo} \leftarrow \text{Mofc}(emb_{1:T})$
Latent Space Sampling:
$\mathbf{z}_{\mathbf{x}} \leftarrow \text{sample}(\mathbf{z}_{co})$
$\mathbf{z}_{\mathbf{v}} \leftarrow \text{sample}(\mathbf{z}_{mo})$
Clip Generation:
$\widehat{\mathbf{x}}_{0:T-1} \leftarrow \text{G}(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}})$
Discrimination:
$T/F \leftarrow \text{VD}(\widehat{\mathbf{x}}_{0:T-1})$
$idx \leftarrow \text{random}(0 : T-1)$
$T/F \leftarrow \text{ID}(\widehat{\mathbf{x}}_{idx})$

At inference time, there are two ways for EncGAN3 to generate videos. One generates from random noises, while the other generates from sampled latent codes. The inference procedure is shown in Algorithm 2. Since EncGAN3 does not consider to constrain the content of output by the inputs, clips generated from latent codes are vastly different from their input clips.

3.4 Experiments

In this section, we present a series of experiments conducted to evaluate the performance of the EncGAN3 model in video generation tasks. These experiments demonstrate the effectiveness of EncGAN3 in producing high-quality, realistic videos. The results provided by EncGAN3 are better than those of the baselines used for comparison. This section is organized as follows: we begin with a detailed description of the datasets used, followed by the experimental setup. We then present both quantitative (including the description of used evaluation metrics) and qualitative results, provide an in-depth analysis, and conclude

Algorithm 2 EncGAN3 Inference Procedure

Required: two-stream Encoder (Enc), Clip Generator (G)
Input: A short video $\mathbf{x}_{0:T}$ or two random noises.
Output: A generated short video $\widehat{\mathbf{x}}_{0:T}$ or $\tilde{\mathbf{x}}_{0:T}$

Phase 1: Generation by latent codes
$(\mathbf{z_x}, \mathbf{z_v}) \leftarrow \text{sample}(\text{Enc}(\mathbf{x}_{0:T}))$
$\widehat{\mathbf{x}}_{0:T} \leftarrow \text{G}(\mathbf{z_x}, \mathbf{z_v})$

Phase 2: Generation by random noise
$(\tilde{\mathbf{z}}_\mathbf{x}, \tilde{\mathbf{z}}_\mathbf{v}) \leftarrow \text{sample}(\mathcal{N}(0, \mathbf{I}))$
$\tilde{\mathbf{x}}_{0:T} \leftarrow \text{G}(\tilde{\mathbf{z}}_\mathbf{x}, \tilde{\mathbf{z}}_\mathbf{v})$

with ablation studies to understand the contributions of different components of the model.

3.4.1 Datasets

We utilized four commonly used datasets for our experiments: UvA-NEMO [86], Weizmann [87], KTH [88] and UCF101 [89]. Each dataset contains videos of varying complexity and motion patterns, providing a comprehensive benchmark for evaluating our model. The UvA-NEMO dataset (UvA) focuses on spontaneous facial expressions with detailed annotations. We consider the preprocessed version from [5], consisting of 1240 videos at a resolution of 128x128 pixels, each with over 100 frames. The Weizmann dataset comprises 90 video sequences, featuring nine subjects performing ten different actions. Videos have a duration of 1 to 3 seconds with a frame rate of 25 frames per second (FPS), with a resolution of 180×144 (weight by height) pixels. The KTH dataset contains 599 video sequences of 25 subjects performing six types of actions (walk, jog, run, box, hand-wave, and hand-clap) in various scenarios (indoor, outdoor, outdoor with scale variation, outdoor with different clothing). Most videos are 10 60 seconds long at 25 FPS, with a resolution of 160×120 pixels. The UCF101 dataset includes 13320 realistic videos covering 101 human action categories, collected from YouTube. Most videos last 5 10 seconds at 25 FPS, with a resolution of 320×240 pixels. The characteristics of the datasets used in the experiments are provided in Table 3.1. For the generation task, we do not split data into training, validation and testing sets as in classification tasks unless the dataset is already split into such sections. We use the entire dataset as the training set and use the entire dataset to calculate the metrics during testing.

Data Preprocessing. The videos used in the training are firstly sampled both spatially and temporally in order to reduce the data size used for training and implicitly reduce the training time. Temporal processing involves sampling a certain number of frames from each video, while spatial processing includes resizing frames to a uniform resolution and normalizing pixel values. Firstly, we do temporal processing by randomly sampling 16 frames from each video. The video generation task generally samples 16 frames from each video for the training, because the GPU memory requirement increases exponentially with the length of the training video. The first frame was selected randomly, and consider a fixed time step for the next frames, varying randomly from 1 to 3 or 1 to 6, depending on the dataset's

Dataset	Video Specifications			Categories	
	Videos	Resolution	Frame Range†	Subjects	Actions
UvA-NEMO [86]	1,240	128×128	(100-400)	564	1
Weizmann [87]	93	180×144	(20-75)	9	10
KTH [88]	599	160×120	(200-1,025)	25	6
UCF101 [89]	13,320	320×240	(26-2,060)	2,525	101

Table 3.1: Datasets used in the short video generation task. †Frame ranges are estimated from duration and frame rate, with potential variations due to data transfer losses.

general video lengths. This sampling strategy is called step sampling, to identify from the other video sampling strategies adopted in order to extract the 16 frames from each video used for training. We do an ablation study of various video sampling strategies in Section 3.4.5 and decide to use step sampling. Next, we do spatial processing by resizing sampled video frames to a uniform resolution. We scale the frames such that we preserve their aspect ratios, ensuring that the minimum height or width is 64 pixels. Then, those scaled frames are cropped to 64×64 pixels. For the Weizmann dataset, to balance frame sharpness and object completeness, we cropped 20 pixels from the edges before scaling. We consider centered cropping to ensure that the entire moving object was captured. Finally, we normalize pixel values by first scaling the pixel values to the range of 0 to 1. Then, to accelerate training convergence, we normalized the scaled pixel values using a mean and variance of 0.5, mapping the pixel values to the range of -1 to 1.

3.4.2 Implementation

The EncGAN3 model consists of an Encoder, a Generator, and a Discriminator. The Encoder is built with multiple convolutional (conv) layers to extract features representing content and motion from input videos and ends with several fully connected (fc) layers to produce latent spaces [57]. Compared to the content latent space, the motion latent space is made by more fully connected layers to compress features from a sequence of difference maps. In detail, the content encoder consists of six two-dimensional (2D) convolutional (conv2d) layers and one fully connected (fc) layer. Each conv2d or fc layer is followed by a batch normalization (bn) layer and a Rectified Linear Unit (ReLU) activation function [90, 91]. The bn layer is to speed up training and improve the generalization performance of the model. We name the structure composed of a conv2d layer followed by a bn layer and then, a ReLU activation function as a conv2d layer set, similar to a fc layer set. Meanwhile, the motion encoder first processes each difference frame map v_{ij} through the same sub-network to produce an embedding. Similar to the content encoder, the sub-network of the motion encoder contains six conv2d layers and one fc layer. Different from the content encoder, a 2D max pooling (maxpool2d) layer is applied after every two conv2d layer sets. So, the sub-network also contains three maxpool2d layers. The embeddings from every two frame difference maps are compressed through one fc layer. The last embedding is not compressed if there is an odd number of embeddings. After the compression, all embeddings are concatenated and further compressed through two fc layer sets. The output of the content encoder will pass through one fc layer to produce the mean vector and another fc layer to produce the variance vector.

The mean and variance vectors represent the content latent space. The output of the motion encoder passes through two fc layers to produce the mean vector and another two fc layers to produce the variance vector. The mean and variance vectors present the motion latent space.

The Generator consists of transposed convolutional (deconv) layers to produce realistic video frames from the latent codes and random noise (G^3), with a small block doing the attention operation on the outputs of convolutional layers, named as F-SA module. The Generator contains four G^3 blocks and one F-SA module. Each G^3 block contains two conv2d and two 1D convolutional (conv1d) layer sets. The layer set in the GAN generator and below discriminator uses the spectral normalization layer instead of the batch normalization layer, to better stabilize the training of adversarial object [92]. Those layer sets within the G^3 block are organized as shown in Figure 3.1. The F-SA module is used on the video stream output of the fourth G^3 block. The module contains a convolutional temporal-wise self-attention (TSA) followed by a convolutional spatial-wise self-attention (SSA). TSA contains three 3D convolutional (conv3d) layers. SSA also contains three conv3d layers [5]. The Discriminator uses a series of convolutional layers to evaluate the generated videos. In detail, the Video Discriminator contains ten conv3d layer sets and the Image Discriminator contains five conv2d layer sets. Different from the Encoder and Generator, the layer set of the Discriminator uses Leaky ReLU instead of the ReLU as the activation function. The Leaky ReLU function allows negative values to pass through by multiplying them with a negative slope instead of setting them as zero as in the ReLU function. Hence, it avoids the problem of gradient vanishing, helps the discriminator better utilize the information in the negative area, and improves the expressiveness of the model [93].

EncGAN3 is able to generate higher-resolution videos by adding extra convolutional and transposed convolutional layers. For example, the Generator shown in Figure 3.1 consists of five G^3 blocks to generate videos at a resolution of 64×64 pixels. By adding one more G^3 block to the end of the Generator, and incorporating appropriate extra convolutional layers in both the Encoder and Discriminator, the model can generate higher resolution (128×128 pixels) videos, as shown in Figure 3.4 and 3.5. Every time the resolution doubles, such as from 64×64 pixels to 128×128 pixels, the EncGAN3 model should add more layers. In detail, the Encoder should add two sets of conv2d layers and one set of fc layers to both the content and motion stream encoders. The Generator should add an additional G^3 block. The Discriminator should add one more set of conv layers to both the Image Stream Discriminator and the Video Stream Discriminator.

The EncGAN3 model is implemented in PyTorch and uses the ADAM optimizer [94] with the exponential decay rate of first-order and second-order moment estimation of β_1=0.5 and β_2=0.999, and consider a training rate of $2e^{-4}$ for all modules: Discriminator, Encoder and Generator. For EncGAN3 we consider the same hyper-parameter initialization as for G^{3}AN.

For the training time cost, we provide a rough time cost account when training on the UvA dataset and using a single V100 GPU with 32 GiB GPU memory (actually 31.75 GiB memory total capacity in practice, if considering the other system storages). With a proper batch size (here is 10 for EncGAN3 and 25 for G^{3}AN) to make full usage of the GPU memory, EncGAN3 trains the beginning 100 epochs for about two and a half hours, and the baseline model G^{3}AN trains the beginning 100 epochs for two hours. Though EncGAN3 takes more time to train the same epochs due to the extra encoder module, EncGAN3 needs to be

trained for fewer epochs than G^3AN to reach its best performance due to the benefit of the extra encoder module. In detail, EncGAN3 performs well after training 3000 epochs but G^3AN needs to train for 5000 epochs. EncGAN3 trains 3000 epochs for about three days with six and a half hours (3d and 6.5h) to four and a half days (4.5d), and 5000 epochs for five and a half to six and a half days (5.5 to 6.5d). The G^3AN trains 5000 epochs for about four days and nineteen hours (4d and 19h). So, considering the total training time to reach the best performance, EncGAN3 takes significantly less training time than G^3AN, about one day less.

We train the EncGAN3 model, according to the methodology from Section 3.3. First, we optimize loss functions L_{D_I} from Equation (3.5) and L_{D_V} from Equation (3.6), for the Discriminator. Then we optimize L_{Enc} from Equation (3.1) for the Encoder. Thirdly, we run the model again with the optimized Discriminator and Encoder modules and optimize the loss L_G from Equation (3.4) for the Generator. The training proceeds with a new batch of data for each iteration, optimizing the Encoder, Generator, and Discriminator, until the validation results indicate stable results.

3.4.3 Quantitative Evaluation

Evaluation Metrics. To quantitatively assess the performance of the EncGAN3 model, we employed the video-level extensions of two commonly used metrics in image generation: Fréchet Inception Distance (FID) [39] and Inception Score (IS) [41], known as video FID and video IS [62, 5, 22, 9, 36]. These metrics were adapted for the video generation by modifying the Inception Network from 2D to 3D pre-trained classifiers [95] to serve as the feature extractor. For the video FID, similar to [5, 40], we utilized a single pre-trained 3D CNN as the Inception network across all datasets. For video IS, following the settings in [42, 22], we employed the same network structure but used different parameters that were pre-trained individually on each dataset. The class amount of the network output is adjusted based on the number of classes in each dataset, such as action classes or actor classes. Both metrics are computed as distance measures between distributions of real and generated videos. To simplify, all results below that described as FID or IS indicate video FID or video IS.

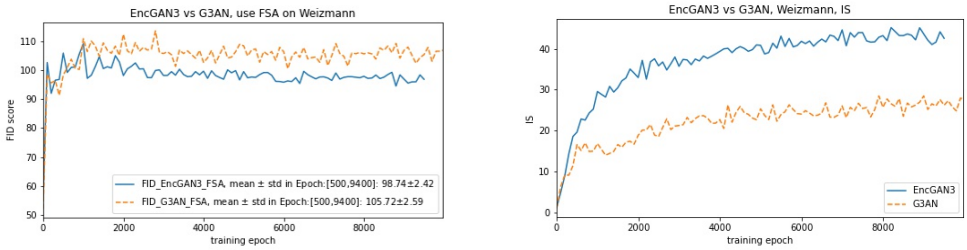
The video FID quantifies the similarity between the distributions of generated and real videos by computing the distance between their feature distributions, obtained from a pre-trained video recognition model. A lower video FID score indicates better visual quality and spatio-temporal consistency in the generated videos. The FID score is calculated as follows:

$$\text{FID} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|^2 + \text{Tr}(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2}) \tag{3.7}$$

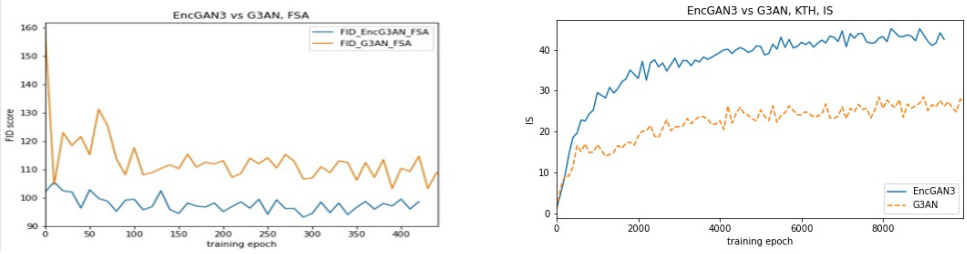
where $\boldsymbol{\mu}_r$ and $\boldsymbol{\mu}_g$ are the means, and $\boldsymbol{\Sigma}_r$ and $\boldsymbol{\Sigma}_g$ are the covariances of the feature representations of the real and generated videos, respectively.

The video IS evaluates the quality and diversity of the generated videos based on the entropy of their predicted class distributions. Higher video IS values reflect higher quality and diversity. The Inception Score is calculated as:

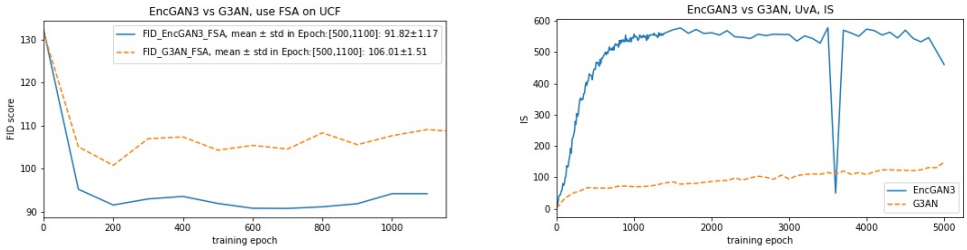
$$\text{IS} = \exp\left(\mathbb{E}_x\left[D_{KL}(p(y|x)\|p(y))\right]\right) \tag{3.8}$$



(a) Weizmann



(b) KTH



(c) UCF101, video FID (d) UvA, video IS

Figure 3.2: Video FID scores (Left column) and video IS scores (Right column) of EncGAN3 (blue line) and G^{3}AN (yellow line), calculated every 100 epochs for Weizmann (a), KTH (b), and UvA (d), and every 10 epochs for UCF101 (c). Lower FID values indicate better visual quality and spatial-temporal consistency, while higher IS values suggest better visual quality and diversity. The outlier for EncGAN3 on UvA in panel (d) may result from an unexpected fluctuation during resumed training.

where $p(y|x)$ is the conditional label distribution given the generated video x, simplified from $\{\mathbf{x}_{ij}\}_{j=0}^{T_i}$ in Section 3.3.1, and $p(y)$ is the marginal label distribution over all generated videos. Besides, the two components in video IS, Intra-Entropy H(y|x) [42] and Inter-Entropy H(y) [42], are used to separately measure the visual quality and diversity of generated videos. A higher H(y) indicates better diversity, while a lower H(y|x) means better visual quality.

Additionally, due to the absence of a universally accepted Inception Network for calculating the video IS, we pre-train the network individually on each dataset. This pre-training can be based on either the action classes or the actor classes within the dataset. The UvA dataset for facial expressions contains only one action class, smiling, but several actor classes. Therefore, we compute the video IS based on the Inception Network pre-trained on the actor classes of the UvA dataset. The human action datasets Weizmann, KTH, and UCF101 are primarily used for action recognition and contain a rich variety of action classes. Thus, we compute the video IS based on their action classes. As these three datasets also contain multiple actor

classes, we further provide the video IS results for the Weizmann and KTH datasets based on the network pre-trained on their actor classes in the ablation study, as shown in Table 3.4.

Evaluation Results. In Table 3.2, we compare the video FID for EncGAN3 with G^{3}AN [5], ImaGINator [44], VGAN [43], TGAN [60] and MoCoGAN [61]. EncGAN3 consistently has the lowest FID scores on all these datasets, indicating that videos generated by EncGAN3 perform better in both visual quality and spatio-temporal consistency. Table 3.3 presents IS and its corresponding Inter-Entropy and Intra-Entropy. EncGAN3 has the best results for those metrics in both Weizmann and KTH datasets, suggesting that the video generated by EncGAN3 performs best in visual quality and diversity in the generation of Human action videos. Note that all Tables in this thesis that shows FID and IS indicates the metrics of video FID and video IS, respectively, for simplification.

	UvA	Weizmann	KTH	UCF101
	FID↓	FID↓	FID↓	FID↓
VGAN*[43] (NeurIPS 2016)	235.01	158.04	-	115.06
TGAN*[60] (ICCV 2017)	216.41	99.85	-	110.58
MoCoGAN*[61] (CVPR 2018)	197.32	92.18	-	104.14
ImaGINator*[44] (WACV 2020)	-	99.80	-	-
G^{3}AN [5] (CVPR 2020)	91.77(119.22)	98.27(86.01)	111.99	108.36(91.21)
EncGAN3 (ours) [1] (ICIP 2022)	**86.21**	**78.93**	**66.62**	**91.18**

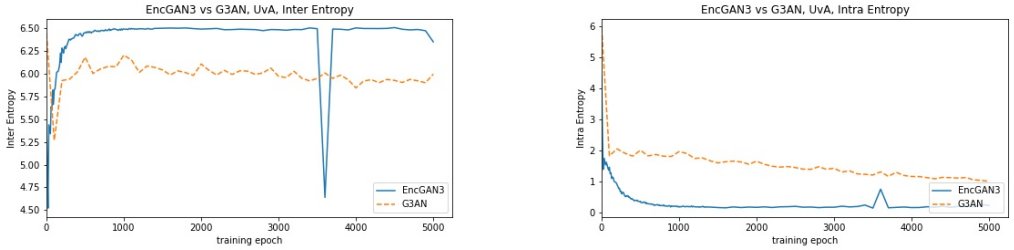
Table 3.2: Results for video FID, where * indicates that the results are referred from [5, 44] and ↓ indicates that lower value is better. We retrain G^{3}AN and provide the results from [5] in parentheses for a fair comparison.

	IS↑	Inter-Entropy ↑	Intra-Entropy ↓	Dataset (classes type)
G^{3}AN [5] (CVPR 2020)	85.44	6.041	1.593	UvA (actor)
	25.54	3.924	0.684	Weizmann (action)
	24.19	4.538	1.352	KTH (action)
	30.01	6.903	3.501	UCF101 (action)
EncGAN3 (ours) [1] (ICIP 2022)	**571.29**	6.499	0.151	UvA (actor)
	42.60	3.959	0.207	Weizmann (action)
	50.48	4.812	0.891	KTH (action)
	33.87	6.699	3.177	UCF101 (action)

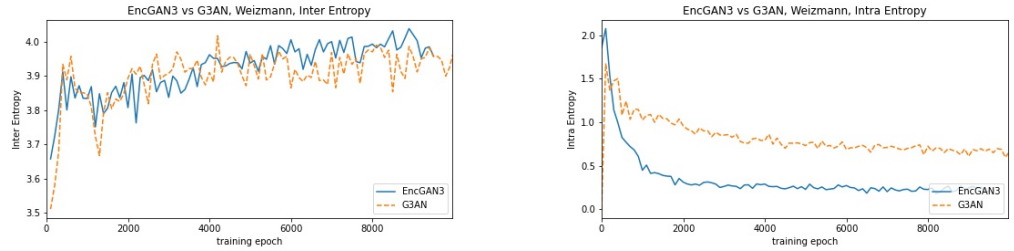
Table 3.3: Results for video IS and its components, where ↑ means that higher value is better. IS↑ means higher IS representing better visual quality and diversity. The inter-entropy H(y) measures the diversity among generated videos. A higher H(y) indicates more diversity. The intra-entropy H(y|x) measures the visual quality and the lower means better.

In addition, the metrics video FID and video IS are sensitive to the generated videos. Their scores change when computed on different generated video sets, even if those sets are produced by the same model. For a more robust measurement, we compute FID and IS after training the model for a specified number of epochs. The variation of FID during training is shown in Figure 3.2, which combines the FID scores for the Weizmann, KTH, and UCF101 datasets on the left column, and the IS scores for the UvA, Weizmann, and KTH datasets

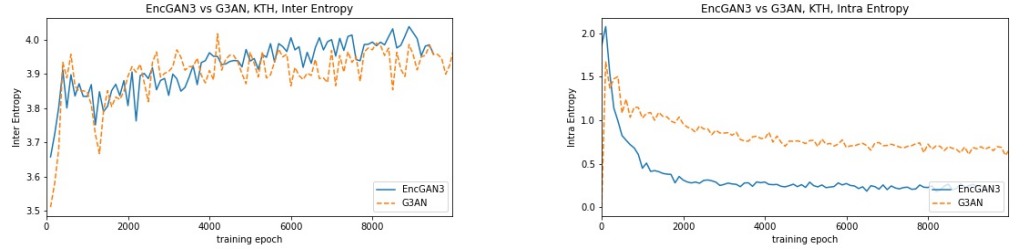
on the right column. These results indicate that EncGAN3 performs better than G^{3}AN, with lower FID scores and a more stable and smoother convergence throughout the training. Notably, the initial FID scores on the UvA dataset are lower than those in later epochs; however, the quality of the generated videos is poorer during the initial training phases compared to the later epochs, when the model better learns from the data and consequently generates higher-quality videos.



(a) UvA



(b) Weizmann



(c) KTH

Figure 3.3: IS components of Inter-entropy H(y) (Left column) and intra-entropy H(y|x) (Right column) for EncGAN3 (blue line) and G^{3}AN (yellow line), calculated every 100 epochs for Weizmann, KTH, and UvA datasets (from top to bottom panels). Higher inter-entropy H(y) values (left column) indicate better diversity, while lower intra-entropy H(y|x) values (right column) suggest better visual quality. The outlier for EncGAN3 on UvA in the top panel of both columns may result from an unexpected fluctuation when resuming training.

In Figure 3.3, we present the results of IS components: Inter-Entropy and Intra-Entropy. The results for EncGAN3 are shown in solid blue lines, while the results for G^{3}AN are indicated by dashed yellow lines. The left column displays the Inter-entropy values for the UvA, Weizmann, and KTH datasets, while the right column shows the results of Intra-Entropy. All panels in Figure 3.3 demonstrate substantially better results for EncGAN3. In the middle and bottom panels of left column in Figure 3.3, the inter-entropy H(y) of EncGAN3 gradually goes better than G^{3}AN, which implies that EncGAN3 gradually provides more diverse video

results than G^3AN during the training. The outlier of EncGAN3 on UvA in Figures 3.2 (d) and 3.3 (a) may come from an unexpected fluctuation when resuming the training.

3.4.4 Qualitative Evaluation

For the qualitative evaluation, we provide results of videos generated by EncGAN3 trained on KTH, Weizmann and UvA-NEMO datasets with the resolution 128×128. The baseline model G^3AN is designed to generate videos at 64×64 pixel resolution. We provide videos generated by EncGAN3 at 64×64 pixel resolution for a fair comparison.

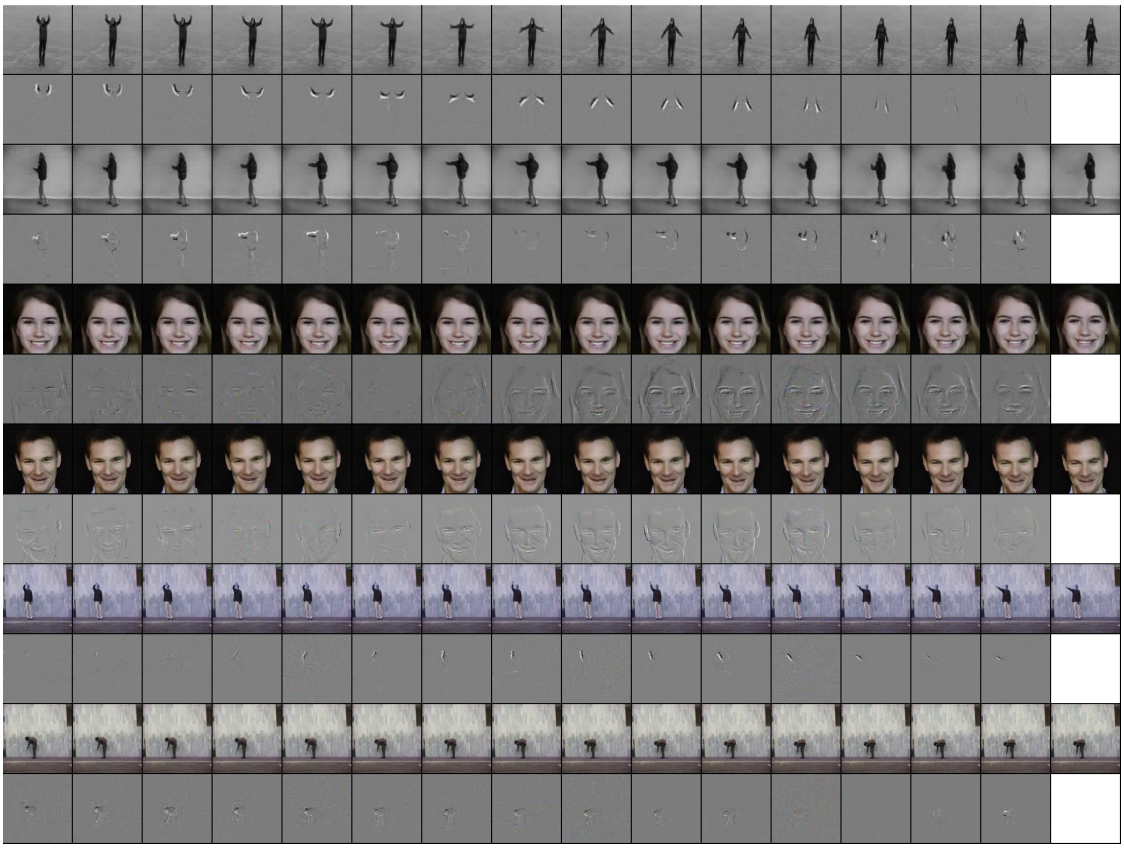


Figure 3.4: Complete generated videos by EncGAN3 at the resolution 128×128.

In Figure 3.4, the first and third rows show the frames of the generated videos while the second and fourth rows show their corresponding difference maps (calculated by Equation (3.3)), indicating the content and movement separately, after training on the KTH dataset. Similarly, the next four rows and the last four rows show results corresponding to generated video data after training on the UvA and Weizmann datasets, respectively. As can be observed from the first row in Figure 3.4, the body of the person exercising the movement is well generated and the frame differences from the second row indicate a sharp representation of the movement. The frames from the third row also show the clear movement of the two moving hands in the boxing action. The fifth and seventh rows show well generated faces with well defined facial features indicating smiling and nodding of the head. The movement can be identified through their corresponding difference maps in the sixth and eighth rows,

Figure 3.5: Enlarged part of the generated videos.

respectively. Moreover, we can observe that the last eight frames in the seventh row perform a delicate movement of eyesight, which shows a good modelling ability of subtle expressions by EncGAN3. Such results indicate good potential for modelling and generating facial expressions displaying micro-expressions [96, 97]. The ninth and eleventh rows show persons waving their hands and bending, respectively. On the ninth row, it can be observed that the moving hand and the head of the person are well defined in the generated video clip. From the eleventh row, it can be observed that the hand that does not touch the ground is also well synthesized. For more details, the enlarged part in Figure 3.5 highlights specific features of the generated videos.

In Figure 3.6, we provide four examples, one on each row, of generated videos with two persons in the scene. Those generated frames show two persons moving independently from each other, of moving direction and movement type. The generation of two-object videos started from training EncGAN3 on the Weizmann dataset after training for rather many epochs, over

Figure 3.6: Generated videos by EncGAN3, of resolution 128 × 128 trained on the Weizmann dataset that show two persons doing similar or different movements simultaneously.

9000 epochs actually. Such results do not replicate anything seen in the training datasets. However, the results show realistic representations of the two persons, performing similar or different activities. We consider that this behaviour may result from the deep compression of content and motion information through the encoder. The deep compression allows the generator to learn richer representations by combining information from different videos. Additionally, the model's large size and efficient representation mechanism, relative to the small training data, seems to create sufficient representation redundancy. This redundancy, guided by the two-stream encoder, allows the model to represent more than one content type simultaneously. For different contents, the motion information directs distinct types of movements, enabling the generation of videos that exhibit complex interactions between multiple moving objects. Notably, such multi-object dynamics were not observed in the results produced by G^3AN, even when trained for a similar or longer duration. These findings suggest that EncGAN3 can generate videos displaying intricate motions and interactions among multiple objects.

(a) EncGAN3, Weizmann

(b) EncGAN3, UvA

Figure 3.7: Generated videos of EncGAN3 on Weizmann (a) and UvA (b) datasets with the resolution of 64 × 64 pixels.

Figures 3.7, 3.8 and 3.9 show more generated video results at lower 64×64 resolution, for comparing to G^{3}AN. Firstly, we provide the full video frames generated by EncGAN3 after training separately on Weizmann and UvA datasets, to show the generation ability of Human body action and Facial expressions, as in Figures 3.7 (a) and (b). In detail, the rows in Figure 3.7 (a) contain the frames from the video showing body movements such as bending, jumping up, waving one hand and waving two hands, respectively. Figure 3.7 (b) displays various people smiling and changing their facial expressions.

(a) KTH (b) UvA (c) Weizmann (d) UCF101

Figure 3.8: Comparing EncGAN3 (upper panels, each representing three rows with four frames sampled from a video) with G^{3}AN (bottom panels) after being trained on various datasets where the generated videos have a resolution of 64×64 pixels.

