Promptus: Can Prompt Streaming Replace Video Streaming

Jiangkai Wu[1], **Liming Liu**[1], **Yunpeng Tan**[1], **Junlin Hao**[1], **Liang Zhang**[2], **Xinggong Zhang**[1]

[1]Wangxuan Institute of Computer Technology, Peking University
[2]Huawei Technologies Ltd.
{jiangkai.wu, llm, yunpengtan, junlin.hao}@stu.pku.edu.cn, zhangliang1@huawei.com, zhangxg@pku.edu.cn

Abstract

With the exponential growth of video traffic, traditional video streaming systems are approaching their limits in communication capacity. To further reduce bitrate while maintaining quality, we propose **Promptus**, a disruptive semantic communication system that *streams prompts instead of videos*. Promptus represents the real-world video with a series of "prompts" for delivery and employs Stable Diffusion to generate the same video at the receiver. To ensure that the generated video is pixel-aligned with the original video, a gradient descent-based prompt fitting framework is proposed. Further, a low-rank decomposition-based bitrate control algorithm is introduced to achieve adaptive bitrate. For inter-frame compression, an interpolation-aware fitting algorithm is proposed. Evaluations across various video genres demonstrate that, compared to H.265, Promptus can achieve more than a 4x bandwidth reduction while preserving the same perceptual quality. On the other hand, at extremely low bitrates, Promptus can enhance the perceptual quality by 0.139 and 0.118 (in LPIPS) compared to VAE and H.265, respectively, and decreases the ratio of severely distorted frames by 89.3% and 91.7%. Our work opens up a new paradigm for efficient video communication.

Code — https://github.com/pku-netvideo/Promptus
Extended version — https://arxiv.org/abs/2405.20032

1 Introduction

With the rapid development of streaming applications (such as YouTube), the traffic of internet video has been continuously growing. To reduce traffic, video codecs represented by VP9 (Mukherjee et al. 2015), H.265 and H.266 (Wieckowski et al. 2021) are widely used to compress videos. These codecs achieve compression by removing redundancies (e.g., spatial and temporal redundancy). However, these redundancies are limited, so there is an upper bound on the transmission efficiency. To further compress the video, non-redundant content will be discarded, which will greatly degrade the video quality, such as causing blurring and blocking artifacts. In recent years, some deep learning-based codecs (Lu et al. 2019; Lin et al. 2020) and streaming frameworks (Yeo et al. 2022; Sivaraman et al. 2024; Li et al. 2023) have been proposed to improve transmission efficiency, but they are limited either by performance or generality.

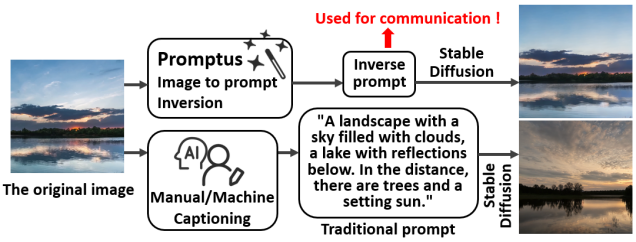


Figure 1: Promptus can invert a given video into prompts. Based on these prompts, Stable Diffusion can generate an almost identical video to the original. So Promptus streams prompts instead of videos, significantly saving bandwidth.

With the popularity of generative AI, Stable Diffusion (SD) (Stability.ai 2024b) has attracted extensive attention thanks to its powerful text-to-image generation. By pre-training on internet-scale datasets, SD learns prior knowledge of almost all visual domains, and also learns the mapping from text to images. So SD can generate high-fidelity images based on a brief prompt composed of a few words.

We are motivated by this question: Is it possible for SD to replace the video codecs? During streaming, the sender transmits prompts instead of encoded videos, and the receiver generates videos instead of decoding videos. In this way, the traffic of internet video is reduced from the video scale to the text scale, greatly improving transmission efficiency. In this paper, we propose Promptus, a system that represents video frames with SD prompts, achieving ultra-low bitrate video streaming. To bring this vision to fruition, we address the following challenges:

First, how to ensure **pixel alignment** between the generated frames and the real frames. To invert a frame into a prompt, the most straightforward and powerful approach is to manually write a textual description. But the frame generated using this description can only guarantee semantic consistency with the real frame, while the differences in pixels are often substantial, as shown in Figure 1. To achieve pixel alignment, Promptus proposes a gradient descent-based prompt fitting framework. Specifically, the prompt is randomly initialized and then used to generate a frame. The

pixel-wise loss between the generated frame and the real frame is calculated. The gradient of the loss with respect to the prompt will be calculated, and the prompt will be iteratively optimized using gradient descent. To implement this framework, Promptus employs single-step denoising instead of iterative denoising to avoid unrolled long gradient chains. Second, Promptus uses embeddings as prompts instead of text to avoid non-differentiability. Third, Promptus uses a noisy previous frame instead of random noise to reduce latent distance. Fourth, Promptus combines reconstruction and perceptual loss to enhance the perceptual quality.

