RICHSPACE: ENRICHING TEXT-TO-VIDEO PROMPT SPACE VIA TEXT EMBEDDING INTERPOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-video generation models have made impressive progress, but they still struggle with generating videos with complex features. This limitation often arises from the inability of the text encoder to produce accurate embeddings, which hinders the video generation model. In this work, we propose a novel approach to overcome this challenge by selecting the optimal text embedding through interpolation in the embedding space. We demonstrate that this method enables the video generation model to produce the desired videos. Additionally, we introduce a simple algorithm using perpendicular foot embeddings and cosine similarity to identify the optimal interpolation embedding. Our findings highlight the importance of accurate text embeddings and offer a pathway for improving text-to-video generation performance.

1 INTRODUCTION

Text-to-video models have developed rapidly in recent years, driven by the advancement of Transformer architectures (Vaswani, 2017) and diffusion models (Ho et al., 2020). Early attempts at text-to-video generation focused on scaling up Transformers, with notable works such as CogVideo (Hong et al., 2022) and Phenaki (Villegas et al., 2022), which demonstrated promising results. More recently, the appearance of DiT (Peebles & Xie, 2023), which incorporates Transformers as the backbone of Diffusion Models, has pushed the capabilities of text-to-video generation models to new heights. Models like Sora (OpenAI, 2024), MovieGen (Meta, 2024), CogVideoX (Yang et al., 2024), and Veo 2 (Google, 2024) have further showcased the potential of these approaches. Despite the impressive progress made in recent years, current state-of-the-art text-to-video generation models still face challenges in effectively following complex instructions in user-provided text prompts. For example, when users describe unusual real-world scenarios, such as "a tiger with zebra-like stripes walking on grassland," the text encoder may struggle to fully capture the intended meaning. This results in text embeddings that fail to guide the video generation model toward producing the desired output. This issue is also observed in the text-to-image generation domain, where a notable work, Stable Diffusion V3 (Esser et al., 2024), addresses this challenge by incorporating multiple text encoders to improve understanding. Although their approach, which combines embeddings from different encoders, yields effective results, it comes at a significant computational cost due to the need to compute embeddings from multiple sources.

In this work, we first study the problem that prompt space is not enough to cover all video space from a theoretical perspective. We provide an informal theorem of our theoretical findings as follows:

Theorem 1.1 (Word Embeddings being Insufficient to Represent for All Videos, informal version of Theorem 4.9)**.** *Let n, d denote two integers, where n denotes the maximum length of the sentence, and all videos are in $\mathbb{R}^d$ space. Let $V \in \mathbb{N}$ denote the vocabulary size. Let $\mathcal{U} = \{u_1, u_2, \cdots, u_V\}$ denote the word embedding space, where for $i \in [V]$, the word embedding $u_i \in \mathbb{R}^k$. Let $\delta_{\min} = \min_{i,j \in [V], i \neq j} \|u_i - u_j\|_2$ denote the minimum ℓ_2 distance of two word embedding. Let $f : \mathbb{R}^{nk} \to \mathbb{R}^d$ denote the text-to-video generation model, which is also a mapping from sentence space (discrete space $\{u_1, \ldots, u_V\}^n$) to video space $\mathbb{R}^d$. Let $M := \max_x \|f(x)\|_2, m := \min_x \|f(x)\|_2$. Let $\epsilon = ((M^d - m^d)/V^n)^{1/d}$. Then, we can show that there is a video $y \in \mathbb{R}^d$, satisfying $m \leq \|y\|_2 \leq M$, such that for any sentence $x \in \{u_1, u_2, \cdots, u_V\}^n$, $\|f(x) - y\|_2 \geq \epsilon$.*

Additionally, we take a different approach by exploring whether we can obtain a powerful text embedding capable of guiding the video generation model through interpolation within the text embedding space. Through empirical experiments, we demonstrate that by selecting the optimal text embedding, the video generation model can successfully generate the desired video. Additionally, we propose an algorithm that takes advantage of perpendicular foot embeddings and cosine similarity to capture both global and local information in order to identify the optimal embedding of interpolation text (Fig. 1 and Algorithm 1).

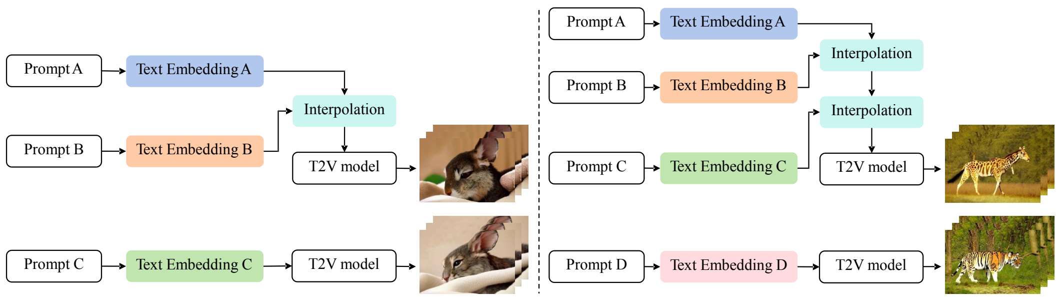


Figure 1: Two kinds of Text Prompts Mixture. **Left: Mixture of Two Prompts.** We set two prompts, A and B, and apply linear interpolation to two corresponding text embeddings. After that, we use one of the interpolation results to generate a video. To evaluate the effect of video interpolation, we set another prompt C, which describes the generated video to generate a video to compare with the interpolated video. **Right: Mixture of Three Prompts**. We set two prompts A and B and apply linear interpolation to two corresponding text embeddings. We manually choose one text embedding interpolated from A and B, then apply linear interpolation to this text embedding and text embedding C. After that, we use one of the interpolation results to generate a video. To evaluate the effect of video interpolation, we set another prompt D which describes the generated video to generate a video to compare with the interpolated video.

In summary, our main contributions are as follows:

- We demonstrate that selecting the correct text embedding can effectively guide a video generation model to produce the desired video.
- We propose a simple yet effective algorithm to find the optimal text embedding through the use of perpendicular foot embeddings and cosine similarity.

Roadmap. Our paper is organized as follows: Section 2 introduces our main algorithm for finding the optimal interpolation embedding. Section 3 presents the experiment result of this work. Section 4 presents the theoretical analysis, including the preliminary of our notation, key concepts of our video algorithm, model formulation, and our definition of an optimal interpolation embedding finder. In Section 5, we conclude our paper.

2 Our Methods

Section 2.1 introduces the problem formulation. In Section 2.2, we present our algorithm for finding the optimal interpolation embedding.

2.1 Problem Formulation

In this section, we introduce the formal definition for finding the optimal interpolation embedding as follows:

Definition 2.1 (Finding Optimal Interpolation Embedding Problem)**.** *Let P_a, P_b, P_c denote three text prompts. Our goal is to generate a video that contains features mentioned in P_a and P_b, and P_c is a text description of the feature combination of P_a and P_b. Let $E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}$ denote the text embedding of P_a, P_b, P_c. Let $f_\theta(E_t, z)$ be defined in Definition 4.8. We define the "Finding optimal interpolation embedding" problem as: According to $E_{t_a}, E_{t_b}, E_{t_c}$, find the optimal interpolation embedding $E_{\rm opt}$ that can make the text-to-video generation model $f_\theta(E_{\rm opt}, z)$ generate video contains features mentioned in P_a and P_b.*

Algorithm 1 Find Optimal Interpolation

datastructure OPTIMALINTERPFINDER
members
  $n \in \mathbb{N}$: the length of input sequence.
  $n_{\text{ids}} \in \mathbb{N}$: the ids length of input sequence.
  $d \in \mathbb{R}$: the hidden dimension.
  $E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}$: the text embedding.
  $\phi_{\cos}(X, Y)$: the cosine similarity calculator. ▷ Definition 4.2
end members

procedure OPTIMALFINDER($E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}$,$n_{a_{\text{ids}}}, n_{b_{\text{ids}}}, n_{c_{\text{ids}}} \in \mathbb{N}$)
  /* Calculate the max ids length. */
  $n_{\text{ids}} \leftarrow \max\{n_{a_{\text{ids}}}, n_{b_{\text{ids}}}, n_{c_{\text{ids}}}\}$
  /* Truncated text embeddings. */
  $E_{a_{\text{truc}}} \in \mathbb{R}^{n_{\text{ids}} \times d} \leftarrow E_{t_a}[: n_{\text{ids}}, :]$
  $E_{b_{\text{truc}}} \in \mathbb{R}^{n_{\text{ids}} \times d} \leftarrow E_{t_b}[: n_{\text{ids}}, :]$
  $E_{c_{\text{truc}}} \in \mathbb{R}^{n_{\text{ids}} \times d} \leftarrow E_{t_c}[: n_{\text{ids}}, :]$
  /* Calculate cosine similarity, Algorithm 2. */
  $L_{\text{CosTruc}} \leftarrow \textsc{CosineSim}(E_{a_{\text{truc}}}, E_{b_{\text{truc}}}, E_{c_{\text{truc}}})$
  $L_{\text{CosFull}} \leftarrow \textsc{CosineSim}(E_{t_a}, E_{t_b}, E_{t_c})$
  /* Add ConsineTruc and CosineFull. */
  $L_{\text{CosAdd}} \leftarrow [\,]$
  for $i = 1 \to k$ **do**
    $L_{\text{CosAdd}}[i] \leftarrow L_{\text{CosTruc}}[i] + L_{\text{CosFull}}[i]$
  end for
  /* Find the optimal interpolation index. */
  $i_{\text{opt}} \leftarrow \text{maxindex}(L_{\text{CosAdd}})$
  /* Calculate optimal interpolation embedding. */
  $E_{\text{opt}} \leftarrow \frac{i_{\text{opt}}}{k} \cdot E_{t_a} + \frac{k - i_{\text{opt}}}{k} \cdot E_{t_b}$
  Return E_{opt}
end procedure

We would like to refer the readers to Figure 2 (a) as an example of Definition 2.1. In Figure 2 (a), we set prompt P_a to *"The tiger, moves gracefully through the forest, its fur flowing in the breeze."* and prompt P_b to:*"The zebra, moves gracefully through the forest, its fur flowing in the breeze."*. Our goal is to generate a video that contains both features of "tiger" and "zebra", where we set prompt P_c to *"The tiger, with black and white stripes like zebra, moves gracefully through the forest, its fur flowing in the breeze."*, to describe the mixture features of tiger and zebra. However, the text-to-video model fails to generate the expected video. Therefore, it is essential to find the optimal interpolation embedding E_{opt} to make the model generate the expected video. In Figure 2 (a), the E_{opt} is the 14-th interpolation embedding of E_{t_a} and E_{t_b}.

2.2 OPTIMAL INTERPOLATION EMBEDDING FINDER

In this section, we introduce our main algorithm (Algorithm 3 and Algorithm 1), which is also depicted in Fig. 1. The algorithm is designed to identify the optimal interpolation embedding (as defined in Definition 2.1) and generate the corresponding video. The algorithm consists of three key steps:

1. Compute the perpendicular foot embedding (Line 9 in Algorithm 2).
2. Calculate the cosine similarity between the interpolation embeddings and the perpendicular foot embedding (Line 22 in Algorithm 2).
3. Select the optimal interpolation embedding based on the cosine similarity results (Algorithm 1).

We will now provide a detailed explanation of each part of the algorithm and the underlying intuitions.

Algorithm 2 Calculate Cosine Similarity

datastructure COSINESIMILARITYCALCULATOR
members
  $n \in \mathbb{N}$: the length of input sequence.
  $d \in \mathbb{N}$: the hidden dimension.
  $E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}$: the text embedding.
  $\phi_{\cos}(X, Y)$: the cosine similarity calculator. ▷ Definition 4.2
end members

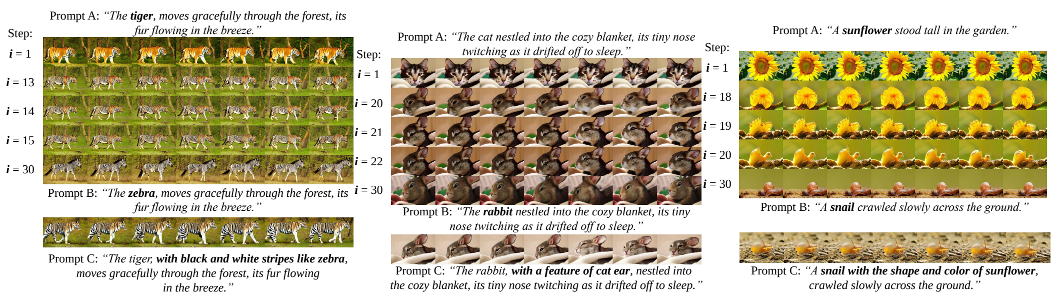
procedure PERPENDICULARFOOT($E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}$)
  /* Find perpendicular foot of E_{t_c} on $E_{t_b} - E_{t_b}$. */
  $E_{ac} \leftarrow E_{t_c} - E_{t_a}$
  $E_{ab} \leftarrow E_{t_b} - E_{t_a}$
  /* Calculate the projection length. */
  $l_{\text{proj}} \leftarrow \langle E_{ab}, E_{ac} \rangle / \langle E_{ab}, E_{ab} \rangle$
  /* Calculate the projection vector. */
  $E_{\text{proj}} \leftarrow l_{\text{proj}} \cdot E_{ab}$
  /* Calculate the perpendicular foot. */
  $E_{\text{foot}} \leftarrow E_{t_a} + E_{\text{proj}}$
  Return E_{foot}
end procedure

procedure COSINESIM($E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}$)
  /* Calculate perpendicular foot. */
  $E_{\text{foot}} \leftarrow$ PERPENDICULARFOOT($E_{t_a}, E_{t_b}, E_{t_c}$)
  /* Init cosine similarity list. */
  $L_{\text{CosSim}} \leftarrow [\,]$
  for $i = 1 \to k$ **do**
    /* Compute interpolation embedding. */
    $E_{\text{interp}} \leftarrow \frac{i}{k} \cdot E_{t_1} + \frac{k-i}{k} \cdot E_{t_2}$
    /* Calculate and store cosine similarity. */
    $L_{\text{CosSim}}[i] \leftarrow \phi_{\cos}(E_{\text{interp}}, E_{\text{foot}})$
  end for
  Return L_{CosSim}
end procedure

Perpendicular Foot Embedding. As outlined in the problem definition (Definition 2.1), our objective is to identify the optimal interpolation embedding that allows the text-to-video generation model to generate a video containing the features described in P_a and P_b. The combination of these features is represented by P_c, which typically does not lead to the desired video output. Consequently, we seek an interpolation embedding of E_{t_a} and E_{t_b} guided by E_{t_c}. The first step involves finding the perpendicular foot of E_{t_c} onto the vector $E_{t_b} - E_{t_a}$, also known as the projection of E_{t_c}. This perpendicular foot embedding, denoted as E_{foot}, is not the optimal embedding in itself, as the information within E_{t_c} alone does not enable the generation of the expected video. However, E_{foot} serves as a useful anchor, guiding us toward the optimal interpolation embedding. Further details of this approach will be discussed in the subsequent paragraph.

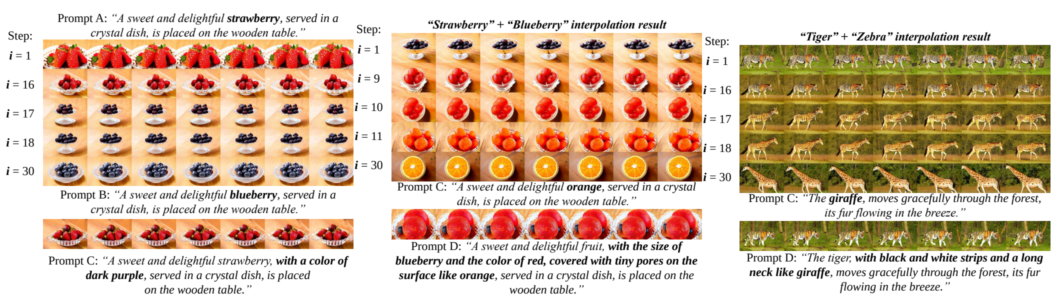
Cosine Similarity and Optimal Interpolation Embedding. To assess the similarity of each interpolation embedding to the anchor perpendicular foot embedding E_{foot}, we employ the straightforward yet effective metric of cosine similarity (Definition 4.2). It is important to note that the input text prompts are padded to a fixed maximum length, $n = 266$, before being encoded by the T5 model. However, in real-world scenarios, the actual length of text prompts is typically much shorter than $n = 266$, which results in a substantial number of padding embeddings being appended to the original text prompt. The inclusion or exclusion of these padding embeddings can lead to significant differences in the perpendicular foot embedding, as their presence introduces a shift in the distribution of the text embeddings. To account for this, we treat text embeddings with and without padding separately. Specifically, we define "full text embeddings" $E_{a_t}, E_{b_t}, E_{c_t} \in \mathbb{R}^{n \times d}$ to represent the embeddings that include padding, and "truncated text embeddings" $E_{a_{\text{truc}}}, E_{b_{\text{truc}}}, E_{c_{\text{truc}}} \in \mathbb{R}^{n_{\text{ids}} \times d}$ to represent the embeddings without padding (Line 13 in Algorithm 1). The full-text embeddings capture global information, whereas the truncated text embeddings focus on local information. We compute the perpendicular foot and cosine similarity separately for both types of text embeddings (Line 17) and then combine the results by summing the cosine similarities from the full and truncated embeddings. Finally, we select the optimal interpolation embedding based on the aggregated cosine similarity scores (Line 25).