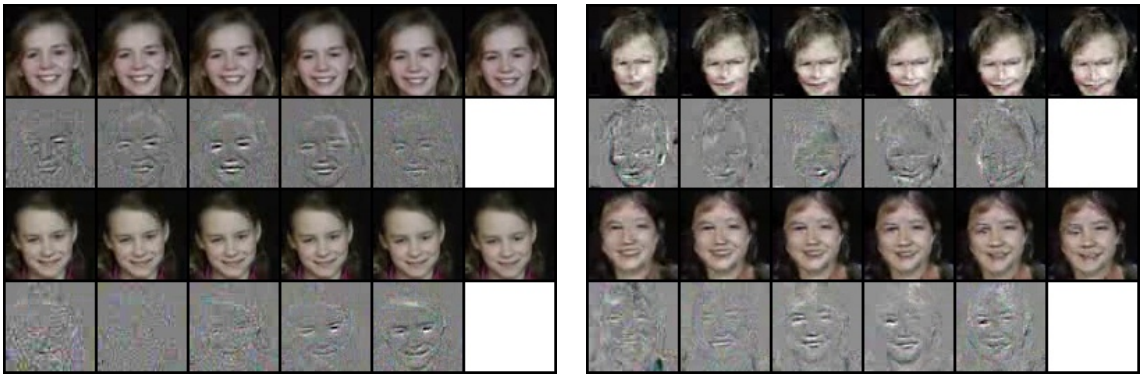
Next, we compare the visual results of EncGAN3 with G^{3}AN in Figure 3.8. In the upper panels of Figure 3.8 (a), (b), (c) and (d), we show on each row four frames from videos generated by EncGAN3 following the training with KTH, UvA, Weizmann and UCF101 datasets, respectively. While the videos generated by G^{3}AN[1] are provided in the bottom four panels of Figure 3.8. The video frames generated by EncGAN3 entail fewer artifacts with less distortion while displaying smooth movement when compared to the frames generated by G^{3}AN. The fewer artifacts can be observed from the first and second row of Figure 3.8 (a), the second row of Figure 3.8 (b) and in Figure 3.8 (c) when compared to the frames generated by G^{3}AN from underneath.

Then, we compare the movement generation results of EncGAN3 with G^{3}AN in Figure 3.9, after training on the UvA dataset. The generated movement of facial expressions displays micro-expressions. The movement is shown by the difference between successive frames in the second and fourth rows in Figure 3.9, indicating our EncGAN3 preserves well the face structure through the facial movement.

3.4.5 Ablation Study

In this section, we provide the ablation studies to explore: the contributions of various components within the EncGAN3 architecture in Figure 3.1, the effect of different ways to assess the video reconstruction term in training objectives, the relationships between

[1]The code used is provided at https://github.com/wyhsirius/g3an-project.



(a) EncGAN3,UvA (b) G^{3}AN3,UvA

Figure 3.9: Video frames generated by EncGAN3 (a) and G^{3}AN (b) on the UvA dataset. Every even row shows frame difference maps used to represent the movement.

input latent codes and generated videos, as well as the impact of training video frames from different sampling strategies.

Architecture	UvA		Weizmann		KTH	
	FID↓	IS↑ (actor)	FID↓	IS↑ (actor)	FID↓	IS↑ (actor)
no G_C, G_T	95.500	63.926	101.638	2.244	73.220	2.867
no G_C	88.058	133.352	89.004	7.020	75.309	3.853
no G_T	90.713	537.852	97.554	5.564	74.963	4.966
no F-SA	87.526	-	82.821	-	73.792	-
no Enc	93.258	148.216	98.564	6.303	75.388	2.328
EncGAN3	**86.210**	**571.29**	**78.935**	**8.906**	**66.621**	**5.986**

Table 3.4: Ablating the contribution of various components of the EncGAN3 architecture.

Contribution of various components within EncGAN3 model. We ablate the Encoder and the three-stream processing pipeline, F-SA module in the Generator to test the contribution of those components to EncGAN3. In this ablation study, we consider three datasets for the training, UvA, Weizmann and KTH. We provide the ablation results using two metrics, the video FID and video IS, in Table 3.4. The video IS on the Weizmann and KTH datasets is computed based on the Inception Network pre-trained on actor classes of the corresponding dataset, instead of on action classes as in the quantitative evaluation. Consequently, the IS values on Weizmann and KTH datasets on Table 3.4 are different from those on Table 3.3. From Table 3.4, we observe that the presence of two auxiliary streams G_S and G_T, as well as the Encoder and F-SA module, benefit the performance of EncGAN3. In detail, the Generator without the temporal stream G_T generates static images without movement while the Generator without the spatial stream G_S generates frames with bad content.

Furthermore, EncGAN3 can generate videos using just random noise or using the latent code provided by the Encoder. Considering the diverse generation capabilities of EncGAN3, we further ablate the impact of the Encoder and F-SA modules in detail with results provided in Table 3.5. In Table 3.5, we provide the video FID scores of our model trained without the Encoder in the first and second rows, trained with the Encoder but tested without the

Encoder	FSA	UvA	Weizmann	KTH
	(yes/no)	FID↓	FID↓	FID↓
no	no	95.47	89.98	79.36
no	yes	91.77	98.27	111.99
train	no	89.46	88.00	**62.53**
train	yes	93.65	102.36	83.51
train, test	no	87.52	82.43	73.79
train, test	yes	**86.21**	**78.93**	66.62

Table 3.5: Ablating the contribution of the Encoder and the F-SA module. The first column indicates whether to use the Encoder during either the training or testing time or not.

Encoder in the third and fourth rows, and using the Encoder at both training and testing time in the last two rows. We alternatively show the results when considering or not the F-SA module. From Table 3.5, we observe that the best FID results all come from the usage of Encoder at training or testing time, indicating the advantages of using the Encoder.

Changing the way how the video reconstruction is assessed. We study the performance difference when assessing the reconstruction error term used in the training objectives described in Section 3.3.1. As the two-stream Encoder uses the first frame and frame difference maps as the input to the content and motion streams, we consider replacing the reconstruction term of videos to be the reconstruction of the first frame and frame difference maps for a more detailed match. The replacement changes the objective function of the Encoder and Generator. The loss function of the Encoder in this case is :

$$\begin{aligned} L_{Enc,\mathbf{v}} = & \sum_{i=1}^{N} \|\mathbf{x}_{i0} - \widehat{\mathbf{x}}_{i0}\| + \sum_{i=1}^{N}\sum_{j=1}^{T_i} \|\mathbf{v}_{ij} - \widehat{\mathbf{v}}_{ij}\| \\ & - D_{KL}(q_{\theta_{\mathbf{x}}}(\mathbf{z}_{\mathbf{x}}|\mathbf{x}))||p(\mathbf{z}_{\mathbf{x}})) - D_{KL}(q_{\theta_{\mathbf{v}}}(\mathbf{z}_{\mathbf{v}}|\mathbf{v}))||p(\mathbf{z}_{\mathbf{v}})), \end{aligned} \tag{3.9}$$

where the first and second term represents the error in the first frame and the difference maps in the ith sequence of T_i frames and N video sequences. Similarly, the objective function of the Generator in this case becomes:

$$\begin{aligned} L_{G,\mathbf{v}} = & \mathbb{E}_{\widehat{\mathbf{x}}_n \sim G(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}})} \log[D(\widehat{\mathbf{x}}_n)] + \mathbb{E}_{\tilde{\mathbf{x}}_n \sim G(\tilde{\mathbf{z}}_{\mathbf{x}}, \tilde{\mathbf{z}}_{\mathbf{v}})} \log[D(\tilde{\mathbf{x}}_n)] \\ & + \mathbb{E}_{\mathbf{z}_{\mathbf{x}} \sim p(\mathbf{z}_{\mathbf{x}}), \mathbf{z}_{\mathbf{v}} \sim p(\mathbf{z}_{\mathbf{v}})} \log[D(G(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}}))] \\ & + \mathbb{E}_{\tilde{\mathbf{z}}_{\mathbf{x}} \sim \mathcal{N}(0,\mathbf{I}), \tilde{\mathbf{z}}_{\mathbf{v}} \sim \mathcal{N}(0,\mathbf{I})} \log[D(G(\tilde{\mathbf{z}}_{\mathbf{x}}, \tilde{\mathbf{z}}_{\mathbf{v}}))] \\ & - \sum_{i=1}^{N} \|\mathbf{x}_{i0} - \widehat{\mathbf{x}}_{i0}\| - \sum_{i=1}^{N}\sum_{j=1}^{T_i} \|\mathbf{v}_{ij} - \widehat{\mathbf{v}}_{ij}\| \end{aligned} \tag{3.10}$$

where the last term is the replacement term, measuring the reconstruction error between real and generated frame difference maps.

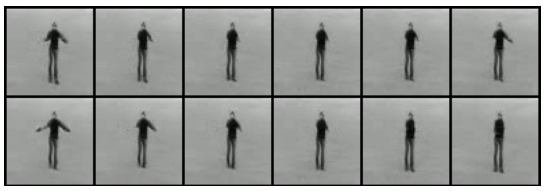
In this study, we consider the replacement of the reconstruction term only in the Encoder, as $L_{Enc,\mathbf{v}}$ in Equation (3.9), or in both Encoder and Generator, marked as $L_{Enc,\mathbf{v}} + L_{G,\mathbf{v}}$ where $L_{G,\mathbf{v}}$ is in Equation (3.10). Besides, we consider decreasing the learning rate from $2e^{-4}$ to $4e^{-5}$ when training all modules, allowing the model to focus on finer details with fewer oscillations. The learning function described in Section 3.3.1 is regarded as the baseline.

Variants	Learning rate	FID↓
Baseline	2e-4	**86.21**
$L_{Enc,\mathbf{v}}$	2e-4	90.77
$L_{Enc,\mathbf{v}}$+$L_{G,\mathbf{v}}$	2e-4	95.02
$L_{Enc,\mathbf{v}}$+$L_{G,\mathbf{v}}$	4e-5	89.71
Baseline	4e-5	88.68

Table 3.6: Ablating changes in the loss functions and learning rate.

As shown by the results from Table 3.6, the baseline version of the objective functions described in Section 3.3.1 achieves better video FID scores than $L_{Enc,\mathbf{v}} + L_{G,\mathbf{v}}$ on both learning rates considered, indicating that it is important to consider the full frame reconstruction in the training objective functions instead of the expressions from the Equations (3.9) and (3.10). The training objective functions such as L_{Enc} from Equation (3.1) and L_G from Equation (3.4) do not only optimize the motion stream reconstruction, but they also consider how the movement representation is employed to reconstruct realistic video streams, frame after frame. As for the learning rate, a decreased learning rate causes a worse video FID result for the baseline objective while causing a better FID result for training objective $L_{Enc,\mathbf{v}} + L_{G,\mathbf{v}}$, indicating that the setting of the learning rate depends on specific training objectives.

Video generation effects following latent code manipulation. To explore the relationship between latent codes and generated frames, we interchange the latent codes of the movement and combine them with different content latent codes, observing the changes in the generated video clips, as displayed in Figure 3.10 and 3.11. As shown in Figure 3.10 (a), we fixed the content latent code z_{c_1} while combining with different motion latent codes z_{m_1}, z_{m_2} as inputs for the Generator, with the video results shown on top and bottom of Figure 3.10 (a), respectively. From Figure 3.10 (a) and (b), we can observe that the generated frames display the same subject clapping hands while having different particular movements.

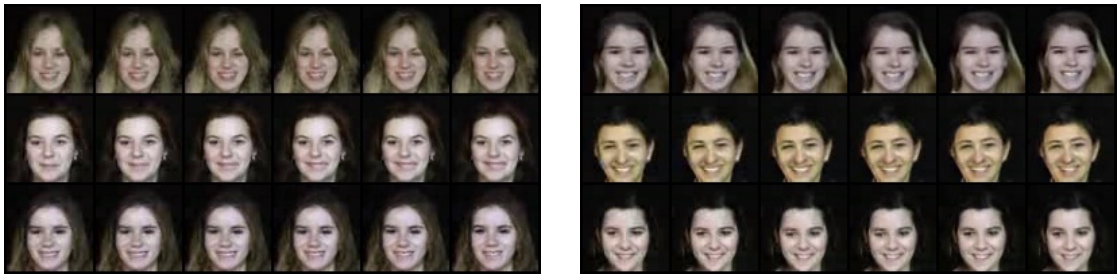


(a) z_{c_1},z_{m_1} (top) and z_{c_1},z_{m_2} (bottom)

(b) z_{c_3},z_{m_3} (top) and z_{c_3},z_{m_4} (bottom)

Figure 3.10: Frames from the first and second rows of (a) and (b) are generated using the same content latent code with different motion latent codes.

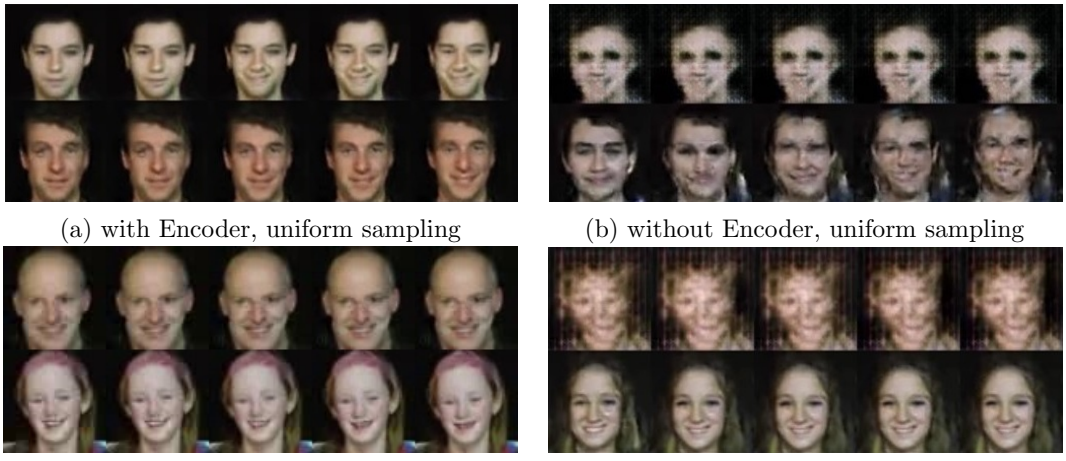
Meanwhile, we consider summing another two latent codes (z_{c_1},z_{m_1}) and (z_{c_2},z_{m_2}) of both their content and motion parts with the results provided in Figure 3.11. The first and second rows are frames generated by latent codes (z_{c_1},z_{m_1}) and (z_{c_2},z_{m_2}), respectively. The third row corresponds to the frames generated by their summations, as $z_{c_1} + z_{c_2}$ and $z_{m_1} + z_{m_2}$ of content and motion latent codes. From Figure 3.11 (a) and (b), We observe that the frames generated using the sum of the two latent codes inherit and combine some properties of the two video sequences corresponding to the two latent codes.



(a) z_{c_1},z_{m_1} (top), z_{c_2},z_{m_2} (middle) and their sum (bottom), respectively of the left and right panels.

Figure 3.11: The latent codes used to generate the frames in the third row in each panel are obtained by summing the latent codes used to generate the frames in the first and second rows.

Sampling strategies for selecting data to produce training set. We conducted an ablation study on two sampling strategies to create training sets with different complexity levels and trained the model with and without using the Encoder, in order to explore whether the usage of the Encoder improves the learning ability of the model to learn efficiently from more complex data. We trained on the UvA dataset using these sampling strategies. The two sampling strategies are step sampling and uniform sampling. The step sampling randomly selects a starting frame and then samples the following video frames with a sampling step of 2 or 3 for the entire number of frames sampled. The sampling step is randomly selected at each sampling time. The uniform sampling divides a video clip into 16 sets with equal numbers of frames, and then sample randomly one frame from within each set. Hence, frames sampled from uniform sampling contain more complex temporal information due to the fluctuation of the frame rate. Compared to the step sampling, the uniform sampling records better the movement happening throughout the video clip while making the data more complex.



(a) with Encoder, uniform sampling (b) without Encoder, uniform sampling

(c) with Encoder, step sampling (d) without Encoder, step sampling

Figure 3.12: Generated frames for EncGAN3 in (a), (c) and G^{3}AN in (b), (d) when using uniformly or step sampled training sets trained for 100 (top row) and 5000 epochs (bottom row).

According to the results from Figure 3.12 (a) and (b), when considering the uniformly sampled training set, the model trained when considering the Encoder generates good video

results soon after training for 100 epochs while the one when not considering the Encoder generates bad results even after training for 5000 epochs, indicating that the Encoder benefits the learning ability of the model to learn more efficiently on more complex data. The first and second rows of all panels in Figure 3.12 are the results achieved after training for 100 and 5000 epochs, respectively. Comparing Figure 3.12 (a) and (c), frames generated by the model trained when considering the step sampled training set show smoother movement and better content after training for 5000 epochs, indicating the benefits of maintaining the frame rate for the training video. As a consequence of these results, for all other experiments, we use step sampling and Encoder during the training.

3.5 Conclusion

In this research chapter, we introduced an encoder-enabled method to enhance GAN-based video generation models with an inference mechanism. This proposed method combines the inference capabilities of the Encoder with the generative abilities of a video GAN. Following this method, we proposed EncGAN3, a VAE-GAN hybrid network specifically designed for high-quality video generation. EncGAN3 leverages an inference mechanism to stabilize learning and improve the video GAN generator's performance, enabling the generation of realistic videos at resolutions of 64×64 and 128×128 pixels. The generated videos can depict single or sometimes even two moving objects without additional restrictions on multi-object generation. Our experiments demonstrated that EncGAN3 achieves competitive performance on several benchmark datasets, including UvA-NEMO, Weizmann, KTH, and UCF101, in terms of video FID and video IS metrics. The results indicate that EncGAN3 produces videos with superior visual quality and diversity compared to other models discussed in this chapter. Future work will focus on enhancing the temporal representation efficiency to enable the generation of longer videos.

(a) Weizmann

(b) UvA

Figure 3.13: Frames in each row are sampled for every 5 frames from 90-frame videos at 128×128 resolution generated by EncGAN3.

3.5.1 Limitation and further work

EncGAN3 is designed to generate 16-frame videos and is specifically trained for this fixed duration. While it is possible to extend the video length by increasing the size of the motion latent code, as shown in Figure 3.13, this approach goes beyond the original design of the model. Forcing EncGAN3 to generate videos longer than its training duration leads to a significant decline in frame quality compared to the standard generation method. This degradation is especially evident in human action videos. For example, in Figure 3.13 (a), the hand

is rendered poorly, compared to the more detailed frames in Figure 3.5. Additionally, facial expressions in longer videos show unnatural distortions, such as the structural deformation of the left eye in Figure 3.13 (b), resulting in unrealistic and awkward movements.

Video length (frames)	Resolution (pixels)	Batch size (16-frame videos)
16	64×64	10
16	128×128	1
20	64×64	CUDA-OOM

Table 3.7: EncGAN3 training cost on V100 with a memory of 32GiB.

Simply extending the training video length to capture longer sequences would cause a steep rise in GPU memory requirements. As detailed in Table 3.7, increasing spatial resolution leads to a relatively moderate increase in memory usage. In contrast, even a small increase in video length results in an exponential rise in memory demand, highlighting the inefficiency of temporal representation compared to spatial representation. This inefficiency motivates further research into developing a more efficient video representation mechanism for the temporal dimension, which could preserve the ability to model diverse types of video data while reducing the substantial cost of such dense temporal representations.

These challenges lead to the development of the recall mechanism, which is introduced in the next chapter. This mechanism allows the model to generate much longer videos, up to hundreds of frames, while achieving better temporal dynamics and maintaining high visual quality without increasing the training video length.

Chapter 4

Longer video generation using REncGAN

4.1 Introduction

Recent advancements in deep learning have significantly improved video generation, leading to a wide range of applications from virtual reality to artistic production and scientific research. Traditional approaches to video generation have predominantly relied on models such as GANs, VAEs and their hybrids. These models have proven effective for generating short video sequences, typically up to 16 frames, which correspond to less than a second of video at standard frame rates [5, 43, 60, 61, 42, 62, 98, 99].

While recent diffusion models have shown remarkable progress in generating high-quality images and short video clips, generating long-duration videos with high fidelity remains a significant challenge [33, 34, 35, 36]. Current industry products like Sora (1 minute), Open-Sora (16 seconds), Runway (4 to 16 seconds), Stable Video (4 seconds), and Pika (3 seconds) highlight the state-of-the-art capabilities in video generation using diffusion models [100, 51, 81, 21]. Most of them use 8 to 30 frame-per-second (FPS), reaching mostly tens of frames, such as CogVideo [52] allows to generate maximum 64 frames for 8 FPS from our experiment. Despite their success in producing high-resolution and complex content, these models often face limitations in generating long videos due to challenges in maintaining temporal consistency, avoiding content repetition, and managing increased computational demands.

In Section 3.5.1 of the previous chapter, we discussed potential directions for further developing EncGAN3. One direction focuses on the spatial dimension, such as expanding the frame resolution or increasing the complexity of generated content (e.g., multi-object movements). The other direction focuses on the temporal dimension, specifically extending the generated video length. Considering that the temporal dimension is the most distinctive aspect of video generation compared to image generation, we decided to prioritize enhancing the long-duration video generation capabilities of VAE-GAN-based models with minimal computational overhead in this work.

While models like EncGAN3 [1] excel at generating sharp and coherent short-term video clips, their performance deteriorates significantly when tasked with sequences beyond their training length, as discussed in Section 3.5.1. This limitation is not unique to EncGAN3; many state-of-the-art models face similar challenges when generating longer videos. For instance, models such as VGAN [43], TGAN [60], and MoCoGAN [61] exhibit severe frame quality degradation when generating sequences exceeding 100 frames. Even models specifically designed for long video generation, such as TATS [9], struggle to maintain temporal dynamics and avoid content repetition, particularly in complex scenarios like human body movements [7, 8]. These issues often stem from the models' limited understanding of temporal progression beyond their training data, leading to blurred or repetitive content in longer sequences.

To address these challenges, we propose two novel methods to enhance the long-duration video generation capabilities of EncGAN3. The first method, LEncGAN, integrates Long Short-Term Memory (LSTM) networks [30, 31] to model temporal dependencies across longer sequences, preserving diversity and coherence. The second method, Recall EncGAN (REncGAN), shifts the focus from individual frames to short-term clips, modelling temporal relationships between these clips. While LEncGAN represents an initial model that we developed for long-term video generation, later we developed a better and more efficient REncGAN model. Both approaches reduce computational demands while enabling the generation of extended sequences with consistent quality and diversity.

Both methods significantly extend the capabilities of EncGAN3, enabling it to generate hundreds or even thousands of frames, whereas most existing methods struggle to generate even one hundred frames [6, 7]. Moreover, our methods achieve this with minimal additional computational overhead. For instance, training REncGAN with a batch size of one requires GPU memory comparable to training EncGAN3 with a batch size of 2.5. In contrast, directly increasing the training video length of EncGAN3 from 16 to 20 frames results in GPU out-of-memory errors. By leveraging the recall mechanism, our approach REncGAN achieves a substantial increase in frame generation capacity—from tens to hundreds or thousands of frames—with only a marginal increase in GPU memory requirements.

The main contributions of this research are as follows:

1. We introduce REncGAN, a novel model incorporating a recall mechanism into a VAE-GAN framework, facilitating the generation of long-duration videos with hundreds or even thousands of frames.
2. We extend EncGAN3 with an LSTM-based architecture (LEncGAN) to produce longer, temporally diverse videos of hundreds of frames.
3. We propose new loss functions for LEncGAN and REncGAN that enhance synchronization between encoding and generation modules, improving long-video generation performance while minimizing memory requirements.
4. We conduct extensive quantitative and qualitative evaluations, demonstrating the superior performance of REncGAN and LEncGAN in generating long videos with high visual quality and spatial-temporal consistency compared to existing methods.

In this chapter, we detail these approaches and their effectiveness in generating long videos

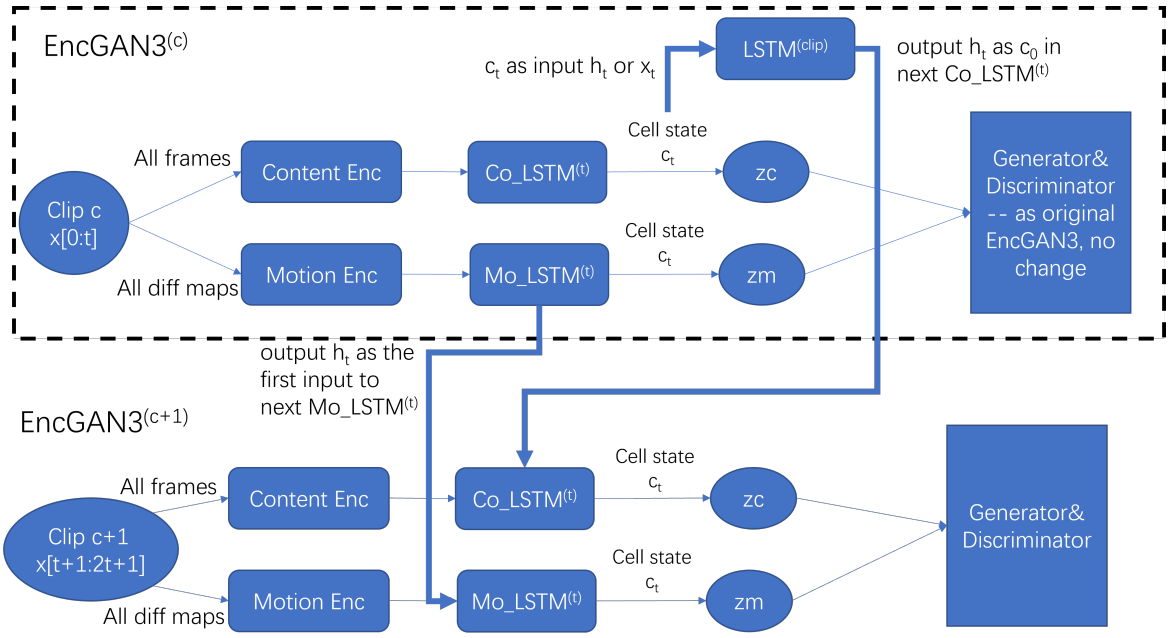


Figure 4.1: LEncGAN3: a Markov chain consists of multiple EncGAN3 states that are connected by LSTM in the Encoder part. To maintain clarity and conciseness, only the connections for the first two states are shown here.

with hundreds of frames, marking a significant advancement over traditional short-term video generation methods.

4.2 LEncGAN: Applying LSTM to Enable EncGAN3 for Long Video Generation

4.2.1 LEncGAN Structure

The development of LEncGAN focuses on integrating Long Short Term Memory (LSTM) modules into the EncGAN3 framework to facilitate long video generation by capturing temporal dependencies between video frames. EncGAN3 utilizes a VAE-GAN hybrid approach with a two-stream encoder, a three-stream generator and a two-stream discriminator, capable of producing 16-frame video clips after training on a corresponding dataset while challenging to generate longer videos. The structure of LEncGAN is depicted in Figure 4.1.

In LEncGAN, the fully connected (fc) layers (FC) of the Motion Encoder (MoEnc) in the original EncGAN3, show as a gray cube at the end of E_m, are replaced by LSTM modules. This modification allows the model to learn temporal information from the difference maps of input video clips. Specifically, the final cell state c_t and hidden state h_t of the Motion LSTM (MoLSTM) are passed to the next state $c+1$, maintaining temporal continuity across video clips.

Similarly, the Content Encoder in the LEncGAN includes an LSTM module that processes

the sequence of video frames. The output states c_t and hidden state h_t from the Content LSTM (CoLSTM) are used as the initial states for the subsequent EncGAN3 state $c+1$, ensuring that content features are temporally coherent across clips.

LEncGAN retains the original generator and discriminator components of EncGAN3 without significant modifications. The key innovation lies in the incorporation of LSTM-enhanced encoders, which provide temporally enriched latent spaces to the generator, enabling it to produce coherent video sequences over extended lengths of time.

This structure aims to utilize the sequential modelling advance of LSTM to capture long-term dependencies, ensuring that the generated video sequences are both temporally and contextually consistent.

Algorithm 3 LEncGAN Training Procedure for the First Clip of a Long Video

Required: Content Encoder (CoEnc), Content LSTM (CoLSTM), Motion Encoder (MoEnc), Motion LSTM (MoLSTM), Clip Generator (G), Video Discriminator (VD), Image Discriminator (ID)
Input: The first clip of a video $\mathbf{x}_{0:T_c}$
Output: Discrimination results of True or False T/F

$(h_0, c_0) \leftarrow (0, 0)$
$T_c \leftarrow 15$

Content Encoding:
for $t = 1 \rightarrow T_c$ **do**
    $(h_{0:t}, (h_t, c_t)) \leftarrow \text{CoLSTM}(\text{CoEnc}(\mathbf{x}_{t-1}), (h_{t-1}, c_{t-1}))$
end for
$\mathbf{z}_c \leftarrow c_t$
Motion Encoding:
for $t = 1 \rightarrow T_c$ **do**
    $\mathbf{v}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{x}_t$
    $(h_{0:t}, (h_t, c_t)) \leftarrow \text{MoLSTM}(\text{MoEnc}(\mathbf{v}_t), (h_{t-1}, c_{t-1}))$
end for
$\mathbf{z}_m \leftarrow c_t$
Latent Space Sampling:
$\mathbf{z}_\mathbf{x} \leftarrow \text{sample}(\mathbf{z}_c)$
$\mathbf{z}_\mathbf{v} \leftarrow \text{sample}(\mathbf{z}_m)$
Clip Generation:
$\widehat{\mathbf{x}}_{0:T_c} \leftarrow \text{G}(\mathbf{z}_\mathbf{x}, \mathbf{z}_\mathbf{v})$
Discrimination:
$T/F \leftarrow \text{VD}(\widehat{\mathbf{x}}_{0:T_c})$
$idx \leftarrow \text{random}(0 : T_c)$
$T/F \leftarrow \text{ID}(\widehat{\mathbf{x}}_{idx})$

4.2.2 LEncGAN Training

The LEncGAN inherits the training objective functions of EncGAN3, as detailed in Section 3.3.1. The key difference lies in the integration of LSTM modules, resulting in different Encoder inputting. LSTM has three inputs: the hidden state, the cell state, and the current input frame embedding. The hidden state is the output from the previous input frame embedding, while the cell state, representing memory, is calculated based on all previous input frame embeddings. Therefore, LEncGAN uses the cell state of the last input frame as the latent space, encapsulating information from all input frames.

In LEncGAN, a long video is split into several 16-frame clips, with each clip serving as the input for EncGAN. The data processing in the Encoder differs between the initial clip and subsequent clips. The process for the subsequent clips is the same as for the second clip. Figure 4.1 illustrates the data flow for the first two clips, showcasing both processing methods.

For the initial clip, the hidden state and cell state of each LSTM module are initialized to zero. The calculation process for the two-stream Encoder for the initial clip is detailed in the lines 5 to 18 of Algorithm 3, where T_c indicate clip length and T is the length of entire video.

For the subsequent clips, the initial hidden and cell states of the LSTM inherit the state from the previous clip. The calculation process for these non-initial clips is provided in Algorithm 4. The inherited state and current clip are calculated on the model with the same parameters. The $.detach()$ in lines 14 and 15 means the backpropagation for gradient calculation stops on the inherited state (h_{T_c}, c_{T_c}). The subsequent clips are processed the same as the second clip with an updated frame range for t.

As shown in Algorithm 1 in Chapter 3, in EncGAN3, the Encoder uses the first frame as the content frame instead of all frames to process the content stream and fc layers (the Mofc) instead of MoLSTM to process the motion stream. Except for the Encoding part, LEncGAN uses the same processing pipeline as EncGAN3, while targeting the long video generation.

4.2.3 LEncGAN Inference

At inference time, LEncGAN generates clip-by-clip with inherited states to ensure consistent of generated clips for stitching them to be a long video. The inference procedure of LEncGAN is shown in Algorithm 5, where diversity in the generated clips is introduced by sampling from the latent space, a feature inherited from EncGAN3, where clips generated from sampled latent codes differ significantly from their input clips.

