NUWA-Infinity: Autoregressive over Autoregressive Generation for Infinite Visual Synthesis

Chenfei Wu[1*] **Jian Liang**[2*] **Xiaowei Hu**[3] **Zhe Gan**[3] **Jianfeng Wang**[3]
Lijuan Wang[3] **Zicheng Liu**[3] **Yuejian Fang**[2] **Nan Duan**[1†]
[1]Microsoft Research Asia [2]Peking University [3]Microsoft Azure AI
{chewu,xiaowei.hu,zhe.gan,jianfw,lijuanw,zliu,nanduan}@microsoft.com
{j.liang@stu,fangyj@ss}.pku.edu.cn

Abstract

In this paper, we present NUWA-Infinity, a generative model for infinite visual synthesis, which is defined as the task of generating arbitrarily-sized high-resolution images or long-duration videos. An autoregressive over autoregressive generation mechanism is proposed to deal with this variable-size generation task, where a global patch-level autoregressive model considers the dependencies between patches, and a local token-level autoregressive model considers dependencies between visual tokens within each patch. A Nearby Context Pool (NCP) is introduced to cache-related patches already generated as the context for the current patch being generated, which can significantly save computation costs without sacrificing patch-level dependency modeling. An Arbitrary Direction Controller (ADC) is used to decide suitable generation orders for different visual synthesis tasks and learn order-aware positional embeddings. Compared to DALL·E, Imagen and Parti, NUWA-Infinity can generate high-resolution images with arbitrary sizes and support long-duration video generation additionally. Compared to NUWA, which also covers images and videos, NUWA-Infinity has superior visual synthesis capabilities in terms of resolution and variable-size generation. The GitHub link is `https://github.com/microsoft/NUWA`. The homepage link is `https://nuwa-infinity.microsoft.com`.

Figure 1: The painting at the top (38912 × 2048) is created as Unconditional Image Generation[HD] by NUWA-Infinity, which is trained on the famous painting *Along the River During the Qingming Festival*. The patches in the middle or at the bottom highlight more details of this AI-created painting.

[*]Both authors contributed equally to this research.
[†]Corresponding author.

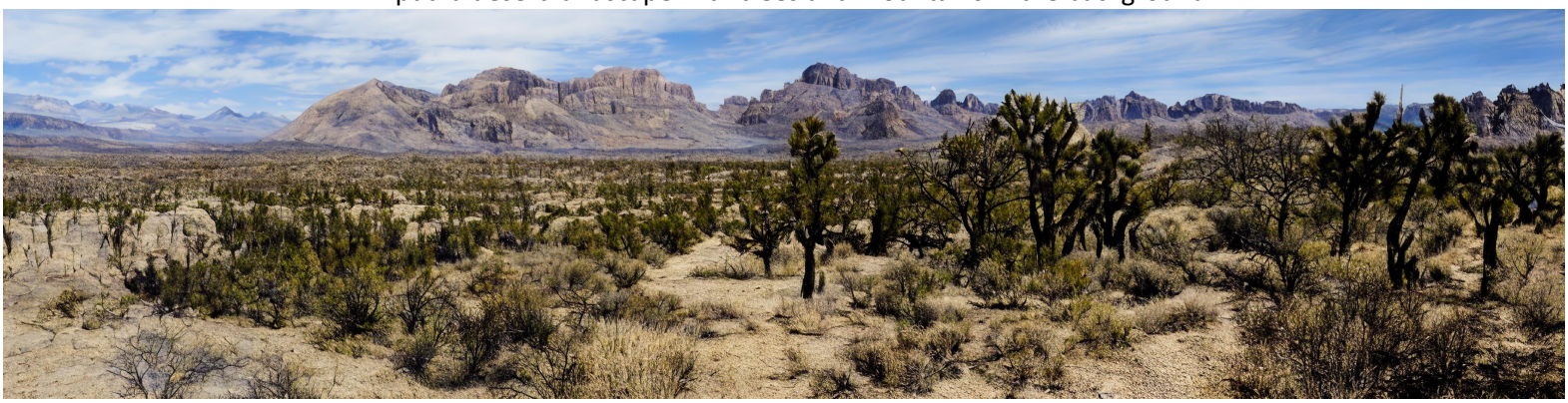
Preprint.

Figure 2: Image Outpainting$^{\mathbf{HD}}$ examples. Four examples in the upper part (2048×1024) show the outpainting of four different directions. The example in the lower part (3328×2048) shows outpainting in all directions of the famous painting of Monet, *Garden at Sainte-Adresse*.

Input: a field with a house and a cloudy sky

Input: a large lake surrounded by green vegetation

Input: a desert landscape with trees and mountains in the background



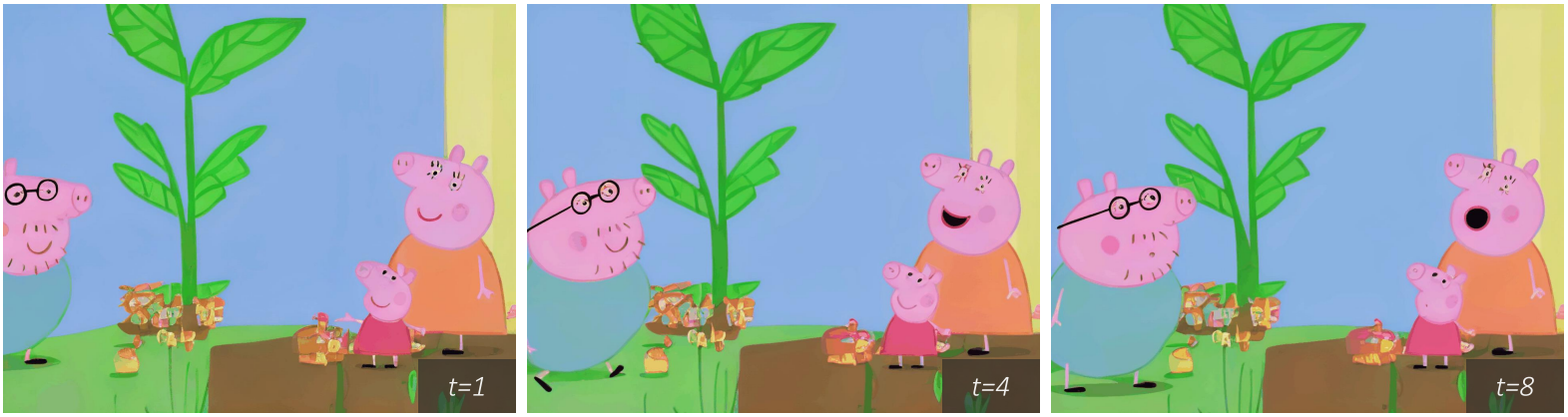
Input: a cliff with a large canyon

Figure 3: Text-to-Image$^{\mathbf{HD}}$ examples in the resolution of 4096×1024.

Figure 4: Image Animation$^{\mathbf{HD}}$ examples in the resolution of 2560×1536 with 20 frames.

Input: Daddy Pig, Mummy Pig and Peppa Pig are walking in the woods

Input: Daddy Pig and Mummy Pig are teaching Peppa Pig how to grow plants



Input: Peppa Pig is playing with her parents in the mud

Input: Daddy Pig, Mummy Pig and Peppa Pig are relaxing in the pool

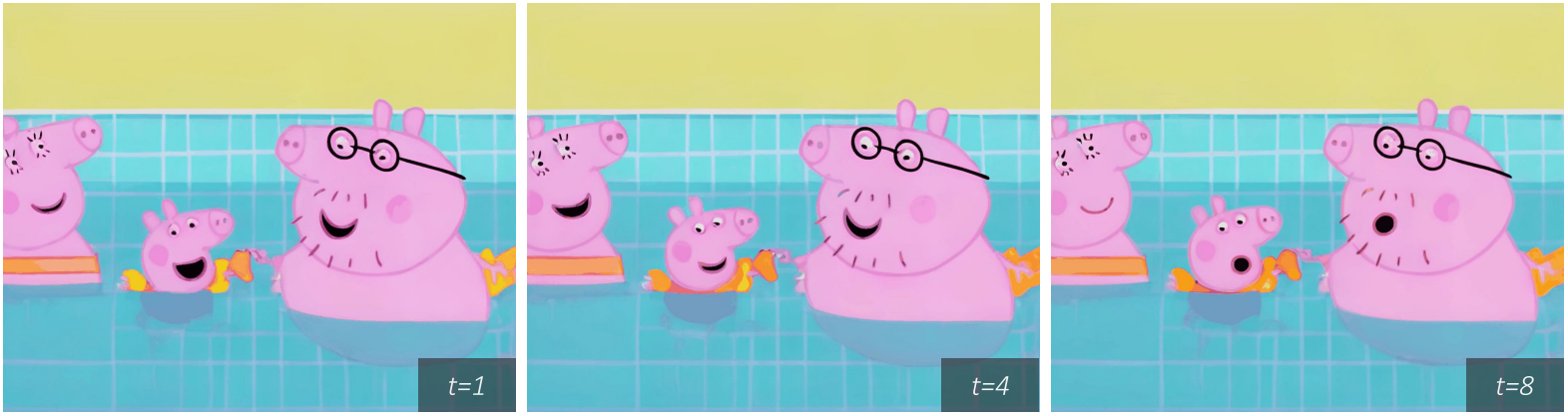


Figure 5: Text-to-Video$^{\text{HD}}$ examples in the resolution of 1280×1024 with 8 frames.