3 EXPERIMENTS



(a) (b) (c)

Figure 2: **Qualitative results of mixture of two features.** Figure (a): Mixture of ["Tiger"] and ["Zebra"]; Figure (b): Mixture of ["Cat"] and ["Rabbit"]; Figure (c): Mixture of ["Sunflower"] and ["Snail"]. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.



(a) (b) (c)

Figure 3: **Extending from two prompts mixture to three prompts mixture.** Figure (a): Mixture of ["Strawberry"] and ["Blueberry"]. Figure (b): Mixture of ["Strawberry" + "Blueberry"] and ["Orange"]. We further apply Algorithm 1 to that optimal embedding and Prompt C embedding, with the guidance of Prompt D. We identify 10-th interpolation embedding as the optimal embedding of ["Strawberry" + "Blueberry"] and ["Orange"] and generate the corresponding video. The video generated directly from Prompt D does not exhibit the desired mixed features. Figure (c): Mixture of ["Tiger" + "Zebra"] and ["Giraffe"]. We present another example of a mixture of three prompts to demonstrate the effectiveness of our algorithm.

In this section, we will first present our qualitative evaluation results of the proposed method in Section 3.1. Then, in Section 3.2, we present our quantitative evaluation.

3.1 QUALITATIVE EVALUATION

Our experiments are conducted on the CogVideoX-2B (Yang et al., 2024). We investigate the performance of our optimal embedding finder algorithm in the following two scenarios:

Mixture of Features from Two Initial Prompts. As outlined in Definition 2.1, we conduct experiments where the goal is to generate a mixture of features described in two text prompts, P_a and P_b. We construct a third prompt, P_c, to specify the desired features. Following Algorithm 1, we identify the optimal text embedding and use it for the text-to-video generation with our base model. We conducted experiments using a variety of text prompts. In Figure 2 (a) and Figure 2 (b), we investigate the mixture of features from different animals, demonstrating that a video containing the mixture of tiger and zebra features, as well as the mixture of rabbit and cat features can only be generated using the optimal embedding, not directly from the text prompts. Similarly, in Figure 3 (a), we show that a video combining features from strawberry and blueberry can only be generated through the optimal embedding, highlighting a similar phenomenon in the context of fruits. Furthermore, in Figure 2 (c), we observe the same behavior in the domain of plants, specifically with the combination of rose and cactus features.

Mixture of Features from Three Initial Prompts. We will investigate further to see if we can add one additional feature to the video. The high-level approach involves applying our optimal interpolation embedding algorithm (Algorithm 1) twice. Given three text embeddings, E_{t_a}, E_{t_b}, and E_{t_c}, where we aim to blend their features in the generated video, we first apply Algorithm 1 to E_{t_a} and E_{t_b} to obtain the optimal interpolation embedding $E_{\mathrm{opt}_{ab}}$. Next, we apply Algorithm 1 again, this time on $E_{\mathrm{opt}_{ab}}$ and E_{t_c}, resulting in the final optimal interpolation embedding E_{opt}. We then use this embedding in our base model to generate the desired video. Following the method described above, we mix the giraffe feature with the tiger and zebra features, as shown in Figure 3 (c). Only by using the optimal embedding identified by our algorithm can we enable the video generation model to produce the desired video. Directly generating the video from the text prompt results in the loss of at least one of the intended features. A similar phenomenon is observed in the case of mixing strawberry, blueberry, and orange features, as shown in Figure 3 (b). The video generated directly from the text prompt always renders each object separately, failing to combine the features into a single coherent entity.

3.2 Quantitative Evaluation

In the previous sections, we presented the qualitative results of our method. In this section, we provide a quantitative evaluation. Following the settings used by VBench (Huang et al., 2024), we evaluate the "subject consistency" and "aesthetic quality" of the generated videos. The results for mixtures of two prompts are presented in Table 1. The average Subject Consistency (SC) of the videos generated using optimal embeddings is 0.9787, higher than the SC of the videos generated directly from the prompt description, which is 0.9748. As for Aesthetic Quality (AQ), the videos generated by optimal embeddings achieve a score of 0.5163, which is lower than the 0.5519 obtained by the videos generated from prompts.

Our method generates videos with higher "subject consistency" than those produced directly from the prompt description (i.e., Prompt C). This suggests that the optimal embedding enables the video generation model to better combine the desired features while maintaining coherence in the generated videos.

Another observation is that the "aesthetic quality" of videos generated using the optimal embeddings is lower than that of videos generated directly from text prompts. This indicates that our method better blends the desired features. The aesthetic model is trained on real-world videos, which leads to a bias toward scoring videos that resemble those found in real-world datasets. However, in our setting, we aim to expand the prompt space of the video generation model, enabling it to generate videos that are rarely observed in real-world datasets. Therefore, a lower aesthetic score reflects that our method aligns better with this goal.

4 Theoretical Analysis

We first introduce some basic notations in Section 4.1. In Section 4.2, we introduce formal definitions of key concepts. Then, we introduce the formal definition of each module in the CogvideoX model in Section 4.3. In Section 4.4, we provide our rigorous theoretical analysis showing that word embedding space is not sufficient to represent all videos.

Table 1: **Quantitative Evaluations.** We evaluate the videos generated using our optimal embeddings and those generated directly from the text prompt with two metrics: "Subject Consistency" (SC) and "Aesthetic Quality" (AQ). Let f represent the optimal embedding finding algorithm, and g denote the video generation model. A higher SC score indicates better coherence in the video, which corresponds to higher quality. Conversely, a lower AQ score suggests that the video is rarely observed in the real world, implying that it aligns more closely with the mixture of desired features. We use A to denote PromptA, B to denote PromptB and C to denote PromptC.

Prompts	**SC** (↑)	**AQ** (↓)
;$g(f(\text{Tiger, Zebra}))$	**0.9751**	0.5472
$g(\text{Tiger, Zebra })$	0.9739	**0.5424**
$g(f(\text{Cat, Rabbit}))$	**0.9688**	**0.4649**
$g(\text{Cat, Rabbit})$	0.9608	0.4821
$g(f(\text{Strawberry, Blueberry}))$	**0.9920**	**0.5957**
$g(\text{Strawberry, Blueberry})$	0.9910	0.7256
$g(f(\text{Sunflower, Snail}))$	**0.9790**	**0.4573**
$g(\text{Sunflower, Snail})$	0.9734	0.4575
avg. $g(f(\text{A}, \text{B}))$	**0.9787**	**0.5163**
avg. $g(\text{C})$	0.9748	0.5519

4.1 Notations

For any $k \in \mathbb{N}$, let $[k]$ denote the set $\{1, 2, \cdots, k\}$. For any $n \in \mathbb{N}$, let n denote the length of the input sequence of a model. For any $d \in \mathbb{N}$, let d denote the hidden dimension. For any $c \in \mathbb{N}$, let c denote the channel of a video. For any $n_f \in \mathbb{N}$, we use n_f to denote the video frames. For any $h \in \mathbb{N}$ and $w \in \mathbb{N}$, we use h and w to denote the height and width of a video. For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between x, y. Namely, $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$. For a vector $x \in \mathbb{R}^n$, we use $\|x\|_2$ to denote the ℓ_2 norm of the vector x, i.e., $\|x\|_2 := \sqrt{\sum_{i=1}^n x_i^2}$.

Let $\mathcal{D}$ represent a given distribution. The notation $x \sim \mathcal{D}$ indicates that x is a random variable drawn from the distribution $\mathcal{D}$.

4.2 Key Concepts

We will introduce some essential concepts in this section. We begin with introducing the formal definition of linear interpolation.

Definition 4.1 (Linear Interpolation)**.** *Let $x, y \in \mathbb{R}^d$ denote two vectors. Let $k \in \mathbb{N}$ denote the interpolation step. For $i \in [k]$, we define the i-th interpolation result $z_i \in \mathbb{R}$ as follows:*

$$z_i := \frac{i}{k} \cdot x + \frac{k-i}{k} \cdot y$$

Next, we introduce another key concept used in our paper, the simple yet effective cosine similarity calculator.

Definition 4.2 (Cosine Similarity Calculator)**.** *Let $X, Y \in \mathbb{R}^{n \times d}$ denote two matrices. Let $X_i, Y_i \in \mathbb{R}^d$ denote i-th row of X, Y, respectively. Then, we defined the cosine similarity calculator $\phi_{\cos}(X, Y) : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times d} \to \mathbb{R}$ as follows $\phi_{\cos}(X, Y) := \frac{1}{n} \sum_{i=1}^n \frac{\langle X_i, Y_i \rangle}{\|X_i\|_2 \|Y_i\|_2}$.*

Then, we introduce one crucial fact that we used later in this paper.

Fact 4.3 (Volume of a Ball in d-dimension Space)**.** *The volume of a ℓ_2-ball with radius R in dimension $\mathbb{R}^d$ space is $\frac{\pi^{d/2}}{(d/2)!} R^d$.*

4.3 Model Formulation

In this section, we will introduce the formal definition for the text-to-video generation video we use. We begin with introducing the formal definition of the attention layer as follows:

Algorithm 3 Video Interpolation

datastructure INTERPOLATION
members
  $n \in \mathbb{N}$: the length of input sequence
  $n_f \in \mathbb{N}$: the number of frames
  $h \in \mathbb{N}$: the height of video
  $w \in \mathbb{N}$: the width of video
  $d \in \mathbb{N}$: the hidden dimension
  $c \in \mathbb{N}$: the channel of video
  $k \in \mathbb{N}$: the interpolation steps
  $T \in \mathbb{N}$: the number of inference step
  $E_{\mathrm{opt}} \in \mathbb{R}^{n \times d}$: the optimal interpolation embedding
  $E_t \in \mathbb{R}^{n \times d}$: the text embedding
  $f_\theta(z, E_t, t) : \mathbb{R}^{n_f \times h \times w \times c} \times \mathbb{R}^{n \times d} \times \mathbb{N} \to \mathbb{R}^{n_f \times h \times w \times c}$: the text-to-video generation model
end members

procedure INTERPOLATION($E_{t_a}, E_{t_b}, E_{t_c} \in \mathbb{R}^{n \times d}, k \in \mathbb{N}, T \in \mathbb{N}$)
  /* Find optimal interpolation embedding, Algorithm 1. */
  $E_{\mathrm{opt}} \leftarrow$ OPTIMALFINDER($E_{t_a}, E_{t_b}, E_{t_c}$)
  /* Prepare initial latents.*/
  $z \sim \mathbb{N}(0, I) \in \mathbb{R}^{n_f \times h \times w \times c}$
  for $t = T \to 0$ **do**
    /* One denoise step. */.
    $z \leftarrow f_\theta(z, E_{\mathrm{opt}}, t)$
  end for
  Return z
end procedure

Definition 4.4 (Attention Layer). *Let $X \in \mathbb{R}^{n \times d}$ denote the input matrix. Let $W_K, W_Q, W_V \in \mathbb{R}^{d \times d}$ denote the weighted matrices. Let $Q = XW_Q \in \mathbb{R}^{n \times d}$ and $K = XW_K \in \mathbb{R}^{n \times d}$. Let attention matrix $A = QK^\top$. Let $D := \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$. We define attention layer Attn as follows: $\mathsf{Attn}(X) := D^{-1}AXW_V$.*

Then, we define the convolution layer as follows:

Definition 4.5 (Convolution Layer). *Let $h \in \mathbb{N}$ denote the height of the input and output feature map. Let $w \in \mathbb{N}$ denote the width of the input and output feature map. Let $c_{\mathrm{in}} \in \mathbb{N}$ denote the number of channels of the input feature map. Let $c_{\mathrm{out}} \in \mathbb{N}$ denote the number of channels of the output feature map. Let $X \in \mathbb{R}^{h \times w \times c_{\mathrm{in}}}$ represent the input feature map. For $l \in [c_{\mathrm{out}}]$, we use $K^l \in \mathbb{R}^{3 \times 3 \times c_{\mathrm{in}}}$ to denote the l-th convolution kernel. Let p denote the padding of the convolution layer. Let s denote the stride of the convolution kernel. Let $Y \in \mathbb{R}^{h \times w \times c_{\mathrm{out}}}$ represent the output feature map. We define the convolution layer as follows: We use $\phi_{\mathrm{conv}}(X, c_{\mathrm{in}}, c_{\mathrm{out}}, p, s) : \mathbb{R}^{h \times w \times c_{\mathrm{in}}} \to \mathbb{R}^{h \times w \times c_{\mathrm{out}}}$ to represent the convolution operation. Let $Y = \phi_{\mathrm{conv}}(X, c_{\mathrm{in}}, c_{\mathrm{out}}, p, s)$. Then, for $i \in [h], j \in [w], l \in [c_{\mathrm{out}}]$, we have $Y_{i,j,l} := \sum_{m=1}^{3} \sum_{n=1}^{3} \sum_{c=1}^{c_{\mathrm{in}}} X_{i+m-1, j+n-1, c} \cdot K^l_{m,n,c}$*

We introduce the formal definition of linear projection layer as follows:

Definition 4.6 (Linear Projection). *Let $X \in \mathbb{R}^{n \times d_1}$ denote the input data matrix. Let $W \in \mathbb{R}^{d_1 \times d_2}$ denote the weight matrix. We define the linear projection $\phi_{\mathrm{linear}} : \mathbb{R}^{n \times d_1} \to \mathbb{R}^{n \times d_2}$ as follows:*

$$\phi_{\mathrm{linear}}(X) := XW$$

And we define the 3D full attention layer as follows:

Definition 4.7 (3D Attention). *Let $\mathsf{Attn}(X)$ be defined as in Definition 4.4. Let $\phi_{\mathrm{conv}}(X, c_{\mathrm{in,out},p,s})$ be defined in Definition 4.5. Let $\phi_{\mathrm{linear}}(X)$ be defined as in Definition 4.6. We define the 3D attention $\phi_{\mathrm{3DAttn}}(E_t, E_v)$ containing three components: $\phi_{\mathrm{linear}}(X)$, $\mathsf{Attn}(X)$, $\phi_{\mathrm{conv}}(X, c_{\mathrm{in}}, c_{\mathrm{out}}, p, s)$. Its details are provided in Algorithm 4.*

Finally, we provide the definition of the text-to-video generation model, which consists of a stack of multiple 3D attention layers, as introduced earlier.

Definition 4.8 (Text-to-Video Generation Model)**.** *Let ϕ_{3DAttn} be defined as Definition 4.7. Let $k_{\mathrm{3D}} \in \mathbb{N}$ denote the number of 3D attention layers in the text-to-video generation model. Let θ denote the parameter in the text-to-video generation model. Let $E_t \in \mathbb{R}^{n \times d}$ denote the text embedding. Let $z \sim \mathbb{N}(0, I) \in \mathbb{R}^{n_f \times h \times w \times c}$ denote the initial random Gaussian noise. Then we defined the text-to-video generation model $f_\theta(E_t, z)$ as follows:*

$$f_\theta(E_t, z) := \underbrace{\phi_{\mathrm{3DAttn}} \circ \cdots \circ \phi_{\mathrm{3DAttn}}}_{k_{\mathrm{3D}} \text{ layers}}(E_t, z).$$

4.4 Word Embedding Space being Insufficient to Represent for All Videos

Since the text-to-video generation model only has a finite vocabulary size, it only has finite wording embedding space. However, the space for all videos is infinite. Thus, word embedding space is insufficient to represent all videos in video space. We formalize this phenomenon to a rigorous math problem and provide our findings in the following theorem.

Theorem 4.9 (Word Embeddings being Insufficient to Represent for All Videos, formal version of Theorem 1.1)**.** *Let n, d denote two integers, where n denotes the maximum length of the sentence, and all videos are in $\mathbb{R}^d$ space. Let $V \in \mathbb{N}$ denote the vocabulary size. Let $\mathcal{U} = \{u_1, u_2, \cdots, u_V\}$ denote the word embedding space, where for $i \in [V]$, the word embedding $u_i \in \mathbb{R}^k$. Let $\delta_{\min} = \min_{i,j \in [V], i \neq j} \|u_i - u_j\|_2$ denote the minimum ℓ_2 distance of two word embedding. Let $f : \mathbb{R}^{nk} \to \mathbb{R}^d$ denote the text-to-video generation model, which is also a mapping from sentence space (discrete space $\{u_1, \ldots, u_V\}^n$) to video space $\mathbb{R}^d$. Let $M := \max_x \|f(x)\|_2, m := \min_x \|f(x)\|_2$. Let $\epsilon = ((M^d - m^d)/V^n)^{1/d}$. Then, we can show that there exits a video $y \in \mathbb{R}^d$, satisfying $m \leq \|y\|_2 \leq M$, such that for any sentence $x \in \{u_1, u_2, \cdots, u_V\}^n$, we have $\|f(x) - y\|_2 \geq \epsilon$.*

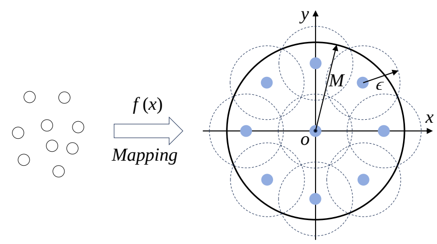


Figure 4: **Mapping from Prompt Space to Video Space.** This figure illustrates the mapping from a prompt space (with discrete prompts) to a video space (with continuous video embeddings) by a video generation model $f(x)$. Regardless of the specific form of the video generation model $f(x)$, there always exists a point in the video embedding space whose distance to all $f(x)$ is at least ϵ.