4.2.4 LEncGAN Implementation

LEncGAN utilizes the same hyper-parameter settings as EncGAN3, including the learning rate and the configuration of the ADAM optimizer. The training is conducted on an Ubuntu system using a single V100 GPU with 32 GB of memory. For the experiments, we use videos

Algorithm 4 LEncGAN Training Procedure for Subsequent Clips of a Long Video

Required: Content Encoder (CoEnc), Content LSTM (CoLSTM), Motion Encoder (MoEnc), Motion LSTM (MoLSTM), Clip Generator (G), Video Discriminator (VD), Image Discriminator (ID)
Input: A video segment $\mathbf{x}_{0:2\times T_c}$
Output: Discrimination results of True or False T/F

$(co\_h_0, co\_c_0) \leftarrow (0, 0)$
$(mo\_h_0, mo\_c_0) \leftarrow (0, 0)$
$T_c \leftarrow 15$

Initial Content and Motion Encoding:
for $t = 1 \rightarrow T_c$ **do**
    $(co\_h_{0:t}, (co\_h_t, co\_c_t)) \leftarrow \text{CoLSTM}(\text{CoEnc}(\mathbf{x}_{t-1}), (co\_h_{t-1}, co\_c_{t-1}))$
    $(mo\_h_{0:t}, (mo\_h_t, mo\_c_t)) \leftarrow \text{MoLSTM}(\text{MoEnc}(\mathbf{x}_{t-1} - \mathbf{x}_t), (mo\_h_{t-1}, mo\_c_{t-1}))$
end for
$co\_h_{T_c}, co\_c_{T_c} \leftarrow co\_h_t.detach(), co\_c_t.detach()$
$mo\_h_{T_c}, mo\_c_{T_c} \leftarrow mo\_h_t.detach(), mo\_c_t.detach()$
Subsequent Content and Motion Encoding:
for $t = T_c + 1 \rightarrow 2 \times T_c$ **do**
    $(co\_h_{0:t}, (co\_h_t, co\_c_t)) \leftarrow \text{CoLSTM}(\text{CoEnc}(\mathbf{x}_{t-1}), (co\_h_{t-1}, co\_c_{t-1}))$
    $(mo\_h_{0:t}, (mo\_h_t, mo\_c_t)) \leftarrow \text{MoLSTM}(\text{MoEnc}(\mathbf{x}_{t-1} - \mathbf{x}_t), (mo\_h_{t-1}, mo\_c_{t-1}))$
end for
Latent Space Sampling:
$\mathbf{z_x} \leftarrow \text{sample}(co\_c_t)$
$\mathbf{z_v} \leftarrow \text{sample}(mo\_c_t)$
Clip Generation:
$\widehat{\mathbf{x}}_{T_c:2\times T_c} \leftarrow \text{G}(\mathbf{z_x}, \mathbf{z_v})$
Discrimination:
$T/F \leftarrow \text{VD}(\widehat{\mathbf{x}}_{T_c:2\times T_c})$
$idx \leftarrow \text{random}(T_c, 2 \times T_c)$
$T/F \leftarrow \text{ID}(\widehat{\mathbf{x}}_{idx})$

Algorithm 5 LEncGAN inference Procedure for Long Video Generation

Input: A video $\mathbf{x}_{0:m\times T_c}$
Output: A generated video $\widehat{\mathbf{x}}_{0:m\times T_c}$
Initialize: $(h_0, c_0) \leftarrow (0, 0)$
$prev\_state \leftarrow (h_0, c_0)$
for $j_c = 1 \rightarrow m$ **do**
    $h_{(j_c-1)\times T_c:j_c\times T_c}, c_{(j_c-1)\times T_c:j_c\times T_c} \leftarrow \text{LSTM}(\text{Enc}(\mathbf{x}_{(j_c-1)\times T_c:j_c\times T_c}), prev\_state)$
    $prev\_state \leftarrow (h_{j_c\times T_c}, c_{j_c\times T_c})$
    $(\mathbf{z_x}, \mathbf{z_v}) \leftarrow \text{sample}(c_{j_c\times T_c})$
    $\widehat{\mathbf{x}}_{(j_c-1)\times T_c:j_c\times T_c} \leftarrow G(\mathbf{z_x}, \mathbf{z_v})$
end for

from the Tai-chi-HD (Taichi) [101] dataset that contains hundreds of frames, such as about 400 frames, to benefit the modelling of long duration. The batch size is set to one due to the varying lengths of videos and the requirement for uniform batch elements. LEncGAN employs the same training objectives as EncGAN3. The differences in the training procedures are detailed in Section 4.2.2.

The LEncGAN model is composed of an Encoder, a Generator, and a Discriminator. The structure of the Generator and Discriminator in LEncGAN is identical to that in EncGAN3. However, LEncGAN introduces two LSTM modules for the content and motion streams of the Encoder. Specifically, the LSTM in the motion stream replaces the fc layers (FC) used for compressing the motion information in EncGAN3. In detail, the content stream encoder of LEncGAN includes six two-dimensional convolutional (conv2d) layers and one fc layer, each followed by a batch normalization layer and a ReLU activation function, similar to EncGAN3. The final fc layer is succeeded by an LSTM module with one recurrent layer. Both the input size and internal feature size of the LSTM module are 512, matching the output feature size of the final fc layer. The motion stream encoder of LEncGAN mirrors the structure of the content stream encoder but processes frame difference maps recursively instead of individual frames.

4.3 Transition from LEncGAN to REncGAN

4.3.1 Motivation for REncGAN

The development of the Recall Encoding GAN3 (REncGAN) was driven by several limitations of LEncGAN. Although LEncGAN successfully introduced temporal coherence through LSTM modules, it faced significant computational and hardware challenges. Specifically, the use of multiple EncGAN3 models resulted in prohibitive GPU memory requirements for training on high-resolution videos, making the model difficult to scale for practical applications.

Moreover, the design of LEncGAN, which involves connecting multiple EncGAN3 states with LSTMs, leads to redundancy and inefficiency. Managing numerous states increased both memory usage and training time. Consequently, there was a need for a more efficient and scalable approach that could retain or enhance the performance achieved by LEncGAN while reducing its computational overhead.

4.3.2 Key Innovations

REncGAN offers several key improvements over LEncGAN, including simplified state management, a novel recall mechanism, and enhanced visual quality.

Firstly, REncGAN simplifies state management by reducing redundancy. It achieves this by sharing weights across different EncGAN3 states, allowing for the optimization of a single state at a time rather than handling multiple states concurrently. This approach significantly decreases memory requirements and improves training efficiency.

Secondly, the recall mechanism in REncGAN replaces the LSTM connections used in LEncGAN. This mechanism overlaps video clips by 8 frames between consecutive states, ensuring temporal coherence and diversity while minimizing memory usage.

In terms of temporal coherence, REncGAN utilizes the middle frame of a short clip sequence (16 frames in our experiments) along with corresponding difference maps as inputs. This method enhances temporal continuity between clips, ensuring smoother transitions and reducing visual inconsistencies. By leveraging inherited state information, the recall mechanism improves the overall coherence of long video sequences.

Furthermore, REncGAN enhances visual quality by addressing both memory and redundancy issues. The improvements in the recall mechanism enable the model to handle more complex training data, resulting in videos with better resolution and smoother transitions.

Finally, REncGAN advances the representation of temporal dimension within and between clips. It models intra-clip and inter-clip relationships to generate long videos with lower GPU memory requirements. This approach maintains continuity and coherence while effectively managing temporal diversity, addressing the challenge of exponential memory growth with video length.

4.4 REncGAN: EncGAN3 with Recall Mechanism

4.4.1 REncGAN Structure

Recall Encoding GAN3 (REncGAN) builds upon the empirical study of LEncGAN, introducing a recall mechanism that enhances the continuity and consistency of generated long videos. The structure of REncGAN is depicted in Figure 4.2. This mechanism overlaps input clips between consecutive states, enabling further the continuity and consistency of generated clips with their connected counterparts. The recall mechanism allows each clip state to inherit the temporal context from the previous clip state, maintaining the continuity across the video sequence. Unlike LEncGAN, which incorporates LSTM modules to capture temporal dependencies, REncGAN achieves temporal coherence without altering the internal structure of EncGAN3. Instead, it modifies the training objective and employs a shared weight configuration across all EncGAN3 states. This approach significantly reduces the memory footprint and enhances training efficiency, particularly for high-resolution videos.

The recall mechanism in REncGAN focuses on relationships between adjacent clips. It follows the Markov chain property, which simplifies long video generation by managing temporal coherence between each pair of consecutive clips, rather than across the entire video. This approach allows REncGAN to generate long videos without needing to increase the training length, which would be impractical. For instance, without the recall mechanism, extending the training length of EncGAN3 from 16 frames to 20 frames would lead to GPU memory overflow, as shown in Table 3.7. Additionally, the memory requirement of REncGAN remains constant, as it depends only on the length of the clips used in training, not the total length of the video. In terms of GPU memory usage, REncGAN requires nearly the same amount of memory as EncGAN3 when both are trained with a batch size of one. Although REncGAN

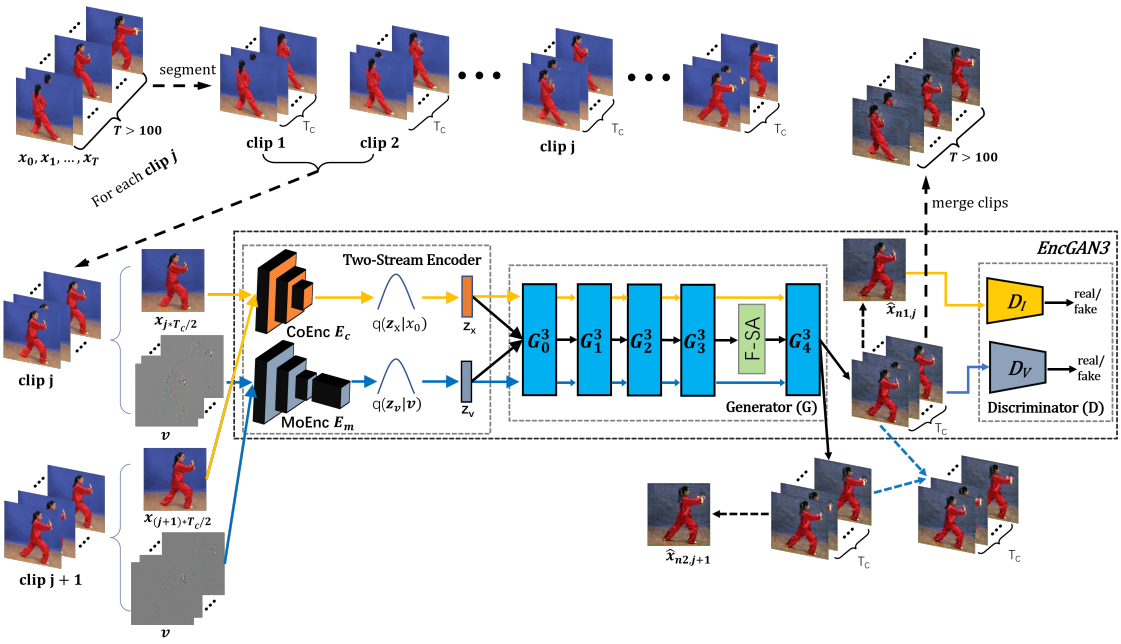


Figure 4.2: Illustration of the Recall Encoding GAN3 (REncGAN).

processes two clips simultaneously, it does not include the noise input processing found in EncGAN3. However, the memory needed to process the second clip, which includes an extra pass through the Encoder, along with the overhead of handling the merged clip through the Video Discriminator (VD), results in REncGAN demanding slightly more memory than EncGAN3. The details of the recall mechanism in REncGAN are illustrated in Figure 4.3.

In practice, REncGAN segments a long video sequence into overlapping short clips. The overlap ensures that each generated clip shares some frames with the next, creating a seamless narrative. A reference frame, often the middle frame of the overlapping segment, is used by the Content Encoder (CoEnc) to maintain continuity and consistency. This frame, along with corresponding difference maps, helps the encoders capture and leverage temporal information, ensuring smooth transitions and coherence in the final output. This operation is built up on the VAE structure and mainly changes the input of the Encoder, named R_Enc.

In addition, the recall mechanism introduces a three-stream video discriminator (3VD) operation, named R_3VD, which processes overlapping video clips. Since R_Enc generates two video clips as input, R_3VD constructs a third clip by combining 8 frames from each of the two generated clips around their overlapping region, representing half of the total frames in each clip. This third clip captures the transition between the two generated clips, ensuring temporal coherence. The Video Discriminator (D_V) then evaluates the three generated clips individually, assessing both their visual quality and the smoothness of their transitions. The outputs of D_V for the three clips are aggregated (e.g., averaged, summed, or otherwise combined. Here is averaged.) to provide learning signals to the generator. This process is crucial for maintaining temporal consistency and visual quality in the generated videos. This operation R_3VD builds upon the traditional GAN framework and mainly modifies the input structure of the Video Discriminator.

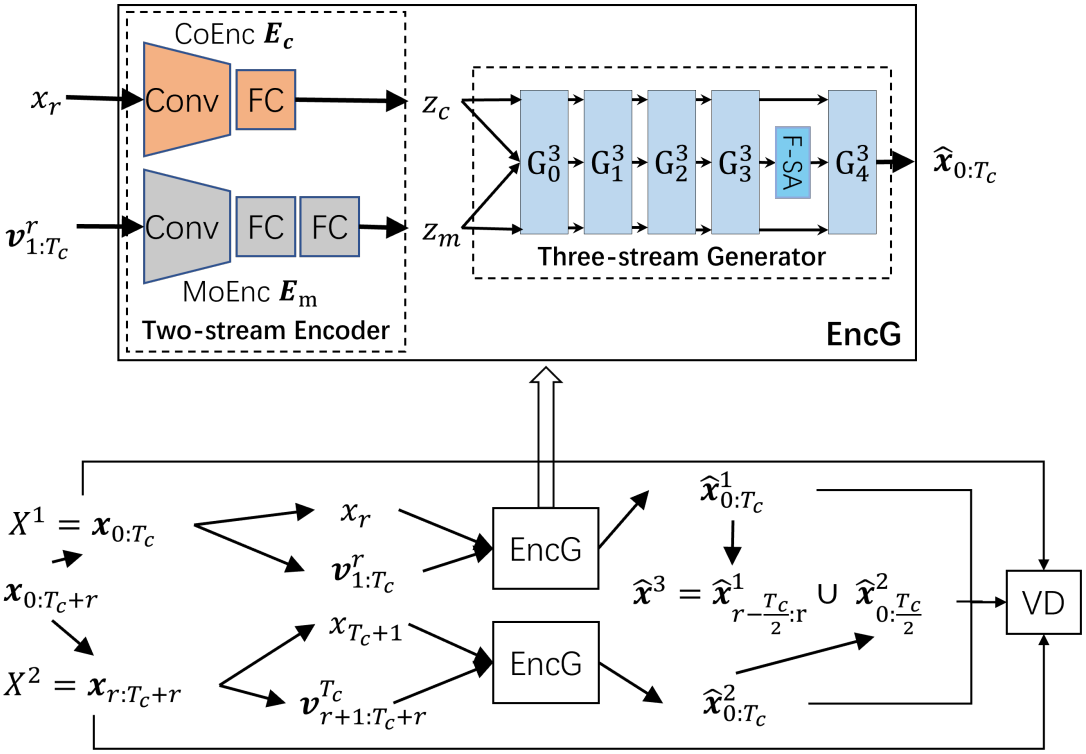


Figure 4.3: Illustration of REncGAN, showcasing the recall mechanism that integrates inputs from the two-stream Encoder and the Video Discriminator (VD) to ensure clip continuity (bottom panel). Each clip has a fixed maximum frame index $T_c = 15$ of index starts from 0, while $T > 100$ for long videos. The reference frame index r denotes the boundary of the overlapping region, calculated as $r = T_c - T_o$, where T_o is the number of overlapping frames. For a 50% overlap($T_o = T_c//2 + 1 = 8$), $r = T_c//2 + 1 = T_o$. The workflow of REncGAN, as depicted in this figure, is further detailed in Algorithm 6, with lines 34 to 40 detailing the main leveraging of VD in the recall mechanism. For instance, a 24-frame input is split into two 16-frame clips (X^1 and X^2) with $T_o = 8$ overlapping frames. The first $r = 8$ frames of their generated clips are stitched to be a new clip $\widehat{X}^3$, and all three clips are fed into VD to optimize the Generator.

In the generation of long videos, REncGAN leverages latent space to provide prior information and does not require the generation of clips from random noise. By removing the noise generation process used in EncGAN3, REncGAN saves GPU memory and better focuses on the core task of generating coherent long videos. This focus is further enhanced by the joint optimization of the encoder and generator in REncGAN, unlike the separate training approach used in EncGAN3. This integrated training improves synchronization between components, leading to higher-quality video outputs. Combined with the efficient use of GPU memory and shared weight configuration, REncGAN can produce long-duration videos with improved coherence and quality, even for high-resolution content.

4.4.2 Training Objective

After implementing the recall mechanism, the loss functions of REncGAN differ from those of the original EncGAN3, introduced in Section 3.2, through the joint training setting of two consecutive video clips, removal of random noise generation terms and processing of the merged video clip for ensuring the video continuation and consistency, resulting in the concatenation of two consecutive short video clips.

In the recall mechanism, the consecutive video clips are all considered to be of the same size or $T_c = 15$ each, while for the sake of continuity and consistency, we consider that they have a number of overlapping frames, defined by the binding frame $\mathbf{x}_r$, which eventually will make up the synthesized long video :

$$\mathbf{y} = \bigcup \{\mathbf{x}_{0:T_c}, \mathbf{x}_{r:r+T_c}, \mathbf{x}_{2r:2r+T_c}, \ldots, \mathbf{x}_{(N_C-1)r:(N_c-1)r+T_c}\} \tag{4.1}$$

where $\mathbf{y}$ represents a training long video sequence, r is the index of the binding frame, used as a reference for connecting clips (hence, the binding frame also called as reference frame), and N_C represents the number of video clips used for synthesizing the long video, which is varied for videos of different lengths.

The generation of clips is adapted to fit the change of the reference frame index r, from 1 in EncGAN3 to $r \in [1, T_c]$. The generated content frame $\widehat{\mathbf{x}}_{j_c,r}$ together with the generated movement are considered for reconstructing all other generated frames from the video clip j_c :

$$\widehat{\mathbf{x}}_{j_c,j} = \widehat{\mathbf{x}}_{j_c,j+1} \ominus \widehat{\mathbf{v}}_{j_c,j+1},\ j = r-1, \ldots, 0 \tag{4.2}$$

$$\widehat{\mathbf{x}}_{j_c,j} = \widehat{\mathbf{x}}_{j_c,j-1} \oplus \widehat{\mathbf{v}}_{j_c,j},\ j = r+1, \ldots, T_c \tag{4.3}$$

where Equation (4.2) and (4.3) utilize pixel-wise addition $\oplus$ and subtraction $\ominus$ operations to generate the preceding and succeeding parts of a video clip, respectively, centered around the reference frame of index r.

The following loss function L_{EncG} is used for training the Encoder and Generator together in REncGAN :

$$\begin{aligned} L_{EncG} = &\sum_{i=1}^{N_L}\sum_{j_c=1}^{N_C} \|\mathbf{x}_{i,j_c,r} - \widehat{\mathbf{x}}_{i,j_c,r}\| + \sum_{i=1}^{N_L}\sum_{j_c=1}^{N_C}\sum_{j=0}^{T_c} \|\mathbf{x}_{i,j_c,j} - \widehat{\mathbf{x}}_{i,j_c,j}(\widehat{\mathbf{v}}_{i,j_c,j}, \widehat{\mathbf{x}}_{i,j_c,r})\| \\ &+ D_{KL}(q_{\theta_\mathbf{x}}(\mathbf{z_x} \mid \mathbf{x}) \| p(\mathbf{z_x})) + D_{KL}(q_{\theta_\mathbf{v}}(\mathbf{z_v} \mid \mathbf{v}) \| p(\mathbf{z_v})) \\ &- \mathbb{E}_{\mathbf{z_x} \sim q_{\theta_\mathbf{x}}(\mathbf{z_x}|\mathbf{x}), \mathbf{z_v} \sim q_{\theta_\mathbf{v}}(\mathbf{z_v}|\mathbf{v})} \log[D(G(\mathbf{z_x}, \mathbf{z_v}))] - \mathbb{E}_{\widehat{\mathbf{x}}_n \sim G(\mathbf{z_x}, \mathbf{z_v})} \log[D(\widehat{\mathbf{x}}_n)] \end{aligned} \tag{4.4}$$

where we consider N_L long videos, each splits into N_C overlapped clips with each clip containing T_c+1 frames with r identifying the reference frame. $\mathbf{x}_{i,j_c,j}$ represents an image frame while $\widehat{\mathbf{x}}_{i,j_c,j}$ is its reconstruction, $\widehat{\mathbf{v}}_{i,j_c,j}$ the reconstruction of the movement, as the difference between consecutive frames, associated with the frame j form clip j_c from the long video i. Meanwhile, $\{\mathbf{z_x}, \mathbf{z_v}\}$ represent the latent spaces of the content and movement, modelled by the encoders E_c and E_m, respectively. They are forced to be probabilistically close to their prior distributions through the Kullback-Leibler Divergence D_{KL}.

The loss function of the image-stream Discriminator $L_{D_I,R}$ in REncGAN is similar to the one used in EncGAN3, after removing the random generator components due to enabling the joint training. The same applies for the loss function of the video-stream Discriminator $L_{D_V,R}$ before applying the R_3VD operation. Their loss functions are :

$$L_{D_I,R} = -\mathbb{E}_{\mathbf{x}_n \sim p(\mathbf{x})} \log[D(\mathbf{x}_n)] - \mathbb{E}_{\widehat{\mathbf{x}}_n \sim G(\mathbf{z_x}, \mathbf{z_v})} \log[1 - D(\widehat{\mathbf{x}}_n)], \tag{4.5}$$

$$L_{D_V,R} = -\mathbb{E}_{\mathbf{x}_{0:T_c} \sim p(\mathbf{x}_{0:T_c})} \log[D(\mathbf{x}_{0:T_c})] - \mathbb{E}_{\widehat{\mathbf{x}}_{0:T_c} \sim p(\widehat{\mathbf{x}}_{0:T_c})} \log[1 - D(\widehat{\mathbf{x}}_{0:T_c})], \tag{4.6}$$

where $\mathbf{x}_n$ is a frame sampled from the real video clip and $\widehat{\mathbf{x}}_n$ is from the video generated by the latent codes, while $\mathbf{x}_{0:T_c}$ and $\widehat{\mathbf{x}}_{0:T_c}$ represent the original and generated image sequences forming clips.

After applying the R_3VD operation which merges two successive short video clips, the video-stream Discriminator loss function $L_{D_V,R2}$ is defined according to :

$$\begin{aligned} L_{D_V,R2} = {} & \mathbb{E}_{\mathbf{x}_{0:T_c} \sim p(\mathbf{x}_{0:T_c})} \log[D(\mathbf{x}_{0:T_c})] + \mathbb{E}_{\widehat{\mathbf{x}}_{0:T_c} \sim p(\widehat{\mathbf{x}}_{0:T_c})} \log[1 - D(\widehat{\mathbf{x}}_{0:T_c})] \\ & + \mathbb{E}_{\widehat{\mathbf{x}}_{stitch} \sim p(\widehat{\mathbf{x}}_{stitch})} \log[1 - D(\widehat{\mathbf{x}}_{stitch})] \end{aligned} \tag{4.7}$$

where $\mathbf{x}_{0:T_c} = \{\mathbf{x}_i\}_{i=0}^{T_c}$ and $\widehat{\mathbf{x}}_{0:T_c} = \{\widehat{\mathbf{x}}_i\}_{i=0}^{T_c}$ represent the real two video inputs and their reconstructions, while $p(\mathbf{x}_{0:T_c})$ and $p(\widehat{\mathbf{x}}_{0:T_c})$ are their probabilities. The reconstructions are stitched together to form $\widehat{\mathbf{x}}_{stitch}$. By using the discriminator $L_{D_V,R2}$ for reconstructing $\widehat{\mathbf{x}}_{stitch}$ enforces the generator to learn the temporal relationships of synthesized clips between consecutive states, defined by the reference frame $\widehat{\mathbf{x}}_r$, thereby enabling the generation of long videos.

As for the numbers of overlapped frames T_o, we consider that only when having half of all frames from a video clip overlapping between two consecutive video clips, that is $T_o = T_c//2 + 1$, the boundary index r of consecutive clips are them similar ($T_o = r$), allowing to use the same generator with the same reference index to generate clips that is connectable to both its former and latter clips. Thus, extending the continuous and coherent between successive clips to the entire video. Furthermore, in the experimental results, we provide an ablation study with results when changing the location of the reference frame r in the sequence of frames defining the number of overlapping frames between consecutive video segments.

4.4.3 Training Procedure

During the training, the Discriminator is updated by optimizing L_{D_I} and $L_{D_V,R2}$ using equations Equation (4.5) and (4.7). If the R_3VD operation is not applied, $L_{D_V,R}$ of Equation (4.6) is used instead of $L_{D_V,R2}$ from (4.7). Then the Encoder and Generator loss L_{EncG} are optimized together according to Equation (4.4). After applying the R_3VD operation, each loss function introduced in Section 4.4.2 (except for $L_{D_V,R}$) is used to pairs of successive clips and the resulting loss values are combined (such as averaged) for back-propagation.

The training procedure of REncGAN is provided in Algorithm 6, showing the continuation in the processing by two consecutive clips. Hence, REncGAN is trained to generate connectable clips for those that come from the latent spaces provided by successive clips. When being

Algorithm 6 REncGAN Training Procedure for Long Videos

Required: Content Encoder (CoEnc), convolution part of the Motion Encoder (Moenc), fc part of the Motion Encoder (Mofc), Clip Generator (G), Image Discriminator (ID), Video Discriminator (VD)
Input: A video $\mathbf{x}_{0:T}$, where $T > 100$
Output: Discrimination results of True or False T/F

$T_c \leftarrow 15$
$r \leftarrow T_c//2 + 1$

Former Clip Encoding:
$\mathbf{z}_{co} \leftarrow \text{CoEnc}(\mathbf{x}_r)$
for $t = 1 \rightarrow T_c$ **do**
    $\mathbf{v}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{x}_t$
    $\mathbf{z}_{\mathbf{v}_t} \leftarrow \text{Moenc}(\mathbf{v}_t)$
end for
$\mathbf{z}_{mo} \leftarrow \text{Mofc}(\mathbf{z}_{\mathbf{v}_{1:t}})$
Latent Space Sampling:
$\mathbf{z}_{\mathbf{x}} \leftarrow \text{sample}(\mathbf{z}_{co})$
$\mathbf{z}_{\mathbf{v}} \leftarrow \text{sample}(\mathbf{z}_{mo})$
Former Clip Generation:
$\widehat{\mathbf{x}}^1_{0:T_c} \leftarrow \text{G}(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}})$

Latter Clip Encoding:
$\mathbf{z}_{co} \leftarrow \text{CoEnc}(\mathbf{x}_r)$
for $t = r + 1 \rightarrow T_c + r$ **do**
    $\mathbf{v}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{x}_t$
    $\mathbf{z}_{\mathbf{v}_t} \leftarrow \text{Moenc}(\mathbf{v}_t)$
end for
$\mathbf{z}_{mo} \leftarrow \text{Mofc}(\mathbf{z}_{\mathbf{v}_{1:t}})$
Latent Space Sampling:
$\mathbf{z}_{\mathbf{x}} \leftarrow \text{sample}(\mathbf{z}_{co})$
$\mathbf{z}_{\mathbf{v}} \leftarrow \text{sample}(\mathbf{z}_{mo})$
Latter Clip Generation:
$\widehat{\mathbf{x}}^2_{0:T_c} \leftarrow \text{G}(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}})$

Merge Generated Clips:
$\widehat{\mathbf{x}}^3 \leftarrow \text{concat}(\widehat{\mathbf{x}}^1_{0:r}, \widehat{\mathbf{x}}^2_{0:r})$

Discrimination:
$T/F \leftarrow \text{avg}(\text{VD}(\widehat{\mathbf{x}}^1), \text{VD}(\widehat{\mathbf{x}}^2), \text{VD}(\widehat{\mathbf{x}}^3))$
$idx \leftarrow \text{random}(0, T_c)$
$T/F \leftarrow \text{avg}(\text{ID}(\widehat{\mathbf{x}}^1_{idx}), \text{ID}(\widehat{\mathbf{x}}^2_{idx}))$

tested, REncGAN follows a similar process for generating connectable clips and merging them into long videos. The temporal diversity within the video is guided by the sequence of latent spaces, preventing repetition in the generated content. Meanwhile, the diversity across different long videos is achieved through sampling from different points in the latent space. As shown in ablation studies of REncGAN, latent codes sampled from the same latent space can provide different generation results.

4.4.4 Inference Procedure

At inference time, REncGAN generates long videos by generating sequential clips with solely sequential input clips, without any extra sequential information, such as the inherit state information in LEncGAN. The inference procedure of REncGAN is shown in Algorithm 7, where the clips generated from sampled latent codes significantly differ from their corresponding input clips as observed in its baseline model EncGAN3.

Algorithm 7 REncGAN Inference Procedure for Long Videos

Required: the two-stream Encoder (Enc), Clip Generator (G)
Input: the entire input video $\mathbf{x}_{0:T}$
Output: Generated long video $\widehat{\mathbf{x}}_{long}$

$\widehat{\mathbf{x}}_{long} \leftarrow \emptyset$
$T_c \leftarrow 15$
$r \leftarrow T_c//2 + 1$
$m \leftarrow T//r$

for $j_c = 1 \rightarrow m - 1$ **do**
  $\mathbf{z}_{co}, \mathbf{z}_{mo} \leftarrow \text{Enc}(x_{(j_c-1)\times r:(j_c+1)\times r})$
  $\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}} \leftarrow \text{sample}(\mathbf{z}_{co}, \mathbf{z}_{mo})$
  $\widehat{\mathbf{x}}_{(j_c-1)\times r:(j_c+1)\times r} \leftarrow \text{G}(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}})$
  if $j_c == 1$ **then**
    $\widehat{\mathbf{x}}_{long} \leftarrow \widehat{\mathbf{x}}_{(j_c-1)\times r:j_c\times r}$
  else if $j_c == m - 1$ **then**
    $\widehat{\mathbf{x}}_{long} \leftarrow \text{concat}(\widehat{\mathbf{x}}_{long}, \widehat{\mathbf{x}}_{(j_c-1)\times r:(j_c+1)\times r})$
  else
    $\widehat{\mathbf{x}}_{long} \leftarrow \text{concat}(\widehat{\mathbf{x}}_{long}, \widehat{\mathbf{x}}_{(j_c-1)\times r:j_c\times r})$
  end if
end for

4.4.5 REncGAN Implementation

Similar to LEncGAN and EncGAN3, REncGAN is implemented using the ADAM optimizer with exponential decay rates for the first-order and second-order moment estimations set to $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is set to 2×10^{-4} for all modules: Discriminator, Encoder, and Generator. The training is conducted on an Ubuntu operation system using a single V100 GPU with 32 GB of memory for ablation studies with the Taichi dataset and

other short-term datasets. For generating 128×128 pixel videos, training is done on the Taichi [101] and Sky-Timelapse (Sky) [71] datasets using one A40 GPU with approximately 46 GB actually available memory of 48 GB total memory on an Ubuntu system.

For evaluation, the model is trained on the Taichi and Sky datasets to generate long videos of arbitrary lengths at 128×128 pixel resolution, and the results are compared with other baseline models. In the ablation study, the model is trained on subsets of the Taichi dataset for long video generation at 64×64 pixel resolution, and on other well-known video datasets such as Weizmann [87], KTH [88], UvA [86], and UCF101 [89] for standard video-length generation of 16 frames at 64×64 pixel resolution.

The evaluation of video generation results uses the Fréchet Video Distance (FVD) [82], Video Fréchet Inception Distance (FID) [5], and Video Inception Score (IS) [22].

As for the GPU memory increment after applying the recall mechanism, train REncGAN on batch size of 1 would require the GPU memory smaller than to train EncGAN3 on batch size as 3. For details, in the implementation, to use the R_3VD operation, the GPU memory should be sufficient to train EncGAN3 with a batch size of at least two. Specifically, to match R_3VD, REncGAN uses two consecutive clip segments as one input and processes these clips in parallel with EncGAN3. While REncGAN removed the noise generation process, it requires GPU memory capable of training the EncGAN3 model with a batch size of over one. Since the clip generation process needs to pass the Encoder while the noise process does not, applying only the R_Enc operation would require a batch size of still a bit over one. But the requirement would be smaller than that of REncGAN, since not consider the merged clip part.

4.5 Experimental Transition from LEncGAN to REncGAN

This section presents the experiments that guided the transition from LEncGAN to R_Enc, the initial version of REncGAN, and finally to the full version of REncGAN described in Section 4.4.

4.5.1 Datasets

In the following, we implement long-term video generation on the Tai-Chi-HD (Taichi) [101] dataset, which features videos with hundreds of frames and the moving object remains consistently in view, benefiting the model to capture extended temporal dependencies of the motion.

Taichi dataset contains 2882 videos of 252 actors playing the Taichi action in its training set. However, due to internet connectivity issues during the download process, some videos were either not successfully downloaded or are no longer available. As a result, we currently have a total of 2,718 videos for this dataset. Although all actors perform Taichi movements, the specific routines and actions they exhibit vary significantly. Videos feature actors performing Taichi motions in simple or complex backgrounds, with frames at 256×256 resolution and

lengths ranging from 128 to 1024 frames.