Second, how to **control the bitrate** of the prompt. Since Promptus uses embeddings as prompts, which are matrices with fixed dimensions, the bitrate of the prompt cannot adapt to dynamic network bandwidth. To address this, Promptus proposes a low-rank bitrate control algorithm. Specifically, Promptus integrates the inverse process of low-rank decomposition into the gradient descent, directly fitting the decomposed prompt. The rank is used to control the trade-off between the quality and bitrate of the prompt. When the rank is higher, the representational capability of the prompt is better, and it can describe more details in the video, but the data size is also larger. So Promptus adaptively selects the prompt rank based on the currently available bandwidth.

Third, how to perform **inter-frame compression** on prompts. Promptus inverts each frame into an independent prompt without considering the correlation across frames. So Promptus needs to transmit prompts for each frame, resulting in the bitrate increasing rapidly as the frame rate rises. To address this, Promptus adds an interpolation-aware fitting strategy, ensuring that consecutive frames also exhibit continuity in the prompt space. With this, Promptus only needs to sparsely transmit prompts for a few keyframes, while the prompts for the remaining frames can be approximated through linear interpolation of the keyframe prompts.

We evaluated Promptus on test videos from different domains and under real-world network traces. The results show that: First, Promptus achieves a scalable bitrate, and its quality advantage over the baselines becomes more significant as the target bitrate decreases. Second, Promptus is general to different video domains, and the more complex the video content, the greater the quality advantage of Promptus compared to the baselines. Third, compared to H.265 (x265 2024), Promptus provides more than a 4x bandwidth saving while preserving the same perceptual quality. Fourth, at extremely low bitrates, Promptus can enhance the perceptual quality by 0.139 and 0.118 (in LPIPS (Zhang et al. 2018)) compared to VAE (Rombach et al. 2022) and H.265, respectively, and decreases the ratio of severely distorted frames by 89.3% and 91.7%.

2 Related Work and Background

2.1 Video Streaming

Video codecs. In internet video traffic, most of the content is encoded using traditional codecs represented by VP8 (Bankoski, Wilkins, and Xu 2011), VP9 (Mukherjee et al. 2015), H.265 and H.266 (Wieckowski et al. 2021). These traditional video codecs achieve compression primarily by eliminating redundancy. So the compression ratio has an upper limit due to the finite amount of redundancy. With the development of deep learning, handcrafted modules in traditional codecs are being replaced by neural networks (Lu et al. 2019; Lin et al. 2020; Djelouah et al. 2019; Rippel et al. 2019; Li, Li, and Lu 2024; Sheng et al. 2024). Compared to handcrafted modules, these embedded neural networks can achieve lower distortion at the same bitrate. However, they still adhere to the traditional coding framework (aiming to remove redundancy), so the improvement is limited.

Neural-enhanced streaming. In contrast to reducing redundancy, neural-enhanced streaming (Park et al. 2023; Zhou et al. 2022) actively discards non-redundant information, transmitting highly distorted low-bitrate videos. Then, at the receiver, neural network-based methods (such as super-resolution) are used to restore high-fidelity details. Thanks to the powerful restoration capabilities, the lost details can be effectively recovered, thus ensuring video quality while achieving substantial compression. However, these neural networks rely on prior knowledge learned from training datasets. Due to the complex and diverse nature of real-world video, there exist domain gaps. The performance often degrades in unseen scenarios (Yang et al. 2021). A feasible solution is to fine-tune the neural networks for each video and send the fine-tuned neural networks along with the video, like NAS (Yeo, Do, and Han 2017; Yeo et al. 2018, 2022, 2020; Kim et al. 2020; Zhang et al. 2022). However, the transmission of the neural networks inevitably increases the overall bitrate. Especially for shorter videos (like TikTok), the overhead of neural networks is unacceptable.

Generative streaming. Compared to restoration, generative streaming directly uses neural networks for generation. For example, in the context of video conferencing, Face-vid2vid (Wang, Mallya, and Liu 2021) extracts facial keypoints for streaming. At the receiver, it generates facial videos using the dynamic keypoints and a static facial image. The bitrate of keypoints is much lower than that of videos, thus achieving low bitrates. Gemino (Sivaraman et al. 2024) sends videos at a low resolution. At the receiver, it estimates facial motion fields from the low-resolution video, then uses the motion fields and a high-resolution facial image to generate high-resolution facial videos. Reparo (Li et al. 2023) proposes a token-based generative streaming. It first uses VQ-GAN (Esser, Rombach, and Ommer 2021) to train a codebook for facial visual features, then maps the video to latent variables using VAE (Kingma and Welling 2014), and finally quantizes the latent variables into token indices according to the codebook. Since only indices need to be sent, the bitrate is very low. In general, generative streaming can greatly compress videos, but it is often designed for specific tasks (such as video conferencing) and lacks generality.