Theorem 4.9 indicates that there always exists a video y, where its ℓ_2 distance to all videos can be represented by the prompt embeddings is larger than ϵ (Fig. 4). This means that there always exists a video that cannot be accurately generated by using only the prompt embeddings from the word embedding space. We defer the proof to Theorem C.6 which is the restatement of Theorem 4.9 in the Appendix.

5 Conclusion

In this work, we propose a novel algorithm to identify the optimal text embedding, enabling a video generation model to produce videos that accurately reflect the features specified in the initial prompts. Our findings reveal that the main bottleneck in text-to-video generation is the text encoder's inability to generate precise text embeddings. By carefully selecting and interpolating text embeddings, we improve the model's ability to generate more accurate and diverse videos. From the theoretical side, we show that text embeddings generated by the text encoder are insufficient to represent all possible video features, which explains why the text encoder becomes a bottleneck in generating videos with mixed desired features. Our proposed algorithm, based on perpendicular foot embeddings and cosine similarity, provides an effective solution to these challenges. These results highlight the importance of refining text embeddings to improve model performance and lay the foundation for future advancements in text-to-video generation by emphasizing the critical role of embedding optimization in bridging the gap between textual descriptions and video synthesis.

Ethic Statement

This paper does not involve human subjects, personally identifiable data, or sensitive applications. We do not foresee direct ethical risks. We follow the ICLR Code of Ethics and affirm that all aspects of this research comply with the principles of fairness, transparency, and integrity.

Reproducibility Statement

We ensure reproducibility on both theoretical and empirical fronts. For theory, we include all formal assumptions, definitions, and complete proofs in the appendix. For experiments, we describe models and algorithms in the main text and appendix.

References

Jie An, Songyang Zhang, Harry Yang, Sonal Gupta, Jia-Bin Huang, Jiebo Luo, and Xi Yin. Latent-shift: Latent diffusion with temporal shift for efficient text-to-video generation. *arXiv preprint arXiv:2304.08477*, 2023.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22563–22575, 2023.

Rumeysa Bodur, Erhan Gundogdu, Binod Bhattarai, Tae-Kyun Kim, Michael Donoser, and Loris Bazzani. iedit: Localised text-guided image editing with weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7426–7435, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22930–22941, 2023.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Google. Veo 2, 2024. URL `https://deepmind.google/technologies/veo/veo-2/`.

Jiaxi Gu, Shicong Wang, Haoyu Zhao, Tianyi Lu, Xing Zhang, Zuxuan Wu, Songcen Xu, Wei Zhang, Yu-Gang Jiang, and Hang Xu. Reuse and diffuse: Iterative denoising for text-to-video generation. *arXiv preprint arXiv:2309.03549*, 2023.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

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

Jerry Yao-Chieh Hu, Weimin Wu, Yi-Chen Lee, Yu-Chao Huang, Minshuo Chen, and Han Liu. On statistical rates of conditional diffusion transformers: Approximation, estimation and minimax optimality. *arXiv preprint arXiv:2411.17522*, 2024a.

Jerry Yao-Chieh Hu, Weimin Wu, Zhuoru Li, Sophia Pi, , Zhao Song, and Han Liu. On statistical rates and provably efficient criteria of latent diffusion transformers (dits). *Advances in Neural Information Processing Systems*, 38, 2024b.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Yupei Lin, Sen Zhang, Xiaojun Yang, Xiao Wang, and Yukai Shi. Regeneration learning of diffusion models with rich prompts for zero-shot image translation. *arXiv preprint arXiv:2305.04651*, 2023.

Chengyi Liu, Jiahao Zhang, Shijie Wang, Wenqi Fan, and Qing Li. Score-based generative diffusion models for social recommendations. *arXiv preprint arXiv:2412.15579*, 2024.

Xingchao Liu, Chengyue Gong, Lemeng Wu, Shujian Zhang, Hao Su, and Qiang Liu. Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. *arXiv preprint arXiv:2112.01573*, 2021.

Haoming Lu, Hazarapet Tunanyan, Kai Wang, Shant Navasardyan, Zhangyang Wang, and Humphrey Shi. Specialist diffusion: Plug-and-play sample-efficient fine-tuning of text-to-image diffusion models to learn any unseen style. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14267–14276, 2023.

Meta. Moviegen: Acastofmediafoundationmodels, 2024. URL `https://ai.meta.com/static-resource/movie-gen-research-paper`.

OpenAI. Sora, 2024. URL `https://openai.com/index/sora/`.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Yanyu Li, Yifan Gong, Kai Zhang, Hao Tan, Jason Kuen, Henghui Ding, et al. Lazydit: Lazy learning for the acceleration of diffusion transformers. *arXiv preprint arXiv:2412.12444*, 2024.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual descriptions. In *International Conference on Learning Representations*, 2022.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in neural information processing systems*, 35:23371–23385, 2022.

Yilin Wang, Haiyang Xu, Xiang Zhang, Zeyuan Chen, Zhizhou Sha, Zirui Wang, and Zhuowen Tu. Omnicontrolnet: Dual-stage integration for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7436–7448, 2024a.

Zirui Wang, Zhizhou Sha, Zheng Ding, Yilin Wang, and Zhuowen Tu. Tokencompose: Text-to-image diffusion with token-level supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8553–8564, 2024b.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

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

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

Appendix

Roadmap. In Section A, we provide a detailed discussion of our work. In Section B, we review related literature. In Section C, we provide detailed proofs for the theorem showing that word embeddings are insufficient to represent all videos. In Section D, we provide more results of our experiments. In Section E, we provide the algorithm for 3D attention.

A Discussion

Identifying the Actual Bottleneck of Generative Models. Our work identifies that the primary bottleneck hindering text-to-video generation models from producing the desired videos is the text encoder's inability to generate accurate text embeddings. Through our proposed algorithm, we can guide the video generation model to produce the desired output. This insight helps the community identify the true bottleneck within cutting-edge generative models, allowing for improvements in model performance and capabilities.

B Related Work

Text-to-Video Generation. Text-to-video generation (Singer et al., 2022; Voleti et al., 2022; Blattmann et al., 2023), as a form of conditional video generation, focuses on the synthesis of high-quality videos using text descriptions as conditioning inputs. Most recent works on video generation jointly synthesize multiple frames based on diffusion models (Song et al., 2020; Ho et al., 2020; Liu et al., 2024; Shen et al., 2024; Hu et al., 2024b;a). Diffusion models implement an iterative refinement process by learning to gradually denoise a sample from a normal distribution, which has been successfully applied to high-quality text-to-video generation. In terms of training strategies, one of the existing approaches uses pre-trained text-to-image models and inserts temporal modules (Ge et al., 2023; An et al., 2023), such as temporal convolutions and temporal attention mechanisms into the pre-trained models to build up correlations between frames in the video (Singer et al., 2022; Gu et al., 2023; Guo et al., 2023). PYoCo (Ge et al., 2023) proposed a noise prior approach and leveraged a pre-trained eDiff-I (Balaji et al., 2022) as initialization. Conversely, other works (Blattmann et al., 2023; Zhou et al., 2022a) build upon Stable Diffusion (Rombach et al., 2022) owing to the accessibility of pre-trained models. This approach aims to leverage the benefits of large-scale pre-trained text-to-image models to accelerate convergence. However, it may lead to unsatisfactory results due to the potential distribution gap between images and videos. Other approaches are training the entire model from scratch on both image and video datasets (Ho et al., 2022). Although this method can yield high-quality results, it demands tremendous computational resources.

Enrich Prompt Space. In the context of conditional tasks, such as text-to-image and text-to-video models, prompts worked as conditions can have a significant influence on the performance of the models. For text-conditioned tasks, refining the user-provided natural provided natural language prompts into keyword-enriched prompts has gained increasing attention. Several recent works have explored the prompt space by the use of prompt learning, such as CoCoOp (Zhou et al., 2022b), which uses conditional prompts to improve the model's generalization capabilities. AutoPrompt (Shin et al., 2020) explores tokens with the most significant gradient changes in the label likelihood to automate the prompt generation process. Fusedream (Liu et al., 2021) manipulates the CLIP (Radford et al., 2021) latent space by using GAN (Goodfellow et al., 2014) optimization to enrich the prompt space. Specialist Diffusion (Lu et al., 2023) augments the prompts to define the same image with multiple captions that convey the same meaning to improve the generalization of the image generation network. Another work (Lin et al., 2023) proposes to generate random sentences, including source and target domain, in order to calculate a mean difference that will serve as a direction while editing. The iEdit (Bodur et al., 2024) generates target prompts by changing words in the input caption in order to retrieve pseudo-target images and guide the model. The TokenCompose (Wang et al., 2024b) and OmniControlNet (Wang et al., 2024a) control the image generation in the token-level space. Compared to the prior works, our work takes a different approach by exploring whether we can obtain a powerful text embedding capable of guiding the video generation model through interpolation within the text embedding space.

C WORD EMBEDDING SPACE BEING INSUFFICIENT TO REPRESENT FOR ALL VIDEOS

In this section, we provide detailed proofs for Theorem C.8, showing that word embeddings are insufficient for representing all videos. We begin with a 1 dimensional case, where we assume all weights in function $f(x)$ are integers.

Lemma C.1 (Integer function bound in 1 dimension)**.** *If the following conditions hold:*

- *Let $V \in \mathbb{N}$ denote a positive integer.*
- *Let $f : [V]^n \to \mathbb{R}$ denote a linear function where weights are all integers.*
- *Let $x \in [V]^n$ denote the input of function f.*
- *Let $M := \max_x f(x), m := \min_x f(x)$.*
- *Let $\epsilon = 0.5$.*

Then we can show there exits a scalar $y \in [m, M]$ such that for any $x \in [V]^n$, $|f(x) - y| \geq \epsilon$.

Proof. Since $x \in [V]^n$, all entries of x are integers. Since function f is a linear function where all weights are integers, the output $f(x) \in \mathcal{Z}$ can only be integer.

Therefore, $m, M \in \mathcal{Z}$. We choose $y = m + 0.5$. Since for all $f(x)$ are integers, then we have $|f(x) - y| \geq 0.5$. $\square$

Then, we extend the above Lemma to d dimensional case.

Lemma C.2 (Integer function bound in d dimension)**.** *If the following conditions hold:*

- *Let $V \in \mathbb{N}$ denote a positive integer.*
- *Let $f : [V]^n \to \mathbb{R}^d$ denote a linear function where weights are all integers.*
- *Let $x \in [V]^n$ denote the input of function f.*
- *Let $M := \max_x \|f(x)\|_2, m := \min_x \|f(x)\|_2$.*
- *Let $\epsilon = 0.5\sqrt{d}$.*

Then we can show there exits a vector $y \in \mathbb{R}^d$, satisfying $m \leq \|y\|_2 \leq M$, such that for any $x \in [V]^n$, $\|f(x) - y\|_2 \geq \epsilon$.

Proof. Let $x_{\min} \in [V]^n$ denote the vector which satisfies $f(x_{\min}) = m$. Since all entries in x and f are integers, all entries in $f(x_{\min})$ are all integers.

For $i \in [d]$, let $z_i \in \mathcal{Z}$ denote the i-th entry of $f(x_{\min})$.

Then, we choose the vector $y \in \mathbb{R}^d$ as

$$y = \begin{bmatrix} z_1 + 0.5 \\ z_2 + 0.5 \\ \vdots \\ z_d + 0.5 \end{bmatrix}$$

Then, since all entries of $f(x)$ are integers, we have $\|f(x) - y\|_2 \geq 0.5\sqrt{d}$. $\square$

Then, we move on to a more complicated case, in which we do not make any assumptions about the function $f(x)$. We still begin by considering the 1 dimensional case.

Definition C.3 (Set Complement)**.** *If the following conditions hold:*

- *Let A, U denote two sets.*

Then, we use $U \backslash A$ to denote the complement of A in U:

$$U \backslash A := \{x \in U : x \notin A\}$$

Definition C.4 (Cover)**.** *If the following conditions hold:*

- *Let X denote a set.*
- *Let A denote an index set.*
- *For $\alpha \in A$, let $U_\alpha \subset X$ denote the subset of X, indexed by A.*
- *Let Let $C = \{U_\alpha : \alpha \in A\}$.*

Then we call C is a cover of X if the following holds:

$$X \subseteq \cup_{\alpha \in A} U_\alpha$$

Lemma C.5 (Any function bound in 1 dimension)**.** *If the following conditions hold:*

- *Let $V \in \mathbb{N}$ denote a positive integer.*
- *Let $f : [V]^n \to \mathbb{R}$ denote a function.*
- *Let $x \in [V]^n$ denote the input of function f.*
- *Let $M := \max_x f(x), m := \min_x f(x)$.*
- *Let $\epsilon = (M - m)/(2V^n)$.*

Then we can show there exits a scalar $y \in [m, M]$ such that for any $x \in [V]^n$, $|f(x) - y| \geq \epsilon$.

Proof. Assuming for all $y \in [m, M]$, there exists one $f(x)$, such that $|f(x)-y| < (M-m)/(2V^n)$. The overall maximum cover of all V^n points should satisfy

$$2 \cdot V^n \cdot |f(x) - y| < (M - m) \tag{1}$$

where the first step follows from there are total V^n possible choices for $f(x)$, and each choice has a region with length less than $2|f(x) - y|$. This is because the y can be either left side of $f(x)$, or can be on the right side of $f(x)$, for both case, we need to have $|f(x) - y| < (M - m)/(2V^n)$. So the length for each region of $f(x)$ should at least be $2|f(x) - y|$.

Eq (1) indicates the overall regions of V^n points can not cover all $[m, M]$ range, i.e. cannot become a cover (Definition C.4) of $[m, M]$. This is because each points can cover at most $2|f(x) - y| < (M - m)/V^n$ length, and there are total V^n points. So the maximum region length is less than $V^n \cdot (M - m)/V^n = (M - m)$. Note that the length of the range $[m, M]$ is $(M - m)$. Therefore, V^n points cannot cover all $[m, M]$ range.

We use $\mathcal{S}$ to denote the union of covers of all possible $f(x)$. Since the length of $\mathcal{S}$ is less than $M - m$, there exists at least one y lies in $[m, M] \backslash \mathcal{S}$ such that $|f(x) - y| \geq (M - m)/(2V^n)$. Here $\backslash$ denotes the set complement operation as defined in Definition C.3.

Then, we complete our proof. $\square$

Here, we introduce an essential fact that states the volume of a ℓ_2-ball in d dimensional space.

Then, we extend our 1 dimensional result on any function $f(x)$ to d dimensional cases.

Theorem C.6 (Word embeddings are insufficient to represent for all videos, restatement of Theorem 4.9)**.** *If the following conditions hold:*

- *Let n, d denote two integers, where n denotes the maximum length of the sentence, and all videos are in $\mathbb{R}^d$ space.*

- *Let $V \in \mathbb{N}$ denote the vocabulary size.*
- *Let $\mathcal{U} = \{u_1, u_2, \cdots, u_V\}$ denote the word embedding space, where for $i \in [V]$, the word embedding $u_i \in \mathbb{R}^k$.*
- *Let $\delta_{\min} = \min_{i,j \in [V], i \neq j} \|u_i - u_j\|_2$ denote the minimum ℓ_2 distance of two word embedding.*
- *Let $f : \mathbb{R}^{nk} \to \mathbb{R}^d$ denote the mapping from sentence space (discrete space $\{u_1, \ldots, u_V\}^n$) to video space $\mathbb{R}^d$.*
- *Let $M := \max_x \|f(x)\|_2, m := \min_x \|f(x)\|_2$.*
- *Let $\epsilon = ((M^d - m^d)/V^n)^{1/d}$.*

Then, we can show that there exits a video $y \in \mathbb{R}^d$, satisfying $m \leq \|y\|_2 \leq M$, such that for any sentence $x \in \{u_1, u_2, \cdots, u_V\}^n$, $\|f(x) - y\|_2 \geq \epsilon$.

Proof. Assuming for all y satisfying $m \leq \|y\|_2 \leq M$, there exists one $f(x)$, such that $|f(x) - y| < ((M^d - m^d)/V^n)^{1/d}$.

Then, according to Fact 4.3, for each $f(x)$, the volume of its cover is $\frac{\pi^{d/2}}{(d/2)!}((M^d - m^d)/V^n)$.

There are maximum total V^n $f(x)$, so the maximum volume of all covers is

$$V^n \cdot \frac{\pi^{d/2}}{(d/2)!}((M^d - m^d)/V^n) < \frac{\pi^{d/2}}{(d/2)!}(M^d - m^d) \quad (2)$$

The entire space of a d-dimensional ℓ_2 ball is $\frac{\pi^{d/2}}{(d/2)!}(M^d - m^d)$. However, according to Eq. (2) the maximum volume of the regions generated by all $f(x)$ is less than $\frac{\pi^{d/2}}{(d/2)!}(M^d - m^d)$. Therefore Eq. (2) indicates the cover of all V^n possible points does not cover the entire space for y.

Therefore, there exists a y satisfying $m \leq \|y\|_2 \leq M$, such that $\|f(x) - y\|_2 \geq ((M^d - m^d)/V^n)^{1/d}$.