Different models employ varied sampling strategies for long video generation, depending on their specific approaches. For instance, DIGAN [7] and StyleGAN-V [8] repeatedly sample 16 frames at different frame rates to cover longer durations of up to 128 frames. StyleInV [11] directly samples 128 frames as the training length, while TATS [9] splits the entire video into multiple 16-frame clips.

In our approach, to capture long-term temporal dependencies, we use as many frames as possible from the entire video by splitting each video into several 16-frame clips. To save computational costs, we resize frames into 64×64 resolution while maintaining their aspect ratio, and randomly sample videos from the Taichi dataset, focusing on those with lengths between 100 and 500 frames. Table 4.1 presents the relevant information of the video data used in the experiments. For the LEncGAN to R_Enc experiments, we exclusively used videos of Subject A. The R_Enc to REncGAN experiments utilized all subjects listed in the table.

Subject ID	Long-term videos	Video length (s)	16-frame segments
A	8	(217, 430)	246
B	7	(140, 335)	158
C	14	(160, 560)	441
D	2	(272, 392)	81
Total	31	(140, 560)	926

Table 4.1: Subsets of Taichi dataset.

4.5.2 From LEncGAN to R_Enc

This section describes the ablation studies that guided the step-by-step development from LEncGAN to R_Enc, the initial version of REncGAN.

Ablating the image reconstruction error. This study ablates the target of the image reconstruction error in LEncGAN, since LEncGAN inherits the middle cell state c_t in LSTM from the previous clip input. We experiment with three settings: (1) using the first frame $\mathbf{x}_{i0}$ of the current input video clip as the target, as in EncGAN3; (2) using its middle frame $\mathbf{x}_{i7}$ as the target, corresponding to middle state (mid-state); and (3) removing the image reconstruction error entirely.

To evaluate the impact of these settings, we conducted experiments by generating videos in three separate runs. In each run, we generated 64 to 100 videos, with each video containing approximately 200 to 300 frames. Qualitative analysis was performed by visually inspecting the generated videos to assess the consistency and quality of the reconstructed frames, since the results are obvious at the beginning stage of the development.

As shown in Figure 4.4, the frames generated by training with the middle frame $\mathbf{x}_{i7}$ as the target are better, showing fewer red artifacts, better preservation of the moving object and better defined movement.

Figure 4.4: Frames from 256-frame videos generated by LEncGAN, trained with the image reconstruction error on the first frame (top row), middle frame (bottom row), and without image reconstruction error (middle row).

Ablating the Inheritance Mechanism. This study investigates the impact of the mid-state inheritance mechanism in LEncGAN. This mechanism addresses the issue that clip-by-clip training can disrupt the sequence continuity learned by the LSTM. Specifically, we examine the effects of using either the LSTM cell state c_t, the LSTM output x_t (both up to the middle frame), or removing the inheritance step entirely on long video generation.

Figure 4.5: Frames sampled from 416-frame videos generated by LEncGAN with no inheritance (top row), inheriting the LSTM cell state (middle row), or inheriting the LSTM output (bottom row).

As shown in Figure 4.5, frames generated by inheriting the LSTM output x_t exhibit the best quality, showing better preservation of the moving object with more consistent and clearer movement.

Ablating the LSTM module. We test the impact of using either the LSTM module or FC. FC means fully connected layers. Since the LSTM module relies on sequence continuity, FC might be better suited for handling the Markovian relationships within the clips.

(a) LSTM.

(b) FC.

Figure 4.6: Frames generated by models using LSTM (a) and FC (b).

As shown in Figure 4.6, both LSTM and FC produce frames with comparable quality in

terms of content and movement that no significant differences observed between the two approaches. However, using LSTM requires setting the batch size to 1, as each video must be processed as a single batch element to preserve sequence continuity, which limits the batch size due to varying video lengths. This results in an inefficient use of computational resources. In contrast, the FC layers do not have this restriction and can handle multiple clips in a batch efficiently. Therefore, given the comparable results, we decided to use FC layers. We conducted additional experiments based on the model using FC, named R_Enc, which ultimately led to the development of REncGAN.

4.5.3 From R_Enc to REncGAN

This section outlines the step-by-step development from R_Enc, the initial version of REncGAN, to the final REncGAN model described in Section 4.4. While most generated frames in R_Enc are clear and have good contrast, the model struggles with realistic subject movement. In many cases, the moving subject either fades away or exhibits abrupt changes in motion, which motivates the development of our enhanced model, REncGAN.

Following this idea of enhancing the connection between states through the GAN structure, we develop the R_3VD operation as described in Section 4.4 and employ it to build an initial model of REncGAN. We then conduct several experiments to explore the most suitable settings for the model.

In the following experiments, we ablate the Markov chain structure components, loss functions, the number of overlapping frame numbers T_o for the recall mechanism, and the dimensions of the motion latent code of REncGAN, which generates long-term videos of 64×64 pixel resolution. In these ablation studies, we use the Frechet Video Distance (FVD) [82] as the evaluation metric, which follows the same calculation as the Frechet Inception Distance (FID) described in Equation (3.7), but with the FVD calculated based on the output of a pre-trained 3-dimensional (3D) video classifier instead of a 2D image classifier. Because long-term videos have different sizes, we sample sets of frames from each long-term video for computing the FVD score.

length	recall components		
	R_Enc	R_3VD	R_Enc, R_3VD
10	2254.30	322.91	199.98
16	2452.01	359.60	230.31
32	3052.51	340.69	283.20
64	2242.24	312.77	259.97
96	2505.06	288.66	243.44
100	2523.67	289.96	232.93
128	2717.66	326.78	252.51
136	2967.59	-	241.25

Table 4.2: Ablating Markov chain structure components with FVD results of different video lengths.

The ablation results when considering variations in the components of the Markov chain structure are shown in Table 4.2. As introduced in Section 4.4, the Markov chain is built

based on two operations, R_Enc and R_3VD. Hence, we ablate the model using only R_Enc or R_3VD. As shown in Table 4.2, both operations are needed in REncGAN to generate long-term videos.

length	loss				
	v1	v2	v3	v4	v5
10	199.98	104.84	97.67	67.81	2525.70
16	230.31	111.87	107.54	81.68	2522.76
32	283.20	131.40	106.10	83.54	2737.70
64	259.97	114.16	125.99	78.91	3041.56
96	243.44	116.19	128.19	88.42	3206.97
100	232.93	118.36	125.92	90.73	3224.24
128	252.51	149.29	128.07	101.26	3299.62
136	241.25	138.17	140.56	104.04	3379.64

Table 4.3: Ablating variations of loss functions with FVD of different video lengths.

We ablate five variations of the loss function and provide the results in Table 4.3. In this ablation study, we set the dimension of motion latent codes $\mathbf{z}_v$ to 10, as in the original EncGAN3, and the number of overlapping frames between the two consequent video clips for the recall to 8. $\widehat{\mathbf{X}}^3$ is the result of merging $\widehat{\mathbf{X}}^1$ and $\widehat{\mathbf{X}}^2$ in the second step of the recall mechanism. Due to the video reconstruction error terms of $\widehat{\mathbf{X}}^1$ and $\widehat{\mathbf{X}}^2$, we consider the necessity of computing the video reconstruction error of $\widehat{\mathbf{X}}^3$, resulting in two versions of loss functions, denoted as $v1$ and $v2$ in Table 4.3. Both $v1$ and $v2$ follow the original loss function of EncGAN3, which trains the Encoder and Generator separately. The loss function $v1$ computes the video reconstruction term of the third video clip $\widehat{\mathbf{X}}^3$ while $v2$ does not.

As observed in the second and third columns of Table 4.3, $v2$ has better FVD scores than $v1$ on all generated videos, irrespective of their length, indicating that there is no need to compute the video reconstruction error of $\widehat{\mathbf{X}}^3$.

The comparison of loss functions $v1$ and $v2$ motivates the ablation of video reconstruction error terms for long-term video generation, resulting in the loss function $v3$ that does not consider any of the video reconstruction error term. As shown in Table 4.3, FVD results of REncGAN using loss $v3$ are more competitive than those using the loss $v2$.

The losses $v1$, $v2$, $v3$ and $v5$ all follow the original loss function of EncGAN3, which trains the Encoder and Generator separately to maintain the independent generation ability of a single generator without the encoder during training. While long-term video generation in REncGAN relies on both Encoder and Generator, it does not need the independent generation ability of a single generator. Hence, we develop the loss $v4$, which trains the Encoder and Generator jointly and removes the term constraining the generating from random noise to reduce the randomness. As shown in Table 4.3, the loss $v4$ achieves the best FVD results for the videos generated of any of the lengths considered in this ablation study, which is used for REncGAN and detailed in Section 4.4.2.

4.5.4 Lessons Learned

Based on the above studies, we propose a method to represent long videos by splitting them into two parts: intra-clip and inter-clip. For intra-clip modelling, the relationships between frames within a single clip are captured in a conventional way, with the model trained on all frames within the clip to understand their dependencies. For inter-clip modelling, we apply the Markov chain principle, where each clip is linked only to its immediate predecessor, rather than to all previous clips. This approach is applied to the short video clips generated by EncGAN3, allowing us to focus on connecting clips rather than individual frame sequences.

As a result, REncGAN, which utilizes a recall mechanism, can generate long videos consisting of hundreds or even thousands of frames, while maintaining a GPU memory footprint similar to that of its backbone EncGAN3, during training. This is achieved by efficiently managing relationships between clips instead of handling extensive frame sequences.

4.6 Evaluation of REncGAN

In the previous section, we presented the results for LEncGAN, when considering the LSTM network in conjunction with EncGAN3 to generate long-term videos. Nevertheless, the results have been rather poor for the sequences of about 200-300 frames generated after training on the Taichi dataset. While most generated frames are clear and have good contrast, the subject lacks proper movement and in many cases, the moving subject fades away in the video, or changes its movement too suddenly.

In this section, we report the results from several experiments, including a series of video generation ablations on REncGAN of hundreds of frames on videos of 128×128 pixel resolution. Additionally, as the visual quality of generated long-term video sequences depends on the video clips generated from the *EncG* module states, we measure the quality of video results from a single *EncG* state under the short-term video generation setting, which consists of using only a single *EncG* state with the two-stream Discriminator.

4.6.1 Datasets

We trained REncGAN on various datasets for both short-term and long-term video generation. For short-term generation, we used UvA-NEMO [86], Weizmann [87], KTH [88], and UCF101 [89] datasets, which contain videos typically under 100 frames. For long-term generation, we utilized the Taichi subset [101] and Sky dataset [71], featuring videos with hundreds to thousands of frames to capture extended temporal dependencies. As detailed in Table 4.4, REncGAN was trained on the 16-frame segmented versions of two datasets: the Taichi subset (hereafter Taichi) and Sky dataset. The UvA-NEMO dataset includes facial expression videos, while the Weizmann, KTH, UCF101, and Taichi datasets focus on human actions in various background settings. The Sky dataset contains sky views, capturing natural phenomena such as cloud movement and light changes. The Sky dataset contains over 5000 videos of frames at the resolution of 640×360 pixels. Details of datasets for short-term video generation are introduced in Section 3.4.1 and of the Taichi dataset is introduced in

Section 4.5.1.

We sampled $T = 16$ frames as input clips from each folder or video file, then cropped and resized the frames to 64×64 and 128×128 pixels, preserving their aspect ratios. For short video generation, we sampled 16 frames from each video with a temporal step size of 1 to 3, depending on the video length, and resized the frames to 64×64 pixels. For long video generation, we used as many frames as possible by splitting each video into multiple 16-frame clips, resizing the frames to 128×128 pixels. In the Taichi dataset, we follow the setting in Section 4.5.1 while resizing frames to 128×128 rather than 64×64 pixels. For the Sky dataset, which includes videos ranging from 3 to 3000 frames, we excluded those shorter than 24 frames, as they did not meet the minimum training requirements for REncGAN. During training on the Taichi dataset, we found that the quality of generated clips was sub-optimal, impacting long video evaluation. To focus on developing the recall mechanism, which aims to generate long videos while maintaining the training video length and not changing the internal structure of the model, we chose a subset of the Taichi dataset to keep the training set size comparable to that used in short video generation tasks. This helps mitigate the impact of training set size on clip quality. Additionally, since REncGAN requires a minimum of 24 frames and some videos in the Sky dataset are shorter than this, we excluded those shorter videos.

Original Video Datasets

Dataset	Videos	Train	Test
UvA-NEMO [86]	1,240	1,173	67
Weizmann [87]	93	83	10
KTH [88]	599	569	30
UCF101 [89]	13,320	12,567	753
Tai-Chi-HD [101]	2,882	2,594	288
Sky-Timelapse [71]	2,618	2,080	232

16-frame Segmented Datasets[†]

Dataset	Segments	Train	Test
Tai-Chi-HD	68,023	61,923	6,100
Taichi subset (used)	926	626	300
Sky-Timelapse (used)	101,340	95,215	6,125

Table 4.4: Video dataset statistics showing both original videos and their 16-frame segmented versions. [†]The amount of Segments is approximate.

4.6.2 Qualitative Evaluation

In the following, we evaluate the visual quality of the video generated. In Figures 4.7 and 4.8, we show sequences of frames of resolution 128×128 pixels resolution, sampled from the videos generated by REncGAN, DIGAN [7], and TATS [9], after training on Taichi and Sky datasets[1].

[1]Baseline results are obtained from the TATS [9] website: `https://songweige.github.io/projects/tats/`.

Figure 4.7: Each row from top to bottom shows Taichi frames of 128 × 128 pixels resolution from long videos generated by REncGAN, DIGAN and TATS, respectively.

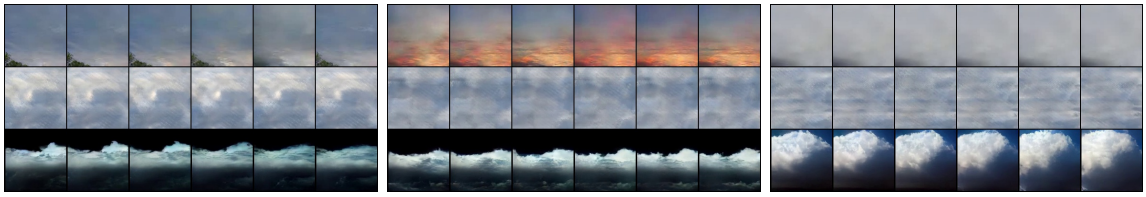


Figure 4.8: Each row from top to bottom shows Sky frames of 128 × 128 pixels resolution from long videos generated by REncGAN, DIGAN and TATS, respectively.

In Figure 4.7, the frames displayed on each row are sampled from generated videos of lengths 424, 1024, and 1024 frames, respectively. These frames are extracted with a step size of 8 from the following sets of frames: 0 to 130 (left), 130 to 260 (middle), and 260 to 400 (right). The frames shown on each row in Figure 4.8, frames are sampled from generated videos of lengths 1324, 1024, and 1024 frames respectively, with a step size of 8 from sets of frames 0 to 300 (left), 300 to 600 (middle), and 600 to 900 (right).

The results from Figure 4.7 indicate that REncGAN generates realistic Taichi movements with a natural speed, whereas TATS output results in videos with movements that look as being too fast, repetitive and less realistic. Additionally, REncGAN maintains the quality of the video generated over longer periods of time without noticeable degradation, outperforming DIGAN in this aspect.

In Figure 4.8, the frames generated by REncGAN show clouds progressively covering the trees in the left panel with minimal noise and without unrealistic artifacts. In contrast, the results of DIGAN, shown in the second row, exhibit some noticeable artifacts and distortions.

4.6.3 Quantitative Evaluation

For the quantitative evaluation, we use FVD, a widely accepted metric for assessing long video generation tasks [8, 9, 11]. Similar to the video FID discussed in the previous chapter on EncGAN3, a lower FVD score indicates better visual quality and spatio-temporal consistency. FVD uses a pre-trained Inception Network different from the one used in video FID, resulting in the measures from different visual perception perspectives.

The FVD results for REncGAN and other models are provided in Table 4.5 when applying on Taichi and on Sky datasets. The results provided consider generating shorter and longer video sequences of 16 and 128 frames, respectively, as indicated on the upper-left side of the dataset name. Table 4.5 shows that REncGAN outperforms other models in terms of FVD

for longer videos on the Taichi sequence, as in Taichi-128^2-128f, highlighting its strength in generating extended video sequences of regular movement. As mentioned in [10], FVD is sensitive primarily to the quality of individual frames and the smoothness of the small-scale motion. Unlike other methods, REncGAN stitches every 8 frames from generated clips. The stitch affects the temporal coherence in the short duration of the 16-frame, resulting in worse performance on FVD-16f. Compared to other methods [6, 7, 8, 9], which prioritize ensuring temporal coherence when generating long videos, our REncGAN model not only takes temporal coherence into account but also focuses on preserving the physical structure of moving objects. This results in weaker FVD performance when generating cloud movement, which consists of irregular motions. However, as shown in Figure 4.8, frames of cloud movement generated by REncGAN perform comparable results to other methods in a long duration of 1000 frames, indicating the advance of our REncGAN in the generation of longer lengths.

	MoCoGAN-HD [6] (ICLR 2021)	DIGAN [7] (ICLR 2022)	StyleGAN-V [8] (CVPR 2022)	TATS [9] (ECCV 2022)	REncGAN [2] (ICIP 2023)
Taichi-128^2-16f	144.7	128.1	143.5	**94.6**	113.5
Taichi-128^2-128f	-	748.0	691.1	-	**145.9**
Sky-128^2-16f	183.6	**114.6**	-	132.5	360.9
Sky-128^2-128f	635.6	**228.6**	-	435.0	587.0

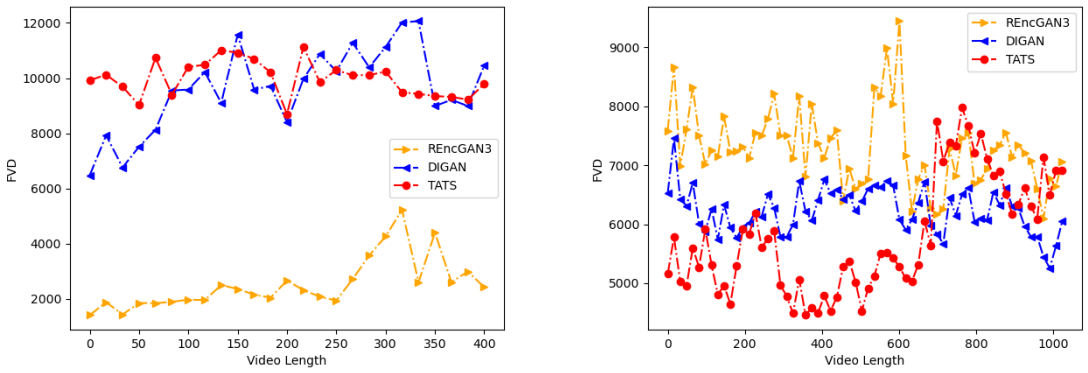
Table 4.5: Evaluation of FVD. Results of other methods (all trained by the same training video length, as 16 frames.) are from [36, 10] ensuring the same resolution and video length of FVD. The notation '-128^2-16f' means the FVD score is calculated on 16 frames with each at 128×128 pixels resolution.

Since REncGAN can be used to generate videos longer than 128 frames, we also assess the quality of long-term videos by analysing short consecutive segments from the generated videos. This method allows us to evaluate the continuity and consistency of video quality across the longer generated videos. We compute FVD for consecutive non-overlapping 16-frame clips extracted from these long-term videos, as shown in Figure 4.9, to evaluate the consistency in the quality of the generated video.

Given that the Taichi dataset includes only a few videos exceeding 1024 frames (specifically, 4 such videos), whereas the Sky dataset contains over 200 videos longer than 1024 frames, we adopt different frame length thresholds for FVD calculation on the two datasets. For the Taichi dataset, we calculate FVD for videos longer than 400 frames, as this threshold captures a sufficient number of videos for meaningful evaluation. In contrast, for the Sky dataset, we calculate FVD for videos longer than 1024 frames, given the abundance of such videos in this dataset.

The differences in the FVD values between Table 4.5 and Figure 4.9 arise from the variations in the ground truth video lengths used for evaluation. Specifically, Table 4.5 reports FVD values for 16-frame and 128-frame videos, where the ground truth videos are filtered to include only those exceeding 12 and 128 frames, respectively. In contrast, Figure 4.9 evaluates FVD for 400-frame and 1024-frame videos, which requires filtering out a significant portion of the dataset. This discrepancy in the ground truth video lengths leads to differences in the FVD values between the two results.

The FVD results calculated on 16-frame segments from the longer generated videos by REnc-



(a) Taichi (b) Sky

Figure 4.9: **Quantitative Evaluation.** FVD of non-overlapping 16-frame clips sliced from long-term videos generated by REncGAN, DIGAN [7] and TATS [9] after training on Taichi (a) and Sky (b) datasets.

GAN, DIGAN, and TATS are shown in Figures 4.9 (a) and 4.9 (b) for the Taichi and Sky datasets, respectively. The results in Figure 4.9 (a) demonstrate that REncGAN outperforms DIGAN and TATS in generating longer video sequences with complex human body movements, such as those in the Taichi dataset. This is because the design of REncGAN involves stitching independently generated clips, which is well-suited for handling rigid movements and reduces frame quality degradation over time. However, in Figure 4.9 (b), the performance of REncGAN on the Sky dataset, particularly for irregular cloud movements, is less competitive compared to DIGAN and TATS. This is due to the emphasis on smooth transitions between frames in irregular movements, which is affected by the stitching operation in REncGAN. Despite this, as shown by the FVD scores in Figure 4.9 (b), the quality of segments generated by REncGAN gradually approaches that of DIGAN and TATS as the sequence length increases, demonstrating its stronger resistance to quality degradation over time. Overall, the ability of REncGAN to generate long and intricate human action sequences, such as those in the Taichi dataset, highlights its broader applicability and effectiveness. While it faces challenges in scenarios requiring highly smooth transitions, such as cloud movements in the Sky dataset, potential improvements to the stitching process—such as incorporating frame interpolation techniques—could further enhance its performance in these scenarios.

4.6.4 Ablation Study

The impact of the recall mechanism in the performance of short-term video generation. The implementation of the recall mechanism extends the capability of EncGAN3 from generating short-term videos of 16 frames to producing long-term videos spanning at least hundreds of frames. However, this adaptation may not be as efficient when aiming to generate short videos. Here, we analyze this effect by video FID and video IS, which were considered in Chapter 3 for the evaluation of EncGAN3. Lower video FID scores indicate better visual quality and spatio-temporal consistency, while higher video IS values indicate

(a) FID results

	UvA FID↓	Weizmann FID↓	KTH FID↓	UCF101 FID↓
VGAN*[43] (NeurIPS 2016)	235.01	158.04	-	115.06
TGAN*[60] (ICCV 2017)	216.41	99.85	-	110.58
MoCoGAN*[61] (CVPR 2018)	197.32	92.18	-	104.14
G^3AN*[5] (CVPR 2020)	91.77	98.27	111.99	108.36
EncGAN3 [1] (ICIP 2022)	86.21	78.93	**66.62**	**91.18**
REncGAN [2] (ICIP 2023)	**70.14**	**70.85**	66.97	95.36

(b) IS and its components

	IS↑	H(y)↑	H(y|x)↓	Dataset (class type)
EncGAN3 [1] (ICIP 2022)	**571.29**	**6.499**	**0.151**	UvA (actor)
	42.60	**3.959**	**0.207**	Weizmann (action)
	50.48	**4.812**	**0.891**	KTH (action)
	33.87	**6.699**	3.177	UCF101 (action)
REncGAN [2] (ICIP 2023)	87.007	4.656	0.190	UvA (actor)
	35.329	3.804	0.239	Weizmann (action)
	11.477	4.087	1.647	KTH (action)
	57.121	5.827	**1.782**	UCF101 (action)

Table 4.6: Quantitative evaluation. "*" results are referred from [5]. ↑ means the higher value is better while ↓ means the lower value is better.

better visual quality and diversity. We also provide Inter-Entropy H(y) and Intra-Entropy H(y|x) as components of the video IS calculation, measuring the diversity and visual quality of the generated videos, respectively [42]. Higher H(y) indicates better diversity, while lower H(y|x) indicates better visual quality. The video FID and video IS results for short-term video generation at a resolution of 64×64 pixels are provided in Table 4.6. As shown in Table 4.6 (a), REncGAN and EncGAN3 yield comparable video FID scores across the four datasets, suggesting that the recall mechanism has a minimal impact on the performance of short video generation when considering the spatial-temporal coherence. However, as shown in Table 4.6 (b), REncGAN exhibits lower IS and H(y) scores compared to EncGAN3 on three datasets. This reduction in diversity can be attributed to the removal of the noise generation term (such as the last term in Equation (3.6)) from the training objectives of REncGAN, which enhances visual quality but reduces diversity. Interestingly, REncGAN achieves better IS and H(y|x) scores on the UCF101 dataset, which is larger and more complex than the other three datasets. This indicates that the closer integration between the Encoder and Generator, achieved when training them together, improves the ability of the model to handle more complex datasets.

Ablating the recall components in long video generation. The results of the ablation study on the recall mechanism components are shown in Table 4.7. As described in Section 4.4, the recall mechanism consists of two main operations: R_Enc and R_3VD. R_Enc refers to the process of feeding video clips considering as having half of frames, overlapped with half of all the generated frames, determined by the reference frame which defines the number of overlapping frames, into the Encoder. R_3VD involves enabling overlaps between

consecutive video clips as a condition for training the Discriminator. In this chapter, the proposed method REncGAN, combining both operations, is referred to as R_Enc + R_3VD as it was explained the structure and training of REncGAN in Section 4.4. The results indicate that both components are essential for REncGAN to effectively generate long videos.

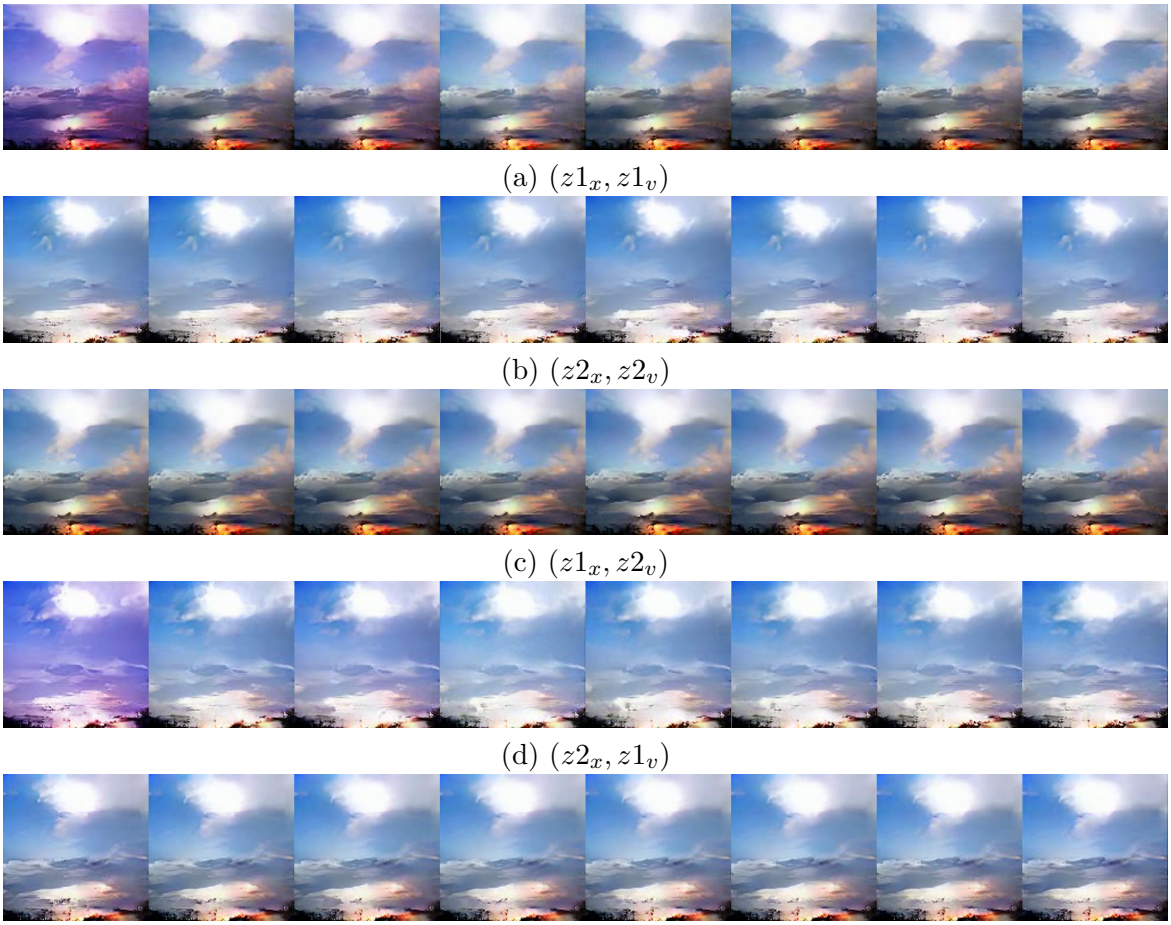
Length	10	16	32	64	96	100	128	136
R_Enc	2254.30	2452.01	3052.51	2242.24	2505.06	2523.67	2717.66	2967.59
R_3VD	322.91	359.60	340.69	312.77	288.66	289.96	326.78	-
R_Enc+R_3VD	199.98	230.31	283.20	259.97	243.44	232.93	252.51	241.25

Table 4.7: FVD for REncGAN components when considering different input video lengths.

Effects of Manipulating Latent Codes on Video Generation. To study the impact of latent code manipulation on generated video frames, we sampled two pairs of latent codes from the same latent space, of $(z1_x, z1_v)$ and $(z2_x, z2_v)$. Figure 4.10 illustrates the frames generated by these latent codes and their manipulated versions. Each row in Figure 4.10 shows frames sampled every two frames from a video clip. Note that frames in Figure 4.10 show subtle movement, as the cloud movement is relatively slow and may not exhibit significant changes within the 16-frame clip.

Figure 4.10 (a) and (b) display the frames generated by $(z1_x, z1_v)$ and $(z2_x, z2_v)$, respectively. As seen in these figures, the frames generated by different latent codes from the same latent space exhibit distinct content. When exchanging their motion latent codes, as shown in Figure 4.10 (c) and (d), frames generated by the same content latent code retained the same content, such as the cloud structure and the land. For example, in Figure 4.10 (d), frames generated by latent code $(z2_x, z1_v)$ preserve the cloud structure from $z2_x$, as shown in Figure 4.10 (b), and the lighting changes from $z1_v$, as seen in Figure 4.10 (a). Figure 4.10 (e) shows the frames generated by the summation of the two latent codes, which incorporate features from both. For instance, the cloud structures in these frames are a combination of characteristics from the frames in Figure 4.10 (a) and (b).

Ablating the sampling step for the training set. By default, we sample every frame from the video (step size of 1) to capture as much detail as possible. The frame rate of the training videos is 29.97 frames per second. In video processing, temporal sub-sampling is often applied to reduce redundancy in the frame sequence. Table 4.8 shows an evaluation of FVD scores using a temporal sampling step of 4. The results demonstrate that using a step size of 1 performs better than sampling every fourth frame, as it preserves the finer details of movement. However, using larger temporal sampling steps introduces additional challenges. For instance, when sampling every 4 frames to obtain 64 frames, the original video must be over 256 frames long. This requirement is too restrictive and excludes a large portion of ground truth videos, making the FVD calculation unreliable. Therefore, when using a step size of 4, we limit the evaluation to 32 sampled frames instead.



(a) $(z1_x, z1_v)$

(b) $(z2_x, z2_v)$

(c) $(z1_x, z2_v)$

(d) $(z2_x, z1_v)$

(e) $(z1_x + z2_x, z1_v + z2_x)$

Figure 4.10: Frames generated by latent code manipulation.