1 Introduction

The ongoing convergence of vision-language representation and modeling techniques brings new opportunities to visual synthesis research. By learning visual knowledge and patterns from a large-scale visual and multimodal corpus, recent visual synthesis models can generate images or videos based on given text, visual, or multimodal inputs and support various visual content creation tasks, such as text-to-image or video generation, image inpainting or outpainting, video prediction, etc. We also witness a notable trend in this area, where more and more work begin to explore how to generate images with higher resolutions [1, 5, 13, 18, 22], or generate videos with longer durations [3, 14, 26]. This is because high-resolution images and long-duration videos can provide better visual effects to practical applications, such as design, advertisement, presentation, entertainment, etc.

However, generating arbitrarily-sized high-quality images (in resolution) or videos (in both resolution and duration) is not a trivial task. First, compared to text generation in the NLP field, the research of generating variable-size visual content is still in its early stage, and therefore is not well studied. Some existing works [1, 6, 22, 23] try to solve this problem with a divide-and-conquer strategy, which divides images or videos into patches, trains the model to generate patches separately without considering their dependencies, and composes the generated patches to form the final image or video. As such methods do not model the dependencies between the generated patches explicitly, they struggle to guarantee the consistency of generated contents, especially when generating high-resolution images or long-duration videos. Second, different from text generation that usually follows a fixed order (e.g., left-to-right), images have two dimensions (i.e., width and height), and videos have three (i.e., width, height, and duration). These suggest that visual synthesis models should consider and model different generation orders and directions for different types of tasks.

In this paper, we use **Infinite Visual Synthesis** to denote the task of generating arbitrarily-sized high-quality images or videos, and propose **NUWA-Infinity** as a general visual synthesis model that can solve the two challenges of this task mentioned before. First, NUWA-Infinity is based on an autoregressive over autoregressive generation mechanism, where a global patch-level autoregressive model considers the dependencies between patches, and a local token-level autoregressive model considers the dependencies between visual tokens within each patch. Compared to diffusion-based approaches [4, 7, 10] that are only able to generate images with a fixed size, the autoregressive formulation naturally considers different levels of dependencies and perfectly deals with the variable-size generation task. We also introduce a Nearby Context Pool (NCP) to cache-related patches already generated as the context for the current patch being generated, which can significantly save the computation cost without sacrificing the patch-level dependency modeling. Second, we propose an Arbitrary Direction Controller (ADC) to decide suitable generation orders and learn order-aware position embeddings, which is extremely useful for image outpainting.

We evaluate NUWA-Infinity on five high-resolution visual synthesis tasks, including Unconditional Image Generation$^{\textbf{HD}}$, Text-to-Image$^{\textbf{HD}}$, Text-to-Video$^{\textbf{HD}}$, Image Animation$^{\textbf{HD}}$ and Image Outpainting$^{\textbf{HD}}$. Compared to DALL·E [19], Imagen [20] and Parti [30], which generate images with a fixed resolution (i.e., 1024×1024), NUWA-Infinity can generate high-resolution images with arbitrary sizes and support long-duration video generation additionally. Compared to NUWA [28], which also supports image and video synthesis at the same time, the generation quality of NUWA-Infinity has been improved significantly. We also show the huge application potential of NUWA-Infinity on creative visual synthesis tasks, such as image outpainting and cartoon creation from natural language descriptions. We hope this technique can help visual content creators to save time, cut costs and improve their productivity and creativity.

2 Related Work

Autoregressive Methods DALL·E [19] tokenizes each image into discrete visual tokens and trains an autoregressive model to generate visual tokens from the corresponding text. The output image is reconstructed by the VQVAE decoder, which takes the visual tokens generated by the autoregressive model as inputs. Parti [30] follows the same architecture of DALL·E, but uses ViT-VQGAN [29] to discretize and reconstruct images, which is an improved version of VQGAN from both architecture and codebook learning aspects. NUWA [28] is the first autoregressive visual synthesis pre-trained model to support both image and video generation tasks. Compared to these previous works, NUWA-Infinity introduces the autoregressive over autoregressive mechanism into the generation procedure, which enables the capability of generating variable-size images and videos.

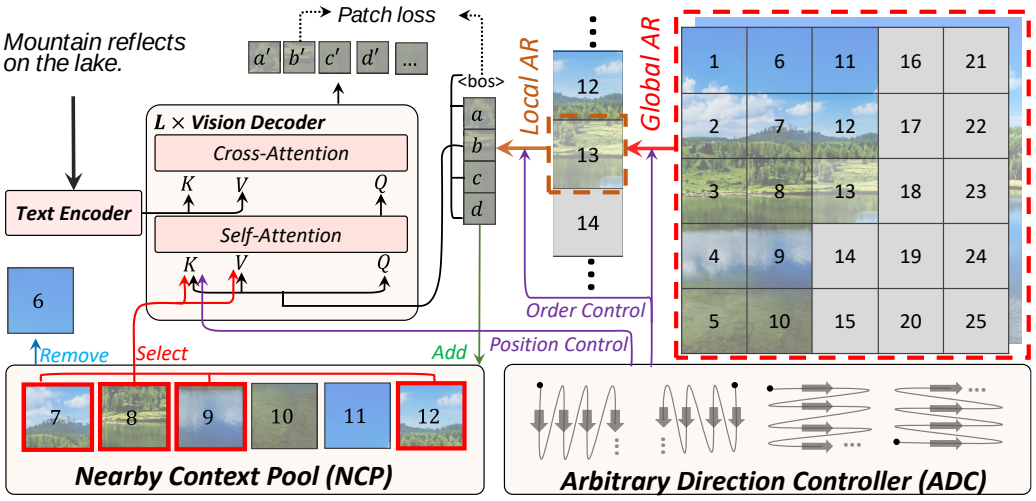


Figure 6: An overview of the proposed NUWA-Infinity model during the training process.

Diffusion Methods DALL·E 2 [18] generates image embedding from an input text based on either an autoregressive or a diffusion model, and uses a diffusion model to produce the output image. Imagen [20] uses a frozen large-scale pre-trained language model T5-XXL [17] to encode each input text, and uses two diffusion models to generate high-resolution images based on the text embeddings. Both of these two diffusion-based text-to-image generation methods cannot support arbitrarily-sized image generation, as the size of the output images is pre-defined before training and inference.

Infinite Visual Synthesis To support infinite visual synthesis, most existing works follow the divide-and-conquer strategy to first divide a large image into several patches, and then train them in an independent way. GAN-based models [22, 23] attempt to divide large images into patches and optimize each of them from global or coordinated latent space independently. Since different patches have no explicit dependency, these models struggle to merge different patches during inference, and can easily lead to inconsistent results. To address this issue, autoregressive models [1, 6] incorporate a sliding window to enforce dependencies between different patches during inference. Recently, Mask-Predict [1, 2, 31] also uses the sliding window approach, but incorporates a progressively mask-and-predict strategy to model dependencies between patches during the window sliding in inference. However, both autoregressive models and Mask-Predict models bring a huge gap between training and inference, since different patches are still trained independently but inferred in a dependent way. By introducing the autoregressive over autoregressive mechanism with Nearby Context Pool and Arbitrary Direction Controller, NUWA-Infinity can enable variable-size image and video generation, and save the computation cost without losing global dependency and consistency modeling.

3 Model

Given an input y, which can be a text or an image, the infinite visual synthesis task aims to generate an image or a video $x \in \mathbb{R}^{W \times H \times C \times F}$ with a user-specified resolution and duration, i.e., $\mathbb{P}(x|y)$, where W, H and C denote the width, height and channel of each image or video frame, respectively, F denotes the frame number. x denotes an image if $F = 1$ and denotes a video if $F > 1$.

In general, NUWA-Infinity follows an autoregressive over autoregressive model to solve this task:

$$\mathbb{P}\left(x|y\right) = \prod_{n=1}^{N} \mathbb{P}\left(p_n|p_{<n}, y\right) = \prod_{n=1}^{N} \prod_{m=1}^{M} \mathbb{P}\left(p_n^{(m)}|p_{<n}, p_n^{(<m)}, y\right) \tag{1}$$

$\prod_{n=1}^{N} \mathbb{P}\left(p_n|p_{<n}, y\right)$ denotes the global autoregressive generation procedure, where p_n is the n^{th} patch being generated, $p_{<n}$ is the previous $n-1$ patches already generated, N is the total number of patches. $\prod_{m=1}^{M} \mathbb{P}\left(p_n^{(m)}|p_{<n}, p_n^{(<m)}, y\right)$ denotes the local autoregressive generation procedure,

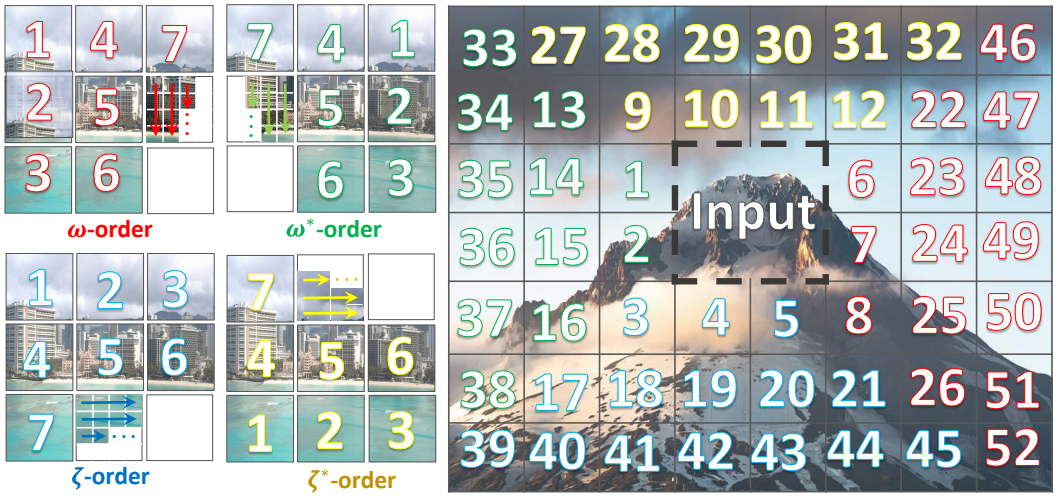


Figure 7: Illustration of patch order control in NUWA-Infinity. The left part shows four basic patch generation orders ($\omega, \omega^*, \zeta, \zeta^*$) during training. The right part shows how NUWA-Infinity performs the image outpainting task by composing these four orders. Arabic numerals indicate the order of global autoregression, arrows indicate the order of local autoregression.

where $p_n^{(m)}$ is the m^{th} visual token being generated in p_n, $p_n^{(<m)}$ is the previous $m-1$ visual tokens already generated, M is the total number of visual tokens in each patch. Each p_n is reconstructed by a pre-trained VQGAN decoder [6], which takes as input the visual token sequence $\{p_n^{(1)}, ..., p_n^{(M)}\}$. The final image or video is formed by composing all generated patches $\{p_1, ..., p_N\}$ based on the specified resolution (i.e., $W \times H$) and duration (i.e., F).