Then, we complete our proof.

□

Definition C.7 (Bi-Lipschitzness)**.** *We say a function $f : \mathbb{R}^n \to \mathbb{R}^d$ is L-bi-Lipschitz if for all $x, y \in \mathbb{R}^n$, we have*

$$L^{-1}\|x - y\|_2 \leq \|f(x) - f(y)\|_2 \leq L\|x - y\|_2.$$

Then, we state our main result as follows

Theorem C.8 (Word embeddings are insufficient to represent for all videos, with Bi-Lipschitz condition)**.** *If the following conditions hold:*

- *Let n, d denote two integers, where n denotes the maximum length of the sentence, and all videos are in $\mathbb{R}^d$ space.*
- *Let $V \in \mathbb{N}$ denote the vocabulary size.*
- *Let $\mathcal{U} = \{u_1, u_2, \cdots, u_V\}$ denote the word embedding space, where for $i \in [V]$, the word embedding $u_i \in \mathbb{R}^k$.*
- *Let $\delta_{\min} = \min_{i,j \in [V], i \neq j} \|u_i - u_j\|_2$ denote the minimum ℓ_2 distance of two word embedding.*
- *Let $f : \mathbb{R}^{nk} \to \mathbb{R}^d$ denote the text-to-video generation model, which is also a mapping from sentence space (discrete space $\{u_1, \ldots, u_V\}^n$) to video space $\mathbb{R}^d$.*
- *Assuming $f : \mathbb{R}^{nk} \to \mathbb{R}^d$ satisfies the L-bi-Lipschitz condition (Definition C.7).*

- *Let* $M := \max_x \|f(x)\|_2, m := \min_x \|f(x)\|_2$.
- *Let* $\epsilon = \max\{0.5 \cdot \delta_{\min}/L, ((M^d - m^d)/V^n)^{1/d}\}$.

Then, we can show that there exits a video $y \in \mathbb{R}^d$*, satisfying* $m \leq \|y\|_2 \leq M$*, such that for any sentence* $x \in \{u_1, u_2, \cdots, u_V\}^n$*,* $\|f(x) - y\|_2 \geq \epsilon$*.*

Proof. Our goal is to prove that when the bi-Lipschitz condition (Definition C.7) holds for $f(x)$, the statement can be held with $\epsilon = \max\{0.5 \cdot \delta_{\min}/L, ((M^d - m^d)/V^n)^{1/d}\}$.

According to Lemma 4.9, we have that $\epsilon \geq ((M^d - m^d)/V^n)^{1/d}$. Then, we only need to prove that when $0.5 \cdot \delta_{\min}/L > ((M^d - m^d)/V^n)^{1/d}$, holds, $\epsilon = \max\{0.5 \cdot \delta_{\min}/L, ((M^d - m^d)/V^n)^{1/d}\} = 0.5 \cdot \delta_{\min}$, our statement still holds.

Since we have assume that the function $f(x)$ satisfies that for all $x, y \in \mathbb{R}^{nk}$, such that

$$\|f(x) - f(y)\|_2 \geq \|x - y\|_2/L. \tag{3}$$

According to the definition of $\delta_{\min}$, we have for all $i, j \in [V], i \neq j$, such that

$$\|u_i - u_j\|_2 \geq \delta_{\min} \tag{4}$$

Combining Eq. (3) and (4), we have for all $i, j \in [V], i \neq j$

$$\|f(u_i) - f(u_j)\|_2 \geq \delta_{\min}/L \tag{5}$$

We choose $y = f(\frac{1}{2}(u_i + u_j))$ for any $i, j \in [V], i \neq j$

Then, for all $k \in [V]$, we have

$$\begin{aligned} \|y - f(u_i)\|_2 &\geq \|\frac{1}{2}(u_i + u_j) - u_k\|_2/L \\ &\geq 0.5 \cdot \delta_{\min}/L \end{aligned}$$

where the first step follows from $f(x)$ satisfies the bi-Lipschitz condition, the second step follows from Eq. (5).

Therefore, when we have $0.5 \cdot \delta_{\min}/L > ((M^d - m^d)/V^n)^{1/d}$ holds, then we must have $\epsilon = 0.5 \cdot \delta_{\min}/L$.

Considering all conditions we discussed above, we are safe to conclude that $\epsilon = \max\{0.5 \cdot \delta_{\min}/L, ((M^d - m^d)/V^n)^{1/d}\}$ ∎

Table 2: **Statement Reference Table.** This table shows the relationship between definitions and algorithms used in the paper, helping readers easily track where each term is defined and referenced.

Statements	Comment	Call	Called by
Def. 4.1	Define linear interpolation	None	Alg. 2, Alg. 1
Def. 4.2	Define cosine similarity calculator	None	Alg. 2, Alg. 1
Def. 4.4	Define attention layer	None	Alg. 4, Def. 4.7
Def. 4.5	Define convolution layer	None	Alg. 4, Def. 4.7
Def. 4.6	Define linear projection	None	Alg. 4, Def. 4.7
Def. 4.7	Define 3D attention	Def. 4.4, Def. 4.5, Def. 4.6	Alg. 4, Def. 4.8
Def. 4.8	Define text to video generation model	Def. 4.7	Def. 2.1
Def. 2.1	Define optimal interpolation embedding	Def. 4.8	Alg. 3
Alg. 4	3D Attention algorithm	Def. 4.4, Def. 4.5, Def. 4.6, Def. 4.7	None
Alg. 2	Cosine similarity calculator algorithm	Def. 4.1, Def. 4.2	Alg. 1
Alg. 1	Find optimal interpolation algorithm	Def. 4.1, Def. 4.2, Alg. 2	Alg. 3
Alg. 3	Video interpolation algorithm	Alg. 1	None

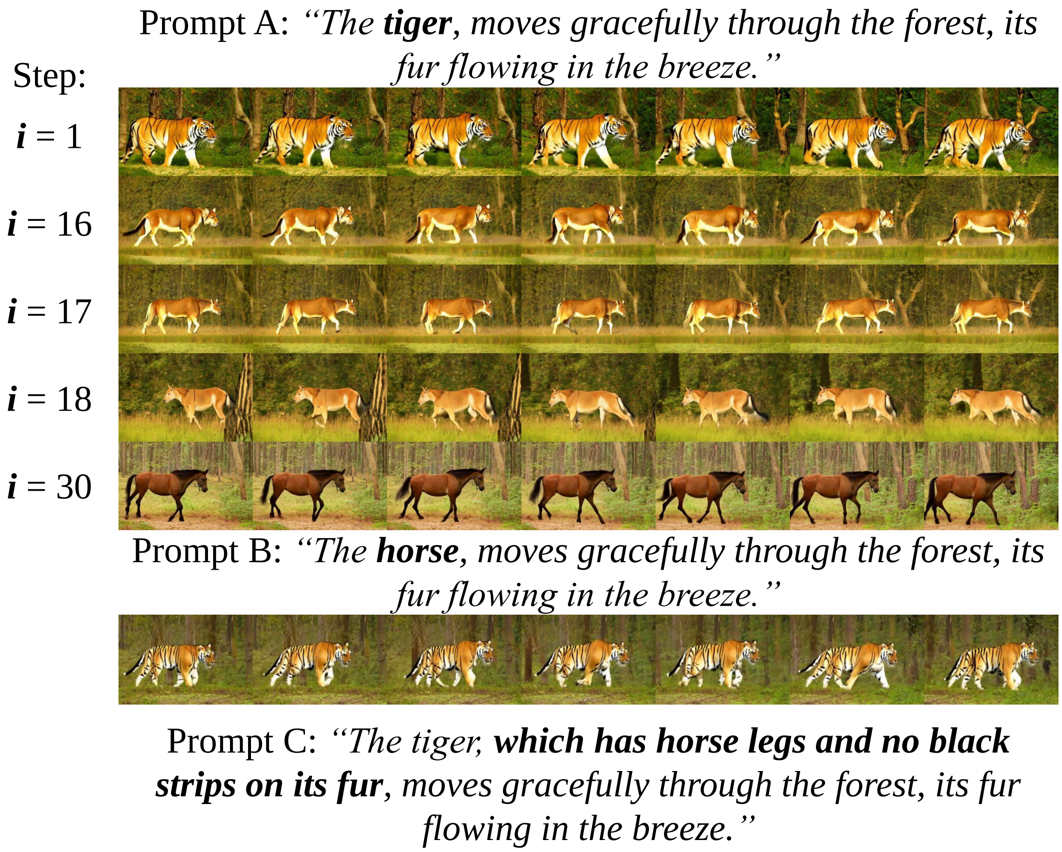


Figure 5: **Mixture of ["Tiger"] and ["Horse"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 17-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

D More Examples

In this section, we will show more experimental results that the video generated directly from the guidance prompt does not exhibit the desired mixed features from the prompts.

E Full Algorithm

In this section, we provide the algorithm for 3D attention in Algorithm 4.

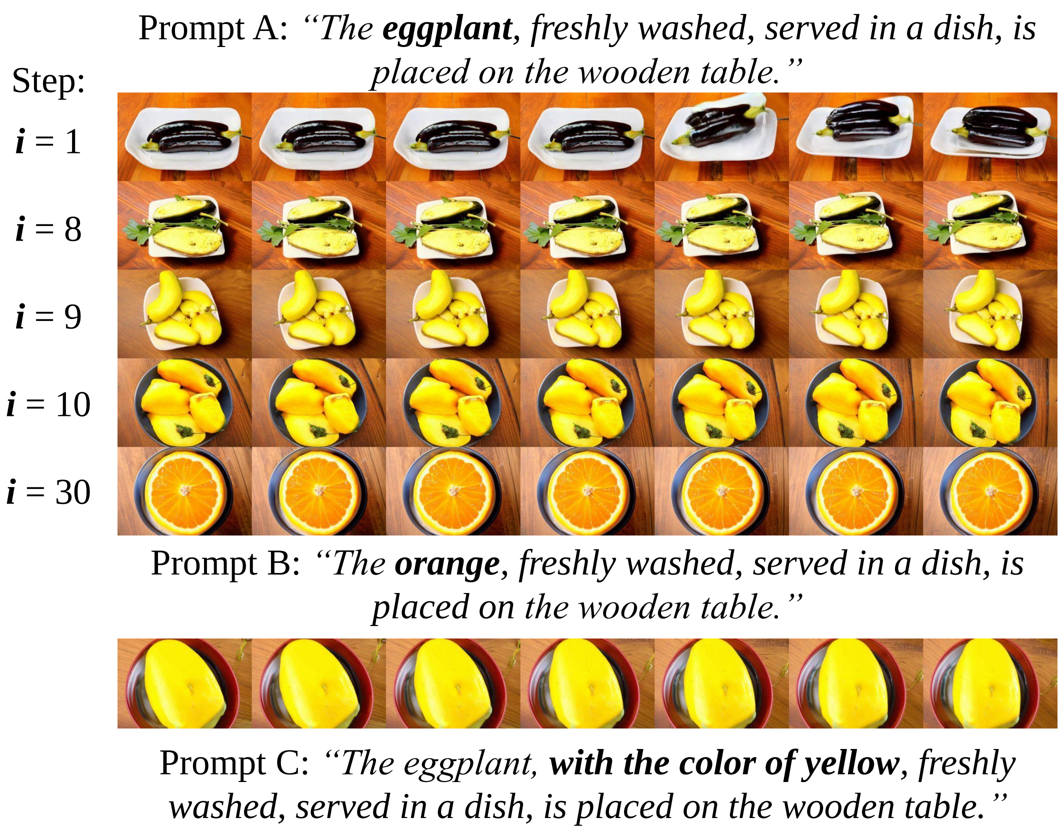


Figure 6: **Mixture of ["Eggplant"] and ["Orange"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 9-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

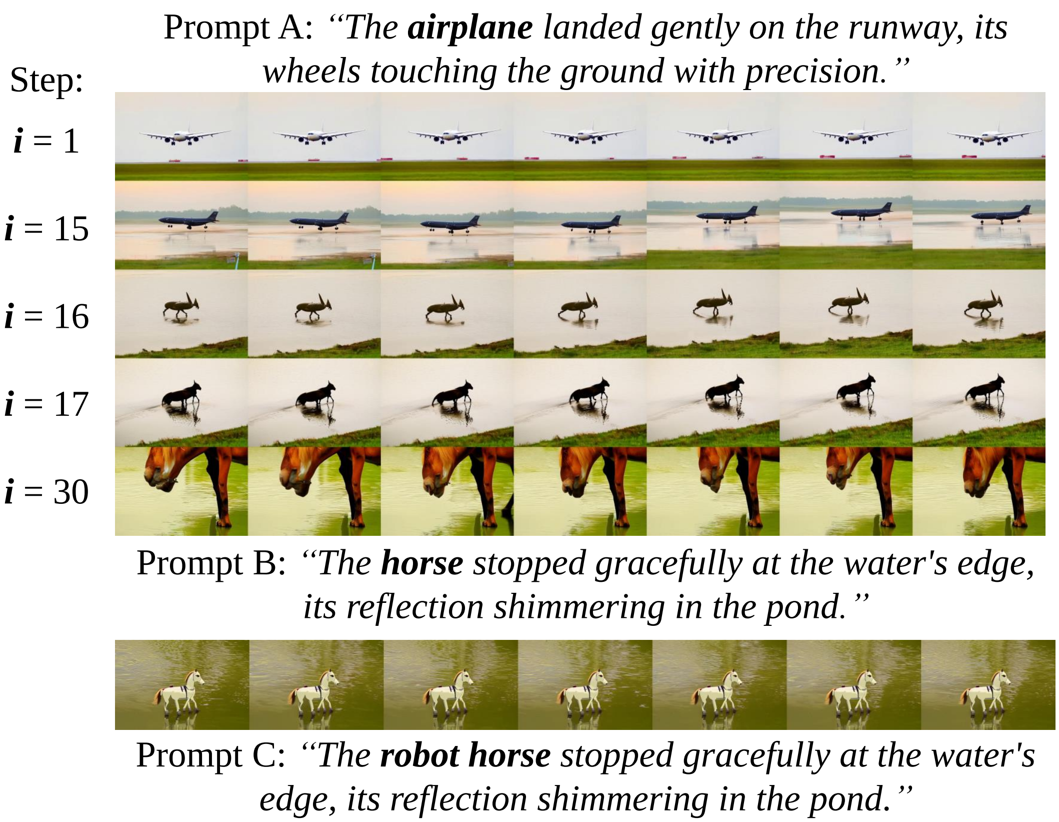


Figure 7: **Mixture of [“Airplane”] and [“Horse”]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

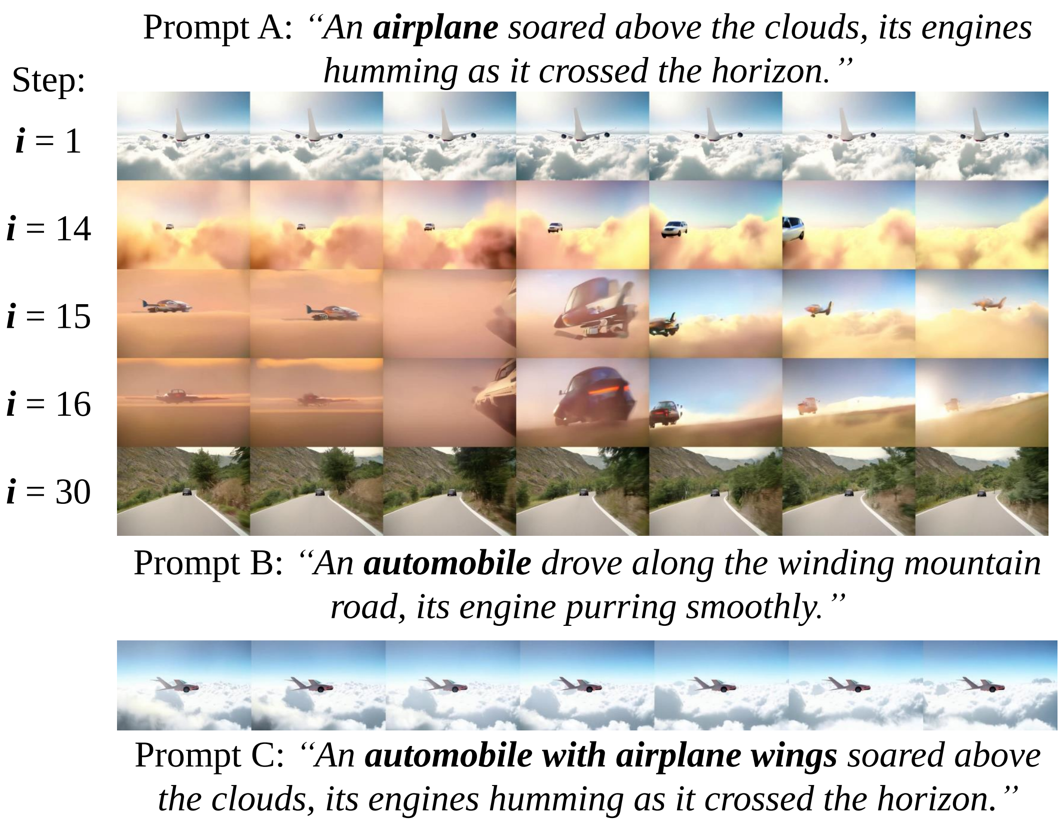


Figure 8: **Mixture of ["Airplane"] and ["Automobile"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