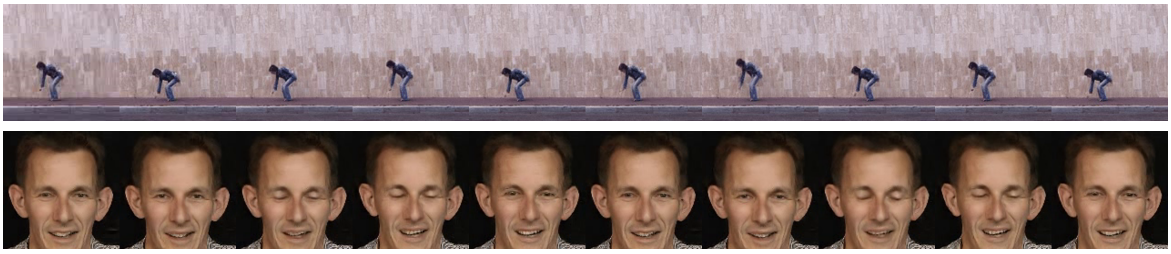
step	FVD of certain video length							
	10	16	32	64	96	100	128	136
1	199.98	230.31	283.20	259.97	243.44	232.93	252.51	241.25
4	1256.66	1923.33	2403.67	-	-	-	-	-

Table 4.8: Ablating the sampling step of the training set with FVD results of different video lengths.

4.7 Comparison of LEncGAN and REncGAN

In this section, we conduct a comparative analysis between REncGAN and LEncGAN in the context of long video generation, also considering the performance of EncGAN3 as a reference.

Figure 4.11 presents frames sampled from videos longer than 16 frames generated by both REncGAN and EncGAN3. The results demonstrate that REncGAN outperforms EncGAN3 in generating long-duration videos, particularly in terms of maintaining high-quality details, such as sharp human actions and clear facial expressions with noticeable blinking. A compar-



(a) REncGAN

(b) EncGAN3

Figure 4.11: Comparison of REncGAN and EncGAN3 on generating human action and facial expression videos. Each row shows 10 frames at 128×128 resolution, with each frame sampled per 5 frames from generated videos, covering a duration of 50 frames.

ative analysis of REncGAN and EncGAN3 on short video generation (16 frames) is provided in Table 4.6.

(a) REncGAN

(b) LEncGAN

Figure 4.12: Comparison of REncGAN and LEncGAN on generating Taichi sequence of videos contains over 300 frames. Frames in each row are sampled per 30 frames from a generated video, covering a duration of 300 frames. Each frame is at a resolution of 128×128 pixels.

Given that the videos in trained datasets are typically around 100 frames in length, and most methods that expand the latent code size struggle to maintain frame quality beyond this range [7, 8], our primary comparison between EncGAN3 and REncGAN involves generating videos of approximately 100 frames. However, when comparing REncGAN with LEncGAN, which are both specifically designed for long video generation, we extend the generated video length based on the two models to over 400 frames. The Taichi dataset, used for training, offers longer duration video data, which aids the models in learning and generating extended sequences. As shown in Figure 4.12, although both REncGAN and LEncGAN successfully generate videos with hundreds of frames, REncGAN consistently produces higher quality

videos, such as the generation of face and hands, showcasing its superiority in long video generation tasks.

4.8 Conclusion

In this chapter, we introduced REncGAN and LEncGAN, designed to tackle the challenges of generating long-duration videos, building upon the foundation of EncGAN3. LEncGAN extends the capability of the model from generating short video clips to producing coherent long-term sequences by incorporating an LSTM module. In contrast, REncGAN removes the LSTM module and implements a recall mechanism to enhance long video generation performance. Both models efficiently reduce memory requirements by modelling temporal relationships between short video clips instead of individual frames, resulting in significant improvements in video consistency and quality.

The recall mechanism in REncGAN offers notable advantages, achieving good temporal coherence by simulating a Markov chain for modelling temporal relationships between clips. This approach allows for the effective generation of long-duration videos while addressing challenges posed by datasets with varying video lengths. REncGAN can generate long videos with hundreds of frames while maintaining realistic content and movement.

However, REncGAN has limitations. While it excels on datasets featuring human actions, such as Taichi, it struggles with coherence and sharpness in simpler or more dynamic datasets, like sky movements. This indicates a need for improved fine-tuning between clips, especially when dealing with videos that exhibit high movement complexity. Future work should explore advanced fine-tuning techniques to enhance this aspect.

Additionally, the recall mechanism requires optimization to lessen its dependency on prior information, potentially bringing REncGAN closer to an unconditional setting while maintaining high performance. To further address sharpness and movement consistency issues, integrating advanced generative techniques, such as diffusion models or updating the generator architecture, may be beneficial.

In conclusion, REncGAN marks a significant advancement in long-term video generation by combining effective temporal modelling with reduced computational overhead. Its ability to generate high-quality long videos with realistic movements establishes a new benchmark in the field. Future research will focus on refining these techniques, addressing current limitations, and exploring new methods to enhance long-duration video generation across diverse datasets.

Chapter 5

Long Video Generation on Less Prior Information

5.1 Introduction

Generating high-quality, long videos is a complex task, requiring both spatial detail and temporal coherence. In this chapter, we introduce two advanced methods built upon the REncGAN framework: AR2 and R3. These methods address key challenges in long video generation, particularly the reliance on prior information and maintaining consistency over extended video sequences.

Background and Motivation. Generating long videos with high quality and temporal coherence presents significant challenges. While generating short videos, such as 16 frames, has been attempted by several models, extending the duration significantly increases memory demands. The previously developed REncGAN (R2) model tackled this by introducing a recall mechanism, which efficiently manages the required memory by focusing on local clip-level relationships rather than modelling the entire video sequence at once. This allowed R2 to generate long videos without increasing the memory requirement for training.

However, despite its success in long video generation, R2 has limitations. The model relies heavily on the prior information from the training data to maintain temporal coherence and video quality, which restricts the diversity of the generated content and caps the video length. Moreover, this reliance on local relationships results in a local bias over long-term coherence, meaning the model captures clip-level interactions well but struggles to represent global video sequences. While the prior information in R2 mitigated these issues, its removal revealed significant repetition problems during long video generation.

This motivates the need for new methods that can overcome the reliance on prior information while maintaining high-quality video generation over extended sequences. To address these challenges, we propose two methods: AR2 and R3.

Problem Definition and Objectives. The key problem addressed in this chapter is the trade-off between reducing the reliance on prior information and ensuring long-term

coherence in video generation. AR2 aims to generate long videos without prior inputs but results in repetitions of short sequences of frames due to the lack of global feature guidance during the training. To resolve this, R3 incorporates global guidance through a GPT module while preserving the memory efficiency of the recall mechanism. While AR2 performs fully unconditional video generation by predicting each clip based on the previous one, R3 requires the first clip as input to ensure that the generated sequence maintains higher quality from the outset. Although R3 is developed to address the repetition problem encountered by AR2, the two models are both built on an improved version of REncGAN (improved R2).

Contributions. This chapter makes the following contributions:

1. **Auto-Regressive Mechanism in AR2:** AR2 modifies the recall mechanism to an auto-regressive setting, removing the reliance on prior video information and enabling the generation of long videos. However, sometimes it produces unrealistic repetitions of short sequences of frames, caused by the local focus of the recall mechanism.
2. **Global Feature Guidance in R3:** R3 introduces a GPT-based module to generate latent space sequences with global feature guidance. This addresses the repetition issue present in the output of AR2 and provides long-term coherence, while still requiring only the first clip as input, maintaining a balance between quality and memory efficiency.
3. **Enhanced Base Model (improved R2):** Improved R2 serves as the foundation for both AR2 and R3, featuring two key enhancements: (1) an optimized generator architecture for handling complex data, and (2) the Video Discriminator Rejection (VDrej) operation, which ensures consistent video quality over extended durations by filtering low-quality frames during auto-regressively generation.

Chapter Outline:

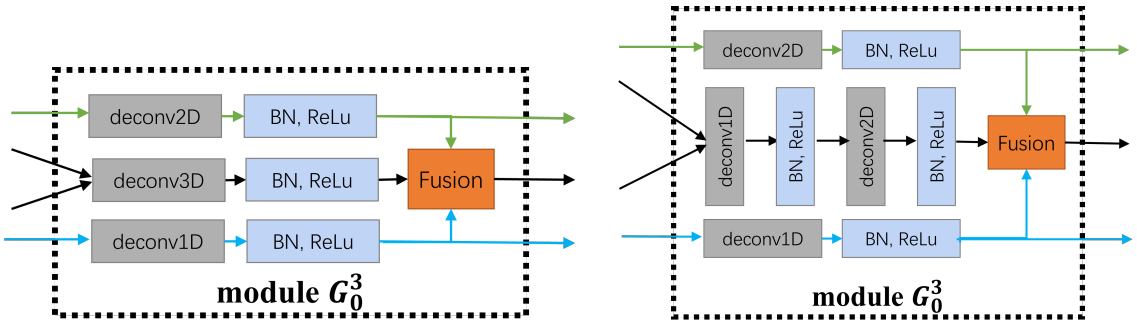
- **Improved R2, AR2 and R3 Methods:** A detailed explanation of Improved R2, AR2 and R3. The latter two also including their innovations and how they address the challenges of R2.
- **Experimental Results:** Qualitative and quantitative evaluations of AR2 and R3, demonstrating their performance.
- **Summary and Discussion:** A summary of findings and potential directions for future research in long video generation.

5.2 Improved R2: Enhancements to the Base Model R2

Before introducing AR2 and R3, we first describe the enhancements made to the base R2 model, referred to as improved R2. These modifications were initially developed during the implementation of AR2 and R3 but were later found to significantly improve the performance of R2 itself. The key enhancements include: 1) the modifications of the Generator; and 2) the VDrej operation.

5.2.1 Generator Modifications

The improved R2 introduces two key modifications to the Generator, aimed at enhancing its ability to generate high-resolution and long-duration videos. These modifications are also incorporated into AR2 and R3.



(a) g_mode: 3d (b) g_mode: 1p2d

Figure 5.1: Generative modes of 3-streams within the G^3 block in the Generator.

Additional G^3 Block for Higher Resolution. To enable the generation of video frames at a higher resolution of 128×128 pixels, we introduce an additional G^3 block, named G_5^3, at the end of the Generator (G) module. This modification is accompanied by extra layers in the Encoder and Discriminator, as described in the third paragraph of Section 3.4.2. The addition of G_5^3 allows the model to produce higher-resolution frames, whereas the original R2 structure illustrated in Figure 4.2 was limited to generating frames at a lower resolution of 64×64 pixels. The placement of G_5^3 block is illustrated in Figure 5.2.

3D Convolutions for Enhanced Spatial-Temporal Processing. The second modification involves changing the internal structure of all G^3 blocks. Each G^3 block consists of three streams: temporal, video, and spatial. In the original R2, the video stream used a combination of 1-dimensional (1D) convolutional layers followed by 2D convolutions, as depicted in Figure 5.1 (b). While this approach was effective for short video generation, as demonstrated in EncGAN3 (Chapter 3), it struggles to handle the increased complexity of spatial-temporal information required for long-duration videos.

To address this limitation, we replace the 1D and 2D convolutions in the video stream with 3D convolutions, as shown in Figure 5.1 (a). The use of 3D convolutions allows the model to process spatial and temporal features in a unified manner, improving its ability to manage the intricate relationships in long-duration videos. This adjustment not only provides more trainable parameters but also enhances the capacity of model to generate high-resolution, long video sequences with better spatial-temporal coherence.

5.2.2 VDrej for Inference

During the video generation phase, the original R2 stitches generated clips sequentially to assemble longer videos. However, the randomness introduced by sampling latent codes for diversity in R2 can disrupt temporal coherence when connecting successive clips. To mitigate

these issues, the improved R2 employs the VDrej operation at Inference time, resulting in a new Inference procedure shown in Algorithm 8.

Algorithm 8 REncGAN Inference Procedure for Long Videos

Required: the two-stream Encoder (Enc), Clip Generator (G)
Input: the entire input video $\mathbf{x}_{0:T}$
Output: Generated long video $\hat{x}_{long}$

$\widehat{\mathbf{x}}_{long} \leftarrow \emptyset$
$T_c \leftarrow 15$
$r \leftarrow T_c//2 + 1$
$m \leftarrow T//r$
$num \leftarrow 10$

for $j_c = 1 \rightarrow m - 1$ **do**
  $\mathbf{z}_{co}, \mathbf{z}_{mo} \leftarrow \text{Enc}(x_{(j_c-1)\times r:(j_c+1)\times r})$
  for $j_{vd} = 1 \rightarrow num$ **do**
    $\mathbf{z}_{\mathbf{x}_{j_{vd}}}, \mathbf{z}_{\mathbf{v}_{j_{vd}}} \leftarrow \text{sample}(\mathbf{z}_{co}, \mathbf{z}_{mo})$
    $\hat{x}_{j_{vd},(j_c-1)\times r:(j_c+1)\times r} \leftarrow G(\mathbf{z}_{\mathbf{x}_{j_{vd}}}, \mathbf{z}_{\mathbf{v}_{j_{vd}}})$
  end for
  if $j_c == 1$ **then**
    $\hat{x}_{best,(j_c-1)\times r:(j_c+1)\times r} \leftarrow \max_{j_{vd}\in[1,num]} \text{VD}(\hat{x}_{j_{vd},(j_c-1)\times r:(j_c+1)\times r})$
    $\hat{x}_{long} \leftarrow \hat{x}_{best,(j_c-1)\times r:j_c\times r}$
  else
    $\hat{x}_{best,(j_c-1)\times r:(j_c+1)\times r} \leftarrow \max_{j_{vd}\in[1,num]} \text{VD}(\text{concat}(prev\_best, \hat{x}_{j_{vd},(j_c-1)\times r:j_c\times r}))$
    if $j_c == m - 1$ **then**
      $\hat{x}_{long} \leftarrow \text{concat}(\hat{x}_{long}, \hat{x}_{best,(j_c-1)\times r:(j_c+1)\times r})$
    else
      $\hat{x}_{long} \leftarrow \text{concat}(\hat{x}_{long}, \hat{x}_{best,(j_c-1)\times r:j_c\times r})$
    end if
  end if
  $prev\_best \leftarrow \hat{x}_{best,(j_c-1)\times r:j_c\times r}$
end for

Unlike the original R2, which generates one clip per latent space, VDrej generates multiple clips (e.g., 20 clips) from the same latent space and selects the best one based on visual quality and connectivity to the previously selected generated clip. Specifically, the clip with the highest Video Discriminator (VD) output—indicating the greatest likelihood of being perceived as real (i.e., the highest value approaching 1)—is chosen. To evaluate connectivity, each newly generated clip is concatenated with the previously selected clip (using the first 8 frames of each), and the stitched clips are input into the VD for assessment. This process significantly improves the quality and stability of the generated videos, ensuring that the majority of long videos maintain high quality while reducing the proportion of poor-quality clips.

In implementation of VDrej, for the first clip, it is directly input into the VD for quality assessment. For subsequent clips, all newly generated clips are concatenated with the

previously selected clip and evaluated by the VD to ensure connectivity and quality. This approach alleviates the impact of errors arising from stitching clips, resulting in more stable and higher-quality video generation.

VDrej specifically targets quality degradation errors and mitigates temporal incoherence while preserving desirable temporal dynamics. By leveraging the VD, which is trained during the training phase and repurposed for testing, this method effectively utilizes existing resources without requiring additional components.

5.3 AR2: Auto-Regressive REncGAN for Unconditional Long Video Generation

Auto-Regressive Recall Encoding GAN3 (AR2) was developed as an extension of REncGAN (R2) to overcome the limitations associated with the reliance on prior information for long video generation. The main design goal of AR2 is to enable unconditional video generation starting from random noise, while still maintaining the efficient internal video representation mechanism established by REncGAN.

To achieve this goal, AR2 introduces an auto-regressive setting within the recall mechanism, which allows for auto-regressive prediction, thereby reducing the dependency on the prior information from an entire video to just a single clip. Additionally, AR2 incorporates a noise generation constraint at the training phase, enabling the generation of clips directly from random noise, similar to traditional GANs. Consequently, AR2 initiates video generation by using noise to produce the first clip, and then auto-regressively predicts each subsequent clip based on the previously generated one. These generated clips are then sequentially stitched to form a long video.

Further modifications include the modification of the Generator to enhance spatio-temporal feature learning ability, as well as the use of a Video Discriminator (VD) at the generation stage to filter and refine the generated clips for improved performance and for ensuring video continuity.

5.3.1 Motivation for AR2

While REncGAN (R2) successfully extended the EncGAN3 model for long video generation, it still presents significant limitations, particularly in its reliance on prior information and its tendency to generate repetitive sequences. The R2 model, with its recall mechanism, models intra-clip relationships using the original EncGAN3 structure and introduces inter-clip relationship modelling through a Markov chain-based approach. This allowed R2 to generate hundreds or even thousands of frames, overcoming the limitations of short video generation.

However, R2 also has notable constraints. One of the primary issues is that R2's video generation is heavily conditioned on the training data, requiring prior information from training video clips to provide the corresponding latent spaces. This reliance on prior information

significantly limits the diversity and length of generated videos. As shown in the ablation studies that we pursued on latent code manipulation, although videos generated from the same latent space exhibit some diversity, the extent of this diversity remains constrained by the model's reliance on specific latent spaces. Additionally, R2 struggles to generate videos beyond the length of the training data, which hinders its potential for truly long-duration, unconditional video generation.

Another limitation arises from the fact that the inter-clip relationship modelled by the recall mechanism is inherently local. This Markov chain-based approach allows the current clip to depend only on the preceding clip, which leads to the issue of generating repetitive sequences over long video durations. Since the model does not account for global video features, it struggles to maintain coherence and lacks in providing novel information across long sequences of clips.

To address these limitations, we propose AR2 (Auto-Regressive REncGAN), a novel approach designed to reduce dependency on prior information and enable unconditional long video generation. AR2 enhances the ability of the model to process temporal dependencies over extended sequences, thereby addressing the repetition problem and improving the diversity of generated videos. By eliminating the need for explicit training clips as priors, AR2 offers a more flexible and scalable solution for long-duration video generation.

In the following sections, we introduce the AR2 structure, training methodology, and implementation details, demonstrating how it overcomes the challenges faced by R2.

5.3.2 AR2 Structure

AR2 builds upon the foundational architecture of improved R2, as described in Section 5.2, aiming to advance the ability of the model to generate long videos without relying on prior real video information. This section explores the structural innovations that AR2 incorporates to achieve this goal, focusing on the integration of an auto-regressive recall mechanism and adaption on the VDrej operation. These changes collectively enable AR2 to generate video sequences without conditioning on prior clips, thereby achieving true unconditional generation.

Auto-regressive recall mechanism. As shown in Figure 5.2, AR2 retains the R_Enc and R_3VD operations of REncGAN but adapts them for an auto-regressive setting. R_Enc and R_3VD are core components of the original recall mechanism in REncGAN. By maintaining the two core operations, AR2 inherits Markov chain based inter-clip relationship modelling, which allows the model to generate long videos of arbitrary lengths from short training clips. Since the training video length is fixed as the short clip length, this approach mitigates the increase in GPU memory requirements typically associated with increasing the length of the training videos.

The R_Enc and R_3VD operations, along with their modifications, are illustrated in the top and bottom green dashed boxes in Figure 5.2. For R_Enc, similar to REncGAN, the long video is divided into overlapping clips, where each pair of adjacent clips forms an input pair, as shown in the left top green dashed box in Figure 5.2. For example, Clip 1 and Clip 2 form one input pair, and Clip 2 and Clip 3 form another. In AR2, only the "former" clip

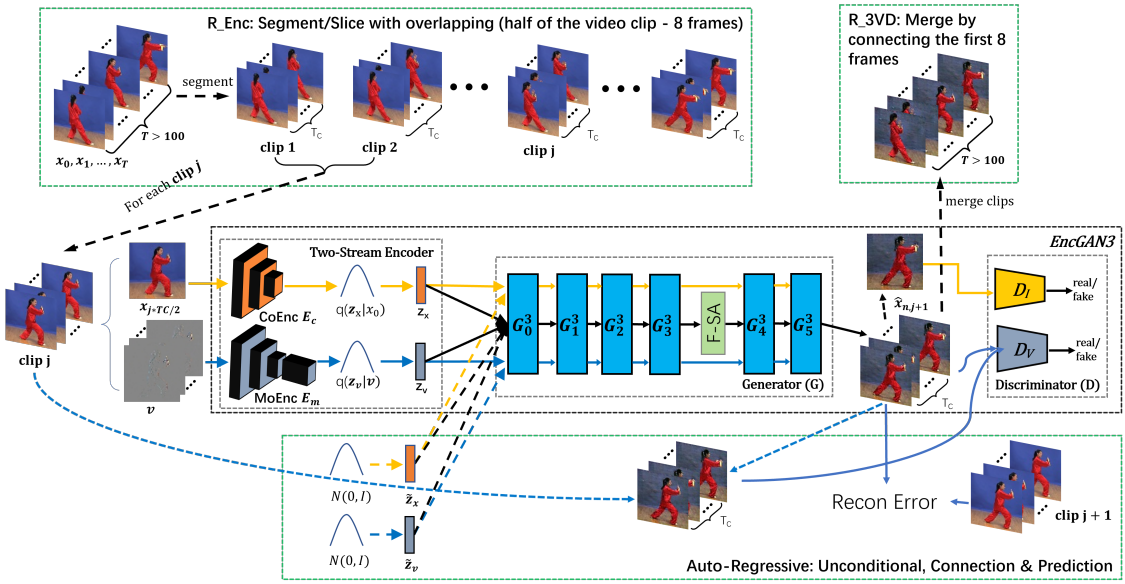


Figure 5.2: Illustration of the Auto-Regressive Recall Encoding GAN3 (AR2).

(e.g., Clip 1 in the pair) is fed into the internal EncGAN3 model, whereas in REncGAN, both clips in the pair were used as inputs. The "latter" clip (e.g., Clip 2 in the pair) is utilized to evaluate the continuation of the video by considering the sequence of the two segmented clips, enabling the prediction of the long video, as shown in the right part of the bottom green dashed box in Figure 5.2. This reconstruction error is calculated using Smooth L1 loss, which combines the benefits of both L1 loss (absolute error) and L2 loss (squared error), offering robustness to outliers while maintaining smoothness.

Such modifications as those detailed above in R_Enc allow AR2 to perform longer video predictions while preserving the Markov chain property between clips. As a result, AR2 can generate long videos by recursively predicting subsequent clips, starting with an initial clip during the video generation stage, while maintaining the training length as a clip length considered of 16 frames in the experiments. Despite reducing the reliance on prior information from the entire video to just an initial clip, AR2 still requires to be provided with an initial clip as input, as the starting frames. To achieve unconditional generation, where the process begins from random noise, AR2 introduces an additional step. It considers noise sampled from a standard Gaussian distribution as input to the Generator, enabling the creation of the initial clip directly from random noise, as shown in the left part of the bottom green dashed box in Figure 5.2.

In conjunction with these changes, the merging process in R_3VD is adapted to align with the new structure of R_Enc. Unlike in REncGAN, where both clips from the input pair were used to generate the stitched clip, AR2 no longer utilizes the latter clip in the generation process. Instead, R_3VD in AR2 constructs the stitched clip by concatenating the first 8 frames of the former clip input with the first 8 frames of its corresponding generated clip, as shown in the middle part of the bottom green dashed box in Figure 5.2. This approach leverages the VD to ensure continuity and address blending issues between clips, preserving the original functionality of R_3VD within the auto-regressive setting in AR2 and hence could be connected as shown in the right top green dashed box in Figure 5.2.

The modifications in both R_Enc and R_3VD enable AR2 to generate long videos without relying on prior information, while maintaining temporal consistency. AR2 starts the video generation process by using noise to generate the initial clip and then recursively predict subsequent clips, facilitating unconditional video generation from scratch. This approach not only preserves the memory efficiency advantages of REncGAN but also extends its capability to produce long, coherent video sequences without requiring real video data. In fact, the memory efficiency of AR2 is closer to that of EncGAN3 than REncGAN. This is because AR2 restores the noise generation pathway removed in REncGAN, which only requires the Generator and Discriminator, bypassing the Encoder entirely. Unlike REncGAN, which processes two clips simultaneously and requires additional memory for handling the second clip through the Encoder, AR2 processes only one clip at a time and incorporates the noise generation function, which does not involve the Encoder.

Additionally, AR2, like REncGAN, processes a stitched clip through VD to ensure temporal consistency. However, since AR2 no longer requires simultaneous processing of two clips and does not need to run a second clip through the Encoder, its overall GPU memory requirements are lower than those of REncGAN. Consequently, the GPU memory requirement to train AR2 is much closer to that of EncGAN3, making AR2 more memory-efficient than REncGAN for long video generation tasks.

VDrej in AR2: Adaptations for Auto-Regressive Generation. While the VDrej operation in improved R2 focuses on improving clip quality and connectivity during sequential clip generation, AR2 adapts VDrej to better suit its auto-regressive generation process. In AR2, the generated clip selected by VDrej operation is not only used for connectivity assessment but also as input to predict the next clip, enabling the VDrej operation to mitigate the error occurred at each prediction step. Therefore, VDrej could alleviate the quality degradation errors accumulated through the auto-regressive process, preserving the desirable temporal dynamics while reducing the impact of prediction errors on clip quality.

5.3.3 Training Objective

After implementing the auto-regressive setting, the training objectives of AR2 differ from those of REncGAN. The differences result from AR2 employing the random noise generation terms and removing the generator process of the latter clip. Similar to REncGAN, AR2 optimizes the encoder and generator jointly to enhance their cooperation. AR2 follows the same process as R2 for generating clips, as shown in Equations (4.2) and (4.3), and connects them as described in Equation (4.1). The binding frame is considered as in R2 as the middle frame of a generated small clip, which corresponds to the index $r = 8$ in the experiments.

The training loss function for jointly optimizing the Encoder and Generator is given by :

$$\begin{aligned} L_{EncG,AR} = & \sum_{i=1}^{N_L}\sum_{j_c=1}^{N_C} \|\mathbf{x}_{i,j_c+1,r} - \widehat{\mathbf{x}}_{i,j_c,r}\| + \sum_{i=1}^{N_L}\sum_{j_c=1}^{N_C}\sum_{j=0}^{T_c} \|\mathbf{x}_{i,j_c+1,j} - \widehat{\mathbf{x}}_{i,j_c,j}(\widehat{\mathbf{v}}_{i,j_c,j}, \widehat{\mathbf{x}}_{i,j_c,r})\| \\ & + D_{KL}(q_{\theta_{\mathbf{x}}}(\mathbf{z}_{\mathbf{x}} \mid \mathbf{x}) \| p(\mathbf{z}_{\mathbf{x}})) + D_{KL}(q_{\theta_{\mathbf{v}}}(\mathbf{z}_{\mathbf{v}} \mid \mathbf{v}) \| p(\mathbf{z}_{\mathbf{v}})) \\ & - \mathbb{E}_{\mathbf{z}_{\mathbf{x}} \sim q_{\theta_{\mathbf{x}}}(\mathbf{z}_{\mathbf{x}}|\mathbf{x}), \mathbf{z}_{\mathbf{v}} \sim q_{\theta_{\mathbf{v}}}(\mathbf{z}_{\mathbf{v}}|\mathbf{v})} \log[D(G(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}}))] - \mathbb{E}_{\widehat{\mathbf{x}}_n \sim G(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{v}})} \log[D(\widehat{\mathbf{x}}_n)] \\ & - \mathbb{E}_{\tilde{\mathbf{z}}_{\mathbf{x}} \sim \mathcal{N}(0,\mathbf{I}), \tilde{\mathbf{z}}_{\mathbf{v}} \sim \mathcal{N}(0,\mathbf{I})} \log[D(G(\tilde{\mathbf{z}}_{\mathbf{x}}, \tilde{\mathbf{z}}_{\mathbf{v}}))] - \mathbb{E}_{\tilde{\mathbf{x}}_n \sim G(\tilde{\mathbf{z}}_{\mathbf{x}}, \tilde{\mathbf{z}}_{\mathbf{v}})} \log[D(\tilde{\mathbf{x}}_n)] \end{aligned} \tag{5.1}$$

where considering N_L long videos, each split into N_C overlapped clips with each clip containing $T_c + 1$ frames with r identifying index of the binding frame of each clip. The clip amount N_C is varied for each long video, depending on their lengths. Frames $\widehat{\mathbf{x}}_{i,j_c,j}$, including $j = r$, are generated from $\mathbf{x}_{i,j_c,j}$ of clip j_c in video i. In the first two loss terms on the right side of the equal sign, these generated frames are used to compute the reconstruction error with frames $\mathbf{x}_{i,j_c+1,r}$ of clip $j_c + 1$ in video i, to enable the prediction function. In contrast, frames $\tilde{\mathbf{x}}_{i,j_c,j}$ in the last two terms on the right side of the equation are generated from random noise, to enable the function of generating clips from random noise enabled by the generator and discriminator modules. These terms deriving from using random noise enables the synthesis of diverse sections of the long video, whose viability is confirmed by the video discriminator VD in both image frames as well as in the latent space domain. The middle four loss terms take similar role as the terms from Equation (4.4).

Due to the reintroducing of the noise generation term characteristic to GAN generators, the training objective of the image-stream Discriminator in AR2 is different from the one used in R2 ($L_{D_I,R}$ in Equation (4.5)) but the same as the one used in EncGAN3 (L_{D_I} in Equation (3.5)). Meanwhile, the training objective of the video-stream Discriminator in AR2, named as $L_{D_V,AR}$, builds upon the one used in R2 while introducing the auto-regressive setting and noise generation term, defined as :

$$\begin{aligned} L_{D_V,AR} = & \mathbb{E}_{\mathbf{x}_{0:T_c} \sim p(\mathbf{x}_{0:T_c})} \log[D(\mathbf{x}_{0:T_c})] + \mathbb{E}_{\widehat{\mathbf{x}}_{0:T_c} \sim p(\widehat{\mathbf{x}}_{0:T_c})} \log[1 - D(\widehat{\mathbf{x}}_{0:T_c})] \\ & + \mathbb{E}_{\widehat{\mathbf{x}}_{stitch} \sim p(\widehat{\mathbf{x}}_{stitch})} \log[1 - D(\widehat{\mathbf{x}}_{stitch})] \\ & + \mathbb{E}_{\tilde{\mathbf{z}}_{\mathbf{x}} \sim \mathcal{N}(0,\mathbf{I}), \tilde{\mathbf{z}}_{\mathbf{v}} \sim \mathcal{N}(0,\mathbf{I})} \log[1 - D(G(\tilde{\mathbf{z}}_{\mathbf{x}}, \tilde{\mathbf{z}}_{\mathbf{v}})] \end{aligned} \tag{5.2}$$

where the stitched clip $\widehat{\mathbf{x}}_{stitch}$ consists of frames from input clip j_c and its generated clip. The last loss term provides learning signals for noise generation ability.

5.3.4 Training and Inference

During the training, the Discriminator is updated by optimizing L_{D_I} and L_{D_V} using equations Equation (3.5) and (5.2). Then, based on the optimized Discriminator, the Encoder and Generator loss L_{EncG} are optimized jointly according to Equation (5.1). We show the Training and Inference procedure of AR2 as pseudo code on Algorithm 9 and 10, respectively.