NUWA-Infinity uses an encoder-decoder architecture to model the above generation procedure. In this paper, we mainly focus on five types of high-definition (**HD**) visual synthesis tasks, including Unconditional Image Generation$^{\textbf{HD}}$, Image Outpainting$^{\textbf{HD}}$, Image Animation$^{\textbf{HD}}$, Text-to-Image$^{\textbf{HD}}$ and Text-to-Video$^{\textbf{HD}}$. In the 1^{st} task, the output image is generated by the vision decoder without any input. In the 2^{nd} and 3^{rd} tasks, the input image is fed into the vision decoder directly as the prefix to generate the output image or video. In the 4^{th} and 5^{th} tasks, the input text is encoded by the text encoder, and the output image or video is generated by the vision decoder.

One problem is that, different from text generation that follows the left-to-right order, images have two dimensions (i.e., width and height), and videos have three (i.e., width, height, and duration). These suggest the model should consider and handle different patch generation orders for different visual synthesis tasks. Motivated by this, we propose an Arbitrary Direction Controller (ADC) (Section 3.1) that can plan proper patch generation orders and learn order-aware positional embeddings.

Another problem is that, the length (i.e., $N \times M$) of the visual tokens to be generated could be extremely long, which is challenging for most existing sequence generation models. To alleviate this issue, we propose a Nearby Context Pool (NCP) (Section 3.2) to cache-related patches already generated as the context for the current patch being generated, which can significantly save the computation cost without sacrificing the patch-level dependency modeling.

We train NUWA-Infinity (Section 3.3) using high-quality image-text pairs crawled from the web, and image-video pairs extracted from high-quality videos.

3.1 Arbitrary Direction Controller (ADC)

In this subsection, we introduce Arbitrary Direction Controller (ADC), which provides two functions: **Split**, which splits images/videos and decides the patch generation order for training and inference procedures; **Emb**, which assigns order-aware positional embeddings based on the current context.

- **Split**. This function takes the shape of an existing or to-be-generated image or video x as input and returns an ordered patch sequence:

$$p_{1:N} = \text{ADC.Split}(x) \tag{2}$$

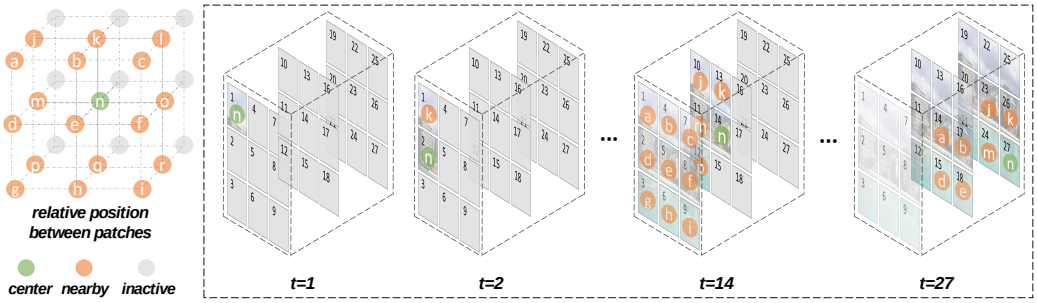


Figure 8: Illustration of dynamic position control in NUWA-Infinity.

$p_{1:N} = [p_1, ..., p_N]$ denotes the ordered patch sequence. For simplicity, we use Fig. 7 to explain how this function splits an image into ordered patches in the training and inference stages. It is straightforward to extend from images to videos by considering the temporal dimension.

The left part of Fig. 7 shows how Split$(\cdot)$ works in the training stage. We define four basic generation orders and represent them using four Greek letters, respectively, according to their writing orders: ω-order ($\downarrow\rightarrow$), ω^*-order ($\downarrow\leftarrow$), ζ-order ($\rightarrow\downarrow$), ζ^*-order($\rightarrow\uparrow$), where $*$ denotes the reversed writing order. When choosing the ω-order in training, Split$(\cdot)$ will return patches in order of the red numerical sequence (top-left example in Fig. 7). Similarly, the other three options will let NUWA-Infinity learn how to generate patches based on the corresponding orders. Note that there are more orders such as ($\uparrow\leftarrow$) or a snake-like order, but the above four basic orders and their compositions are enough to generate images in arbitrary resolutions or shapes.

The right part of Fig. 7 shows how Split$(\cdot)$ works in the inference stage for Image Outpainting$^{\mathbf{HD}}$. Given a small image of a volcanic vent as input and its relative position in the targeted image with a specified larger resolution, the goal is to synthesize this targeted image by generating all the surrounding patches of the input. In order to leverage as many contextual patches as possible when generating new patches, Split$(\cdot)$ selects a patch generation order illustrated as a numerical sequence from 1 to 52.

- **Emb**. This function assigns positional embeddings to the patch p_n being generated and the patches in c_n that are already generated and selected as the context of p_n:

$$e_n = \text{ADC.Emb}([p_n; c_n]) \tag{3}$$

It is crucial to design a proper positional embedding for the order-aware patch generation procedure, since the absolute positional embedding [25] is unable to consider and model all relative positions in image and video generation tasks. Motivated by the recent work on relative positional embedding [11, 15], we propose a dynamic positional embedding in ADC, where dynamic means the positional embeddings could change according to different situations. Fig. 8 shows 18 dynamic relative embeddings from "a" to "r". The center patch being generated (in green color) is always labeled as "n" and the relative positions of previously generated patches based on "n" are labeled by other symbols (in orange color). As a result, an embedding matrix of the size of $18 \times d$ is formed. The right part shows how the embeddings are dynamically assigned to different patches when generating a specific patch. For example, when generating the last (i.e., 27^{th}) patch, the positional embedding "n" is assigned to the current patch and the positional embeddings of "a","d","b","e","j","m","k" are assigned to patch 14, 15, 17, 18, 23, 24 and 26 in c_{27}, respectively.

3.2 Nearby Context Pool (NCP)

An image or a video could be extremely large, thus the previous patches $p_{<n}$ in Eq. 1 could have a large size. A natural idea is to only consider nearby patches as contexts. However, simply ignoring distant patches will lose long-term memory and thus harm the global consistency of the generated image or video. To address this issue, we propose a Nearby Context Pool (NCP) with three functions illustrated in Fig. 9: **Select**, which dynamically selects nearby patches as the context to promote infinite generation; **Add**, which saves multi-layer hidden states of previously generated patches to help long-term memory; **Remove**, which removes expired caches for self-cleaning.

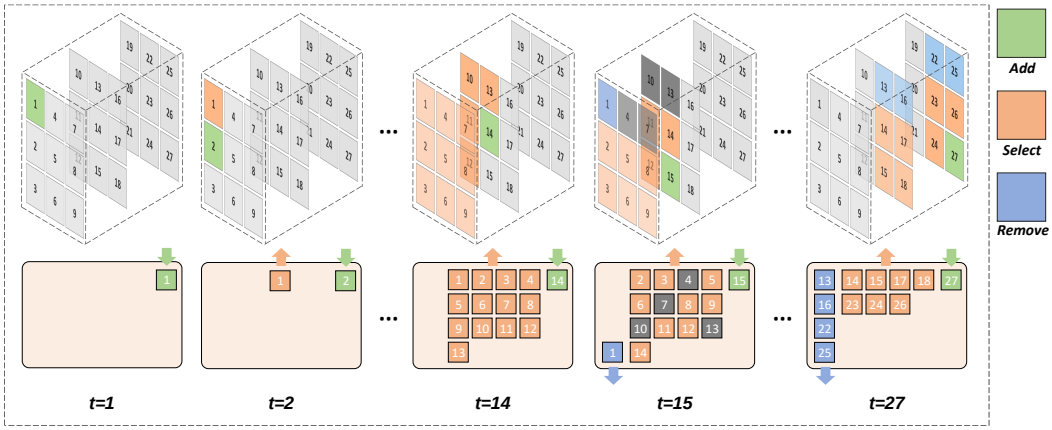


Figure 9: Illustration of NCP in ω-order with a context extent of (1,1,1).

- **Add**. This function adds the cache a_n of the patch p_n already generated into NCP:

$$\text{NCP.Add}(a_n) \tag{4}$$

  In NCP, the cache a_n of the patch p_n is defined as all the resulting multi-layer hidden states from the generation of p_n. Since NUWA-Infinity will not retain all generation history, this cache mechanism ensures the transmission of the necessary information in the whole generation procedure.