2.2 Stable Diffusion (SD)

SD (Stability.ai 2024b) is a popular text-to-image generative model. It learns a denoising process from 5.85 billion image-text pairs, enabling it to generate high-quality images by denoising the pure noise images. To control the content of the generated images, SD also receives user-input prompts (in natural language) as the condition for denois-

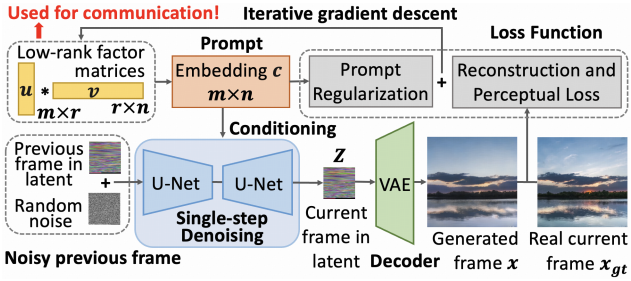


Figure 2: Workflow of our video to prompt inversion.

ing, thereby generating images that align with the semantic of the prompts. Specifically, let p represent the prompt. SD first performs word embedding and semantic extraction using the CLIP (Radford et al. 2021), converting discrete natural language p into a continuous text embedding c (an $m * n$ matrix). The text embeddings c and the randomly sampled noise image N are input into the denoising process, generating the denoised N'. Next, N' is input into the denoising again for T iterations to obtain the denoised Z. Since the denoising occurs in the latent space, Z needs to be input into the VAE Decoder to generate the image x in the pixel space.

However, SD cannot meet the fidelity of video streaming. SD only defines the generation from prompt to image, without the inversion from image to prompt. The most straightforward solution is to extract text descriptions using Captioning algorithms (Yang et al. 2023) and use these descriptions as prompts to generate images. Although the generated images can semantically align with the original images, these descriptions are too high-level, resulting in generated images having large structural differences, as shown in Figure 1. Besides text, ControlNet (Zhang, Rao, and Agrawala 2023) can also use images (such as masks and edges) as prompts, controlling SD to generate images that align with the contours of the image prompts. However, apart from the contours, details such as color and texture cannot be aligned (Qiao et al. 2024). In conclusion, the frames generated by SD cannot faithfully reproduce the original ones at the pixel level, making them unsuitable for video streaming.

3 Video to Prompt Inversion

3.1 Gradient Descent Based Prompt Fitting

Gradient Descent Framework. To obtain the inverse prompt, instead of training inverse priors (which is expensive and difficult to guarantee pixel alignment), we propose to fully leverage SD's built-in knowledge to infer the prompt. To this end, we adopt gradient descent to iteratively fit the prompt, with the framework shown in Figure 2. Specifically, at the beginning, the prompt is randomly initialized. SD generates a frame based on this prompt. Since the prompt is random, the generated frame is meaningless. Third, the pixel-wise difference between the generated frame and the target frame is calculated as the loss value. Fourth, backpropagation is used to compute the gradient of the loss with respect to the prompt. Finally, the prompt is updated using gradient descent. The above steps are iteratively executed until the loss is sufficiently small, and the resulting prompt can satisfy pixel-aligned generation. In the above steps, SD is pre-trained and frozen, so its prior knowledge is distilled into the prompt via gradient descent.

To realize this framework, there are five key components:
Single-step denoising to avoid unrolled long gradient chains. As described in §2.2, SD uses iterative denoising to generate images. So the prompt recursively influences the generated image. It causes the gradient of the loss with respect to the prompt to involve unrolling numerous iterations (e.g., 50 steps), where the accumulated long gradient chain introduces prohibitive computational and memory overhead. Thus, instead of using the traditional SD, we adopt SD Turbo 2.1 (Stability.ai 2024a), a variant that can generate images through single-step denoising. It allows gradient descent to require only a single forward-backward pass, greatly improving efficiency. Although the quality of the generated images is weaker than the traditional SD, these quality losses can be compensated for during the end-to-end fitting.
Employing embeddings as prompts to prevent non-differentiability. Computing the gradient through backpropagation requires the forward pass from the prompt to the loss to be completely differentiable. However, as described in §2.2, the forward pass includes the CLIP (Radford et al. 2021), which converts text from discrete natural language to continuous text embeddings. This conversion involves indexing and table lookup, making it non-differentiable. So gradients cannot be propagated to the natural language, preventing gradient descent. To address this, we discard the non-differentiable CLIP and directly use text embeddings as prompts for conditioning. In this case, the forward pass is fully differentiable. In the following sections, prompt refers to the text in embedding rather than in natural language[1].
Using a noisy previous frame instead of random noise to reduce latent space distance. The input noise can be viewed as a point in the latent space, and the denoising process actually moves this point under the control of the prompt. The goal of Promptus is to find the inverse prompt that can move the noise point to the point represented by the target image. Since we adopt a single-step denoising, if the input noise is far from the target image in the latent space, this movement cannot be completed in a single step, making it impossible to fit the inverse prompt. As shown in Figure 3(a), using random noise as input, after the loss value converges, the generated image still has noticeable differences from the target image, including blurring, artifacts, and inconsistent details. So we need to reduce the distance between the input noise and the target image. We observe that in a video, adjacent frames are close in the latent space. Thus, we manually add noise to the previous frame:

$$N^t = (1-\gamma) * Z^{t-1} + \gamma * N^0 \quad (1)$$

Where Z^{t-1} is the previous frame in the latent space. N^0 is a fixed noise. γ is a hyperparameter that controls the

[1]It is important to note that, even though it is not natural language, embeddings can still serve as prompts. In fact, prompts can take multiple modalities, such as images (Zhang, Rao, and Agrawala 2023), audio (Tang et al. 2024), embeddings (Ruiz et al. 2023; Kawar et al. 2023), fMRI (Chen, Qing, and Zhou 2024), etc.