Algorithm 9 AR2 Training Procedure

Required: Encoder (Enc), Generator (G), Image Discriminator (ID), and Video Discriminator (VD). Operations of concatenation (concat), computing reconstruction error (recon), random sample (sample) and average (avg)
Input: Adjacent clips $\mathbf{x}_{j_c,0:T_c}$ and $\mathbf{x}_{j_c+1,0:T_c}$ of clip length T_c, Random Noise $\tilde{\mathbf{z}}$
Output: Discrimination results of True or False T/F

$T_c \leftarrow 15$
$r \leftarrow T_c//2 + 1$

Generation:
$\mathbf{z} \leftarrow \text{sample}(\text{Enc}(x_{j_c,0:T_c}))$
$\widehat{\mathbf{x}}_{j_c,0:T_c} \leftarrow \text{G}(\mathbf{z})$
$\tilde{\mathbf{x}}_{j_c,0:T_c} \leftarrow \text{G}(\tilde{\mathbf{z}})$
$recon\_error \leftarrow \text{recon}(\mathbf{x}_{j_c+1,0:T_c} - \widehat{\mathbf{x}}_{j_c,0:T_c})$

Stitch Clip:
$\widehat{\mathbf{x}}_{stitch} \leftarrow \text{concat}(\mathbf{x}_{j_c,0:r}, \widehat{\mathbf{x}}_{j_c,0:r})$

Discrimination:
$T/F \leftarrow \text{avg}(\text{ID}(\text{sample}(\widehat{\mathbf{x}}_{j_c,0:T_c})), \mathbf{ID}(\text{sample}(\tilde{\mathbf{x}}_{j_c,0:T_c})))$
$T/F \leftarrow \text{avg}(\text{VD}(\mathbf{x}_{j_c,0:T_c}), \text{VD}(\widehat{\mathbf{x}}_{j,0:T_c}), \text{VD}(\widehat{\mathbf{x}}_{stitch}), \text{VD}(\tilde{\mathbf{x}}_{j_c,0:T_c}))$

Algorithm 10 AR2 Inference Procedure

Required: Encoder (Enc), Generator (G) and Video Discriminator (VD). Operations of the random sample (sample), stitch clips (stitch) and VDrej shown in Algorithm 8.
Input: Normal distribution $\mathcal{N}(0, \mathbf{I})$, generated clip number (num).
Output: Generated video $\widehat{\mathbf{x}}_{0:T}$

$T_c = 15$

$\widehat{\mathbf{x}}_{1,0:T_c} \leftarrow \text{VDrej}(\text{G}(\text{sample}(\mathcal{N}(0, \mathbf{I}))))$
$\text{prev\_clip} \leftarrow \widehat{\mathbf{x}}_{1,0:T_c//2+1}$
for $j_c = 1 \rightarrow num$ **do**
    $\widehat{\mathbf{x}}_{j_c+1,0:T_c} \leftarrow \text{VDrej}(\text{G}(\text{sample}(\text{Enc}(\widehat{\mathbf{x}}_{j_c,0:T_c}))), \text{prev\_clip})$
end for
$\widehat{\mathbf{x}}_{0:T} \leftarrow \text{stitch}(\widehat{\mathbf{x}}_{j_c,0:T_c}$ for j_c from 1 to $num)$

5.3.5 AR2 Implementation

For the structure details, AR2 is based on the improved R2 without altering the internal structure. Compared to the structure used by EncGAN3 when generating 128×128 pixel resolution frames, the Generator in AR2 also includes five G3 blocks and one F-SA module. However, since the improved R2 enhances model performance by replacing 1D+2D convolutions with 3D convolutions in the video stream of the Generator, each G3 block in AR2 contains one 3D convolution, one 2D convolution, and one 1D convolution, instead of the

original two 2D and two 1D convolutions. The F-SA module remains unchanged, as do the other Encoder and Discriminator modules. The Hyper-parameters such as the learning rate and used optimizer are the same as those used in REncGAN.

Computation cost. At training time, the training video data is split into several batches. The model optimizes its trainable parameters once based on the learning signals from one batch instead of one video input, saving time. The model after optimized on one batch is called one iteration. Train for one epoch means to feed the entire training set once to the model.

AR2 is trained on Taichi and Sky datasets. For the Taichi dataset, in practice, AR2 is trained on 3 A40 GPUs with 48 GB, which costs about 64 hours to train for 5000 epochs. The GPU memory of A40 allows a batch size of 10. With such a batch size, the training set produces 10 batches. When batching video data, it does not drop videos that are not enough to form a batch. Here, train one epoch means the model is optimized for 10 iterations. For the Sky dataset, in practice, AR2 is trained on 2 H100 GPUs with 80 GB, which costs about 72 hours to train for 75 epochs. The Sky database has been pre-divided into training sets and test sets. AR2 is trained on the train set provided in the Sky dataset. The GPU memory of H100 allows a batch size of 55. With such a batch size, the used training set could produce 1284 batches and also not drop out videos that are not enough to form a batch.

Due to the usage of adversarial loss, the loss curve is unstable and not useful for finding out the best training epoch. To find out the best version of the model parameters, we calculate the FVD sores based on sequences of both 16 frames and 128 frames sampled from the generated videos, to measure the generation quality in both short and long duration. To reduce fluctuations of the score, we sample the initial certain frames with a sampling step as 1.

For the Taichi dataset, we calculate the FVD sores for every 10 epochs, which is about 100 iterations. The best FVD results appeared around 550 to 700 epochs, about 7000 iterations maximum. Hence, the training time cost to reach the best performance of AR2 on the Taichi dataset should be $64 \times 3 \times (700 \div 5000) \approx 27$ hours if trained on one A40 GPU. For the Sky training set, we compute the FVD scores for every 1 epoch, which is 1284 iterations. The best FVD results appeared around 5 to 13 epochs, about $13 \times 1284 = 16692$ iterations maximum. Hence, the training time cost to reach the best performance of AR2 when training on the Sky training set should be $72 \times 2 \times (13 \div 75) \approx 25$ hours if trained on one H100 GPU.

	AR2	Time unit	GPU type (num=1)	batch size	batch num
Taichi	27	hours	A40	10	10
Sky	25	hours	H100	55	1284

Table 5.1: Training time cost of AR2.

Video generation time. We generate long videos and then compute the FVD. Each time, we generate 64 long videos, with each video of 1000 frames in length. In practice, after training on the Sky dataset, it takes 87.5 minutes to generate videos in this way 8 times and we compute FVD 29 times. Each time we compute FVD for sequences of 16 frames and 128 frames. Roughly, regardless of the time cost for computing the FVD score, the generation of one 1000-frame video takes about $87.5 \div (8 \times 64) \approx 0.17$ minutes, about 10 seconds. In

addition, applying the VDrej mechanism would require more time for the video generation. Because VDrej generates a certain number (such as 20) of clips from each latent space and uses VD to pick up the best one. In practice, after training on the Taichi dataset, it takes 309.5 minutes to generate 64 long videos of 1000 frames with using the VDrej mechanism 9 times and compute FVD 500 times. Roughly, regardless time cost of computing FVD, the generation of one 1000-frame video takes about $309 \div (8 \times 64) \approx 0.6$ minute, about 36 seconds.

5.4 R3: REncGAN with GPT Directs Long Duration

5.4.1 Motivation for R3

The development of R3 is primarily motivated by the need to address the limitations of R2, specifically its reliance on prior information and lack of global perspective of entire long video sequences. In R2, the VAE loss constrains the latent spaces of different clips to approximate a standard normal distribution $\mathcal{N}(0, \mathbf{I})$, which makes those latent spaces similar. However, the recall mechanism in R2 ensures that the latent space of each clip encodes specific temporal information. This approach intentionally introduces subtle differences between the latent spaces of different clips, which makes those latent spaces to be different. These differences allow the sequence of these latent spaces to carry the necessary temporal information for generating extended videos. The trade-off, however, is that this approach inherently depends on the availability of training videos. To address this dependency, R3 incorporates a GPT module that learns long-duration temporal information directly from sequences of latent spaces [16, 17, 18, 19]. This GPT module generates the latent spaces required for producing long videos, effectively eliminating the need for training data.

The internal video representation in R3 is structured to address different temporal relationships: non-Markov chain based intra-clip relationships within each clip, Markov chain based inter-clip relationships between adjacent clips, and long-duration features that span and integrate all clips within the entire video. The non-Markov chain based intra-clip relationships, which model the connections between frames within a clip, are managed by EncGAN3. This non-Markov chain based representation way is commonly used in video generation models, particularly for short video generation tasks. Though provide good generation results, it is inefficiency in temporal dimension, represented as the GPU memory requirements exponentially increased by the training length. The inter-clip relationships, modelled using a Markov chain approach, are handled by the original recall mechanism in R2. This Markov chain-based representation is a key aspect of R2, enabling memory-efficient generation of long videos with arbitrary lengths from a fixed short training length. Finally, the long-duration features that span all clips within a video are captured by the GPT module. This representation is crucial in R3, as it provides the global perspective that R2 lacks. The first two types of relationships are learned from pixel-wise information to capture finer details, while the long-duration feature modelling by GPT uses latent-wise information to save computational costs.

Hence, R3 retains the advantages of R2 while addressing its limitations. The key advantage of R2 is its efficient representation of long videos in the temporal dimension, demonstrated by

its lower GPU memory requirements during training for long video generation. However, the reliance of R2 on prior information stems from its focus on local features due to the Markov chain-based inter-clip relationship modelling, which limits model from having a global perspective of the entire video. By integrating GPT, R3 overcomes this limitation, allowing for both efficient internal representation and a more comprehensive global understanding of the video content.

5.4.2 R3 Structure

As illustrated in Figure 5.3, the R3 model consists of improved R2 (as described in Section 5.2) and a GPT module [17]. The improved R2 model, which incorporates enhancements such as the additional G^3 block and VDrej, serves as the foundation for R3. The training of R3 involves two main steps. First, improved R2 is trained according to the procedures outlined in Section 4.4.3 of the previous chapter, same as the original R2. Once improved R2 is trained, its Encoder (Enc) is used to encode the training videos into sequences of latent spaces. These sequences are then fed into the GPT module, which is trained to learn sequences of video segments representing long-duration information. This allows R3 to generate sequences of latent spaces using GPT, rather than relying on training videos, as improved R2 does. As a result, R3 reduces the dependency on prior information, and the maximum length of the generated videos is no longer constrained by the training data.

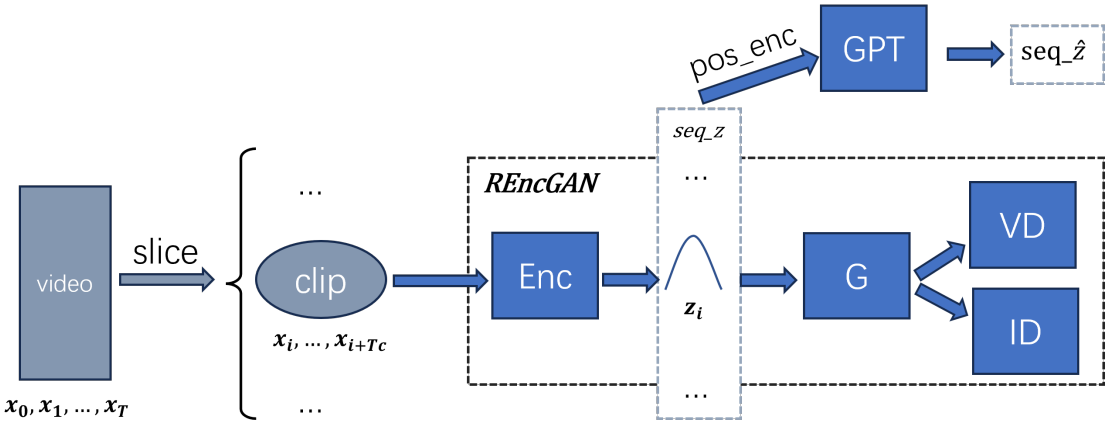


Figure 5.3: Illustration of the R3. z_i indicates a latent space and seq_z is the sequence of latent spaces. T is video length while T_c is clip length. The former length is arbitrary but longer than 100 frames at least, such as 500, 1000 frames. The latter length is fixed to 16 frames.

The GPT module in R3 is adapted from the original GPT used in natural language processing, where it is trained on discrete data (i.e., text). In that context, GPT maps each discrete word to a continuous vector, known as a token, through a process called tokenization. The reverse operation, mapping the vector back to the word, is called de-tokenization. In R3, since the data consists of continuous latent spaces rather than discrete tokens, the tokenization and de-tokenization processes are removed. Instead, each latent space is treated as a token, with its size corresponding to the size of the latent space. The sequences of latent spaces, after positional encoding (denoted as pos_enc in Figure 5.3), are fed directly into

the GPT module. The positional encoding is crucial because GPT cannot learn sequential information directly from the sequence of input tokens unless the position information is embedded within the tokens themselves.

During training, R3 computes the reconstruction error between the input and output sequences of latent spaces using the Smooth L1 loss function, similar to EncGAN3, R2, improved R2 or AR2. However, in contrast to those models, where the reconstruction error is calculated on pixel-wise clips, R3 applies this error to sequences of latent spaces. To manage sequences of varying lengths, R3 pads all sequences to a uniform length. This allows the GPT module to process a batch of sequences simultaneously, enhancing training efficiency and reliability. R3 identifies the maximum sequence length in the training set and pads all shorter sequences to this length by adding zeroes. A mask within GPT is then used to ignore these padding values, ensuring that the module focuses only on the actual latent spaces. During training, the GPT module is trained independently, with the parameters of the Encoder and other improved R2 modules frozen.

At inference time, R3 utilizes all modules except the Image Discriminator to refine the latent spaces generated by GPT. This helps to reduce accumulated errors through an auto-regressive process, thereby maintaining the quality of the generated clips. As described in Algorithm 11, R3 begins by encoding an initial clip into a latent space z, which is then used as input for GPT to predict the next latent space. The predicted latent space generates several latent codes, each of which is used to produce a clip via the Generator. Each of these generated clips is stitched with the previous clip, starting with the initial clip. The merging consists in stitching together the first 8 frames of the generated clip with the first 8 frames of the previous clip. The VD then evaluates the quality of these stitched clips, selecting the best one based on both clip quality and video continuity. This selection process, known as VDrej, is discussed in improved R2 (introduced in Section 5.2). Unlike the VDrej in improved R2, where the best clip is selected for continuity only, in R3, the best clip is then encoded into a latent space, which along with the previous GPT input, is used to predict the next latent space, continuing the process. This process enables R3 to generate long videos with improved temporal coherence, less quality degradation and reduced dependency on prior information.

5.4.3 Training and Inference

In brief, R3 is trained in two steps. First, R3 trains inside improved R2 component. Secondly, R3 trains the GPT module based on latent spaces encoded from clips, using the Encoder from the trained improved R2. The training objectives and procedure of improved R2 are the same as the original R2, which is described in Section 4.4.2 and 4.4.3 of the previous chapter. For GPT, the training focuses on minimizing the difference between the input and output latent space sequences. The training objective for GPT L_{GPT} is defined as :

$$L_{GPT} = \sum_{i=1}^{N_L} \sum_{j_c=1}^{N_C} \|\mathbf{z}_{ij_c} - \widehat{\mathbf{z}}_{ij_c}\| \tag{5.3}$$

where $\mathbf{z}_{ij_c}$ represents the j_c-th latent space in the i-th sequence, and $\widehat{\mathbf{z}}_{ij_c}$ is the latent space generated by the GPT module. The training procedure of GPT is as usual GPT, where inputting a sequence of latent spaces and output a sequence of latent spaces where contains

the prediction of the next latent space. The inference procedure of R3 is shown in Algorithm 11. The details is described in the last paragraph in the last sub-section.

Algorithm 11 R3 Inference Procedure

Required: GPT, Encoder (Enc), Generator (G) and Video Discriminator (VD). Operations of getting the last token from a sequence (last), sample, concatenation (concat) and VDrej
Input: an initial clip ($\mathbf{x}_{0:T_c}$), numbers of generated clips (num)
Output: the generated long video (gen_vid)

$T_c = 15$
$r = T_c//2 + 1$

$z_0 \leftarrow \text{Enc}(\mathbf{x}_{0:T_c})$
$input\_toks \leftarrow z_0$
$gen\_vid \leftarrow \mathbf{x}_{0:r}$
$prev\_clip \leftarrow \mathbf{x}_{0:r}$
for $j_c = 1 \rightarrow num$ **do**
    $gen\_toks \leftarrow \text{GPT}(input\_toks)$
    $z_{j_c} \leftarrow \text{last}(gen\_toks)$
    $\widehat{\mathbf{x}}_{j_c,0:T_c} \leftarrow \text{VDrej}(\text{G}(\text{sample}(z_{j_c})), prev\_clip)$
    $gen\_vid \leftarrow \text{concat}(gen\_vid, \widehat{\mathbf{x}}_{j_c,0:r})$
    $prev\_clip \leftarrow \widehat{\mathbf{x}}_{j_c,0:r}$
    $z_{j_c} \leftarrow \text{Enc}(\widehat{\mathbf{x}}_{j_c,0:T_c})$
    $input\_toks \leftarrow \text{concat}(input\_toks, z_{j_c})$
end for

5.4.4 R3 Implementation

Structure details. R3 consists of two main components: improved R2 and an additional GPT module. The key parts of improved R2 are the modifications of the Generator and the VDrej operation. Similar to AR2, the modification to the Generator in the improved R2 compoent are retained in R3 but the VDrej operation is adapted for R3. The retained modifications includes the replacement of 1D and 2D convolutions with 3D convolutions and the ability to generate video frames at a resolution of 128×128 pixels. While the VDrej in R3 includes extra usage of the selected clip for predicting the next clip, as described in the last paragraph of Section 5.4.2.

The GPT module integrated into R3 is composed of 48 transformer blocks. Each of these blocks contains multi-head attention with 24 heads, and each token is represented as a vector with a dimension of 1536. The GPT module is based on the Transformer decoder architecture but is simplified, with each transformer block including only one multi-head self-attention mechanism and two fully connected (fc) layers, removing the cross-attention component as not required in the auto-regressive setting. The length of the input sequence for GPT is determined by the longest sequence present in the training video set, corresponding to the longest video from the dataset used for training.

In terms of the size of the GPT module, the GPT used in R3 is comparable to the GPT-2 [16] and significantly larger than GPT-1 [16]. GPT-1 features 12 transformer blocks, each with 12 heads in the multi-head attention, and token vectors with a dimension of 768. These parameters are notably smaller than those used in the GPT module of R3. On the other hand, GPT-2 consists of 48 transformer blocks, each with 12 heads in the multi-head attention, and token vectors with a dimension of 1600. Compared to GPT-2, the GPT module in R3 increases the number of heads to 24 to enhance the reconstruction of continuous variables. This adjustment is crucial as generating continuous variables demands greater sensitivity to the values within each dimension of the variable. By analysing token relationships from multiple perspectives, the GPT in R3 is better equipped to generate accurate values across all dimensions of the variables. Since R3 directly uses the latent space as tokens, the dimension of the GPT tokens in R3 is determined by the latent space, which is smaller than the token dimension used in GPT-2.

Training setting. R3 uses a two-step training. The first step is training the improved R2, following the settings of the original R2 introduced in Section 4.4.5. The second step consists in the training of the GPT. The training of improved R2 implements the ADAM optimizer with exponential decay rates for the first-order and second-order moment estimations set to $\beta_1 = 0.5$ and $\beta_2 = 0.999$ [94]. The GPT model is optimized using the AdamW optimizer, which is a variant of the Adam optimizer that includes weight decay regularization that is more generally used in Transformer training [102]. This optimizer uses the exponential decay rates for the first-order and second-order moment estimations are configured to $\beta_1 = 0.9$ and $\beta_2 = 0.95$. Due to their sensitivity to the gradient updates of the Transformer, the learning rate of GPT is 5×10^{-5}, which is smaller than the one used in R2 which was considered as 2×10^{-4}. As the input and output of the GPT used in R3 are continuous variables, R3 uses the Smooth L1 loss to compute the reconstruction error between its input and output instead of the cross entropy loss used for GPT in natural language tasks. The former loss is suitable for regression tasks and the latter one is for multi-class classification tasks.

Computation cost. R3 is trained on Taichi and Sky datasets. The training of R3 is split into two steps: the training of R2 and then, GPT. Details of the definitions such as epoch or FVD are described when introducing the implementation of AR2.

For the Taichi dataset, the training of R2 takes about 45.5 hours for 3000 epochs on 3 A40 GPUs. We consider a batch size of 25 and split the training set into 12 batches. In all experiments, we also consider for the training the video segments that are left over after splitting the original video into batches. By training R2 between 3000 epochs to 5000 epochs takes 13 hours on 2 H100 GPUs. For the used batch size of 50, the training set is split into 9 batches. The best FVD was obtained at 3970 epochs. Thus, the required training time cost to reach the best performance is about $45.5 \times 3 = 136.5$ hours on single A40 and $13 \times 2 \times (970 \div 2000) \approx 12.6$ hours on single H100. The training of GPT takes 29 hours for 10,000 epochs on 2 H100 GPUs. We consider a batch size of 30 and split the training set into 1 batch. The best FVD was obtained at 4800 epochs. Thus, the training time required to reach the best performance was about $29 \times 2 \times (4800 \div 10000) \approx 27.8$ hours on a single H100.

For the Sky training set, the training of R2 costs about 12 days on 3 A40 and 6 days hours on 2 H100 for 234 epochs, which is much longer than the cost training on the Taichi dataset. For the used batch size of 25 on A40, the training set is split into 1883 batches, where we

Model	GPU Configuration	Batch Size/GPU	Batch Num	Single GPU Time
R3's R2	2 × H100 (80GB) / (3 × A40 + 2 × H100)	50 / (25, 50)	12 / (1883+1412)	12.6h / (36+3.5)d
R3's GPT	2 × H100 (80GB)	30 / 5	1 / 187	27.8h / 21d

Table 5.2: Time cost of R3 for training from scratch. Values separated by "/" represent results on different datasets Taichi and Sky, like Taichi/Sky.

filter out videos with less than 24 frames to match the training requirement of R2. For the used batch size of 50 on H100, the training set is split into 1412 batches. The best FVD was obtained from 215 epochs. Thus, the training time cost to reach the best performance is about 36 days on a single A40 plus 3.5 days on a single H100. The training of GPT takes 289.5 hours (a bit over 12 days) for 1470 epochs on 2 H100 GPUs. We consider a batch size of 5 and split the training set into 187 batches. The best FVD was obtained at 1290 epochs. Thus, the training time required to reach the best performance was about $289.5 \times 2 \times (1290 \div 1470) \approx 508.1$ hours (21 days) on a single H100. In addition, the Sky dataset contains some extremely long videos (3000 or even over 4000 frames). The training sequence length depends on the maximum sequence length, although most videos do not require such a long length. This significantly impacts GPU memory requirements, as the model size is also affected by the input sequence length, potentially leading to GPU memory overflow when attempting to process the maximum length. Therefore, we set a threshold for the maximum training sequence length at 1000 frames, and for videos exceeding this length, we discard the excess frames.

Model Size and Storage Requirements. The R3 model consists of two major components: R2 and GPT. The total number of parameters in R3 is approximately 1.49 billion, with R2 contributing 128.7 million parameters and GPT contributing 1.36 billion parameters.

Each parameter in the model is stored as a 32-bit floating point number (float32), which requires 4 bytes of memory. Thus, the storage memory required for a model can be computed as follows:

$$\text{Memory (GB)} = \frac{\text{Number of Parameters} \times 4 \text{ (bytes)}}{1024^3} \tag{5.4}$$

In addition to the raw parameter storage, the final model size in .pth format may slightly vary due to factors such as structural indexing and compression differences across different datasets.

The detailed breakdown of parameter distribution and memory usage is summarized in Table 5.3.

5.5 Experimental Results

This section presents the experimental results for both qualitative and quantitative evaluations. We have applied our models, when considering different types of the recall mechanisms, on the long term video generation, after training on the Taichi and Sky datasets, the current generally used datasets for this task, as introduced in Section 4.6.1. To match the setting of our AR2 and R3, we use the 24-segmented versions of both datasets, as shown in Table 5.4.

Model Component	Parameter Count	Storage Memory (GB)
R2 Sub-components		
Encoder (Enc)	2.06M	0.008 GB
Generator (G)	90.37M	0.35 GB
Variational Decoder (VD)	29.32M	0.11 GB
Identity Module (ID)	6.96M	0.03 GB
Total R2	**128.71M**	**0.50 GB**
GPT Component		
GPT	**1.36B**	**5.08 GB (pure)**
Total R3	**1.49B**	**5.84 - 6.03 GB**

Table 5.3: Model size and storage memory of R3. The total memory usage includes minor overhead from structural indexing.

Dataset	Segments	Train	Test
Taichi	300	230	70
Sky	47,075	42,075	4,800

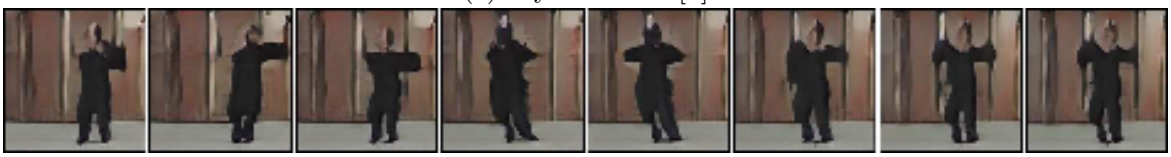
Table 5.4: Video dataset statistics of their 24-frame segmented versions. The amount of Segments is approximate.

For the qualitative evaluation, we show frames sampled at specific intervals to represent longer durations within a shorter frame count. For quantitative evaluation, we use FVD as the metric to measure generated videos of 16-frame and 128-frame lengths, denoted as FVD-16f and FVD-128f, respectively. These metrics assess the quality of short and long-duration video sequences. We also compute the ratio of FVD between these two lengths to quantify the degree of quality degradation as video duration increases.

(a) MoCoGAN-HD [6]

(b) DIGAN [7]

(c) StyleGAN-V [8]



(c) Long-Video-GAN [10]

(d) StyleInV [11]

(e) our improved R2

(f) our AR2

(g) our R3

Figure 5.4: Taichi frames sampled from generated videos of 128 frames each. For comparison of our recall-based methods, improved R2, AR2 and R3, with other long video generation methods. Frames in each row are sampled starting from the first frame, with one frame sampled every 16 frames, representing a video of 128 frames in total. The results of MoCoGAN-HD, DIGAN, StyleGAN-V, Long-Video-GAN and StyleInV are from [11] at 256×256 resolution while our recall-based methods are at 128×128 resolution.

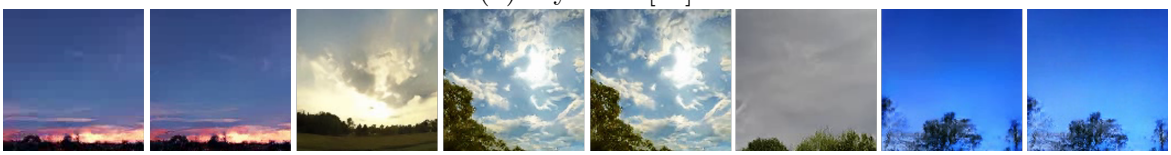
(a) MoCoGAN-HD [6]

(b) DIGAN [7]

(c) StyleGAN-V [8]

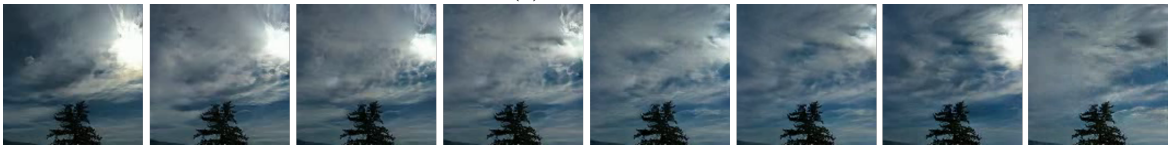
(c) Long-Video-GAN [10]

(d) StyleInV [11]



(e) our improved R2

(f) our AR2



(g) our R3

Figure 5.5: Sampled frames from 128-frame videos generated by our improved R2, AR2 and R3, as well as other methods such as MoCoGAN-HD (ICLR21), DIGAN (ICLR22), StyleGAN-V (CVPR22) and StyleInV (ICCV23) after training on the Sky dataset. Frames in each row are sampled starting from the first frame, with one frame sampled every 16 frames. Frames of other methods are from [11] at 256×256 resolution while those from our AR2 and R3 are at 128×128 resolution.

5.5.1 Qualitative Evaluation

In this section, we evaluate our recall-based methods, AR2 and R3, on 128-frame videos, comparing them with other long video generation methods presented in [11] (ICCV23). We also extend the evaluation to videos of 512-frame, comparing our methods (including improved R2) with results from other long video generation techniques provided on the website[1] of Ge et al.[9] (ECCV22). Due to the randomness generation setting of our methods, as well as for those used for comparison, where the generation process starts from a random noise or an initial clip without fixed constraints, it is not feasible that we consider the exact same scene or background for direct comparative purposes. Therefore, the quantitative results should be interpreted in the context of this limitation.

Figures 5.4 and 5.5 show Taichi and Sky frames, respectively. For both Figures, frames on each row were sampled from 128-frame videos generated by our AR2, R3 at 128×128 resolution and other long video generation methods at 256×256 resolution. Although improved R2, AR2 and R3 have lower resolution compared to other methods, they demonstrate a superior ability to preserve the physical structure of moving objects during motion in Taichi movements and clear cloud movement with good frame quality in Sky frames. Besides, the cloud movement of videos generated by improved R2 sometimes shows sudden changes in the cloud configuration, as shown in Figure 5.5, while AR2 and R3 generate videos with better frame quality and temporal coherence.

Figure 5.6: Frames sampled from 1000-frame videos at 128×128 resolution. Frames from top to bottom rows are generated by DIGAN, TATS, AR2 and R3after training on the Taichi dataset. Frames in each row are sampled every 32 frames from sequences 0 to 300 (left), 300 to 600 (middle), and 600 to 900 (right). The results of DIGAN and TATS are from [9].

Moreover, AR2 and R3 reduce the reliance on prior information as in improved R2, allowing the generation of videos longer than those in the training set. We further evaluate them on videos of over 1000 frames, as shown in Figure 5.6 and 5.7. For both Figures, from left to right columns are frames sampled from sequences 0 to 300, 300 to 600, and 600 to 900. Frames on each row are sampled for every 32 frames from the same video of over 1000 frames. In Figure 5.6, the generated frames of AR2 and R3 maintain better physical structure during the Taichi movement than DIGAN and TATS. And R3 exhibits richer motion content compared to AR2. In Figure 5.7, all methods except TATS generate videos displaying realistic cloud

[1] https://songweige.github.io/projects/tats/

Figure 5.7: Frames from 1000-frame videos at 128 × 128 resolution generated after training on the Sky dataset. Frames from top to bottom rows are generated by DIGAN, TATS, AR2 and R3. Frames in each row are sampled every 32 frames from sequences 0 to 300 (left), 300 to 600 (middle), and 600 to 900 (right). The results of DIGAN and TATS are from [9].

movement in the sky.

	prior	Taichi-128×128			Sky-128×128		
		FVD-16f↓	FVD-128f↓	ratio↑	FVD-16f↓	FVD-128f↓	ratio↑
MoCoGAN-HD [6] (ICLR 2021)	no	144.7	-	-	183.6	635.6	0.28
DIGAN [7] (ICLR 2022)	no	128.1	748.0	0.17	114.6	228.6	0.50
StyleGAN-V [8] (CVPR 2022)	no	143.5	691.1	0.2	-	-	-
TATS [9] (ECCV 2022)	no	**94.6**	-	-	132.5	435.0	0.30
Long-Video-GAN [10] (NeurIPS 2022)	no	-	-	-	**107.5**	**142.6**	**0.75**
VIDM [36] (AAAI 2023)	no	121.9	563.6	0.21	-	-	-
REncGAN (R2) [2] (ICIP 2023)	entire video	113.5	145.9	0.77	360.9	587.0	0.61
Improved R2	entire video	101.1	**108.9**	**0.93**	293.9	591.5	0.49
AR2	no	1258.5	1986.3	0.63	445.8	840.4	0.52
R3	initial clip	160.4	385.1	0.41	256.9	433.8	0.59

Table 5.5: Evaluation of FVD on the generated long-videos, measuring the sub-sequences of 16 and 128 frames, denoted FVD-16f and FVD-128f. The ratio of FVD-16f to FVD-128f quantifies the degradation in frame quality over longer durations, reflecting both the individual frame quality and the temporal coherence of the generated sequences. Results of other methods are from [36, 10] to ensure the same resolution (128 × 128) and video length of FVD. The FVD results available for StyleInV are on a different resolution (256 × 256) and hence not shown here.

5.5.2 Quantitative Evaluation

In Table 5.5, we evaluate the quality of generated videos for both the AR2 and R3 models across different durations using FVD metrics, specifically FVD-16f and FVD-128f, which are calculated using the first 16 frames and the first 128 frames of the generated video sequences, respectively. Some methods used in the qualitative evaluation are not included here in the quantitative comparison because their results were directly cited from [11] at a 256 × 256 resolution, while we focus on 128 × 128 resolution for FVD metrics. The FVD calculation follows Equation (3.7), but instead of using the output of an image classifier, it is now based on the output of a pre-trained Inception Network for video classification. Additionally, we calculate the FVD ratio between the results provided for FVD-16f and FVD-128f to assess

the degree of quality degradation as video duration increases.

As shown in Table 5.5, all methods that apply the recall mechanism, such as the improved R2 and AR2 models, exhibit a low degree of quality degradation, indicating the effectiveness of the recall mechanism in preserving video quality over longer durations. AR2 and R3 are developed based on the improved R2, as described in Section 5.2. R3 is directly incorporating the improved R2, as described in Section 5.4.2.