- **Select**. This function selects the context c_n for the patch p_n to be generated:

$$c_n = \text{NCP.Select}(p_n) \tag{5}$$

  In NCP, the context c_n of p_n is defined as the caches of nearby patches already generated within a pre-defined 3D extent (e^w, e^h, e^f), denoting the width extent, height extent, and frame extent. For example, when NUWA-Infinity generates the 14^{th} patch (as p_n) in Fig. 9 and the maximum extent is set as $(1, 1, 1)$, the context will include the patches from 1 to 13.

- **Remove**. This function removes the caches of those patches in NCP that no longer have any effect on the generation of future patches:

$$\text{NCP.Remove}() \tag{6}$$

  In NCP, a cache is only cleaned when it cannot serve as the context for any patch to be generated. The Remove$(\cdot)$ function will be invoked after each Add$(\cdot)$ function.

3.3 Training and Inference Strategy

This section will introduce the training and inference strategies of NUWA-Infinity in Algorithm 1 and Algorithm 2, respectively.

3.3.1 Training Strategy

Given each input-output pair $\langle y,\ x \in \mathbb{R}^{W\times H\times C\times F}\rangle$ in the pre-training corpus, we first split the visual data x into patches and then randomly select one patch generation order $p_{1:N} = [p_1, ..., p_N]$ from the four orders $\{\omega, \omega^*, \zeta, \zeta^*\}$ described in Section 3.1. A pre-trained VQGAN encoder [6] transforms all images in x into visual tokens $[p_1^{(1)}, ..., p_1^{(M)}, ..., p_N^{(1)}, ..., p_N^{(M)}]$, and each patch $p_n \in \mathbb{R}^{M\times d}$ is represented by its corresponding visual tokens $[p_n^{(1)}, ..., p_n^{(M)}]$. y is encoded by a text encoder as y', which denotes a sequence of token embeddings.

We train NUWA-Infinity based on each ordered patch sequence r. For the n^{th} patch p_n, we first select its context $c_n \in \mathbb{R}^{N^c\times L\times M\times d}$ based on the NCP described in Section 3.2:

$$c_n = \text{NCP.Select}(p_n) \tag{7}$$

where N^c denotes the number of patches in q, L denotes the layer number of the vision decoder, M denotes the visual token number in each context patch, and d denotes the dimension of each

visual token embedding. Note that N^c can be changed during training, as different patches may have different numbers of context patches in q. The positional embeddings $e_n \in \mathbb{R}^{(1+N^c)\times d}$ of p_n and its context c_n are dynamically assigned by the ADC operation described in Section 3.1:

$$e_n = \text{ADC.Emb}([p_n; c_n]) \tag{8}$$

Then, an L-layer vision decoder takes as input $p_n = [p_n^{(1)}, ..., p_n^{(M)}] \in \mathbb{R}^{M\times d}$ and c_n. In the 1^{st} layer, p_n and the 1^{st} layer hidden states $c_n^{(1)} \in \mathbb{R}^{N^c\times M\times d}$ of all patches in c_n are fed into a self-attention module, enhanced by the positional embeddings e_n:

$$\begin{aligned} Q^s &= p_n W^q \\ K^s &= [p_n; c_n^{(1)}]W^k + e_n \\ V^s &= [p_n; c_n^{(1)}]W^v \\ \tilde{Q}^s &= \text{SelfAtt}(Q^s, K^s, V^s) \end{aligned} \tag{9}$$

$Q^s \in \mathbb{R}^{M\times d}, K^s \in \mathbb{R}^{(1+N^c)\times M\times d}, V^s \in \mathbb{R}^{(1+N^c)\times M\times d}$ are queries, keys and values, respectively, $W^q, W^k, W^v \in \mathbb{R}^{d\times d}$ are parameters to be learned, $\tilde{Q}^s$ denotes the attended results.

Algorithm 1: Training Strategy

Input: images or videos x, optional text y
Output: optimized NUWA-Infinity model
Initial Arbitrary Direction Controller **ADC**;
Initial Nearby Context Pool **NCP** $\leftarrow \emptyset$;
$p_{1:N} \leftarrow$ **ADC.Split**(x);
for all n from 1 to N **do**
  $c_n \leftarrow$ **NCP.Select**(p_n);
  $e_n \leftarrow$ **ADC.Emb**$([p_n; c_n])$;
  $a_n, \hat{p}_n \leftarrow$ **NUWA**(p_n, c_n, e_n, y);
  NCP.Add(a_n);
  NCP.Remove$()$;
  $\mathcal{L}_n = \text{CrossEntropy}(p_n, \hat{p}_n)$;
  optimize $\mathcal{L}_n$;
end

For the Text-to-Image**HD** task, $\tilde{Q}^s$ and y' are further fed into a cross-attention module as shown in Eq. (10).

$$\begin{aligned} &Q^c = \tilde{Q}^s W^{q'}, \quad K^c = y' W^{k'}, \quad V^c = y' W^{v'} \\ &\tilde{Q}^c = \text{CrossAtt}(Q^c, K^c, V^c) \end{aligned} \tag{10}$$

where $\tilde{Q}^c \in \mathbb{R}^{M\times d}, K^c \in \mathbb{R}^{T\times d}, V^c \in \mathbb{R}^{T\times d}$ are queries, keys and values, respectively, $W^{q'}, W^{k'}, W^{v'} \in \mathbb{R}^{d\times d}$ are parameters to be learned, T is the number of token embeddings in y', $\tilde{Q}^c$ denotes the attended results.

By feeding $\tilde{Q}^s$ (for tasks without text input) or $\tilde{Q}^c$ (for tasks with text input) into a feed forward network, the output of the 1^{st} layer $\hat{p}_n^{(1)} \in \mathbb{R}^{M\times d}$ is obtained:

$$\hat{p}_n^{(1)} = \text{FFN}(\tilde{Q}^c) \tag{11}$$

By iteratively stacking Eq. (9)$\sim$(11) into L layers, we obtain $\hat{p}_n^{(1)}, \hat{p}_n^{(2)}, ..., \hat{p}_n^{(L)} \in \mathbb{R}^{M\times d}$. p_n and the previous $L-1$ layer outputs are concatenated to obtain a L-layer cache of the n^{th} patch p_n:

$$a_n = [p_n; \hat{p}_n^{(1)}; \hat{p}_n^{(2)}; ..; \hat{p}_n^{(L-1)}] \tag{12}$$

where $a_n \in \mathbb{R}^{L\times M\times d}$. For simplicity, the procedure from Eq. (9) to Eq. (12) is defined as NUWA:

$$\hat{p}_n, a_n = \text{NUWA}(p_n, c_n, e_n, y) \tag{13}$$

where $\hat{p}_n = \hat{p}_n^{(L)}$ denotes the output embeddings. Then, NCP will collect the cache of p_n to help the prediction of the next patches and conduct a self-cleaning to remove useless patches, as shown in Eq. (14).

$$\begin{aligned} &\text{NCP.Add}(a_n) \\ &\text{NCP.Remove}() \end{aligned} \tag{14}$$

Finally, the cross-entropy loss is used to optimize model parameters based on $\hat{p}_n$ and the ground-truth p_n.

Algorithm 2: Inference Strategy

Input: a text y or an image h
Output: generated image/video x
Initial Arbitrary Direction Controller **ADC**;
Initial Nearby Context Pool **NCP** $\leftarrow \emptyset$; $q_{1:K} \leftarrow$ **ADC.Split**(h);
for all k from 1 to K **do**
    $c_k \leftarrow$ **NCP.Select**(q_k);
    $e_k \leftarrow$ **ADC.Emb**$([q_k; c_k])$;
    $a_k, \hat{q}_k \leftarrow$ **NUWA**(q_k, c_k, e_k, y);
    NCP.Add(a_k);
    NCP.Remove$()$;
end
} Image Condition Pre-caching

$p_{1:N} \leftarrow$ **ADC.Split**(x);
for all n from 1 to N **do**
    $c_n \leftarrow$ **NCP.Select**(p_n);
    $e_n \leftarrow$ **ADC.Emb**$([p_n; c_n])$;
    $a_n, \hat{p}_n \leftarrow$ **NUWA**$(\emptyset, c_n, e_n, y)$;
    NCP.Add(a_n);
    NCP.Remove$()$;
    $x' \leftarrow [x'; \hat{p}_n]$;
end
if target x is an image **then**
    return **VQGANDecoder**(x')
else
    return **PixelGuidedVQGANDecoder**(x')
end
} Pixel-Guided VQGAN

3.3.2 Inference Strategy

NUWA-Infinity can support various visual synthesis scenarios, and we focus on five tasks in this paper: Unconditional Image Generation[HD], Image Outpainting[HD], Image Animation[HD], Text-to-Image[HD] and Text-to-Video[HD]. There are two specific designs for the last four tasks: Image Condition Pre-caching and Pixel-Guided VQGAN.

Image Condition Pre-caching For Image Outpainting[HD] and Image Animation[HD], the input is an image condition h and the output is a spatial extended image or a temporal extended video. A VQGAN encoder is used to encode h into a list of patches with corresponding visual tokens, where K denotes the number of conditional patches. Then, these patches and visual tokens are fed into the vision decoder (Eq. 8$\sim$14) as a prefix to initialize NCP, which will be used next to generate the extended image or following video frames.