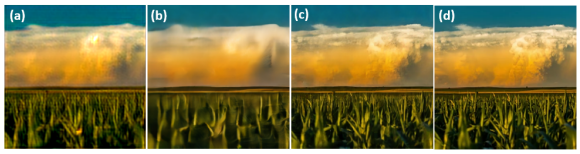


Figure 3: Visualization of fitting results. (a) Using random noise as the input. (b) Only using MSE as the loss. (c) Ours. (d) Ground Truth. It shows that the noisy previous frame and the perceptual loss both contribute to the visual quality.

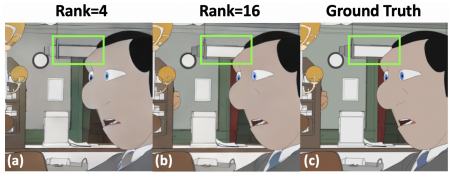


Figure 4: As the rank increases, the prompt fits more details. When the rank is 4, the lamp's color is inconsistent with GT. When the rank increases to 16, the lamp's color is corrected.

degree of noise addition, which we set to 0.95. N^t is the input noise for generating the current frame. As shown in Figure 3(c), compared to random noise, the noisy previous frame can reduce the distance in the latent space, resulting in a generated image that better matches the target image.

Combining reconstruction and perceptual loss functions. To achieve pixel-aligned supervision, the most intuitive loss function is the per-pixel reconstruction loss, such as MSE. These reconstruction losses attempt to minimize the overall pixel error, while errors in details often lead to large pixel errors. Therefore, to reduce the overall error, reconstruction losses tend to abandon the fitting of details, resulting in overly smooth images, as shown in Figure 3(b). To make the generated images sharp, one approach is to use perceptual loss instead of reconstruction loss, such as LPIPS (Zhang et al. 2018). However, perceptual loss aims to maximize the subjective quality without focusing on the exact consistency of each pixel, leading to misalignment between the generated and target images. To simultaneously ensure pixel alignment and subjective quality, we combine the reconstruction and perceptual loss as the fitting loss D:

$$D = \alpha * D_{rec}(x, x_{gt}) + (1 - \alpha) * D_{per}(x, x_{gt}) \quad (2)$$

Where D_{rec} represents the reconstruction loss, which is MSE by default. D_{per} represents the perceptual loss, which is LPIPS by default. α is a hyperparameter that balances pixel alignment and perceptual quality, which we set to 0.8. The result is shown in Figure 3(c). It can be observed that our generated image ensures pixel alignment while maintaining sharpness, making it identical to the ground truth.

Prompt Regularization. Even with real frames as supervision, fitting occasionally fails. This is because, as gradient descent progresses, the distribution of prompts shifts away from the distribution of CLIP embeddings. So we propose prompt regularization to constrain the prompt distribution:

$$\lambda = \|\bar{c} - \mu\|_2 \quad (3)$$

Here, μ is the mean of embeddings output by CLIP (which we obtained through statistics on a large amount of text, specifically $\mu = -0.168$). The final loss function L is:

$$L = \beta * D + (1 - \beta) * \lambda \quad (4)$$

Where β is a hyperparameter used to balance the fitting loss and the prompt regularization, which we set to 0.9.

3.2 Low-Rank Based Prompt Adaptive Bitrate

According to §3.1, each prompt is an $m*n$ embedding (e.g., $1024 * 77$), where each element is a floating-point number (e.g., 32-bit float), resulting in a fixed and high bitrate. To adaptively control the prompt bitrate, there are two directions: Dimensionality reduction decreases the number of parameters. Quantization reduces the bits for each parameter.

Low-rank decomposition. Instead of performing explicit dimensionality reduction, Promptus proposes to directly fit a low-rank prompt end-to-end, thereby reducing the quality degradation caused by prompt degradation. To achieve this, Promptus integrates the inverse process of CP decomposition (Chen et al. 2022) into gradient descent fitting. Specifically, Promptus derives the embedding c as follows:

$$c = (u * v)/\sqrt{r} \quad (5)$$

Where u and v are two low-rank factor matrices with dimensions $m * r$ and $r * n$. r is the rank of the embedding c. u and v compose the c through matrix multiplication and normalization. At this point, the embedding c, as an intermediate variable, is no longer fitted. u and v, as the new prompt representation, will be randomly initialized and fitted.

The rank r determines the trade-off between bitrate and quality. Compared to the embedding size of $m * n$ (e.g., $1024 * 77$), the total size of u and v is $(m+n) * r$. So reducing r significantly lowers the bitrate. However, on the other hand, when rank r is smaller, the embedding is constrained to be a low-rank matrix, resulting in weaker representational capability and inability to fit high-frequency details in the video. As shown in Figure 4, when the rank is 4, the lamp's color is inconsistent with the ground truth. When the rank increases to 16, the lamp's color is corrected. So it is necessary to adaptively select the rank r based on the current network bandwidth to trade off between bitrate and quality.