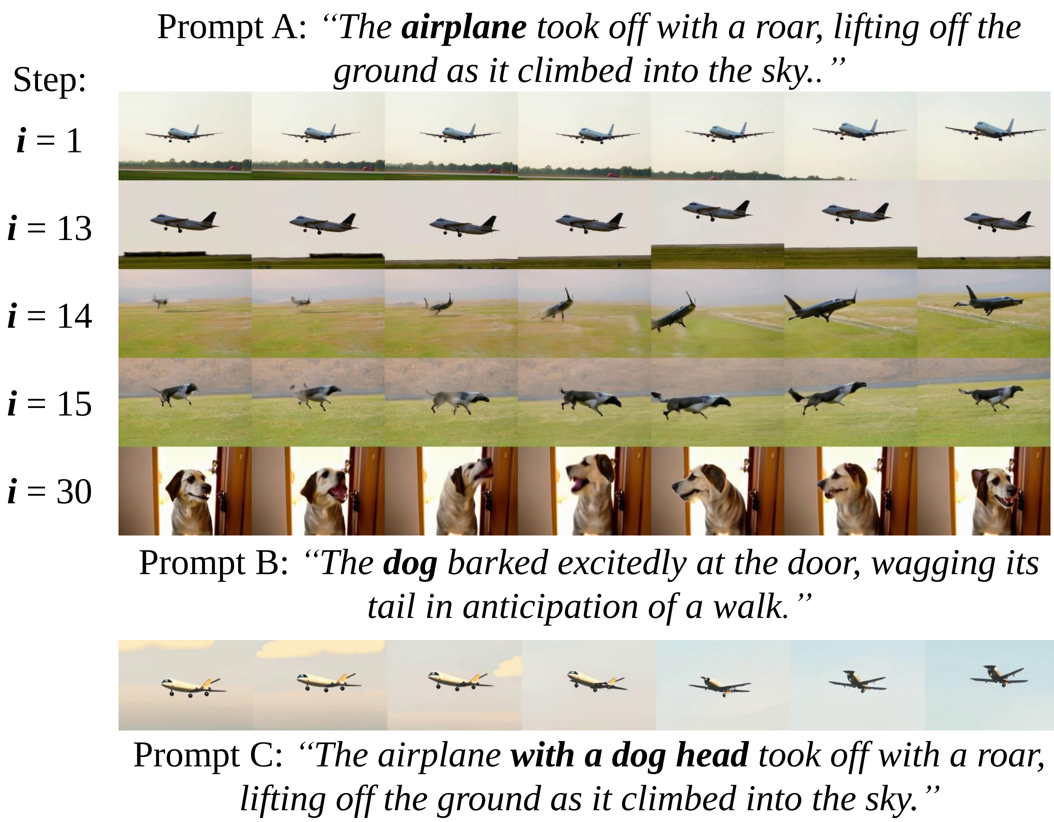


Figure 9: **Mixture of ["Airplane"] and ["Dog"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 14-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

Prompt A: *"An **airplane** glided smoothly in the sky."*

Prompt B: *"A **deer** moved quietly in the woods."*

Prompt C: *"An **airplane with antlers**, glided smoothly in the sky."*

Figure 10: **Mixture of ["Airplane"] and ["Deer"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

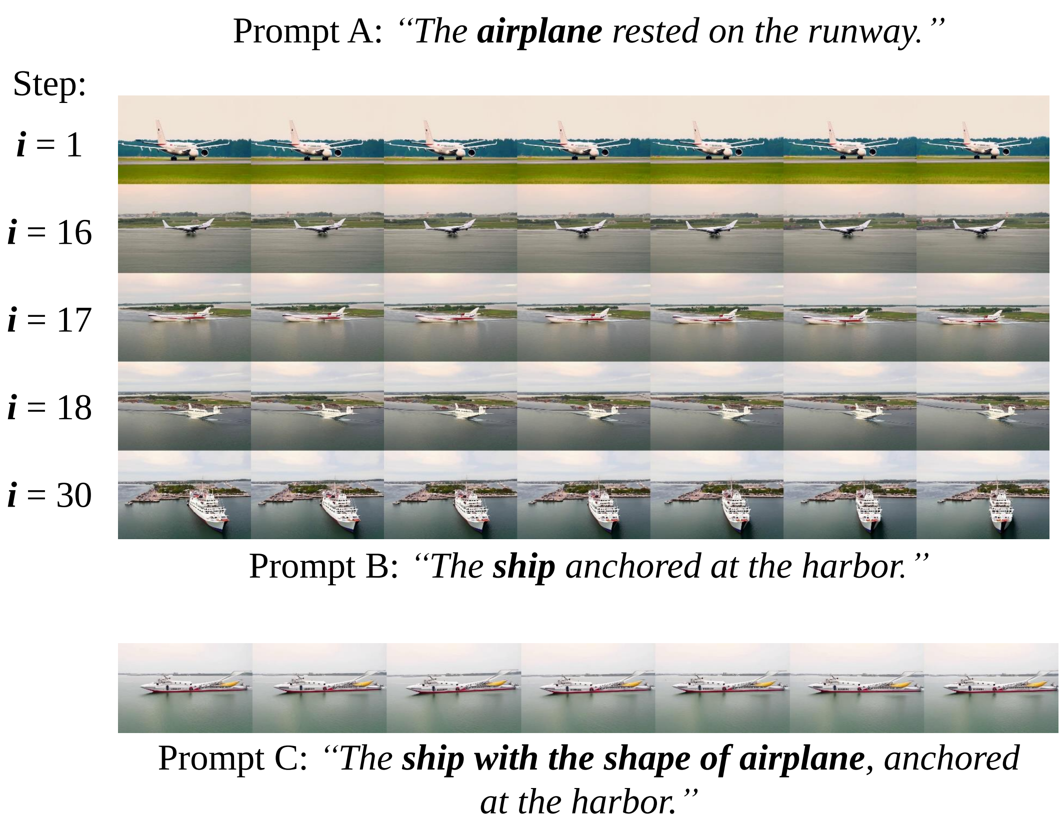


Figure 11: **Mixture of ["Airplane"] and ["Ship"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 17-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

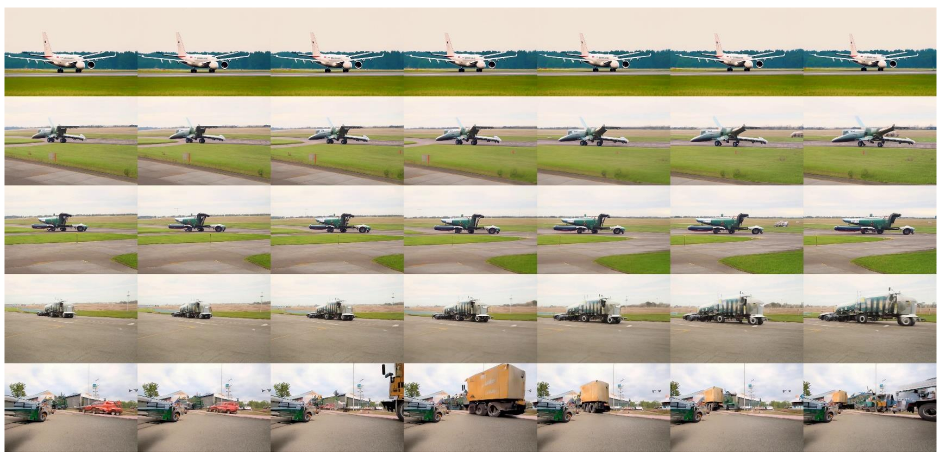


Figure 12: **Mixture of ["Airplane"] and ["Truck"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 17-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

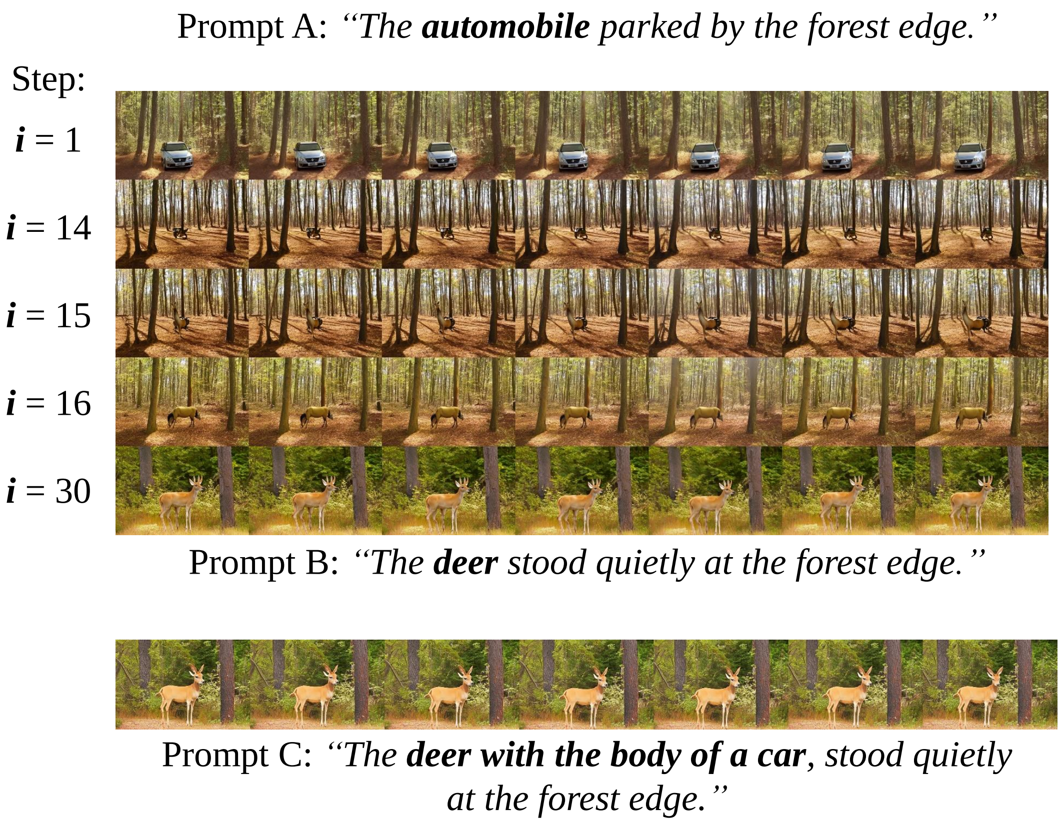


Figure 13: **Mixture of ["Automobile"] and ["Deer"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

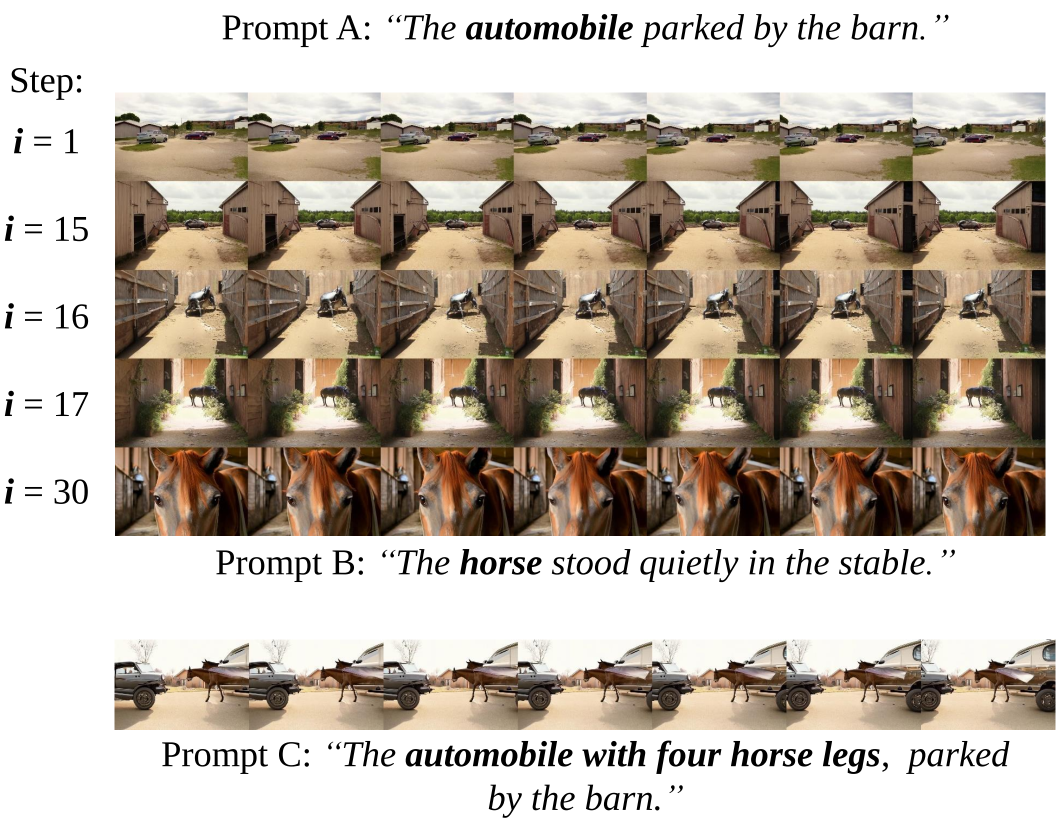


Figure 14: **Mixture of ["Automobile"] and ["Horse"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

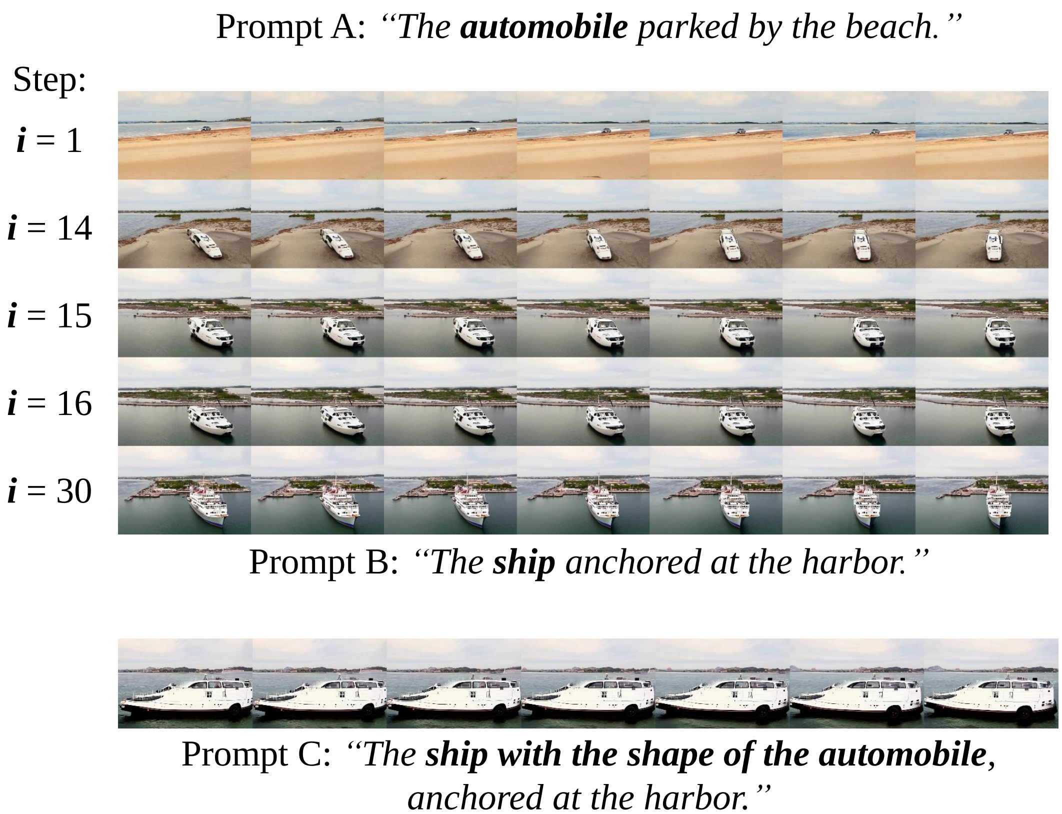


Figure 15: **Mixture of ["Automobile"] and ["Ship"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