Pixel-Guided VQGAN For Image Animation[HD] and Text-to-Video[HD], the output are videos. Since traditional VQVAE is trained only on images, simply decoding a video frame-by-frame will lead to inconsistency between frames. To solve this issue, we propose a Pixel-Guided VQGAN (PG-VQGAN), as shown in Fig. 10. Different from traditional VQGAN which is trained on images independently, we sample 2 consecutive frames $n-1$ and n as a training pair and use the pixel-level information of the $n-1$ frame to enhance the decoder of frame n. In detail, the frame $n-1$ is encoded with the same number of layers as the traditional VQGAN decoder, and the output of each encoder layer is fused with the corresponding output layer of the VQGAN decoder. We simply use an element-sum operation as the fusion strategy and observe promising results (see Fig. 10). When decoding the first frame during inference, Image Animation[HD] has a ground-truth $n-1$ frame. However, For Text-to-Video[HD], since there are no ground-truth frames, we instead use traditional VQGAN to decode the first frame and Pixel-Guided VQGAN to decode the following frames.

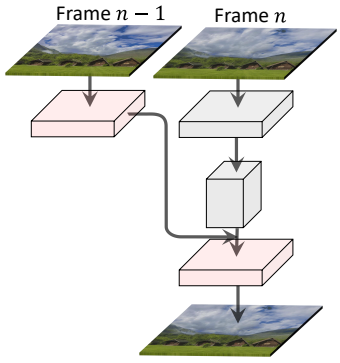


Figure 10: Pixel-Guided VQGAN.

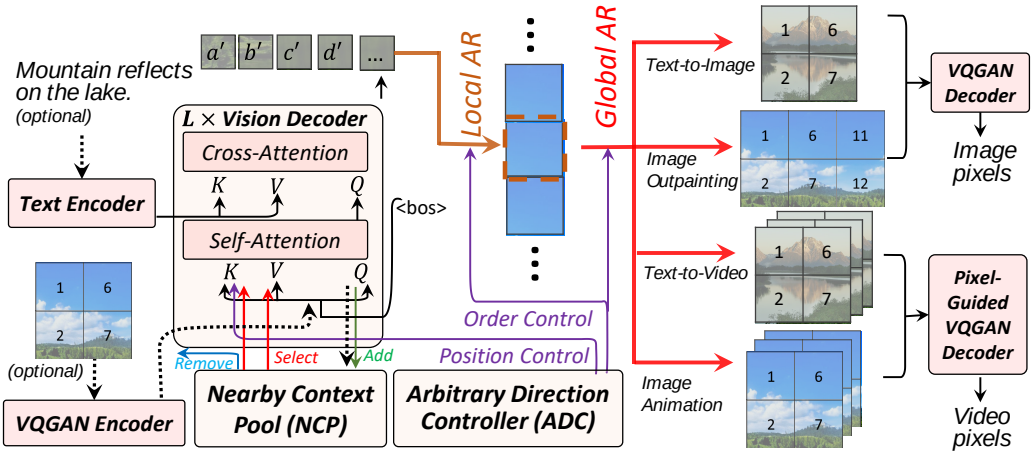


Figure 11: Inference pipeline of NUWA-Infinity for downstream tasks.

4 Experiments

4.1 Datasets

Different from most visual synthesis works, NUWA-Infinity focuses on generating images and videos with high resolutions and long durations. As a result, most existing datasets cannot be used in training or evaluation. To evaluate the ability of NUWA-Infinity on the five tasks mentioned before, we build four datasets with high resolutions ($\geqslant 1024^2$) in the following:

- **RQF**. We build a new dataset, Riverside of Qingming Festival (RQF), based on the version of *Along the River During the Qingming Festival* drawn by Qiu Ying[3]. This version is 30.5 centimeters high and 987 centimeters wide in reality, and we download a digital print with 4200×135912 pixels. We resize the whole image to 2048×66270 resolution and split it into several overlapped 2048×2048 patches instead of non-overlapping ones. We end with a dataset of 128 images. We train NUWA-Infinity with the $< \emptyset, image >$ pairs on the dataset with 2048×2048 resolution but qualitatively evaluate Unconditional Image Generation$^{\mathbf{HD}}$ at higher resolution 2048×38912.
- **LHQC**. We build a new dataset, Landscape High Quality with Captions (LHQC), based on the publicly available dataset LHQ [22]. The original LHQ dataset consists of 90K high-resolution ($\geqslant 1024^2$) nature landscapes. To support the text prompt, we use the image captioning model from [27] to generate the captions for the dataset first and manually fix some errors in the generated results. We finally obtain a dataset with 90K text-image pairs. We spilt the dataset into two splits: train (85K) and test (5K). We train NUWA-Infinity with the $< text, image >$ pairs on the train split and evaluate Text-to-Image$^{\mathbf{HD}}$ and Image Outpainting$^{\mathbf{HD}}$ on the test split.
- **LHQ-V**. We build a new dataset, Landscape High Quality for Videos (LHQ-V), based on the videos scrapped from the www.pexels.com website. We first build a query set with 100 landscape related keywords (e.g., "sky", "forest", "cloud"). Then, we query the website using these keywords and obtain 85K high-resolution videos. To further make the dataset cleaner, we use Mask R-CNN [8] to detect objects from these videos and remove videos containing objects that are not related to landscape (i.e., "table", "human", "computer"). Finally, we obtain a dataset with 40K videos. We split the dataset into two splits: train (38K) and test (2K). We train NUWA-Infinity with the $< \emptyset, video >$ pairs on the train split and evaluate Image Animation$^{\mathbf{HD}}$ on the test split.
- **PeppaPig**. We build a new dataset, PeppaPig, based on the famous cartoon PeppaPig. We collect Season 1-4 videos of PeppaPig and split the videos into multiple clips based on the timeline of subtitles. We then ask 20 trained annotators to annotate the captions for these videos. If the clip is not smooth, the annotators are asked to provide a "N/A" caption. We also ask a meta annotator to check these captions. Finally, we remove videos with the "N/A" caption and obtain 10K text-video pairs. We split the dataset into two splits: train (9K) and test (1K). We train NUWA-Infinity with the $< text, video >$ pairs on the train split and evaluate Text-to-Video$^{\mathbf{HD}}$ on the test split.

[3] https://www.comuseum.com/painting/masters/qiu-ying/

Setting	UIG$^{\mathbf{HD}}$	T2I$^{\mathbf{HD}}$ & IO$^{\mathbf{HD}}$	IA$^{\mathbf{HD}}$	T2V$^{\mathbf{HD}}$
VQVAE				
Backbone	VQGAN	VQGAN	PG-VQGAN	PG-VQGAN
Codebook	16384	16384	16384	16384
Dimension	256	256	256	256
Compression Ratio	16	16	16	16
Transformer				
Layer Number L	24	24	24	24
Hidden Dimension d	1280	1280	1280	1280
Head Number	20	20	20	20
Self-attention	✓	✓	✓	✓
Cross-attention	✗	✓	✗	✓
NCP & ADC				
Patch Number N	64	16	80	80
Patch Tokens M	256	256	256	256
Context Extent	(2, 2, 0)	(2, 2, 0)	(1, 1, 3)	(2, 2, 3)
Dataset				
Name	RQF	LHQC	LHQ-V	PeppaPig
Train Scale	128	85K	38K	10K
Test Scale	N/A	5K	2K	1K
Training & Inference				
Epoch	6000	50	50	150
Visual Size $W \times H$	2048×2048	1024×1024	1024×1024	1024×1024
Frame Length F	N/A	N/A	5	5
Text Length	N/A	77	N/A	77
Batch Size	128	512	256	256
Learning Rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Warmup Ratio	2%	5%	5%	5%

Table 1: Implementation details of model training for different tasks.

4.2 Implementation Details

The training of NUWA-Infinity can be split into two stages. In the first stage, all images are cropped into 1024×1024 and videos are cut into $1024 \times 1024 \times 5$ with 5 fps. Then, they are encoded into discrete visual tokens using the VQGAN model with a compression ratio of 16. In the second stage, we train the model using Adam optimizer [12] with a learning rate of 1×10^{-4}, a batch size of 256, and warm-up 5% of total 50 epochs. We train four models on four datasets and inference on five tasks. More settings for different tasks can be found in Tab. 1.

4.3 Metrics

- **FID/Block-FID**. Fréchet Inception Distance (FID) [9] is used to calculate the quality of the generated image. In this work, we also propose Block-FID, which splits a large image into several blocks and calculates the average Fréchet Inception Distance of all blocks.

- **IS**. Inception Score (IS) [21] is a common metric to calculate the diversity of the generated results. A higher IS indicates the model generates more diversified results.

- **FVD**. Fréchet Video Distance (FVD) [24] is widely used to calculate the quality of generated videos. It measures the distance between the ground-truth video and the generated video. A lower FVD denotes a higher similarity.

- **CLIP-SIM**. Recently, the CLIP Similarity Score (CLIP-SIM) [16] is used to measure the semantic consistency between images and text. We use the CLIP model to calculate the similarity score between the generated image and given text to judge its semantic consistency.

4.4 Main Results

4.4.1 Unconditional Image Generation$^{\mathbf{HD}}$

Unconditional Image Generation$^{\mathbf{HD}}$ aims to generate images without conditions. Fig. 12 and 13 show a single long image generated by the model trained on the RQF dataset for Unconditional Image Generation$^{\mathbf{HD}}$. The generated image has an extremely high resolution of 38912×2048. To better fit the page, we split the complete image into 6 splits, each having a resolution of 6485×2048. The generated results demonstrate the following abilities of NUWA-Infinity:

- **Infinity ability**. NUWA-Infinity can generate visual content with large size. Although trained on 2048×2048 patches as illustrated in Tab. 1, NUWA-Infinity can generate images 19 times longer than each training instance. This is attributed to our proposed NCP module, which makes the computation grow linearly with the output size, as only a small number of previously generated patches in the context are used during inference.
- **Creation ability**. By comparing the generated results with the original painting, we find that NUWA-Infinity has a strong creative ability. For example, for the gate wall of the second split in Fig. 12, it is a composition of multiple walls in different directions. Also, for the first split in Fig. 13, NUWA-Infinity generates a lot of pedestrians and houses. Many different people walk together, which leads to a crowded situation in this split. Note that these scenes do not appear in the original painting, and they are fully created by NUWA-Infinity.