Fitting-aware quantization. We adopt differentiable fake quantization (Zafrir et al. 2019) and incorporate it into the fitting process, automatically compensating for the quantization loss through end-to-end gradient descent. We test the impact of quantization on quality. The results indicate our method reduces the bits from 32 to 8 without quality loss.

3.3 Interpolation-Aware Fitting Based Prompt Inter-Frame Compression

According to §3.1, Promptus fits each frame as an independent prompt, without considering the correlation across frames. Therefore, the bitrate of Promptus increases rapidly as the frame rate rises. However, codec-based streaming can avoid this problem through inter-frame compression. Is it possible to perform inter-frame compression on prompts as well? Our insight is: prompts are high-level semantics, so

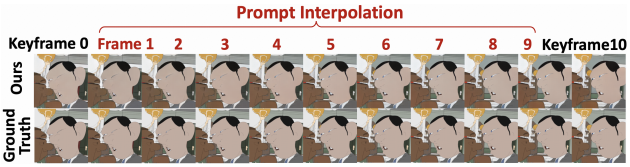


Figure 5: The prompt interpolation not only fully preserves the video details but also successfully fits the motion.

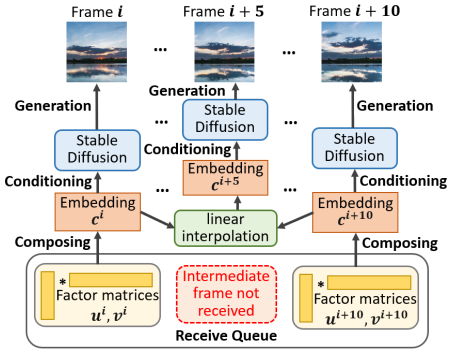


Figure 6: Promptus approximates the unreceived prompts by linear interpolation. Thus, instead of sending the prompt for each frame, only the prompts for keyframes need to be transmitted sparsely, thereby achieving inter-frame compression.

the prompts of continuous video frames can change continuously. With this, during streaming, we only need to sparsely transmit the prompts of a few frames (as keyframes), and the prompts of the remaining frames can be obtained by linear interpolation of the keyframe prompts, as illustrated in Figure 6. Since keyframes account for only a small portion of the total frames, inter-frame compression is achieved.

To ensure that interpolated prompts generate meaningful frames, we integrate the interpolation into the gradient descent fitting. Specifically, we randomly initialize prompts for keyframes, interpolate prompts for each intermediate frame, and generate frames. At this point, the interpolated prompts are intermediate variables, so they are not fitted; instead, gradients are backpropagated to keyframe prompts for fitting.

Prompt interpolation works. As shown in Figure 5, the intermediate frames not only fully preserve the video details but also successfully approximate the motion. In practice, **scene changes** often occur within the video, at which point prompt interpolation no longer works. So we will detect scene changes and treat the new scenes as new videos. *For more system details, please refer to the extended version.*

4 Evaluation

4.1 Experiment Setup

Test videos: To validate the generality across domains, we test Promptus on four video datasets with vastly different genres: QST (Zhang et al. 2020), UVG (Mercat, Viitanen, and Vanne 2020), GTA-IM (Cao et al. 2020) and Animerun (Siyao et al. 2022). The domains of these videos span natural landscapes and human activities, outdoor and indoor scenes, real-world and CG-synthesized scenes, 3D video games and 2D animations. For consistency, videos are cropped and resized to a resolution of 512*512, with a frame rate of 30 FPS. *Note that Promptus supports flexible resolutions and we test **other resolutions** in the extended version.*

Baselines: We compare Promptus with six baselines: **H.265** and **H.266** (Wieckowski et al. 2021) are advanced traditional codecs that achieve compression by utilizing hand-designed modules, while **DCVC** (Li, Li, and Lu 2021) is a neural codec. **VAE** (Rombach et al. 2022) is an autoencoder. Its encoder maps the input image to low-dimensional latent variables, while the decoder reconstructs the image from the latent variables. For inter-frame compression, we encode the latent variables into a video using H.265. **NAS** (Yeo et al. 2022) represents neural-enhanced streaming, which trains overfitted super-resolution DNNs for each video. These DNNs are transmitted along with low-resolution videos. At the receiver, DNNs are employed to enhance the video quality. To control the total bitrate, we adjust the bitrate of the low-resolution video and the number of DNN parameters. **VQGAN** (Li et al. 2023) stands for generative streaming. To compress the video, it first divides the video into patches, maps each patch to a latent variable, and then quantizes each latent variable into a token index using a codebook. At the receiver, the token indices can be reconstructed into a video. For bitrate control, we adjust the number of patches (tokens).