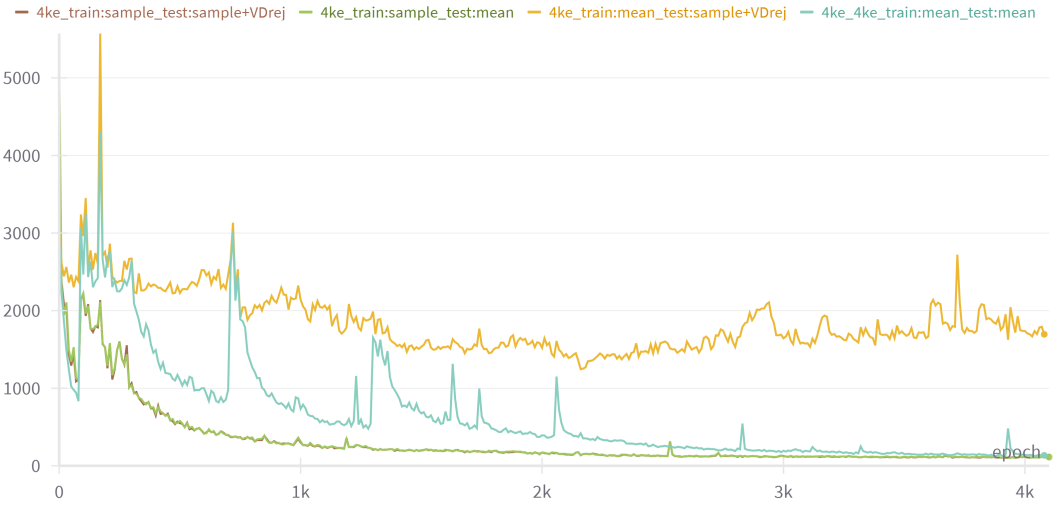
The improved R2 model demonstrates competitive performance on the Taichi dataset, achieving the lowest FVD-128f score (108.9) and the highest ratio (0.93), which indicates superior temporal coherence and minimal quality degradation over longer durations. However, it does not achieve the best FVD-16f score, where TATS performs better with a score of 94.6. In contrast, AR2 and R3 show relatively higher FVD values, primarily due to their reduced reliance on prior information. The high absolute FVD values of AR2 can also be attributed to its model capacity. While the recall mechanism preserves the internal structure of the short video generation model, maintaining its overall capacity, it introduces additional complexity by handling noise and managing other long-term video characteristics. This added complexity shifts focus away from short-clip quality, as reflected by the higher FVD-16f scores. As a result, poorer short-clip quality negatively affects the overall quality of long videos, despite the low degree of quality degradation as indicated by the FVD ratio.

Similarly, the recall mechanism in the R3 model helps preserve video quality over time by splitting the long-video generation process into two stages: first, GPT generates a sequence of latent representations, and second, a short-video generator independently produces each clip based on these latent representations. The VDrej operation is applied to each generated clip, refining the output and reducing error accumulation at each step. This mechanism is evidenced by the generally higher FVD ratios (mostly over 0.5), indicating a lower degree of quality degradation over longer durations. However, despite the effectiveness of recall mechanism in the R3 model in mitigating error propagation, the GPT-based auto-regressive generation in R3 still accumulates more errors than the improved R2 model, leading to higher overall FVD values. This is further exacerbated by the reduced reliance on prior information in R3, which contributes to a decline in generated video quality. Although R3's two-stage approach—GPT generating latent sequences and the short-video generator producing pixel-wise clips—reduces complexity, its greater focus on temporal coherence rather than detailed frame generation results in higher FVD values.

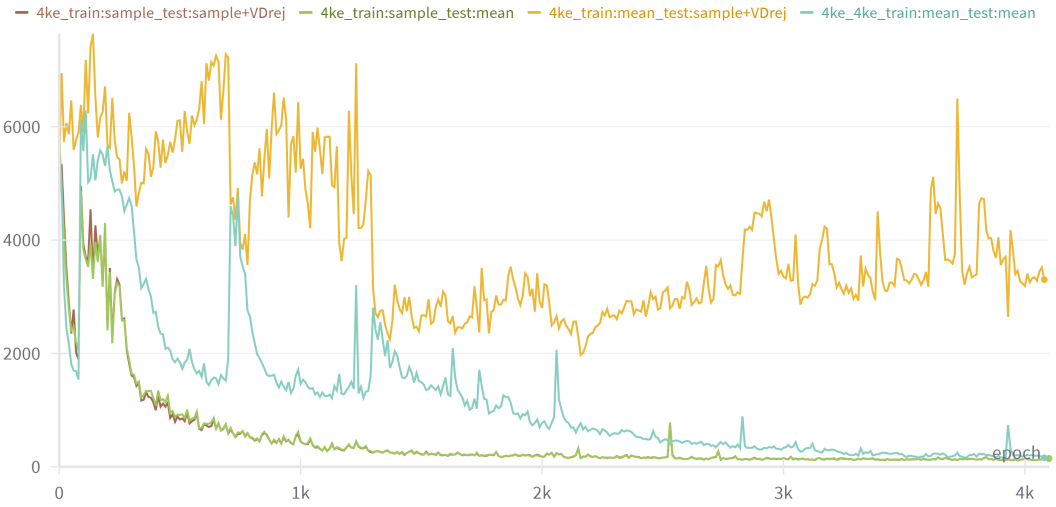
As noted in [10], FVD is particularly sensitive to the quality of individual frames and the smoothness of small-scale motion. Unlike other methods, REncGAN stitches every 8 frames from generated clips. This stitching process impacts temporal coherence over short durations, particularly within 16-frame segments, leading to worse performance on the FVD-16f metric. Additionally, when generating cloud movement, which involves irregular and less structured motion, REncGAN struggles to achieve low FVD values, as it prioritizes preserving the physical structure of moving objects. Consequently, this results in weaker FVD performance in the videos generated after training on the Sky dataset. However, upon visual inspection of the generated video frames, the quality is still rather good, particularly in maintaining content realism and visual consistency.

In summary, while both AR2 and R3 show higher absolute FVD values, these metrics do not fully capture their capability to generate long videos with complex motion, as evidenced

by the results in the Qualitative Evaluation section 5.5.1.



(a) FVD-16f



(b) FVD-128f

Figure 5.8: Ablating the usage of latent space at training and test time by plotting FVD scores across trained model parameters at different training steps. The plot of FVD results across over 4000 epochs and the FVD scores are calculated for every 10 epochs. At each calculation, FVD is calculated based on 16 frames and 128 frames, respectively, denoted as FVD-16f and FVD128f as in (a) and (b).

5.5.3 Ablation of the improved R2

In this section, we test the improved version of R2, with its modifications described in Section 5.2. The improved R2 is the foundation of AR2 and R3. We study the impact of ablating the usage of latent spaces, model structure, and training objectives of this improved R2 model.

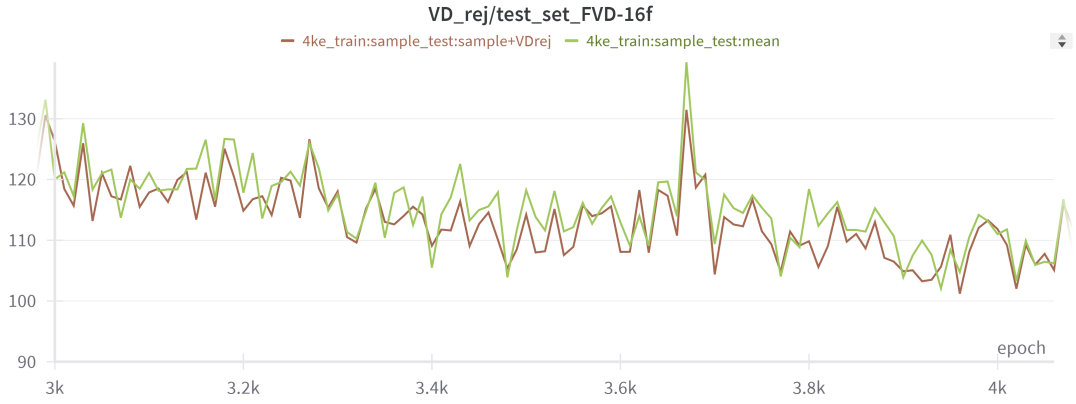
Ablating the usage of latent space at training and testing time. In this ablation study, we evaluate the impact of using the mean of the latent variables versus sampled latent codes during training and testing. As illustrated in Figure 5.8, we explore two training scenarios: one where the mean value of the latent space is used (labeled as $train : mean$) and another where latent codes are sampled from the distribution of the latent space (labeled as $train : sample$). During testing, we evaluate the models trained under these two scenarios using both the mean value ($test : mean$) and sampled latent codes ($test : sample$) for generation, as shown in Figure 5.8.

For the video generation using sampled latent codes, we employ the VDrej technique, which is introduced in Section 5.3.2, to ensure optimal performance, allowing us to effectively compare the best performance between using the mean or by sampling of the latent space for choosing the generative code. Figures 5.8 (a) and (b) display the FVD results calculated on video sub-sequences of 16 frames and 128 frames, respectively. In these figures, the first 16 and 128 frames from each generated video are used to compute the FVD. Although the model trained with the mean of the latent code ($train : mean$) initially exhibits a high variability in results when generating videos considering the mean ($test : mean$), its ($train : mean\_test : mean$) performance gradually approaches that of the model trained with sampled latent codes. However, when the $train : mean$ model is tested with sampled latent codes ($test : sample$), the FVD results are considerably worse. Given the importance of the diversity among the generated videos, we decide to use sampled latent codes during both training and testing.

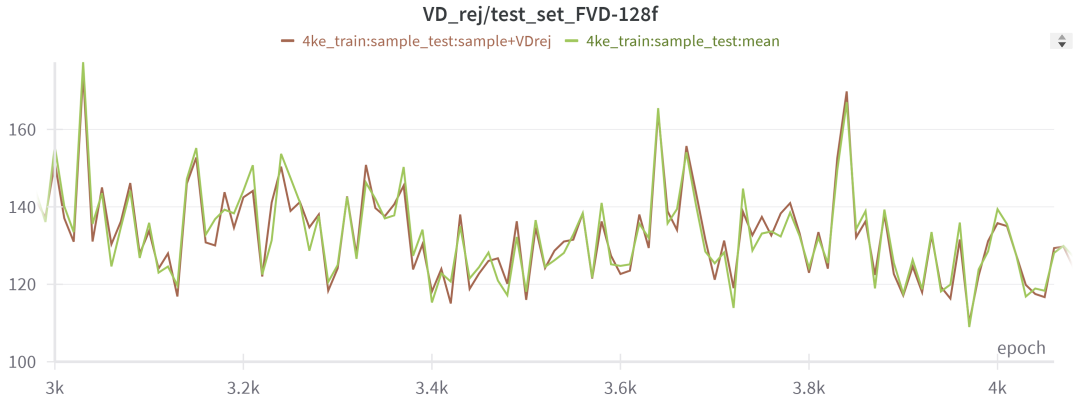
Furthermore, Figure 5.9 presents the results from epochs 3000 to 4060, where the best FVD scores were observed. This figure provides a clearer comparison of the FVD differences when the model trained with latent codes is used to generate when considering either the mean ($test : mean$) or sampled latent codes($test : sample$). It is evident from the figure that generating videos considering sampling the distribution of the latent codes yields better FVD results in most cases, due to the use of VDrej. VDrej cannot be used when generating videos using the mean of the latent space because it works by selecting the best latent code from multiple samples within a latent space. However, the mean-based approach provides only a single option per latent space, for which there is no need for selection. Based on the FVD results shown in Figure 5.9, considering both FVD16f and FVD128f, we selected the model parameters on epoch 3970 as the best R2 model for further training the GPT.

Ablating the structure of video stream in Generator. In this study, we ablate the structure of the video stream in the Generator, when using 3D spatio-temporal data representation together, or 1D with 2D convolutions, denoted as $g\_mode : 3d$ and $g\_mode : 1p2d$, separately. The Generator contains several G^3 blocks, with each processing three streams: spatial, video and temporal. The two structures of the video stream are shown in Figure 5.1 (a) and (b). R3 uses the R2 with the structure $g\_mode : 3d$ that is shown as Figure 5.1 (a). And the structure of $g\_mode : 1p2d$, as in Figure 5.1 (b), is used in the original R2 that is introduced in Section 4.4 in Chapter 4. As shown in Table 5.6, R2 with $g\_mode : 3d$ provides better FVD results than using $g\_mode : 1p2d$ structure, whether considering the generated short videos (FVD-16f), longer videos (FVD-128f), or the degree of frame quality degradation (FVD ratio).

Ablating the noise generation ability. In this study, we explore the impact of including or excluding the ability to generate short video clips from random noise in R2. EncGAN3, as introduced in Chapter 3, incorporates loss terms that constrain the generation of videos



(a) FVD-16f



(b) FVD-128f

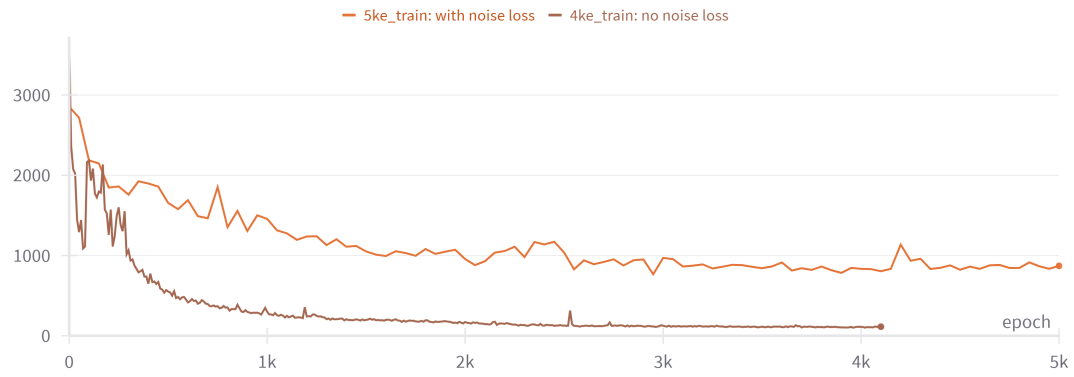
Figure 5.9: Ablating the usage of latent space at test time by plotting the FVD results across fewer epochs to show in detail. The FVD results cover from 3000 epochs to 4060 epochs, where the best FVD result was observed, for FVD-16f and FVD-128f on (a) and (b), respectively.

Processing structure	Taichi-128×128-16f	Taichi-128×128-128f	FVD ratio(16f/128f)
3d	101.1	108.9	0.93
1p2d	113.5	145.9	0.77

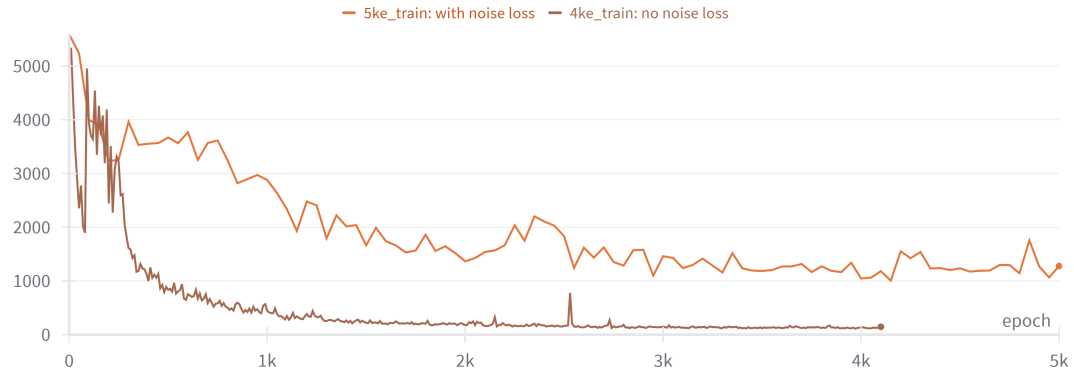
Table 5.6: Ablating the structure of video stream in the Generator.

from random noise. When applying the original recall mechanism to EncGAN3, resulting in R2, we remove this noise generation constraint because R2 generates long videos based on latent spaces and does not require noise. To focus more on modelling long video generation, we eliminate the redundant functionality of generating clips from noise. In R3, we use GPT to generate sequences from the latent space. The auto-regressive generation process of GPT requires an initial input. While we use training clips as the starting input in R3, as described in Section 5.4.2, we also consider reintroducing the noise generation functionality to enable unconditional long video generation by generating the first clip from noise. Thus, in this ablation study, we assess the performance changes of R2 with and without the noise

generation capability to determine if this feature should be included.



(a) FVD-16f



(b) FVD-128f

Figure 5.10: Ablating the noise generation ability by plotting FVD results of FVD-16f and FVD-128f on (a) and (b), respectively.

From the results from Figure 5.10, we can observe that by considering the noise generation function results in a significant drop in the FVD values. To achieve better performance, we choose to use the R2 model without noise generation. Table 5.7 provides a detailed comparison of FVD values. Despite the noticeable drop in FVD values, the FVD ratio, which represents quality degradation, remains superior to other methods compared in Table 5.5 (the FVD ratio of most methods is below 0.5), further demonstrating the advantages of the recall mechanism in long video generation.

noise_gen loss	Taichi-128×128-16f	Taichi-128×128-128f	FVD ratio(16f/128f)
no	101.1	108.9	0.93
yes	766.9	1004.5	0.76

Table 5.7: Considering or not the availability of the noise component for generating the long video.

5.5.4 Ablation of AR2

In this section, we ablate the AR2 model. AR2 builds upon the improved version of R2 by introducing an auto-regressive generation mechanism to enable unconditional generation. The ablation studies for the improved R2 were already presented in the previous section, and the study of the auto-regressive mechanism has been covered in previous evaluation sections. Here, we focus on the ablation studies of the VDrej mechanism, which is used in both AR2 and R3.

Ablating the VDrej mechanism. This study examines the effects of ablating the VDrej mechanism by comparing results generated using the mean value of the latent space, randomly sampled latent codes, and sampled latent codes filtered through the VDrej mechanism. The last one involves sampling 20 latent codes from the latent space and selecting the best one using the VD module.

As shown in Table 5.8, the VDrej mechanism yields the best performance, highlighting its superiority. Random sampling can reduce generation quality due to noise and uncertainty, making the mean value of the latent space a more reliable option in general. However, when the VDrej mechanism is applied to the sampled latent codes, the quality of the generated samples improves significantly. The accumulated error through each step in an auto-regressive manner is also reduced through the VDrej mechanism. Therefore, sampling latent codes with VDrej achieves the best performance.

	mean	sample	sample+VDrej
Taichi-128×128-16f	1353.5	1266.8	**1258.5**
Taichi-128×128-128f	2013.1	2087.1	**1986.3**
Sky-128×128-16f	448.6	455.2	**445.8**
Sky-128×128-128f	857.4	930.3	**840.4**

Table 5.8: Ablation of different generation methods, including using the mean of the latent variables, sampling a latent code from the latent space, or sampling several latent codes from each latent space and using the Video Discriminator (VD) to select the best generated clip.

5.5.5 Ablation of R3

In this section, we ablate the R3 model. R3 is composed of the GPT module and the improved R2. The ablation studies for the improved R2 were provided in an earlier section, and the study on the VDrej mechanism was discussed in the context of AR2. In this section, we present the ablation studies focused on the adaption of the GPT module in R3 model. In the following, we consider several ways to provide the latent spaces as the input to the GPT and how to use GPT in the generation of long videos.

Different ways to format the sequences of latent spaces. In this ablation study, we investigate the impact of different token sequence formats on the performance of GPT models. In the R3 model, the latent space of the videos is used as tokens for the GPT input. Each latent space is obtained by encoding a video clip. By splitting a video into multiple clips and encoding each, we obtain a sequence composed of latent spaces. When considering

this sequence as input for GPT, we explore different designs for the GPT input. One major challenge is that of handling the padding with zeros in order to complete the size of the sequences. The default padding with zeros for GPT is denoted as $pad0$. The input format is specified as:

$$[seq, padding(0)] \tag{5.5}$$

where $padding(0)$ means padding with zeros. However, according to TATS [9], padding with zeros can destroy the temporal consistency, which severely degrades the quality of generated frames beyond the observed video length. Because these frames are represented with zero values, which actually implies poor content quality. This explains the poor performance of VideoGPT [20] when generating long videos. To address this issue, we consider alternative padding methods. One approach involves padding with other latent sequences, denoted as $padzs$, where zs represents the sequence of latent spaces. The input format is specified as:

$$[seq, padding(seq_{class_i})] \tag{5.6}$$

where seq_{cls_i} denotes the latent sequence corresponding to the i-th instance that belongs to the same class as seq. Specifically, we use sequences of latent spaces generated from other videos with the same moving object for padding, aiming to maintain the frame quality when generating videos which are longer than the length of the videos used for training. Such an approach needs to recognize padding when generating longer videos by using VDrej, to avoid forcing connectivity for the frames displaying transitions between different scenes. However, GPT is not able to recognize the padding when trained on continuous variables. Therefore, we design a sequence mixed of discrete and continuous variables as GPT input, aiming to use the discrete variables to help identify padding positions, denoted as $disc + cont$, where $disc$ and $cont$ indicate discrete and continuous variables. The input format is specified as:

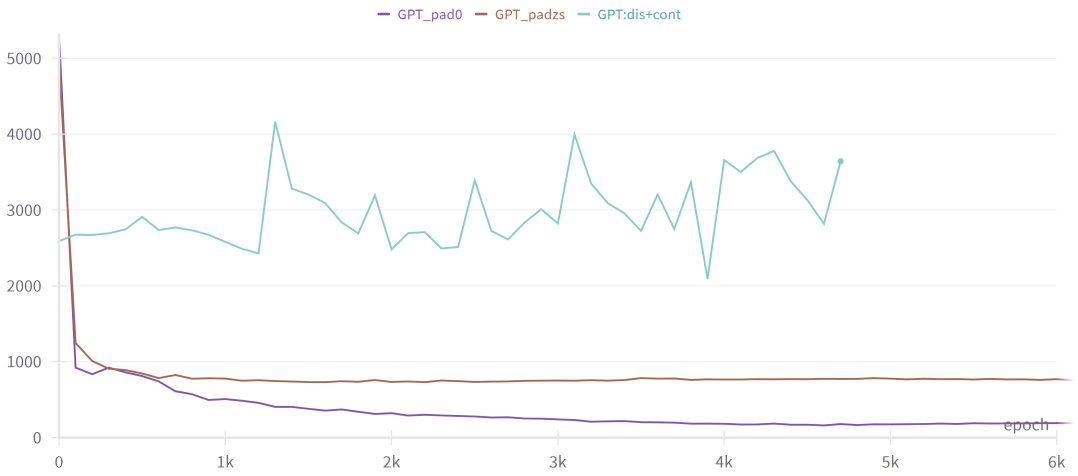
$$[cls, seq1\_len, seq1, seq\_i\_len, seq\_i, padding] \tag{5.7}$$

where $seq1$ and $seq2$ are two sequences of latent spaces while they have the same class cls. And $seq1\_len$, $seq2\_len$ are their respective lengths, which helps determine the padding position. This approach is intended to support the generation of videos having arbitrary lengths, especially when considering scene transitions involving the same moving person or object. During the sequence generation, when the model reaches padding, the input sequence format is updated as:

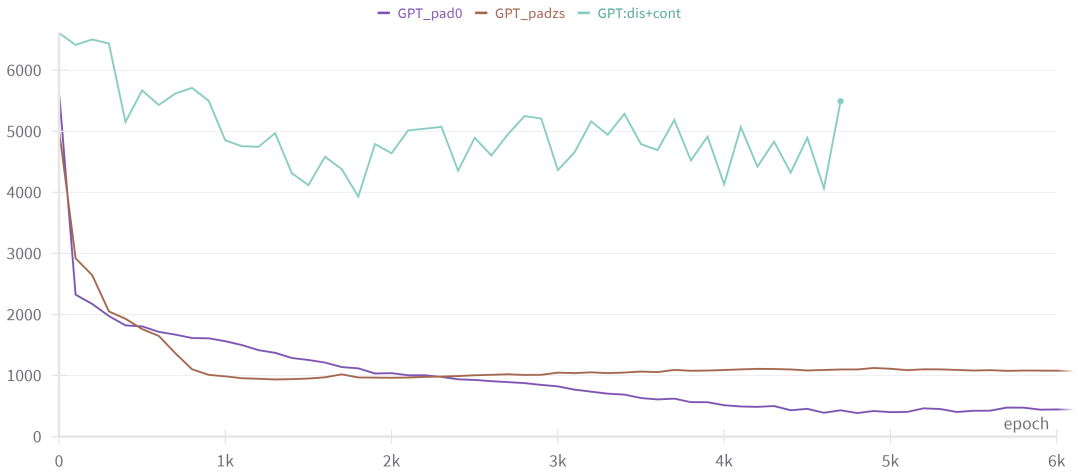
$$[cls, seq\_i\_len, seq\_i, seq] \tag{5.8}$$

allowing the model to continuously generate video content by cycling through different long video sequences. Since all these sequences pertain to the same actor class, transitions between different videos can be interpreted as scene changes.

In brief, we designed three different input video sequence formats based on our approach to latent space sequences for generating video sequences. These formats consider continuous latent variables with no padding as in Equation (5.5), continuous latent variables with padding (zs) as in Equation (5.6), and also considering both continuous and discrete variables with no padding with zeros as in Equation (5.7). The results, as shown in Figure 5.11, indicate that using zeros as padding, the default padding method in GPT, yields the best FVD results. The best FVD values of plots in this Figure are detailed in Table 5.9. The GPT trained on pure sequences of latent spaces, without discrete variables, and padding with zero, instead of other latent space sequences, performs significantly better.



(a) FVD-16f



(b) FVD-128f

Figure 5.11: Results when considering different ways for sequencing the latent spaces between being provided to the GPT. Labels 'GPT_pad0', 'GPT_padzs' and 'GPT:disc+cont' correspond to the format of GPT input shown in Equation (5.5), (5.6) and (5.7), separately. FVD results of FVD-16f and FVD-128f are shown on (a) and (b), respectively. FVD is calculated for every 100 epochs.

The mixed discrete and continuous variable input for GPT requires tokenization to convert inputs into a purely continuous variable token form. The tokenization operation also allows for the adjustment of token dimensions. In the following, we consider changing the size of the latent space (token size), used as the input for the GPT, and evaluate the impact on the long video generation results in Figure 5.12. As shown in Figure 5.12, performance deteriorated when varying token sizes. The corresponding numerical results are provided in Table 5.12.

Results when considering different generation approaches. In this section, we investigate how different generation methods affect both model performance and generation time. Given that R3 is trained in two stages, incorporating both the R2 and GPT modules, we explore two distinct approaches for the generation.

input type	pad value	Taichi-128×128 FVD-16f	Taichi-128×128 FVD-128f	FVD ratio (16f/128f)
cont	0	160.4	385.1	0.41
cont	zs	731.3	935.6	0.78
disc + cont	0	2093.1	3929.2	0.53

Table 5.9: Considering different ways to form latent space sequences to be provided to the GPT. *cont* and *disc* mean continuous and discrete variables, respectively. The corresponding input formats from top to bottom are described as in Equation (5.5), (5.6) and (5.7), separately.

token dimension	Taichi-128×128 FVD-16f	Taichi-128×128 FVD-128f	FVD ratio (16f/128f)
1536	2093.1	3929.2	0.53
512	2125.8	3815.1	0.55
256	2003.8	3878.1	0.51

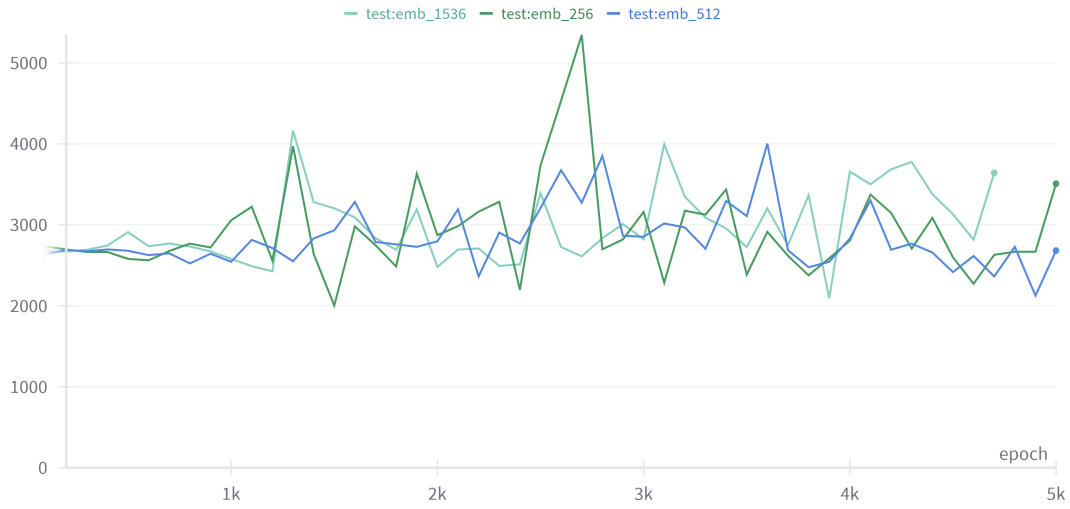
Table 5.10: FVD results of different sizes for the latent space (token vectors) used as input to the GPT module.

The first method, denoted as GPT_{faster}, involves GPT generating the entire sequence of latent spaces, which are then passed to R2 to map these latent spaces into clips. These clips are subsequently stitched together to form a long video. This approach requires auto-regression only during the latent space generation by GPT, allowing R2 to generate all clips in parallel. Consequently, this method is faster and more GPU memory-efficient.

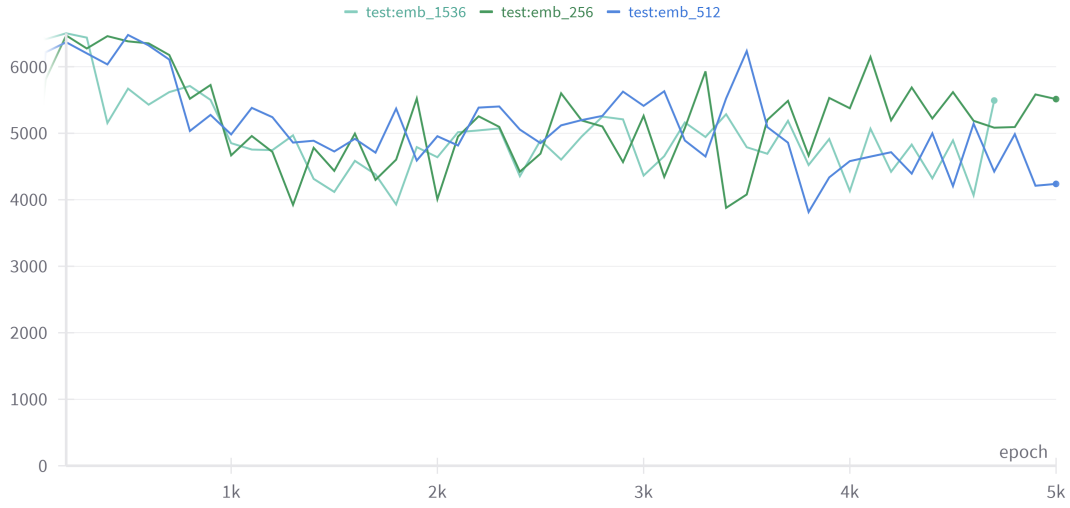
The second method, denoted as GPT_{robust}, takes a different approach. GPT generates one token at a time, and then R2 generates a clip based on that token. This generated clip provides a latent space for predicting the next token to the GPT. R2 utilizes the VDrej mechanism to select the best clip from the generated options, effectively optimizing the latent space at the pixel level to enhance the realism of the video. Although this method takes longer for the generation, as it requires auto-regressive generation of each clip and prevents R2 from generating clips in parallel, it is more robust in maintaining the video quality over longer sequences.

As shown in Figure 5.13, the FVD scores clearly show that the combined generation approach of GPT and R2 (GPT_{robust}) produces superior results compared to the separate generation method (GPT_{faster}). Thus, while GPT and R2 are trained separately, the generation process benefits from their integration (GPT_{robust}), rather than relying on GPT to generate the entire latent sequence independently before passing it to R2 to generate clips (GPT_{faster}).

Additionally, in Figure 5.13, we also compare the performance of generation based on the mean value of latent spaces versus sampled latent codes from their corresponding distribution. It is named as GPT_{faster}, since the usage of VDrej in GPT_{robust} requires to use sampled latent codes. In the first part of the generated video, the results based on sampled latent codes are worse, gradually approaching the FVD results of the generated videos corresponding to using the mean of the latent space for the generation. This suggests that GPT progressively learns to capture key values from the input latent space, mapping to a region within the latent space rather than merely learning how to map to a single value.