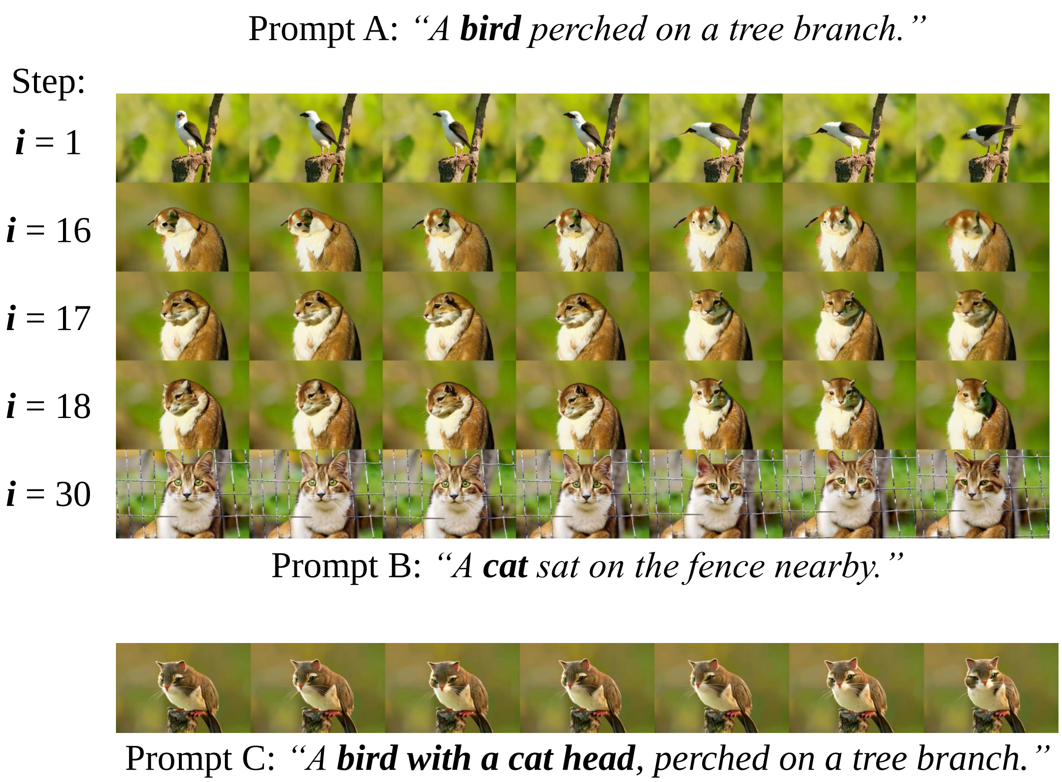


Figure 16: **Mixture of ["Bird"] and ["Cat"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 17-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

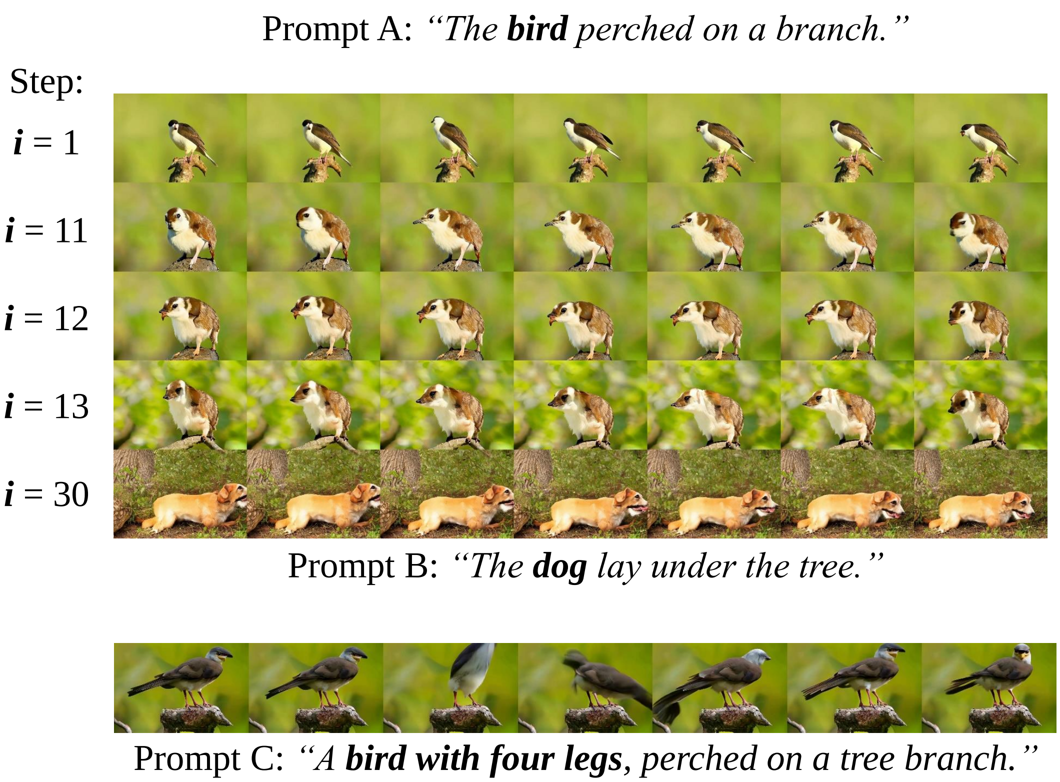


Figure 17: **Mixture of ["Bird"] and ["Dog"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 12-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

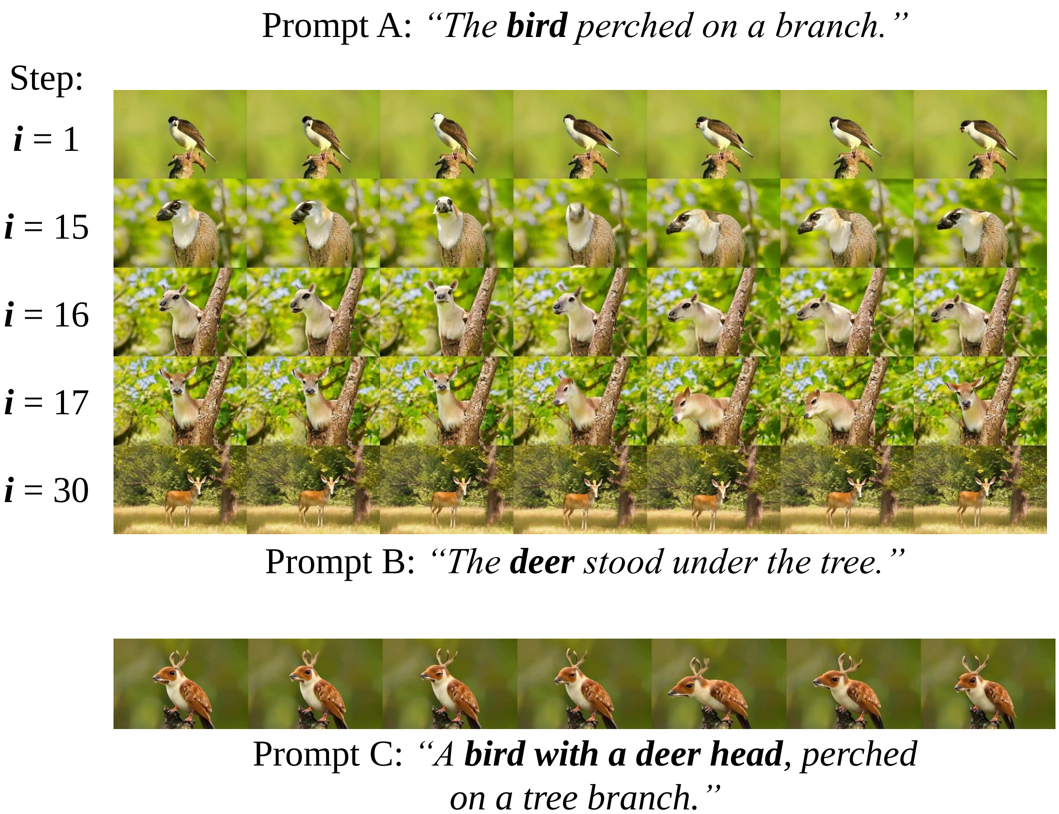


Figure 18: **Mixture of ["Bird"] and ["Deer"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

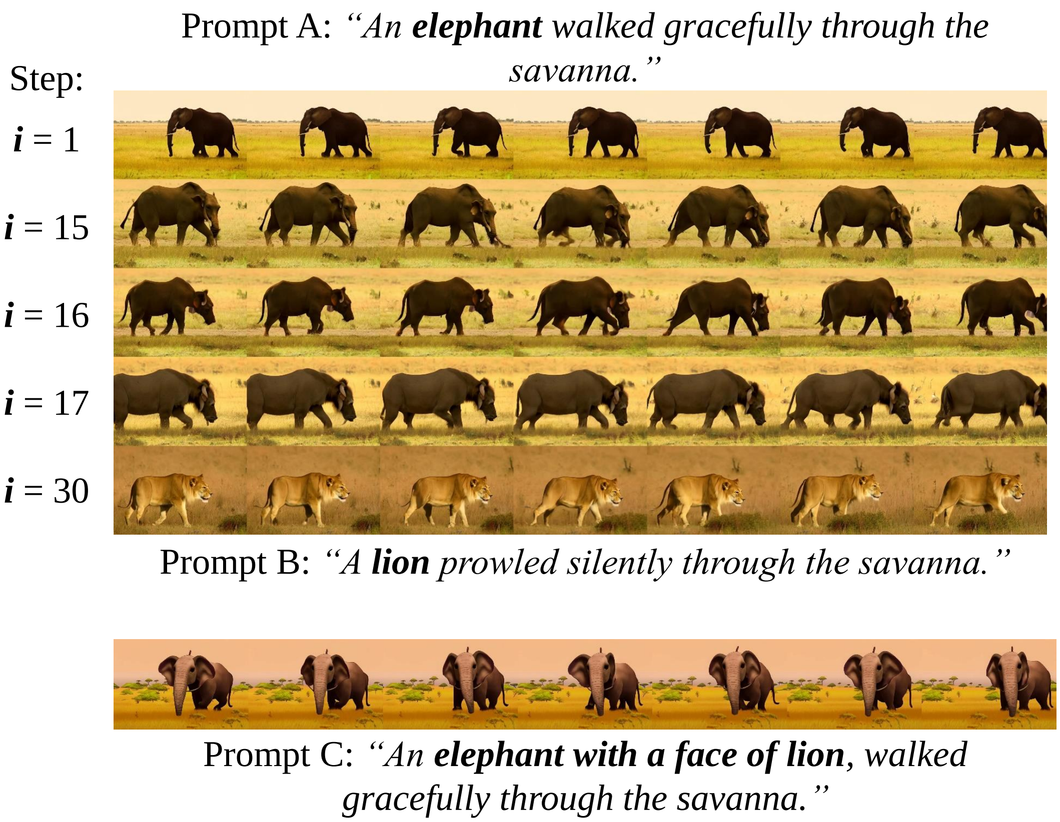


Figure 19: **Mixture of [“Elephant”] and [“Lion”]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

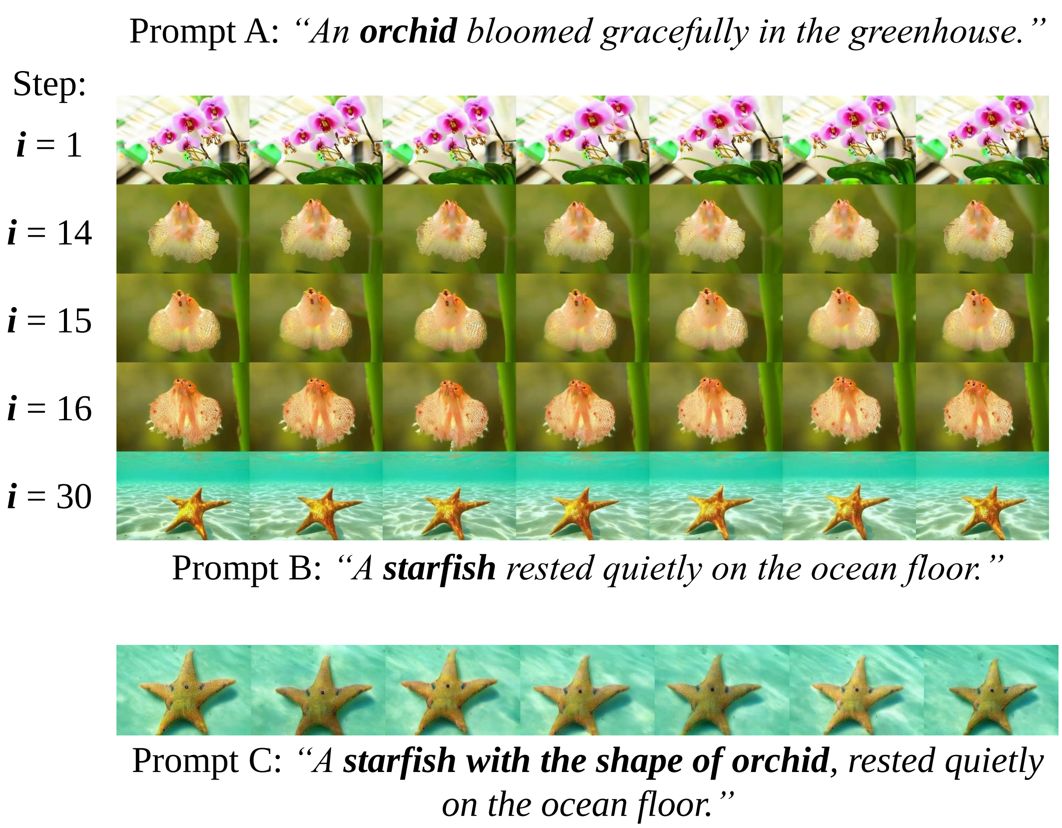


Figure 20: **Mixture of ["Orchid"] and ["Starfish"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

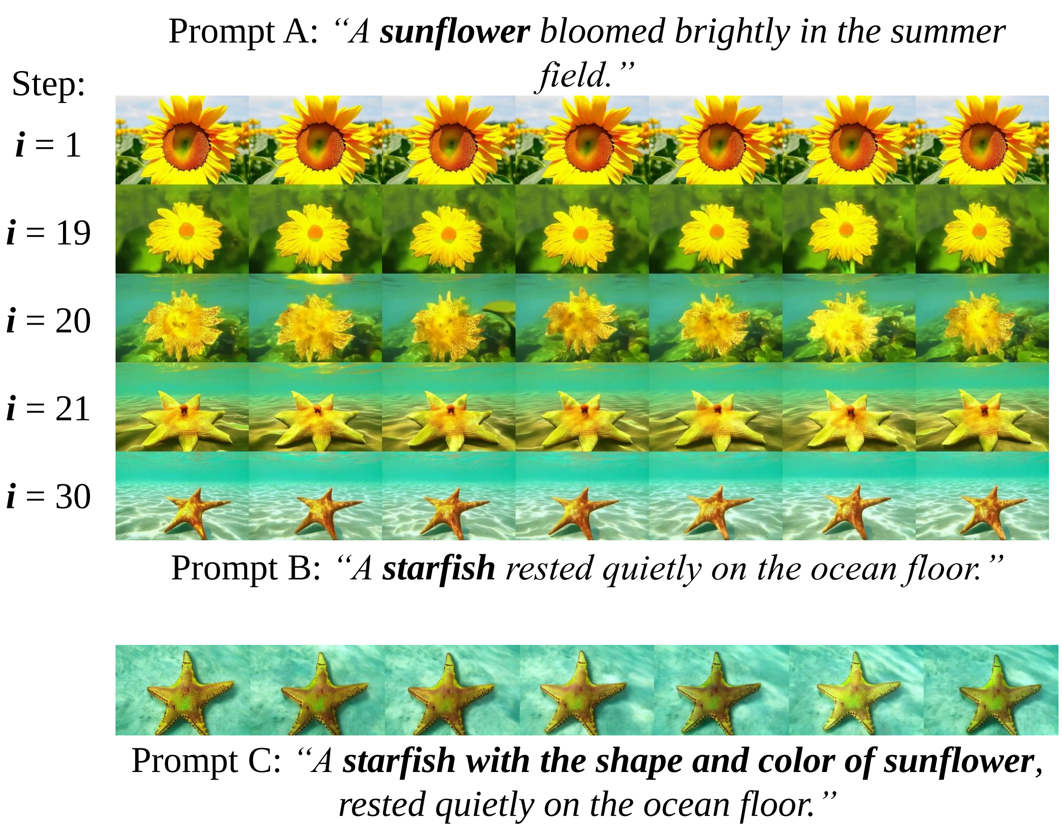


Figure 21: **Mixture of ["Sunflower"] and ["Starfish"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 20-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

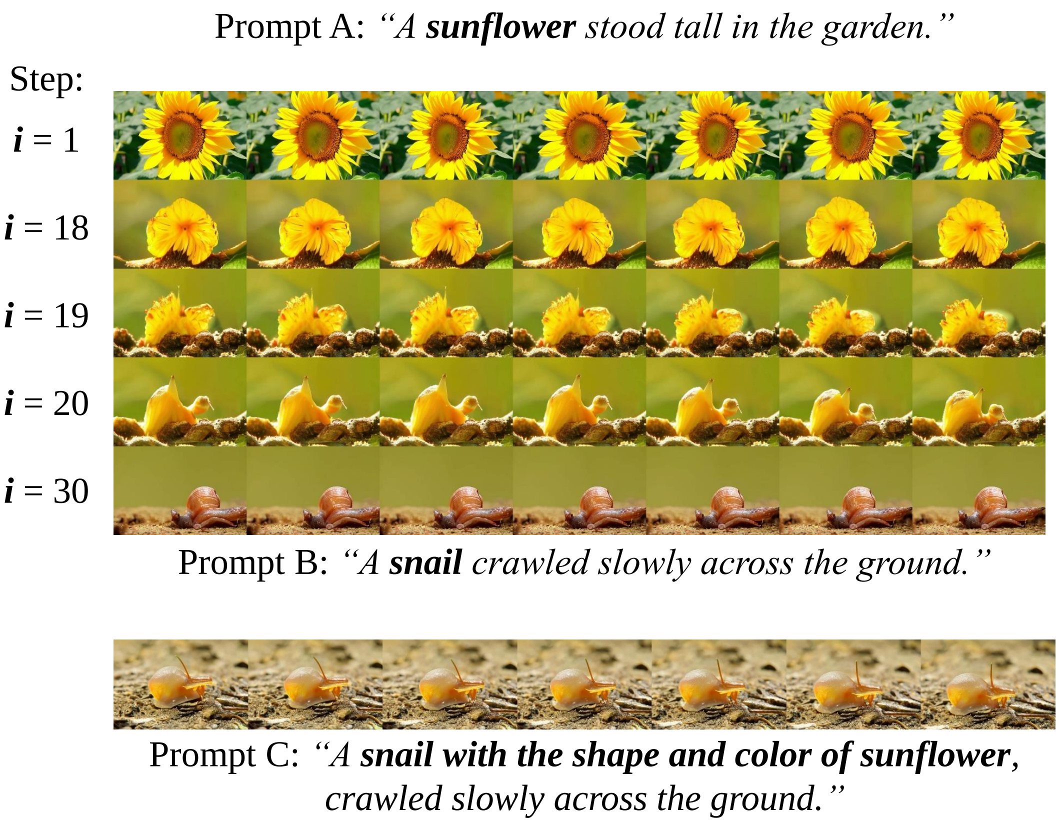


Figure 22: **Mixture of ["Sunflower"] and ["Snail"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 19-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