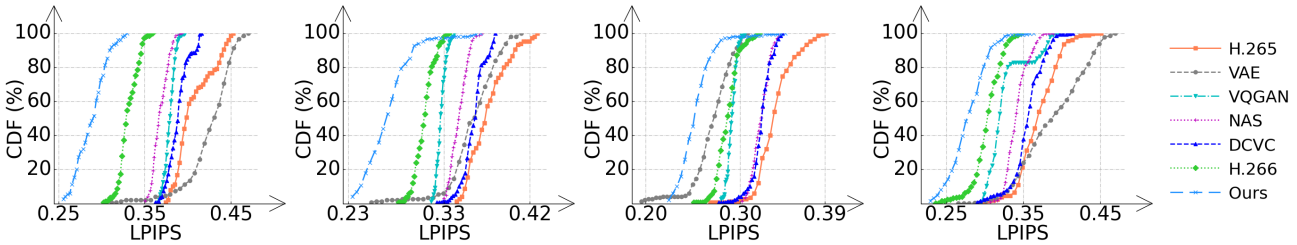
Metrics: To evaluate video quality, we mainly adopt the LPIPS (Zhang et al. 2018), because Promptus is more oriented towards the semantic communication task rather than video compression. Our primary goal is to keep the semantics correct at low bitrates. LPIPS better reflects human subjective perception (Yu et al. 2023), making it more suitable for evaluating semantic distortion. Smaller LPIPS indicates higher quality. *We also evaluate traditional **PSNR** and **SSIM** as objective metrics, provided in the extended version.*

Real-world network traces: We collect traces from real-world scenarios such as subways, aiming to test weak network conditions such as poor signal coverage, network overload, and frequent switching of mobile wireless networks. We use Mahimahi (Netravali et al. 2015) to replay traces.

4.2 Transmission Efficiency

This section proves the transmission efficiency. Figure 7 shows the CDF of the frame quality under 3 bitrate levels and real-world network traces. We also calculate the mean frame quality for each method and bitrate setting in Table 1.

First, Promptus achieves better transmission efficiency across all bitrate settings. For example, in Figure 7, the curves of Promptus are all to the left of the baseline curves. Second, Promptus can significantly reduce the ratio of severely distorted frames. For instance, under real-world network conditions, only 5.2% of frames in Promptus have LPIPS higher than 0.32, while NAS, H.265, VAE, DCVC and VQGAN have 98.2%, 96.9%, 94.5%, 93.0% and 53.2%, respectively. Third, Promptus can achieve more than a 4x bandwidth reduction while preserving the same perceptual quality. For example, in Table 1, the average LPIPS of Promptus under 140 kbps is better than H.265 under 540 kbps. Fourth, the lower the bitrate, the greater the advantage



(a) 140 kbps (b) 280 kbps (c) 540 kbps (d) Real-world traces

Figure 7: Frame quality CDF at three target bitrates and real-world network traces. It indicates Promptus achieves better transmission efficiency across all bitrate settings. The lower the bitrate, the greater the advantage of Promptus. Detail in §4.2.

Bitrate	140k	280k	360k	540k	Real
H.265	0.407	0.372	0.360	0.339	0.372
VAE	0.428	0.358	0.326	0.272	0.391
DCVC	0.389	0.357	0.345	0.324	0.357
VQGAN	0.379	0.322	0.306	0.292	0.328
NAS	0.369	0.341	0.335	0.323	0.343
H.266	0.332	0.305	0.298	0.287	0.305
Ours	**0.289**	**0.268**	**0.264**	**0.255**	**0.280**

Table 1: Mean frame quality (LPIPS ↓). Detail in §4.2.

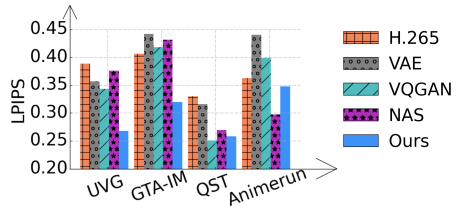


Figure 8: Mean frame quality on 4 datasets. It proves the generality of Promptus across domains. Detail in §4.3.

of Promptus. For example, when the bitrate is 540 kbps, the average LPIPS of Promptus is 0.017, 0.084, 0.032 and 0.037 lower than VAE, H.265, H.266 and VQGAN. When the bitrate is 140 kbps, this difference further increases to 0.139, 0.118, 0.043 and 0.090. These improvements are partly due to Promptus's transmission efficiency, enabling it to provide higher perceptual quality at the same bitrate. Additionally, since Promptus sends prompts without entropy coding, it can precisely control the target bitrate, thus making full use of the real-world network bandwidth. *More details on **bitrate control** and **parameter settings** are in the extended version.*

4.3 Generality and Corner Cases

This section proves the generality across domains. Figure 8 shows the average frame quality on four datasets.

First, Promptus maintains good transmission efficiency on different domains. As shown in Figure 8, Promptus achieves lower LPIPS compared to most baselines on each dataset. Second, the more high-frequency details a video has, the greater the advantage of Promptus. For example, for the Animerun with fewer details, the LPIPS of Promptus is 0.015 lower than H.265. For the detail-rich UVG, this advantage further expands to 0.121. To intuitively demonstrate this gain, we visualize the frames of each method on each dataset at low bitrates (such as 225 kbps) in Figure 9. It can be observed that, compared to the baselines which exhibit blurriness and blocking artifacts, Promptus preserves more high-frequency details, resulting in higher perceptual quality.