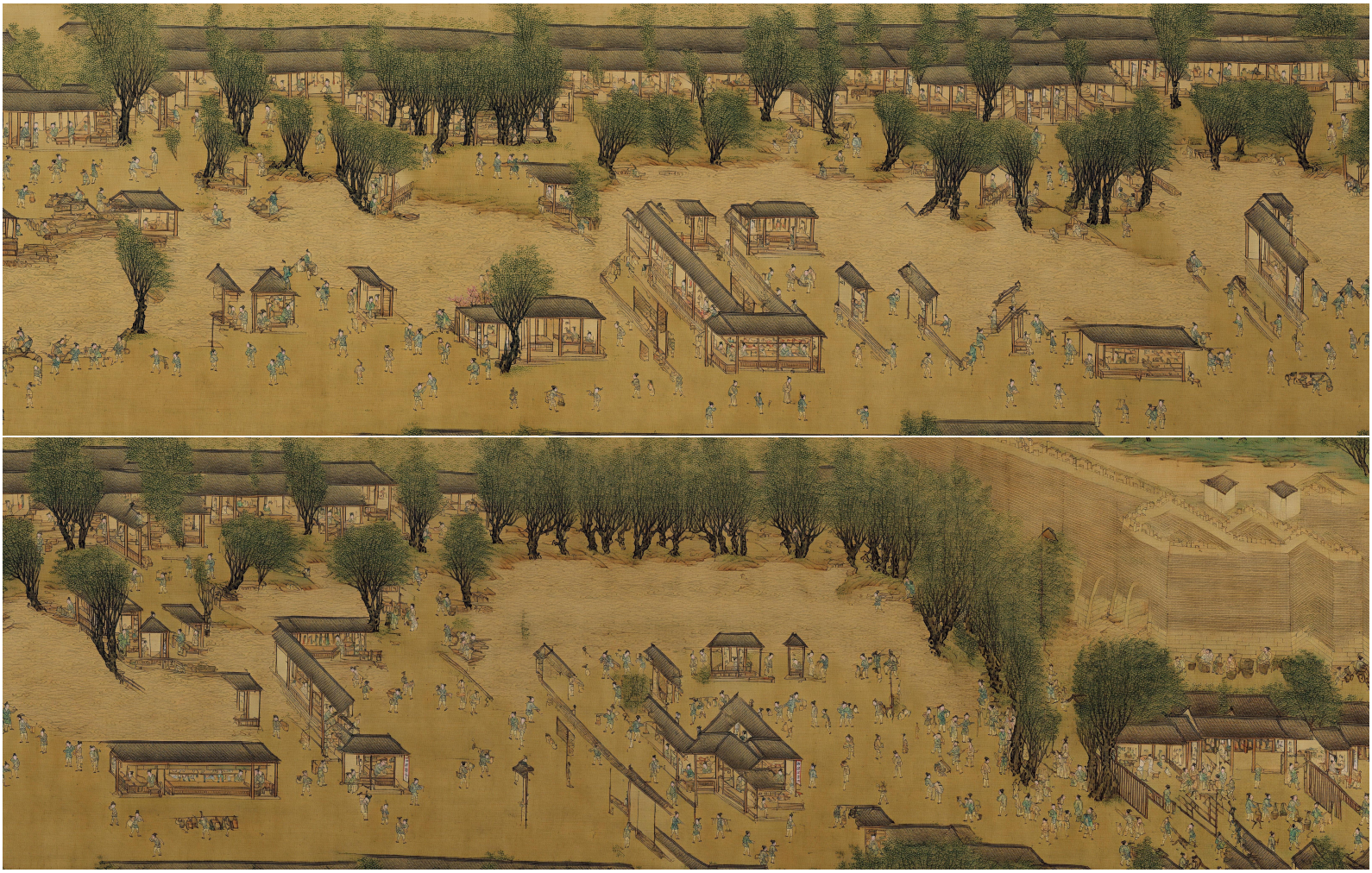


Figure 12: Part I of a huge image (38912×2048) synthesized in Unconditional Image Generation$^{\mathbf{HD}}$ task on RQF dataset. Splits are connectable by row, each has a resolution of 6485×2048.

Figure 13: Part II of the huge image (38912 × 2048) illustrated in Fig. 12.

- **Local details and global consistency**. The generated results also show decent local details and global consistency. In NUWA-Infinity, the local autoregression generates visual tokens one by one, thus the human faces, human gestures, leaves of trees, roof tiles, and many other details are clearly painted. The global autoregression generates patches one-by-one. As shown in Fig. 13, the human figures gradually dwindle, and then the picture transitions to the mountains. The smooth transitions make the picture look natural globally even though it is extremely long.

4.4.2 Text-to-Image$^{\mathbf{HD}}$

Text-to-Image$^{\mathbf{HD}}$ aims to generate an image based on the input text. For a fair comparison, all models are trained from scratch on the LHQC dataset. As shown in Tab. 2, when generating images with 1024×1024 resolution, NUWA-Infinity outperforms AR-based models Taming Transformer [6] and Mask-Predict model MaskGIT [1], in visual quality (Block-FID), semantic consistency (CLIP-SIM), and diversity (IS). When generating images of size 4096×1024 which is 4 times as long as the training images, the performance of MaskGIT [1] decreases rapidly but NUWA-Infinity still maintains excellent visual quality with Block-FID of 15.65. As also shown in Fig. 14, NUWA-Infinity generates significantly better results, and the reflection of the hill can be clearly seen. Note that we did not compare with DALL·E 2 [18], Imagen [20], or Parti [30] directly because of three reasons: (i) all of them do not support arbitrary-large visual synthesis (i.e, generating images with 4096×1024 resolution); (ii) they did not make their pre-trained models public; and (iii) NUWA-Infinity focuses on enabling infinite visual synthesis and is not pre-trained on large-scale datasets, thus it is hard to make a direct comparison.

Method	Block-FID↓	Block-FID(×4)↓	IS↑	CLIP-SIM↑
Taming [6]	38.89	46.37	4.58	0.2662
MaskGIT [1]	24.33	45.76	4.61	0.2754
NUWA-Infinity	**9.71**	**15.65**	**4.98**	**0.2807**

Table 2: Comparisons on LHQC dataset for Text-to-Image$^{\mathbf{HD}}$ task.

Figure 14: Samples on LHQC dataset for Text-to-Image$^{\mathbf{HD}}$. Left: 1024×1024, Right: 1024×4096. For a fair comparison, we did not provide input sketch for Taming Transformer.

Method	FVD↓
NUWA-Infinity (NAR)	368.16
NUWA-Infinity (P-NAR)	165.39
NUWA-Infinity (AR)	**62.57**

Table 3: Comparisons on LHQ-V dataset for Image Animation$^{\mathbf{HD}}$ task.

4.4.3 Image Animation$^{\mathbf{HD}}$

Image Animation$^{\mathbf{HD}}$ aims to generate a video based on an input image. We build two baseline models as a comparison shown in Tab. 3. All three models are globally autoregressive, but they use different local generative methods (shown in round brackets). NUWA-Infinity (AR) is our default autoregressive over autoregressive model with a local autoregressive mechanism to generate a patch. NUWA-Infinity (NAR) can be viewed as an autoregressive over non-autoregressive model, as the tokens inside a patch are generated locally in parallel instead of autoregressively. NUWA-Infinity (P-NAR) can be viewed as an autoregressive over progressive non-autoregressive model, as locally, the tokens inside a patch are generated by a Mask-Predict method [1] introduced in Sec. 2. NUWA-Infinity achieves significantly better performance than baselines, with an FVD score of 62.57. Fig. 15 provides a qualitative comparison between the generated 60-frame videos from the same input image of 1024×1024 resolution. We find that NUWA-Infinity with a default local AR mechanism can generate more realistic images. However, we do find a speed-performance trade-off as AR generation takes more time compared with NAR and P-NAR. We will discuss it in Sec. 4.5.3.

Figure 15: Samples on LHQ-V dataset for Image Animation$^{\mathbf{HD}}$ task. We compare local NAR, P-NAR, and AR by generating 60 frames with a high resolution of 1024×1024.

4.4.4 Image Outpainting$^{\mathbf{HD}}$

Image Outpainting$^{\mathbf{HD}}$ aims to generate an out-painted image based on the input image. We do not train NUWA-Infinity for the Image Outpainting$^{\mathbf{HD}}$ task, but used the model trained for Text-to-Image$^{\mathbf{HD}}$ directly. To evaluate the model's ability of outpainting in all directions, we set up four settings: Right Extend $\Rightarrow$, Left Extend $\Leftarrow$, Down Extend $\Downarrow$ and Up Extend $\Uparrow$. For example, in the Right Extend $\Rightarrow$ setting, we input the image with the left half in the LHQC dataset and ask the model to predict the right half. Since for the output image in this task, half is ground-truth and half is extended, we calculate Block-FID between the extended area and the same area in the ground-truth test set of LHQC. As shown in Tab. 4, NUWA-Infinity significantly outperforms Taming [6] and MaskGIT [1] by a large margin in all four directions. Since the model trained by Text-to-Image$^{\mathbf{HD}}$ also supports a text prompt, we also try to outpaint the image by adding text controlling, and we find better performance on Down Extend $\Downarrow$ and Up Extend $\Uparrow$, while similar performance on Right Extend $\Rightarrow$ and Left Extend $\Leftarrow$ compared with one without text. We hypothesize that it is because the upper or lower half of the image contains less information, while the left or right half has more visual semantic hints. Note that for a fair comparison, Taming and MaskGIT use text prompt by default.