(a) FVD-16f

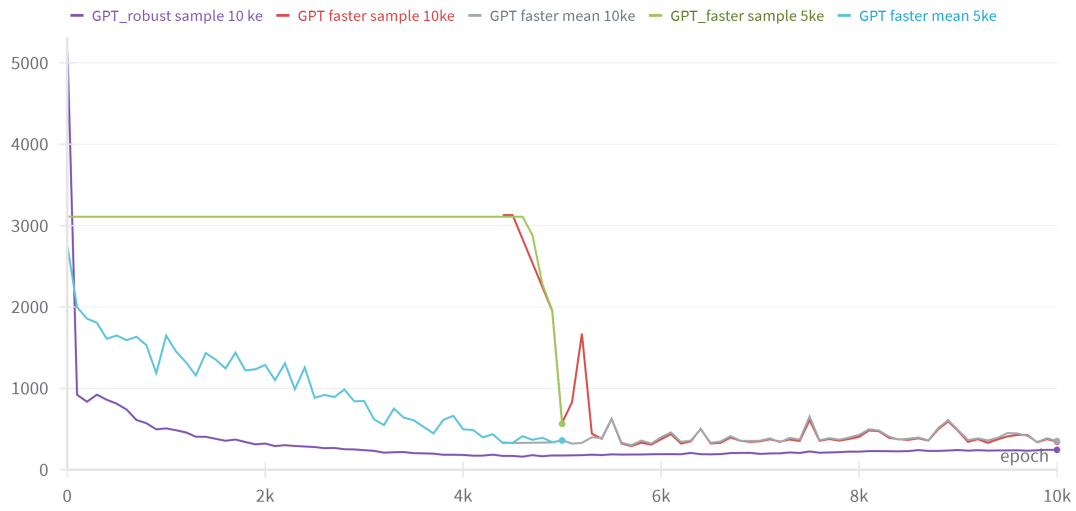


(b) FVD-128f

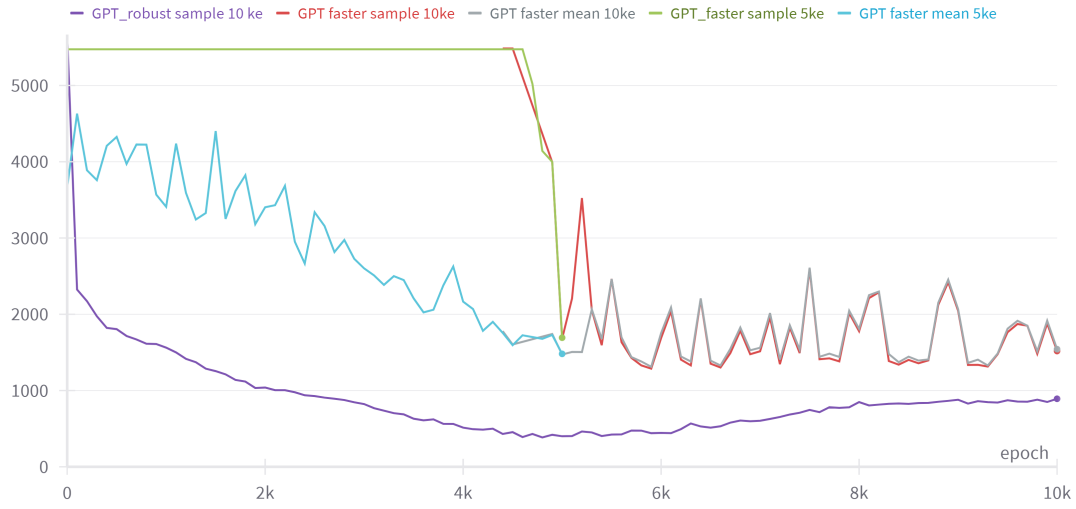
Figure 5.12: Changing the size of the latent space (token vectors) used as input for the GPT. FVD results of FVD-16f and FVD-128f are on (a) and (b), respectively. FVD is calculated for every 100 epochs.

Table 5.11 presents a detailed comparison of FVD scores and their corresponding generation times. As indicated in the table, videos generated using the GPT_{robust} approach achieve significantly better FVD results compared to GPT_{faster}, especially in terms of FVD-128f, which measures the quality over longer durations. The ratio of the FVD calculated on 16 frames over the FVD calculated for 128 frames, in GPT_{faster} compared to GPT_{robust} highlights the robustness of the GPT_{robust} method when considering the quality degradation when generating longer sequences.

Regarding the generation time, as shown in Table 5.11, the use of VDrej incurs the highest time consumption, increasing from 0.06 minutes to 0.5 minutes, a nearly tenfold jump. This is followed by the purely auto-regressive generation in GPT_{robust}, where the removal of



(a) FVD-16f



(b) FVD-128f

Figure 5.13: Ablating different generation approaches. Curves labeled as 'GPT faster sample 5ke' and 'GPT faster sample 10ke' are based on the same setting while run twice, so as the 'GPT faster mean 5ke' and 'GPT faster mean 10ke'.

parallel generation in R2 increases the processing time from 0.5 minutes to 0.72 minutes. Specifically, the time-consuming nature of VDrej stems from the need to sample a specified number of latent codes from each latent space (20 in this study) for each clip generation. Since generating each clip is time-intensive, directly increasing the number of clips generated naturally leads to a significant increase in the time cost. However, likely due to the GPU parallel processing, despite a 20-fold increase in the number of clips generated, the time cost only increases by about tenfold.

Moreover, since we generate multiple videos in parallel in practice, we offer an approximate measure of the time required to generate a 1000-frame video. The time costs shown in Table 5.11 are calculated as follows:

ways	VDrej	Taichi-128×128 FVD-16f	Taichi-128×128 FVD-128f	FVD ratio (16f/128f)	generation time cost (minutes)
faster	yes	291.2	1287.7	0.22	0.06
faster	no	301.7	1311.5	0.22	0.5
robust	yes	160.4	385.1	0.41	0.72

Table 5.11: Ablation study for different ways to generate the long video sequences. The time cost indicates how long it takes to generate a video with 1000 frames.

- For GPT_{faster}, GPT generates the latent sequences, and then R2 generates all clips in parallel. The generation time for the latent spaces by GPT, based on experiment records, is approximately 30 minutes for 26 runs, each generating 64 latent space sequences corresponding to 1000-frame videos. Thus, the generation time for each video by GPT is about 0.02 minutes. The time for R2 to generate clips from latent spaces is measured in two ways: using the mean value from each latent space, or by sampling several values from the latent space distributions, employing the VDrej mechanism. Generating clips by using the mean value takes about 1.3 hours for 31 runs, each generating 64 videos, resulting in 0.04 minutes for each generated video. However, when using VDrej with 20 sampled latent codes, the time cost increases to about 0.48 minutes per video, based on an experiment duration of 26 hours for 51 runs.
- For GPT_{robust}, which uses both GPT and R2 for each clip and the VDrej mechanism to refine each generated clip, the total generation time is approximately 69.5 hours for 90 runs, each producing 64 long videos of 1000 frames each. The average time cost per video is 0.72 minutes. Clearly, GPT_{robust} (0.72 min) requires a longer time for generating videos than GPT_{faster} (0.5 min), with the VDrej mechanism being the primary contributor to the increased generation time (VDrej: 0.5 min versus no VDrej: 0.06 min).

5.5.6 Comparison of the Computational Costs

Table 5.12 compares the training time required to train models from scratch. Both TATS and R3 are computationally expensive, requiring weeks and would be months if considering the single-GPU training time. Additionally, our AR2 model takes longer to train on the Taichi dataset compared to R3's R2 component, despite having the same model structure. This is primarily due to the introduction of auto-regressive mechanisms, which require additional iterations to converge. In contrast, on the Sky dataset, AR2 exhibits a significantly shorter training time than R3's R2 component. This discrepancy arises from the different numbers of training epochs required for convergence. Specifically, the base model EncGAN3, used in R3's R2, was found to have limited capacity for further improvement at that stage, making prolonged training inefficient.

Table 5.13 evaluates the time required to generate a single 1024-frame video across different methods. Our AR2 and R3 models achieve significantly faster generation times (0.6 minutes and 0.72 minutes, respectively) compared to TATS-Base (30 minutes) and TATS-Hierarchical (7.5 minutes + 23 seconds) (23 sec for interpolation), while maintaining comparable generation quality. Although some methods, such as DIGAN and MoCoGAN-HD, exhibit faster

Table 5.12: Training time comparison across different models and hardware settings.

Model	Actual Time	GPU Configuration	Batch Size/GPU	Batch Num	Single GPU Time
TATS's VQ-GAN	57h	8 × V100 (32GB)	2	-	19d
TATS's GPT	10d	8 × V100 (32GB)	3	-	80d
TATS total	12.37d	-	-	-	99d
AR2	13.5h / 12.5h	3 × A40 (48GB) / 2 × H100 (80GB)	10 / 55	10 / 1284	27h / 25h
R3's R2	6.3h / (12+1.75)d	2 × H100 (80GB) / (3 × A40 + 2 × H100)	50 / (25, 50)	12 / (1883+1412)	12.6h / (36+3.5)d
R3's GPT	13.9h / 10.5d	2 × H100 (80GB)	30 / 5	1 / 187	27.8h / 21d
R3 total	20h/(12+12.25)d	-	-	-	40h/(36+24.5)d

Note: "h" is hours and "d" is days, "GB" is Gigabit. Values separated by "/" represent the resource cost (e.g., training time, GPU) on different datasets Taichi and Sky, like Taichi/Sky. "+" indicates cumulative training time across different GPU types. Training time is reported until convergence or when exceeding a reasonable training duration.

generation times (4.2 seconds and 28.5 seconds, respectively), the quality of extended generated frames for our models ourperform theses methods as well as others like VideoGPT.

Model	DIGAN	MoCoGAN-HD	VideoGPT	TATS-Base	TATS-Hierarchical	AR2	R3
Time (sec/min)	4.2s	28.5s	42min	30min	7.5min + 23s	0.6min	0.72min
GPU	-	-	-	1 P6000		1 A40 or H100	

Table 5.13: Time required for generating a 1024-frame video across different models.

5.6 Conclusion

Summary of Findings. In this research chapter, we introduce two methods, AR2 and R3, designed to enhance long video generation by reducing reliance on prior training data as in R2. AR2 implements an auto-regressive framework, while R3 incorporates a GPT module to autonomously generate latent space sequences. Both methods significantly improve video diversity and temporal coherence, and employ the VDrej to maintain high video quality over extended durations.

Experimental results show that both AR2 and R3 inherit the effectiveness of the recall mechanism in preserving video quality over longer time durations and can generate videos of arbitrary lengths. However, AR2 struggles with long-term coherence due to the lack of a global view, while the performance of R3 depends on the effectiveness of the GPT module.

Limitations and Future Work. Despite these advancements, some limitations remain. Firstly, the quality of generated long videos is influenced by the quality of the underlying short clips. While the recall mechanism is designed to handle long video generation with short training videos, it does not inherently enhance the capacity of model to process more complex data. Future work could explore integrating diffusion-based short video generation models with the recall mechanism to improve short clip quality and study its performance on more complex data.

Another limitation is the lack of control over generated content. Future research could enhance the GPT component in R3 to link video generation with text features, enabling more precise control based on text input. Additionally, applying the recall mechanism to

diffusion-based generators may improve both the diversity and quality of generated content, while broadening the scope of controllable features.

In summary, future work should focus on improving short video inputs and exploring methods for more controllable and diverse content generation. Specifically, if we were to approach this study differently, we would prioritize upgrading the short-video generator model using the latest generative technologies (e.g., VQ-GAN [80] or DiT [38] for videos [9, 52]) to achieve clip quality comparable to state-of-the-art methods. This would enable a fairer evaluation of the recall mechanism's performance in generating longer video sequences. We would also design experiments to better isolate the impact of recall on long video generation and explore recall-based implementations using diffusion models, which could further enhance the robustness and flexibility of long video generation frameworks.

Chapter 6

Conclusions

6.1 Restatement of Research Goals and Contributions

This thesis explores and advances the research in the area of generative video modelling, focusing on long-duration video generation using generative adversarial networks (GANs), variational auto-encoders (VAEs), and Transformer-based models. Specifically, three major contributions: (1) the introduction of EncGAN3, a VAE-GAN hybrid for stabilized video generation, (2) the development of REncGAN and LEncGAN, which incorporate a recall mechanism to extend short-term video generation into long-term, and (3) the creation of AR2 and R3, which leverage auto-regressive and GPT-based architectures to further enhance long video generation while reducing the dependence on prior training data.

These contributions address key challenges in video generation, such as improving the temporal consistency of generated videos, maintaining high video quality over extended sequences, and reducing computational overhead. The innovations introduced through these research results provide a basis for future work in generative video modelling, particularly in handling the complexities of long-duration video generation.

6.2 Summary of Key Findings

Contribution 1: EncGAN3. In the first part of this thesis, we introduced the EncGAN3, a VAE-GAN hybrid network designed for video generation. EncGAN3 incorporates an encoder to stabilize the training process, which improved the quality and diversity of generated videos at resolutions of 64×64 and 128×128 pixels. The model successfully generates videos depicting single or occasionally two moving objects across several benchmark datasets. However, EncGAN3 struggled to maintain high video quality when tasked with generating sequences longer than 16 frames, resulting in visual artifacts and inconsistencies in motion over extended durations.

This limitation led to the development of more advanced architectures, which aimed to address the challenges of long-term video generation.

Contribution 2: REncGAN (R2) and LEncGAN. In order to extend video generation to longer sequences, the second contribution introduced REncGAN and LEncGAN, both built on the foundation of EncGAN3. LEncGAN incorporated an LSTM module, allowing it to generate coherent long-term sequences by modelling temporal dependencies across video clips. REncGAN removed the LSTM and introduced the recall mechanism, which modelled temporal relationships between video clips more efficiently.

The recall mechanism, acting as a form of temporal memory (i.e., how the model remembers temporal things: Markov-chain-based inter-clip relationships), significantly improved the consistency and coherence of generated long videos. REncGAN demonstrated the ability to generate hundreds of frames while maintaining visual fidelity. However, the model exhibited performance limitations on datasets with simpler, non-structured content, such as movements of clouds in the sky, highlighting the need for improved fine-tuning between clips and more dynamic adaptation to varied content.

Contribution 3: AR2 and R3. In the third contribution, two advanced methods were introduced to address the reliance on prior training data observed in earlier models. AR2 employed an auto-regressive framework, while R3 incorporated a GPT module to generate long video sequences in a more autonomous manner. These methods leveraged the previously introduced recall mechanism and combined it with the advantages of Transformer-based architectures.

AR2 and R3 significantly enhanced video diversity and temporal coherence in the generated long-term videos. However, AR2 struggled with long-term coherence due to its lack of a global view of the generated sequences. R3, which benefited from the ability of GPT module to model long-range dependencies, displays improved performance in generating arbitrary-length videos. Nonetheless, the performance of R3 was closely tied to the effectiveness of the GPT module and still required refinement in terms of content control.

6.3 Reflection on Key Challenges

Despite the advancements presented in this thesis, several challenges remain in the field of generative video modelling, particularly when aiming to generate long-duration sequences:

1. **Temporal Consistency and Video Quality:** The generation of long, high-resolution videos that maintain both visual realism and motion consistency over time represents a challenge. While the recall mechanism improved the temporal coherence of videos, maintaining this quality over hundreds of frames, especially when generating more complex content, remains difficult.

2. **Computational Overhead:** Video generation, particularly long-term generation, requires significant computational resources. While the proposed models reduced memory usage by modelling relationships between video clips rather than individual frames, further optimization is needed to reduce training time and improve scalability.

3. **Evaluation Metrics:** A major challenge in video generation lies in the evaluation metrics. Fréchet Video Distance (FVD), an extension of FID, aligns well with human

perception for video consistency and coherence but lacks the ability to assess the structural stability of moving objects. Meanwhile, a standardized version of the Inception Score (IS) for videos remains undeveloped, making it unclear how to measure the diversity of generated videos, whether to be calculated across different videos or within a single video over time.

In this thesis, EncGAN3 used video FID and video IS following previous models, while REncGAN incorporated FVD-16f and FVD-128f for better alignment with visual perception. For the final research chapter, only FVD metrics were used, with the FVD ratio assessing quality degradation over longer sequences. Video IS was omitted due to the complexity of retraining models for comparison.

4. **Dataset Complexity:** Models like REncGAN had performance drops when generating videos from datasets with simpler or rapidly changing content, highlighting the need for models that are more adaptable to varied types of motion and scene complexity.

6.4 Further Work

Several promising directions for future research emerged from the results presented in this thesis:

1. **Integration of Diffusion Models:** Diffusion models, which have recently shown strong performance in image and video generation, offer a potential solution to the limitations encountered in generating long-term videos [33, 34, 35, 36]. Integrating diffusion-based short video generation with the recall mechanism could enhance both the quality of short clips and the temporal consistency of longer sequences.

2. **Controllable Generation:** Future research could focus on enhancing the control over generated content, particularly through the use of text-conditioned generation frameworks like CLIP or ControlNet [83, 84, 103]. The incorporation of these frameworks could allow for more precise control of generated video content, linking it more closely to textual descriptions or user input.

3. **Refinement of the Recall Mechanism:** While the recall mechanism was shown to be effective for long-term video generation, further optimization is needed to reduce its dependency on prior information and improve its adaptability across various datasets. Potential improvements include integrating insights from other video generation models or applying the recall mechanism in more advanced generative frameworks.

4. **New Evaluation Metrics:** Future research should also address the need for more robust and standardized evaluation metrics. While FVD is currently the most used video quality assessment measure, it does not properly evaluate the structural stability of moving objects and their complex interactions. Additionally, the lack of a unified video Inception Score (video IS) limits the ability to assess the diversity of generated videos. Research should aim to develop a reliable video IS variant or alternative metric that can more comprehensively assess both diversity across videos as well as the temporal consistency within videos. Moreover, better metrics that capture long-term

quality degradation and object dynamics more effectively can be developed, especially as generative models begin to handle increasingly complex video sequences.

5. **New Dataset for Long Video Generation:** The datasets used for long video generation requires some of the following properties. The first requirement is that the videos should be long enough, such as covering at least minutes, in a single shot. Then, videos containing several shots better contain labels, such as text descriptions of the logical relationships across those shots and their switches, which could benefit semantic coherence. Video sequences with branches in different time steps would benefit from the diversity in the temporal dimension. This could be achieved by videos with many redundancies and similarities, which allow the jumps to different movement sequences, such as the videos showing cloud movement or the records of 2D video games that take place in a simple, fixed background.

6.5 Concluding Remarks

This thesis presents a series of advancements in generative video modelling, particularly for synthesizing long-duration videos. By introducing EncGAN3, LEncGAN, REncGAN (R2), AR2, and R3, this research contributes significantly to improving the temporal consistency, efficiency, and diversity of generated videos. The recall mechanism, in particular, stands out as an important innovation, enabling more effective long-term video generation.

While challenges remain, especially regarding content control and computational efficiency, the techniques developed in this thesis pave the way for future exploration in this rapidly evolving field. Emerging approaches like Diffusion models and advanced Transformer architectures hold great promise for overcoming existing limitations and further pushing the boundaries of generative video technology.

List of References

[1] J. Yang and A. G. Bors, "Encoder enabled gan-based video generators," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 1841–1845, IEEE, 2022.

[2] J. Yang and A. G. Bors, "Enabling the encoder-empowered gan-based video generators for long video generation," in *ICIP*, pp. 1425–1429, IEEE, 2023.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, p. 2672–2680, 2014.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *2nd International Conference on Learning Representations (ICLR)*, 2014.

[5] Y. Wang, P. Bilinski, F. Bremond, and A. Dantcheva, "G3AN: Disentangling Appearance and Motion for Video Generation," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5264–5273, 2020.

[6] Y. Tian, J. Ren, M. Chai, K. Olszewski, X. Peng, D. N. Metaxas, and S. Tulyakov, "A good image generator is what you need for high-resolution video synthesis," in *Int. Conf. on Learning Representations (ICLR)*, 2021.

[7] S. Yu, J. Tack, S. Mo, H. Kim, J. Kim, J.-W. Ha, and J. Shin, "Generating videos with dynamics-aware implicit generative adversarial networks," in *ICLR*, 2022.

[8] I. Skorokhodov, S. Tulyakov, and M. Elhoseiny, "Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2," in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 3626–3636, 2022.

[9] S. Ge, T. Hayes, H. Yang, X. Yin, G. Pang, D. Jacobs, J.-B. Huang, and D. Parikh, "Long video generation with time-agnostic vqgan and time-sensitive transformer," *European Conference on Computer Vision (ECCV)*, 2022.

[10] T. Brooks, J. Hellsten, M. Aittala, T. chun Wang, T. Aila, J. Lehtinen, M.-Y. Liu, A. A. Efros, and T. Karras, "Generating long videos of dynamic scenes," in *NeurIPS*, 2022.

[11] Y. Wang, L. Jiang, and C. C. Loy, "Styleinv: A temporal style modulated inversion network for unconditional video generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22851–22861, 2023.

[12] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," in *arXiv preprint arXiv:1511.05644*, 2015.

[13] A. Larsen, S. Sønderby, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Int. Conf. on Machine Learning (ICML)*, pp. 1558–1566, 2016.

[14] F. Ye and A. G. Bors, "Learning latent representations across multiple data domains using lifelong VAEGAN," in *ECCV, vol. LNCS 12365*, pp. 777–795, 2020.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[16] A. Radford, "Improving language understanding by generative pre-training," 2018. `https://openai.com/index/language-unsupervised/`.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019. `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

[18] "Language models are few-shot learners," 2020. `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[19] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[20] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas, "Videogpt: Video generation using vq-vae and transformers," *arXiv preprint arXiv:2104.10157*, 2021.

[21] Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, and Y. You, "Open-sora: Democratizing efficient video production for all," March 2024. `https://github.com/hpcaitech/Open-Sora`.

[22] X. Sun, H. Xu, and K. Saenko, "TwoStreamVAN: Improving motion modeling in video generation," in *Proc. IEEE/CVF Winter Applic. in Computer Vison (WACV)*, pp. 2744–2753, 2020.

[23] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue, "Evaluating two-stream cnn for video classification," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pp. 435–442, 2015.

[24] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

[25] Y. Wan, Z. Yu, Y. Wang, and X. Li, "Action recognition based on two-stream convolutional networks with long-short-term spatiotemporal features," *IEEE Access*, vol. 8, pp. 85284–85293, 2020.

[26] F. Xiao, Y. J. Lee, K. Grauman, J. Malik, and C. Feichtenhofer, "Audiovisual slowfast networks for video recognition," *arXiv preprint arXiv:2001.08740*, 2020.

[27] A. Nebisoy and S. Malekzadeh, "Video action recognition using spatio-temporal optical flow video frames," *arXiv preprint arXiv:2103.05101*, 2021.

[28] S. Yan, X. Xiong, A. Arnab, Z. Lu, M. Zhang, C. Sun, and C. Schmid, "Multiview transformers for video recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3333–3343, 2022.

[29] M. I. Jordan, "Serial order: A parallel distributed processing approach," in *Advances in psychology*, vol. 121, pp. 471–495, Elsevier, 1997.

[30] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[31] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Inf. Proc. Systems (NeurIPS)*, p. 802–810, 2015.

[32] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[33] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, pp. 2256–2265, Proceedings of Machine Learning Research (PMLR), 2015.

[34] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 6840–6851, 2020.

[35] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *arXiv:2204.03458*, 2022.

[36] K. Mei and V. M. Patel, "Vidm: Video implicit diffusion models," in *Proceedings of the AAAI conference on artificial intelligence*, 2023.

[37] W. Weng, R. Feng, Y. Wang, Q. Dai, C. Wang, D. Yin, Z. Zhao, K. Qiu, J. Bao, Y. Yuan, *et al.*, "Art-v: Auto-regressive text-to-video generation with diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7395–7405, 2024.

[38] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023.

[39] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6626–6637, 2017.

[40] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[41] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[42] J. He, A. Lehrmann, J. Marino, G. Mori, and L. Sigal, "Probabilistic video generation using holistic attribute control," in *Proc. European Conf. on Computer Vision (ECCV), vol. LNCS 11209*, pp. 466–483, 2018.

[43] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating Videos with Scene Dynamics," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 613–621, 2016.

[44] Y. Wang, P. Bilinski, F. Bremond, and A. Dantcheva, "ImaGINator: Conditional Spatio-Temporal GAN for Video Generation," in *Proc. IEEE/CVF Winter Conf. on Applic. of Computer Vision (WACV)*, pp. 1160–1169, 2020.

[45] F. Ye and A. G. Bors, "Learning joint latent representations based on information maximization," *Information Sciences*, vol. 567, no. 8, pp. 216–236, 2021.

[46] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *Int. Conf. on Learning Representations (ICLR)*, 2017.

[47] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," *Int. Conf. on Learning Representations (ICLR), arXiv preprint arXiv:1606.00704*, 2017.

[48] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin, "Alice: Towards understanding adversarial learning for joint distribution matching," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5495–5503, 2017.

[49] K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang, "Visual autoregressive modeling: Scalable image generation via next-scale prediction," *Advances in neural information processing systems*, vol. 37, pp. 84839–84865, 2024.

[50] J. Han, J. Liu, Y. Jiang, B. Yan, Y. Zhang, Z. Yuan, B. Peng, and X. Liu, "Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis," *arXiv preprint arXiv:2412.04431*, 2024.

[51] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, *et al.*, "Stable video diffusion: Scaling latent video diffusion models to large datasets," *arXiv preprint arXiv:2311.15127*, 2023.

[52] W. Hong, M. Ding, W. Zheng, X. Liu, and J. Tang, "Cogvideo: Large-scale pretraining for text-to-video generation via transformers," *arXiv preprint arXiv:2205.15868*, 2022.

[53] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2021.

[54] Z. Xing, Q. Feng, H. Chen, Q. Dai, H. Hu, H. Xu, Z. Wu, and Y.-G. Jiang, "A survey on video diffusion models," *ACM Computing Surveys*, vol. 57, no. 2, pp. 1–42, 2024.

[55] A. Melnik, M. Ljubljanac, C. Lu, Q. Yan, W. Ren, and H. Ritter, "Video diffusion models: A survey," *Transactions on Machine Learning Research (TMLR)*, 2024.

[56] N. Aldausari, A. Sowmya, N. Marcus, and G. Mohammadi, "Video generative adversarial networks: a review," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–25, 2022.

[57] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[58] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, no. 2, 2012.

[59] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6546–6555, 2018.

[60] M. Saito, E. Matsumoto, and S. Saito, "Temporal Generative Adversarial Nets With Singular Value Clipping," in *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pp. 2830–2839, 2017.

[61] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proc. IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, pp. 1526–1535, 2018.

[62] A. Clark, J. Donahue, and K. Simonyan, "Adversarial video generation on complex datasets," in *Int. Conf. on Learning Representations (ICLR)*, 2019.

[63] L. Sheng, J. Pan, J. Guo, J. Shao, and C. C. Loy, "High-quality video generation from static structural annotations," *International Journal Computer Vision*, vol. 128, no. Nov, pp. 2552–2569, 2020.

[64] F.-T. Hong, , L. Shen, and D. Xu, "Dagan++: Depth-aware generative adversarial network for talking head video generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023.

[65] S. Gupta, A. Keshari, and S. Das, "Rv-gan: Recurrent gan for unconditional video generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2024–2033, 2022.

[66] L. Kumar, D. K. Singh, and A. Srivas, "Performance evaluation of video-to-video synthesis gan models on cityscapes dataset," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, 2023.

[67] L. Kumar and D. K. Singh, "Pose image generation for video content creation using controlled human pose image generation gan," *Multimedia Tools and Applications*, vol. 83, no. 20, pp. 59335–59354, 2024.

[68] A. Mittal, B. Kaur, and N. Kaur, "Design and development of gan model for video frame interpolation," in *2024 International Conference on Integrated Circuits, Communication, and Computing Systems (ICIC3S)*, vol. 1, pp. 1–5, IEEE, 2024.

[69] C. Li, D. Huang, Z. Lu, Y. Xiao, Q. Pei, and L. Bai, "A survey on long video generation: Challenges, methods, and prospects," *arXiv preprint arXiv:2403.16407*, 2024.

[70] W. Menapace, S. Lathuiliere, S. Tulyakov, A. Siarohin, and E. Ricci, "Playable video generation," in *CVPR*, pp. 10061–10070, 2021.

[71] J. Zhang, C. Xu, L. Liang, M. Wang, X. Wu, Y. Liu, and Y. Jiang, "DTVNet: Dynamic time-lapse video generation via single still image," in *European Conference on Computer Vision (ECCV)*, pp. 300 – 315, 10 2020.

[72] X. Shen, X. Li, and M. Elhoseiny, "Mostgan-v: Video generation with temporal motion styles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5652–5661, 2023.

[73] Q. Zhang, C. Yang, Y. Shen, Y. Xu, and B. Zhou, "Towards smooth video composition," in *The Eleventh International Conference on Learning Representations*, 2023.

[74] A. Munoz, M. Zolfaghari, M. Argus, and T. Brox, "Temporal shift gan for large scale video generation," in *Proc. of the IEEE/CVF Winter Conf. on Applications of Computer Vision (WACV)*, pp. 3179–3188, January 2021.

[75] Y. He, T. Yang, Y. Zhang, Y. Shan, and Q. Chen, "Latent video diffusion models for high-fidelity long video generation," *arXiv preprint arXiv:2211.13221*, 2022.

[76] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, *et al.*, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[77] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, D. Parikh, S. Gupta, and Y. Taigman, "Make-a-video: Text-to-video generation without text-video data," in *The Eleventh International Conference on Learning Representations*, 2023.

[78] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[79] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International conference on machine learning*, pp. 1691–1703, PMLR, 2020.

[80] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

[81] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh, "Video generation models as world simulators," 2024. `https://openai.com/research/video-generation-models-as-world-simulators`.

[82] T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly, "Towards accurate generative models of video: A new metric & challenges," *arXiv preprint arXiv:1812.01717*, 2018.

[83] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[84] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

[85] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 6299–6308, 2017.

[86] H. Dibeklioğlu, A. A. Salah, and T. Gevers, "Are you really smiling at me? spontaneous versus posed enjoyment smiles," in *Proc. European Conf. on Computer Vision (ECCV), vol. LNCS 7574*, pp. 525–538, 2012.

[87] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and M. Basri, "Actions as space-time shapes," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 1395–1402, 2005.

[88] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. Int. Conf. on Pattern Recog. (ICPR)*, vol. 3, pp. 32 – 36, 2004.

[89] K. Soomro, A. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *Technical report, CRCV-TR-12-01*, 12 2012.

[90] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.

[91] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[92] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[93] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, GA, 2013.

[94] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learning Representations (ICLR), arXiv preprint arXiv:1412.6980*, 2015.

[95] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[96] P. Ekman and D. Cordaro, "What is meant by calling emotions basic," *Emotion Review*, vol. 3, no. 4, p. 2672–2680, 2011.

[97] X. Li, X. Hong, A. Moilanen, X. Huang, T. Pfister, G. Zhao, and M. Pietikäinen, "Towards reading hidden emotions: A comparative study of spontaneous micro-expression spotting and recognition methods," *IEEE Trans. on Affective Computing*, vol. 9, no. 4, pp. 563–577, 2018.

[98] A. Blattmann, T. Milbich, M. Dorkenwald, and B. Ommer, "ipoke: Poking a still image for controlled stochastic video synthesis," in *ICCV*, pp. 14707–14717, October 2021.

[99] P. Ardino, M. De Nadai, B. Lepri, E. Ricci, and S. Lathuilière, "Click to move: Controlling video generation with sparse motion," in *ICCV*, pp. 14749–14758, October 2021.

[100] P. Esser, J. Chiu, P. Atighehchian, J. Granskog, and A. Germanidis, "Structure and content-guided video synthesis with diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7346–7356, 2023.

[101] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe, "First order motion model for image animation," *Advances in Neural Information Processing Systems*, vol. 32, pp. 7137 – 7147, 2019.

[102] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. `https://openreview.net/forum?id=Bkg6RiCqY7`.

[103] Y. Pan, Z. Qiu, T. Yao, H. Li, and T. Mei, "To create what you tell: Generating videos from captions," in *2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, pp. 1789–1798, 10 2017.