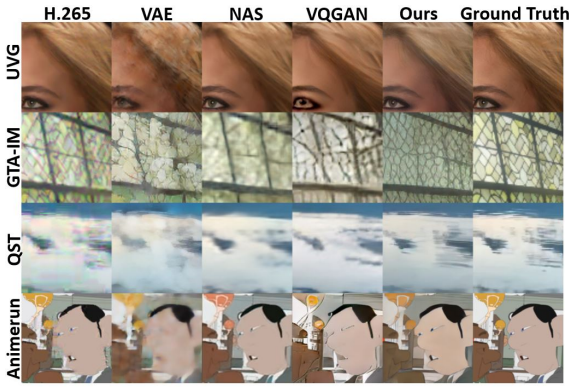


Figure 9: Visualization of the results on different datasets. It can be observed that, compared to the baselines which exhibit blurriness and blocking artifacts caused by compression, Promptus preserves more high-frequency details.

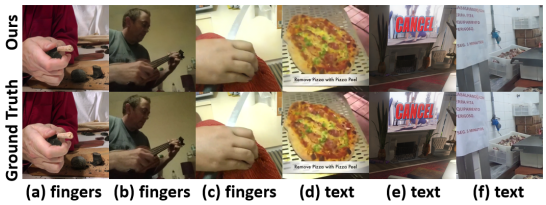


Figure 10: Fitting results for challenging examples, with a bitrate of 225 kbps. Promptus is able to fit them well (§4.3).

Since SD itself struggles to generate specific elements, such as fingers or text. To further evaluate the generality of Promptus, we also show its fitting results for these challenging examples in Figure 10. The results show that Promptus can fit them well. This is because the end-to-end gradient descent can compensate for the limitations of SD itself.

Figure 11: To evaluate the temporal consistency, we visualize the videos as space-time X-t slices (Li et al. 2024). The results indicate that our videos are temporally aligned with the ground truth videos in terms of motion, and exhibit almost no flickering phenomena. More details are in §4.4.

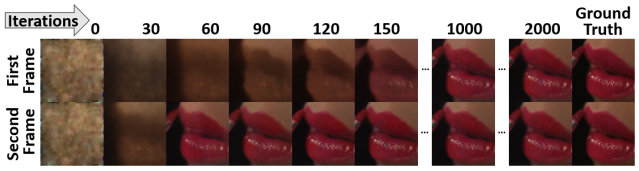


Figure 12: Visualization of fitting results at different iterations. The second frame requires only 120 iterations to achieve a visual quality comparable to that of 2000 iterations for the first frame. More details are discussed in §4.5.

4.4 Temporal Consistency and Flickering

This section evaluates the temporal consistency of the generated videos. As shown in Figure 11, we visualize the entire videos as space-time X-t slices (Li et al. 2024), with three scanning lines at different heights marked on the left. Both of our method and H.265 are at a bitrate of 225 kbps.

Results demonstrate that although Prompus generates video frame by frame, it maintains good temporal consistency. First, it can be seen that our videos are temporally aligned with the ground truth videos in terms of motion. Second, our videos exhibit almost no flickering phenomena. For example, pixels in the X-t slices mostly move smoothly along the t dimension. This is because the proposed interpolation-aware fitting (§3.3) allows each frame to be as similar as possible to the ground truth, and video deflickering (Lei et al. 2023) is also used to further alleviate inconsistency between frames. Third, compared to H.265, our method reduces visual degradation caused by blocking artifacts. As shown in the red box of Figure 11, H.265 exhibits some zig-zag phenomena in X-t slices, while our method is free from these artifacts. ***We provide some demo videos in the open-source repository to showcase the quality.***

4.5 Speed and Overhead

We conduct tests on an NVIDIA 4090D GPU. The resolution of the generated video is 512*512, and the rank of the prompt is 8. The SD model has a total of 867M parameters.

Generation overhead. Table 2 shows the fine-grained overhead of each step in Promptus's frame generation. Specifically, from receiving a prompt to generating a frame, Promptus includes the following steps: prompt dequantization, prompt composition, prompt interpolation, adding noise to the previous frame, and SD generation. Among them, most steps only involve simple linear computations, so the time overhead is almost negligible. In contrast, SD generation accounts for the vast majority of the time overhead. Therefore, following StreamDiffusion (Kodaira et al. 2023), we use TAESD and TensorRT to accelerate the SD to 6.16 ms. In summary, the total time overhead of Promptus's frame generation at the receiver is 6.226 ms, achieving **160 FPS**. *The **real-time engines are provided** in the open-source repository.* Additionally, the total memory usage is 7846 MB.

Steps	Time (ms)
Dequantization	0.016
Prompt composition	0.025
Prompt interpolation	0.013
Noised frame	0.012
SD generation	6.160
Total	6.226

Table 2: Time overhead of generating a frame §4.5.