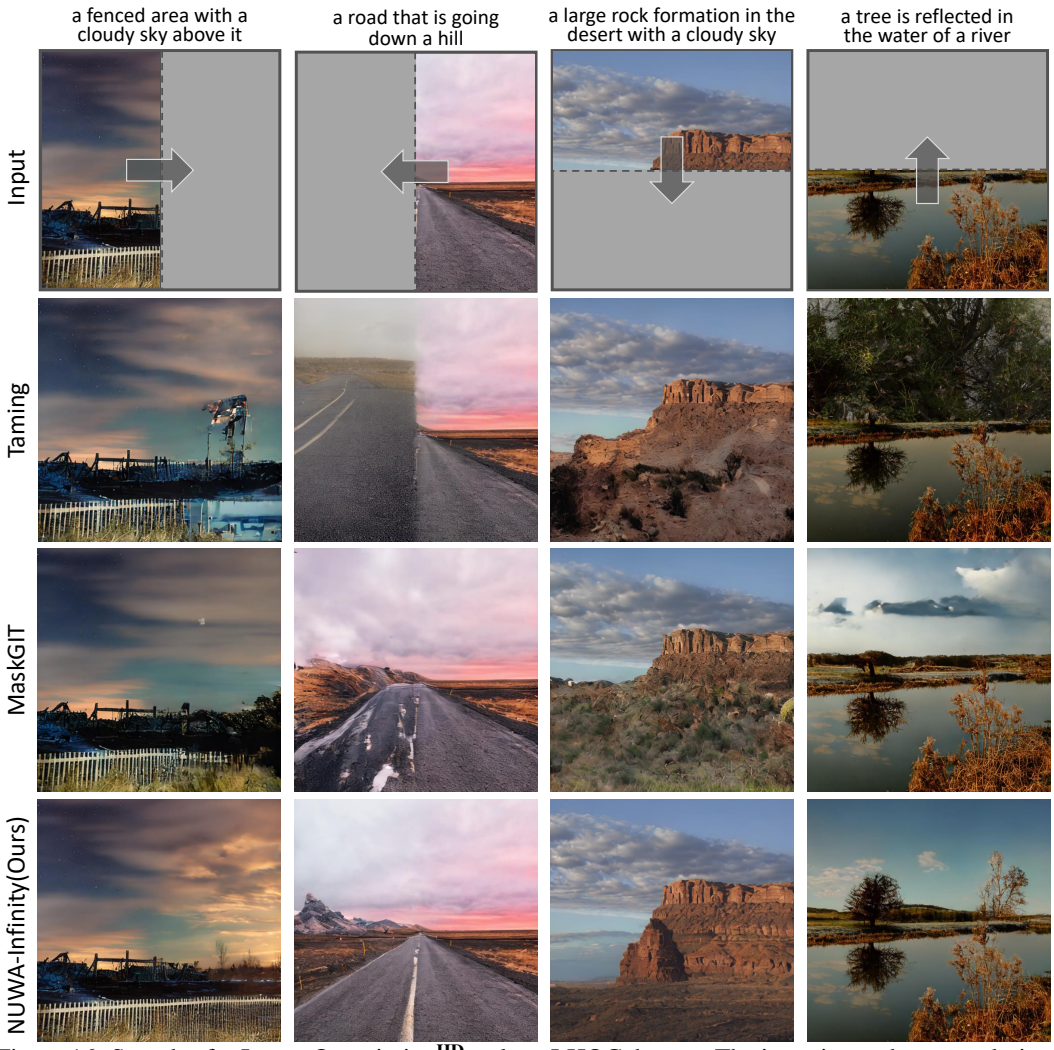


Figure 16: Samples for Image Outpainting$^{\mathbf{HD}}$ task on LHQC dataset. The input image has a resolution of half of 1024×1024.

In Fig. 16, we provide four input images illustrating four directions. Taming Transformer only successfully generates the input image in the third column, as it predicts the lower half of the hill based on the upper half. This is because Taming Transformer only trains token-by-token in $\zeta - order$, and this order only fits the down extension. MaskGIT successfully predicts another half in all directions benefited by its bidirectional masked language model. Compared with MaskGIT, NUWA-Infinity can generate more realistic results. For example, in the fourth column, when inputting the reflection of a tree in the lake, NUWA-Infinity successfully generates the most consistent tree on the shore.

Method	Block-FID↓			
	Right Extend ⇒	Left Extend ⇐	Down Extend ⇓	Up Extend ⇑
Taming [6]	22.53	N/A	26.38	N/A
MaskGIT [1]	14.68	14.81	25.57	25.38
NUWA-Infinity w/o text	**6.43**	**6.71**	11.47	8.03
NUWA-Infinity	6.45	6.72	**9.84**	**7.43**

Table 4: Comparisons on LHQC dataset for Image Outpainting$^{\mathbf{HD}}$ task.

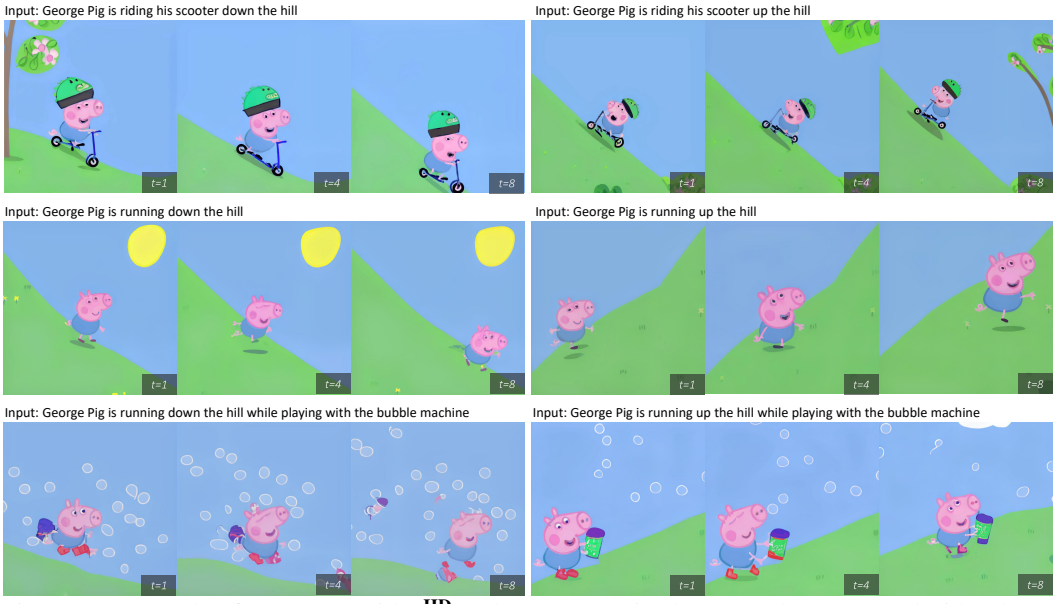


Figure 17: Samples for Text-to-Video$^{\mathbf{HD}}$ task on PeppaPig dataset. The generated video has a resolution of 1024×1024 with 8 frames.

4.4.5 Text-to-Video$^{\mathbf{HD}}$

Text-to-Video$^{\mathbf{HD}}$ aims to generate high-resolution videos given the input texts. As shown in Fig. 8, NUWA-Infinity can generate videos considering both objects and motions, and distinguish the motion "up the hill" from "down the hill". Based on this ability, Fig. 17 shows a simple video story generated by four sentences. We believe this technology could help cartoon design in the future.

4.5 Ablation Studies

We conduct detailed ablations on three components of our model, ADC, NCP, and Vision Decoder.

4.5.1 Ablation Study on ADC

Patch size of ADC.Split As introduced in Sec. 3.3, each patch representation has a size of $p_n \in \mathbb{R}^{M \times d}$. The horizontal axis of Fig. 18 shows different patch sizes M, and the vertical axis shows FID scores for Text-to-Image$^{\mathbf{HD}}$. The blue line shows the FID score of generating image resolution of 1024×1024. The red line shows the Block-FID score of generating image resolution of 1024×4096. A smaller patch size harms the FID score of the generated image, but a larger patch size requires larger GPU memory. When patch size $M = 256$, a good balance between performance and memory can be achieved.

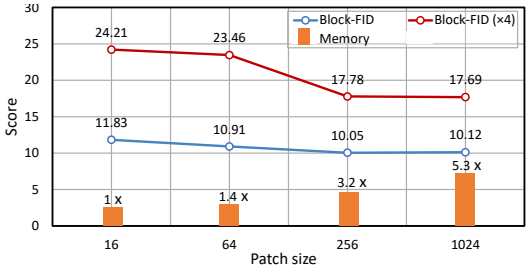
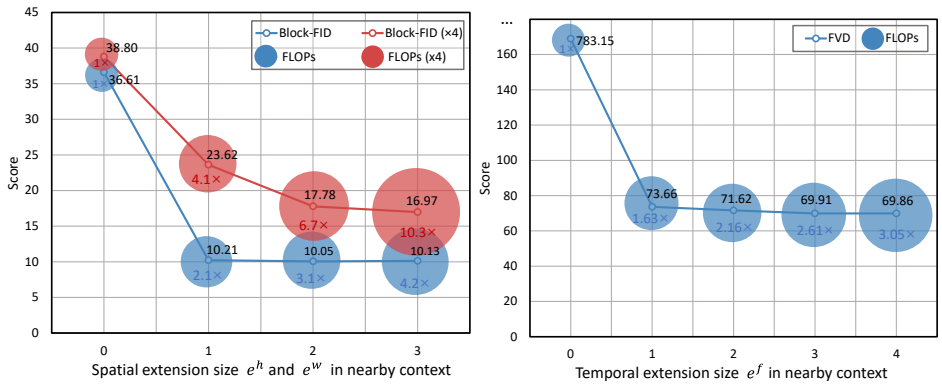


Figure 18: Impact of patch size.