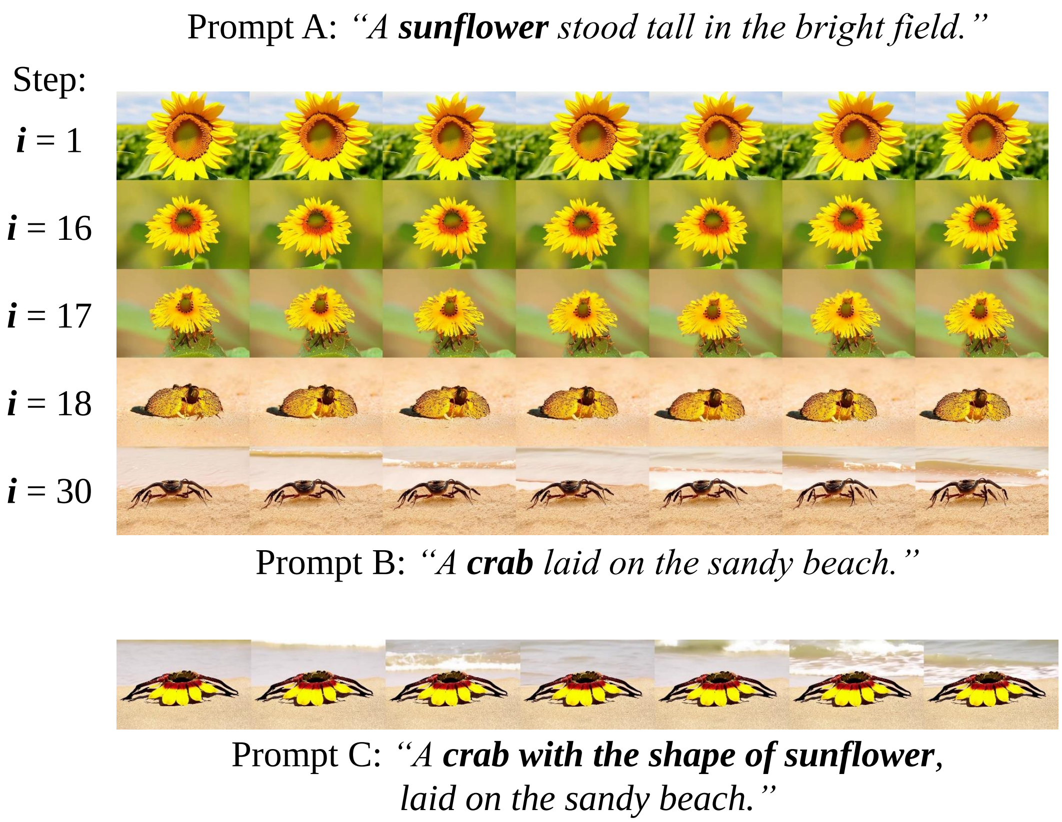


Figure 23: **Mixture of [“Sunflower”] and [“Crab”]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 17-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

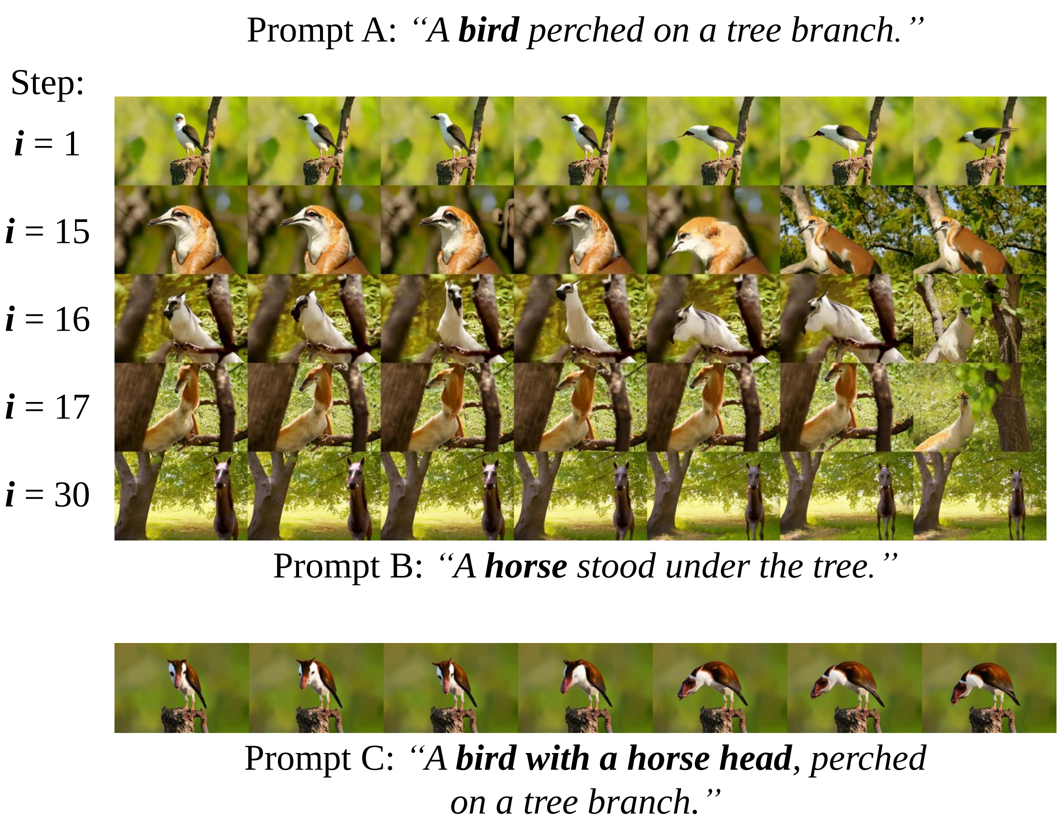


Figure 24: **Mixture of [“Bird”] and [“Horse”]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

Prompt A: *“A **butterfly** landed delicately on a vibrant petal.”*

Step:

i = 1

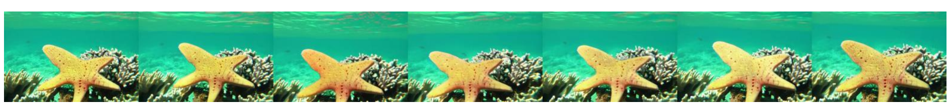
i = 15

i = 16

i = 17

i = 30

Prompt B: *“A **starfish** clung to a coral reef beneath the waves.”*



Prompt C: *“A **starfish with the shape of butterfly**, clung to a coral reef beneath the waves.”*

Figure 25: **Mixture of [“Butterfly”] and [“Starfish”]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

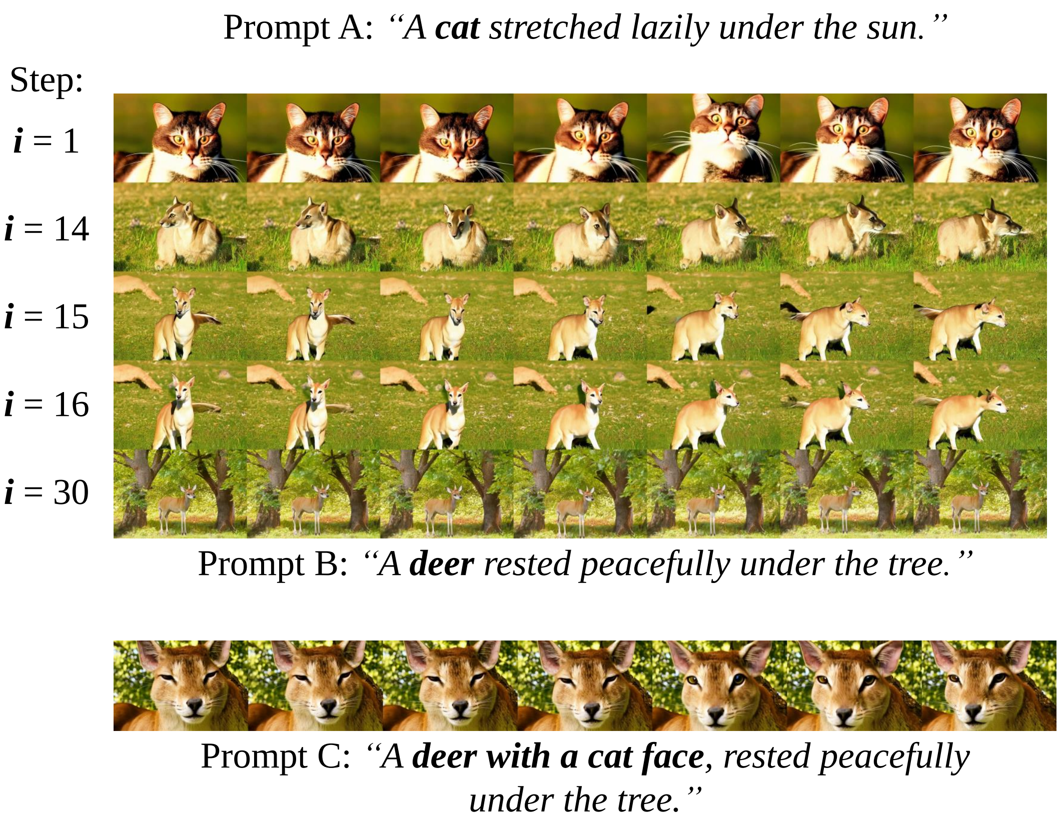


Figure 26: **Mixture of ["Cat"] and ["Deer"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

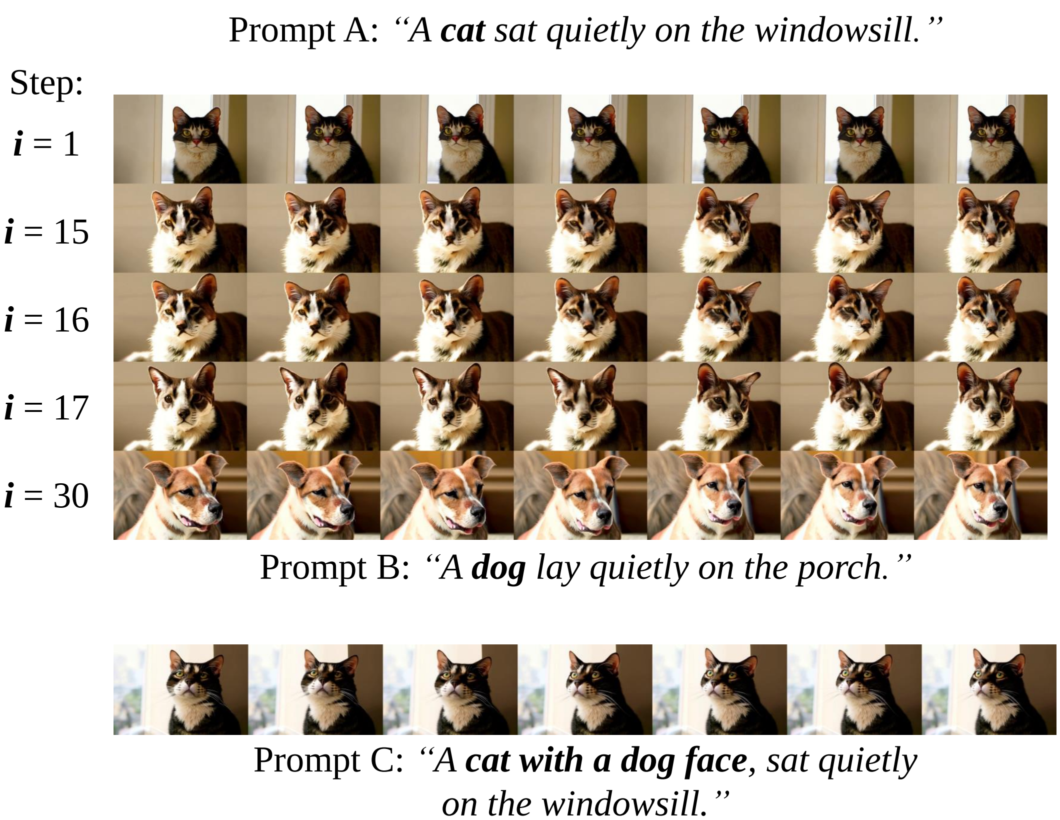


Figure 27: **Mixture of ["Cat"] and ["Dog"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

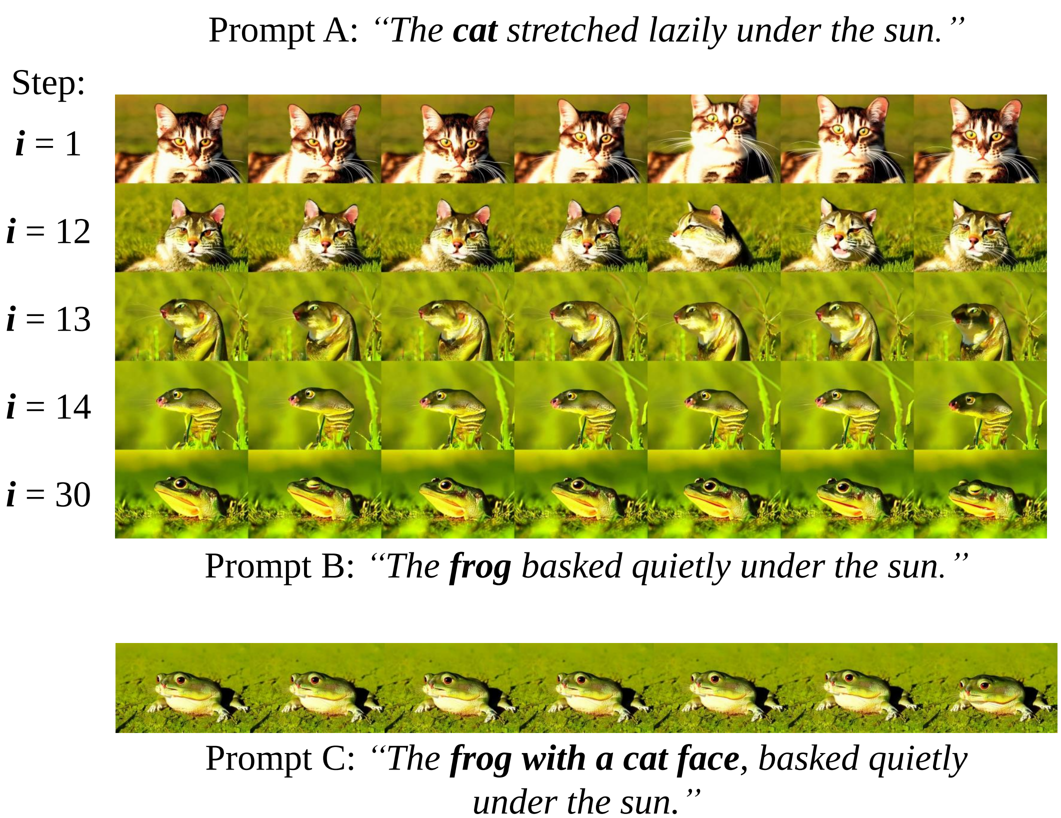


Figure 28: **Mixture of ["Cat"] and ["Frog"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 13-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