Feed position of ADC.Emb As introduced in Sec. 3.1, ADC dynamically provides relative positional embeddings during training and inference stages. In Eq. 9, the relative positional embedding e_n is added to the key of self-attention. We call it pre-feeding, as the positional information is fed before the computation of attention. This is different from traditional post-feeding in Transformers, where the positional information is fed after the computation of attention. Tab. 5 shows that pre-feeding brings better performance than post-feeding. This is because pre-feeding controls which contexts to attend to with positional information while post-feeding only adjusts the attention distributions after the attention.

RPE	Block-FID↓	Block-FID(×4) ↓
Pre	**10.05**	**17.78**
Post	10.47	18.89

Table 5: Impact of feed position.



(a) Spatial extension.

(b) Temporal extension.

Figure 19: Ablation results on extents in NCP.

4.5.2 Ablation Study on NCP

Caches in NCP As introduced in Sec. 3.2, the Add operation saves multi-layer hidden states of previously generated patches as "caches". This allows information transmission between patches during training and inference. In other words, even though distant patches are removed from NCP, their information can be still captured in the hidden states of the nearby patches in NCP. To verify the effectiveness of this design, we train another model without using the caches in NCP and show its result of Text-to-Image$^{\mathbf{HD}}$ in Tab. 6. The NCP design with information transmission significantly outperforms the one without information transmission.

Context	Block-FID↓	Block-FID(×4) ↓
w/o caches	15.80	38.32
w/ caches	**10.21**	**23.62**

Table 6: Ablation results in caches in NCP.

Extents in NCP As introduced in Sec. 3.2, NCP saves caches in a 3D extent (e^w, e^h, e^f). Fig. 19a shows the comparisons between different extent size of (e^w, e^h, e^f) for Text-to-Image$^{\mathbf{HD}}$. Although larger extent size brings better Block-FID scores, the performance becomes limited when $e^h, e^w > 2$ while computation improves significantly. For videos, Fig. 19b shows the impact of temporal extent on Image Animation$^{\mathbf{HD}}$. We find the FVD improvements become limited if the e^f increases to 3. As a result, we choose $(e^w, e^h, e^f) = (2, 2, 3)$ as our default setting.

4.5.3 Ablation Study on Vision Decoder

Tab. 7a shows ablations on different vision decoders. NUWA-Infinity is an autoregressive over autoregressive model and it follows the autoregressive (AR) formulation in the vision decoder. We also try another two vision decoders based on a non-autoregressive (NAR) formulation and a progressive autoregressive (P-NAR) formulation. For these two models, the global autoregression is still maintained, while the local autoregression is changed into NAR and P-NAR, respectively. We find AR-based vision decoder achieves the best performance and NAR-based vision decoder achieves the fastest speed.

Tab. 7b shows two settings of NUWA-Infinity: Base and Large. We find that the base model can also achieve acceptable performance compared with the large model. This is due to the limited training samples in the dataset. In this paper, we focus on the effectiveness of NUWA-Infinity architecture, instead of large-scale pre-training. We will pre-train NUWA-Infinity with more data in the future.

Tab. 7c shows when to optimize the loss between the predicted patch of vision decoder and the ground-truth patches. During training, as long as a patch is predicted, NUWA-Infinity calculates the patch loss and optimizes it immediately instead of accumulating the gradients of all patches. We simply call this mechanism "patch loss" and the accumulated one "accumulated loss". We find that patch loss accelerates convergence from 70 epochs to 50 epochs and improves the Block-FID score compared with the accumulated loss. This is because patch loss will share optimized parameters between each patch, which helps the model learn large images or videos.

Vision Decoder	Block-FID↓	Block-FID(× 4)↓	CLIP-SIM↑	Inference Speed↑
NUWA-Infinity (NAR)	92.34	98.67	0.2451	**95**×
NUWA-Infinity (P-NAR)	19.86	38.59	0.2726	15×
NUWA-Infinity (AR)	**10.05**	**17.78**	**0.2753**	1×

(a) Decoder model.

Parameters	Depth	Dim	Block-FID↓
202M (Base)	16	768	10.05
809M (Large)	24	1280	**9.71**

(b) Decoder size.

Loss	Block-FID↓	Convergence epoch↓
Patch	**10.05**	**50**
Accumulated	11.62	70

(c) Decoder loss.

Table 7: Ablation experiments with text-to-image generation on LHQC. We use a base model with 16 layers and dim of 768 except (b).

5 Discussions

Training Data The model for infinite visual synthesis requires high-resolution images and videos as the training data. Such high-quality visual data are harder to collect duo to quality and license issues. In the future, we will collect large-scale datasets satisfying the quality and license criteria for the development of this research direction.

Evaluation Metric Compared to text generation tasks, such as machine translation and text summarization, visual synthesis is more difficult to evaluate as the number of ground-truths of a model's output could be unlimited. Currently, we follow the traditions (i.e., using FID, FVD, IS, and CLIP-SIM scores) to measure the quality of generated images and videos. In the future, we will explore better evaluation metrics for visual synthesis tasks.

Inference Speed Autoregressive models can deal with dependencies between generated contents well. However, the training and inference efficiency of this generation mechanism is still a blocking issue for deploying such models for practical usage. In the future, we will explore ways to combine the advantages of autoregressive models and non-autoregressive models (such as the diffusion model) to achieve both generation quality and inference (or training) efficiency.

Pre-trained Version In this paper, we train NUWA-Infinity for different downstream tasks directly, due to the lack of large-scale high-quality visual data. In the future, we will pre-train the next version of NUWA-Infinity with more collected visual data and report its generalization capabilities on open-domain inputs.

6 Conclusion

NUWA-Infinity is a visual synthesis framework that can be trained to generate high-quality images and videos from the given text or image input. Different from DALL·E, DALL·E 2, Imagen and Parti, an autoregressive over autoregressive mechanism is proposed to support variable-size visual content generation tasks, such as image outpainting, image animation, text-to-image generation, and text-to-video generation. We hope such models help visual content creators save time, cut costs, and increase productivity and creativity.

Acknowledgements

We'd like to thank Minheng Ni, Xiaodong Wang, and Bei Li for the figure and table formats of this paper. We'd also like to thank Yu Liu, Jieyu Xiao, Scarlett Li, and Jane Ma for the discussion of potential application scenarios. We'd also like to thank Yang Ou and Bella Guo for the design of the homepage, and Tiantian Xue and Daisy Hou for the implementation of the homepage. We'd also like to thank Ting Song, Yan Xia, and Shiyou Ren for the help with the dataset construction. We'd also like to thank Yan Fan and Quanlu Zhang for their system support.

References

[1] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11315–11325, 2022.

[2] Jaemin Cho, Jiasen Lu, Dustin Schwenk, Hannaneh Hajishirzi, and Aniruddha Kembhavi. X-LXMERT: Paint, Caption and Answer Questions with Multi-Modal Transformers. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 8785–8805, Online, 2020. Association for Computational Linguistics.

[3] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. arXiv preprint arXiv:1907.06571, 2019.

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In Advances in Neural Information Processing Systems, volume 34, pages 8780–8794, 2021.

[5] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, and Hongxia Yang. Cogview: Mastering text-to-image generation via transformers. In Advances in Neural Information Processing Systems, volume 34, pages 19822–19835, 2021.

[6] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12873–12883, 2021.

[7] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10696–10706, 2022.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, pages 2961–2969, 2017.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Advances in Neural Information Processing Systems, volume 30, 2017.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Advances in Neural Information Processing Systems, volume 33, pages 6840–6851, 2020.

[11] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve Transformer Models with Better Relative Position Embeddings. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3327–3335, Online, 2020. Association for Computational Linguistics.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[13] Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. InfinityGAN: Towards Infinite-Pixel Image Synthesis. In International Conference on Learning Representations, 2022.

[14] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite Nature: Perpetual View Generation of Natural Scenes From a Single Image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 14458–14467, 2021.

[15] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3202–3211, 2022.

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In International Conference on Learning Representations, pages 8748–8763. PMLR, July 2021.

[17] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 21(140):1–67, 2020.

[18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125, 2022.

[19] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation. In Proceedings of the 38th International Conference on Machine Learning, pages 8821–8831. PMLR, July 2021.

[20] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, and Rapha Gontijo Lopes. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. arXiv preprint arXiv:2205.11487, 2022.

[21] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In Advances in Neural Information Processing Systems, volume 29, pages 2234–2242, 2016.

[22] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 14144–14153, 2021.

[23] Łukasz Struski, Szymon Knop, Przemysław Spurek, Wiktor Daniec, and Jacek Tabor. LocoGAN — Locally convolutional GAN. Computer Vision and Image Understanding, 221:103462, August 2022.

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

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[26] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High Fidelity Video Prediction with Large Stochastic Recurrent Neural Networks. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

[27] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. GIT: A Generative Image-to-text Transformer for Vision and Language. arXiv preprint arXiv:2205.14100, 2022.

[28] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. N\"UWA: Visual Synthesis Pre-training for Neural visUal World creAtion. In Proceedings of the European Conference on Computer Vision (ECCV), 2022.

[29] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized Image Modeling with Improved VQGAN. In International Conference on Learning Representations, March 2022.

[30] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, and Burcu Karagol Ayan. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. arXiv preprint arXiv:2206.10789, 2022.

[31] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. M6-ufc: Unifying multi-modal controls for conditional image synthesis. arXiv preprint arXiv:2105.14211, 2021.