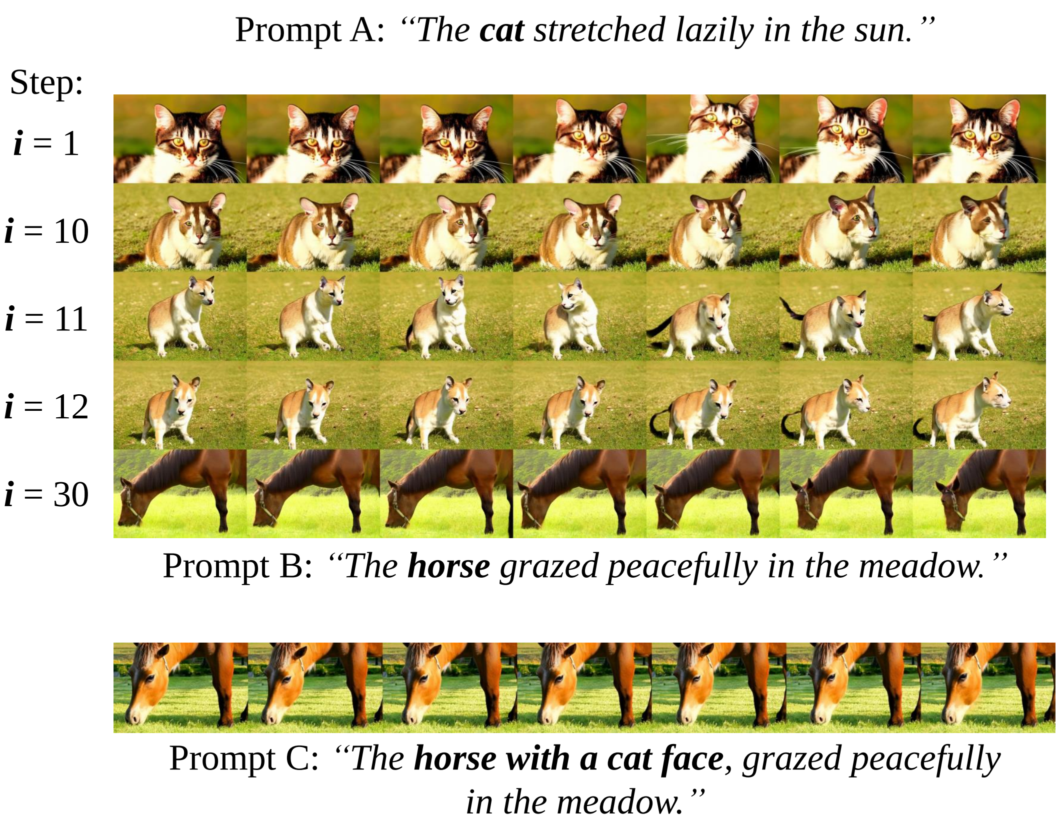


Figure 29: **Mixture of [“Cat”] and [“Horse”]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 11-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

Prompt A: *"The **otter** rested on a smooth stone by the water."*

Step:

i = 1

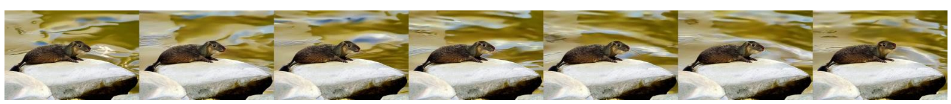
i = 13

i = 14

i = 15

i = 30

Prompt B: *"The **lizard** basked on the tree bark under the sun."*



Prompt C: *"The **otter with lizard skin**, rested on a smooth stone by the water."*

Figure 30: **Mixture of ["Otter"] and ["Lizard"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 14-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

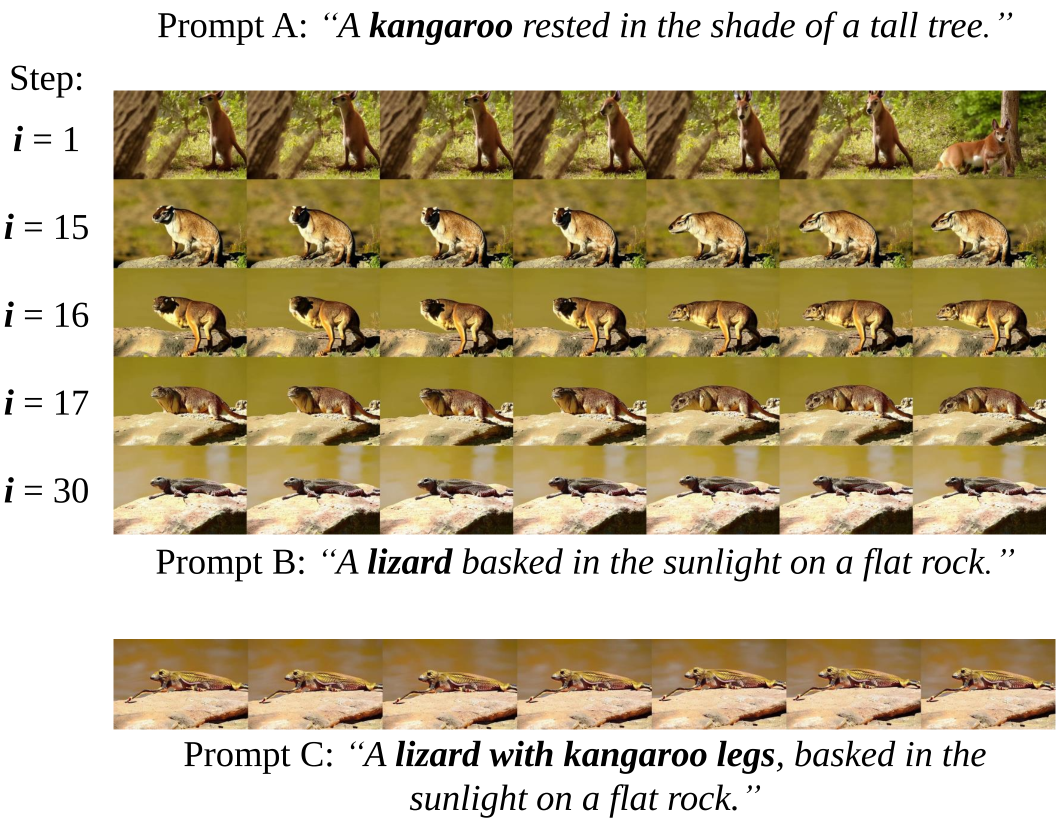


Figure 31: **Mixture of ["Kangaroo"] and ["Lizard"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

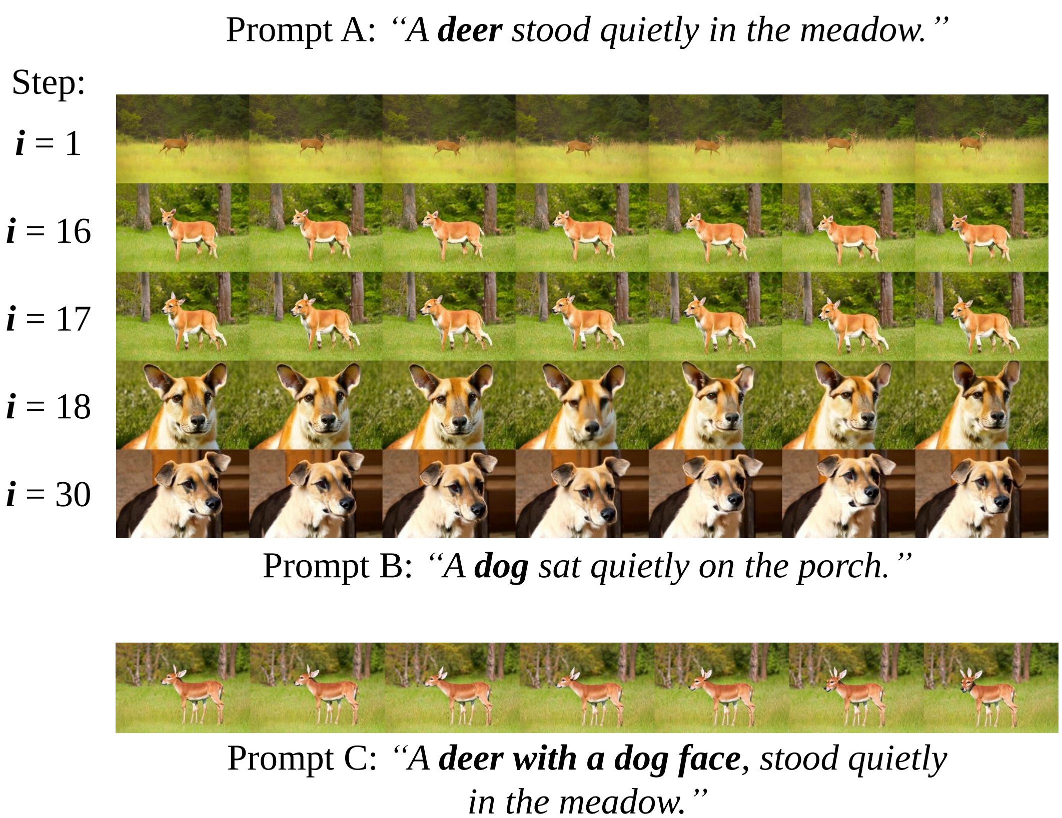


Figure 32: **Mixture of ["Deer"] and ["Dog"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 17-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

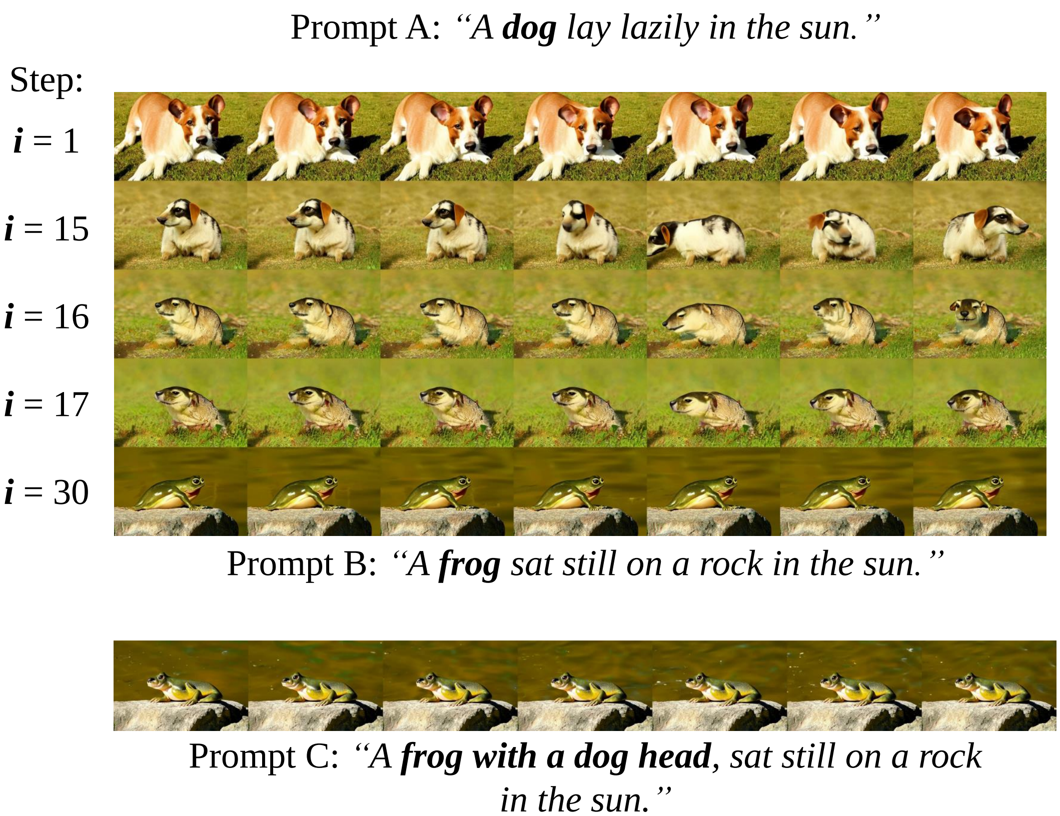


Figure 33: **Mixture of ["Dog"] and ["Frog"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 16-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

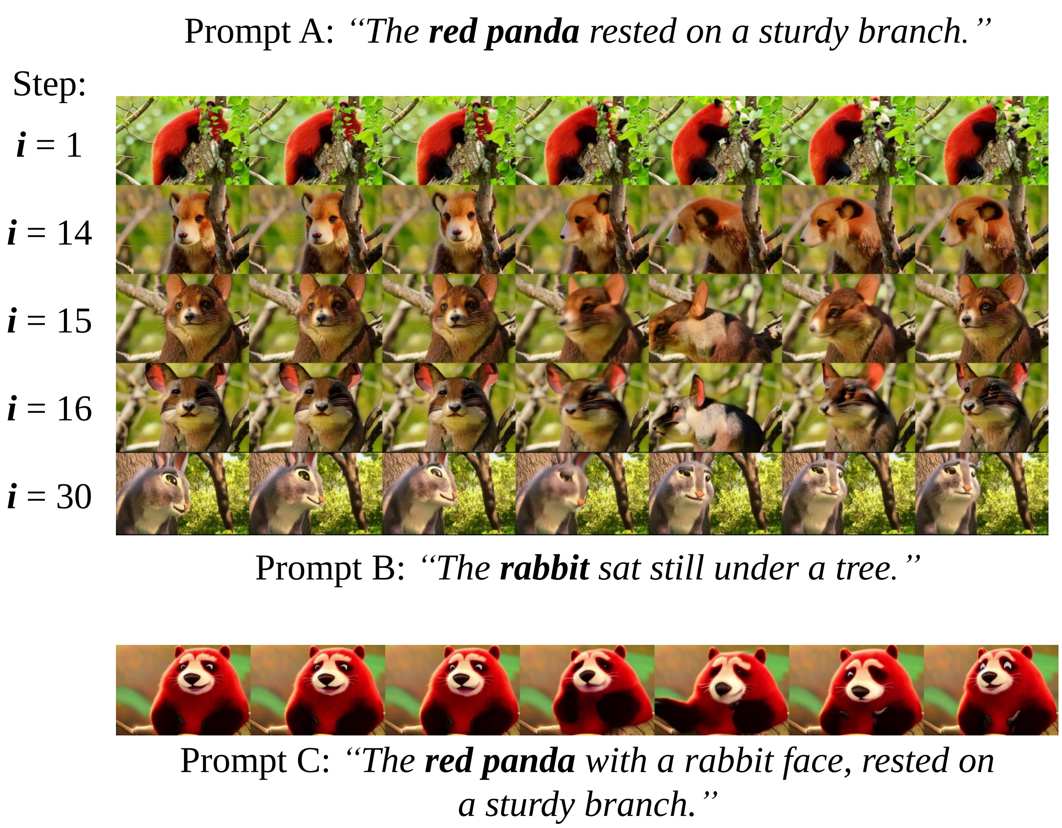


Figure 34: **Mixture of ["Red Panda"] and ["Rabbit"]**. Our objective is to mix the features described in Prompt A and Prompt B with the guidance of Prompt C. We set the total number of interpolation steps to 30. Using Algorithm 1, we identify the 15-th interpolation embedding as the optimal embedding and generate the corresponding video. The video generated directly from Prompt C does not exhibit the desired mixed features from Prompts A and B.

Algorithm 4 3D Attention

datastructure 3D ATTENTION ▷ Definition 4.7
members
$n \in \mathcal{N}$: the length of input sequence
$n_f \in \mathcal{N}$: the number of frames
$h \in \mathcal{N}$: the hight of video
$w \in \mathcal{N}$: the width of video
$d \in \mathcal{N}$: the hidden dimension
$c \in \mathcal{N}$: the channel of video
$c_{\text{patch}} \in \mathbb{R}^{n \times d}$: the channel of patch embedding.
$E_t \in \mathbb{R}^{n \times d}$: the text embedding.
$E_{\text{video}} \in \mathbb{R}^{n_f \times h \times w \times c}$: the video embedding.
$E_{\text{patch}} \in \mathbb{R}^{n_f \times h' \times w' \times c_{\text{patch}}}$: the patch embedding.
$\phi_{\text{conv}}(X, c_{\text{in}}, c_{\text{out}}, p, s)$: the convolution layer. ▷ Definition 4.5
$\mathsf{Attn}(X)$: the attention block. ▷ Definition 4.4
$\phi_{\text{linear}}(X)$: the linear projection. ▷ Definition 4.6
end members

procedure 3D ATTENTION($E_t \in \mathbb{R}^{n \times d}$, $E_v \in \mathbb{R}^{n_f \times h \times w \times c}$)
/* E_{patch} dimension: $[n_f, h, w, c_v] \to [n_f, h', w', c_{\text{patch}}]$ */
$E_{\text{patch}} \leftarrow \phi_{\text{conv}}(E_v, c_v, c_{\text{patch}}, p = 2, s = 2)$
/* E_{patch} dimension: $[n_f, h', w', c_{\text{patch}}] \to [n_f \times h' \times w', c_{\text{patch}}]$ */
$E_{\text{patch}} \leftarrow \text{reshape}(E_{\text{patch}})$
/* E_{hidden} dimension: $[n + n_f \times h' \times w', c_{\text{patch}}]$ */
$E_{\text{hidden}} \leftarrow \text{concat}(E_t, E_{\text{patch}})$
/* E_{hidden} dimension: $[n + n_f \times h' \times w', c_{\text{patch}}]$ */
$E_{\text{hidden}} \leftarrow \mathsf{Attn}(E_{\text{hidden}})$
/* E_t dimension: $[n, d]$ */
/* E_{patch} dimension: $[n_f \times h' \times w', c_{\text{patch}}]$ */
$E_t, E_{\text{patch}} \leftarrow \text{split}(E_{\text{hidden}})$
/* E_v dimension: $[n_f \times h' \times w', c_{\text{patch}}] \to [n_f \times h \times w, c_v]$ */
$E_v \leftarrow \phi_{\text{linear}}(E_{\text{patch}})$
/* E_v dimension: $[n_f \times h \times w, c_v] \to [n_f, h, w, c_v]$ */
$E_v \leftarrow \text{reshape}(E_v)$
Return E_v
end procedure

LLM USAGE DISCLOSURE

LLMs were used only to polish language, such as grammar and wording. These models did not contribute to idea creation or writing, and the authors take full responsibility for this paper's content.