Inversion overhead. The time overhead of inversion depends on two factors: the number of iterations needed for convergence and the time taken for each iteration. Among them, the time taken for each iteration is 150 ms. For the number of iterations, there is a significant difference between different frames. For the first frame of a new video (or new scene), the inversion takes thousands of iterations to converge since it starts from scratch. For subsequent frames, thanks to the continuous prompt space (§3.3) providing sufficient scene priors, the inversion can converge in only a few hundred iterations. We present an example in Figure 12. It shows that the second frame requires only 120 iterations to achieve a quality comparable to that of 2000 iterations for the first frame. In practice, we set the number of iterations for the first frame to 10,000, and for subsequent frames to 500. For memory usage, inversion occupies 10 GB in total.

5 Conclusion and Discussion

This paper proposes Promptus, a real system that replaces video streaming with prompt streaming by representing video frames as prompts. Evaluations across various video domains and real network traces demonstrate that it achieves a 4x bandwidth saving while preserving the same perceptual quality. *We also conduct **ablation studies** to prove the contribution of each module, as detailed in the extended version.*

Promptus extends the boundaries of AIGC to video streaming, offering a semantic communication paradigm. As an initial effort, there are some limitations, such as the time overhead of inversion and the latency of interpolation (*details are in the extended version*). These limitations restrict Promptus to on-demand videos, preventing its use in live videos. So further improving the efficiency is future work.

Acknowledgments

We sincerely thank the anonymous reviewers for their valuable feedback. This work is sponsored by the National Natural Science Foundation of China (62431017). We gratefully acknowledge the support of Key Laboratory of Intelligent Press Media Technology. Xinggong Zhang is the corresponding author (zhangxg@pku.edu.cn).

References

Bankoski, J.; Wilkins, P.; and Xu, Y. 2011. Technical overview of VP8, an open source video codec for the web. In *2011 IEEE International Conference on Multimedia and Expo*, 1–6. IEEE.

Cao, Z.; Gao, H.; Mangalam, K.; Cai, Q.-Z.; Vo, M.; and Malik, J. 2020. Long-term human motion prediction with scene context. In *European Conference on Computer Vision*, 387–404. Springer.

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, 333–350. Springer.

Chen, Z.; Qing, J.; and Zhou, J. H. 2024. Cinematic mindscapes: High-quality video reconstruction from brain activity. *Advances in Neural Information Processing Systems*, 36.

Djelouah, A.; Campos, J.; Schaub-Meyer, S.; and Schroers, C. 2019. Neural inter-frame compression for video coding. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6421–6429.

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

Kawar, B.; Zada, S.; Lang, O.; Tov, O.; Chang, H.; Dekel, T.; Mosseri, I.; and Irani, M. 2023. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6007–6017.

Kim, J.; Jung, Y.; Yeo, H.; Ye, J.; and Han, D. 2020. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 107–125.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. *Journal of Machine Learning Research (JMLR)*, 15(1): 1929–1958.

Kodaira, A.; Xu, C.; Hazama, T.; Yoshimoto, T.; Ohno, K.; Mitsuhori, S.; Sugano, S.; Cho, H.; Liu, Z.; and Keutzer, K. 2023. StreamDiffusion: A Pipeline-level Solution for Real-time Interactive Generation. *arXiv preprint arXiv:2312.12491*.

Lei, C.; Ren, X.; Zhang, Z.; and Chen, Q. 2023. Blind video deflickering by neural filtering with a flawed atlas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10439–10448.

Li, J.; Li, B.; and Lu, Y. 2021. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34: 18114–18125.

Li, J.; Li, B.; and Lu, Y. 2024. Neural video compression with feature modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 26099–26108.

Li, T.; Sivaraman, V.; Fan, L.; Alizadeh, M.; and Katabi, D. 2023. Reparo: Loss-resilient generative codec for video conferencing. *arXiv preprint arXiv:2305.14135*.

Li, Z.; Tucker, R.; Snavely, N.; and Holynski, A. 2024. Generative image dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24142–24153.

Lin, J.; Liu, D.; Li, H.; and Wu, F. 2020. M-LVC: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3546–3554.

Lu, G.; Ouyang, W.; Xu, D.; Zhang, X.; Cai, C.; and Gao, Z. 2019. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11006–11015.

Mercat, A.; Viitanen, M.; and Vanne, J. 2020. UVG dataset: 50/120fps 4K sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, 297–302.

Mukherjee, D.; Han, J.; Bankoski, J.; Bultje, R.; Grange, A.; Koleszar, J.; Wilkins, P.; and Xu, Y. 2015. A technical overview of vp9—the latest open-source video codec. *SMPTE Motion Imaging Journal*, 124(1): 44–54.

Netravali, R.; Sivaraman, A.; Das, S.; Goyal, A.; Winstein, K.; Mickens, J.; and Balakrishnan, H. 2015. Mahimahi: accurate {Record-and-Replay} for {HTTP}. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 417–429.

Park, S.; Cho, Y.; Jun, H.; Lee, J.; and Cha, H. 2023. Omnilive: Super-resolution enhanced 360 video live streaming for mobile devices. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, 261–274.

Qiao, L.; Mashhadi, M. B.; Gao, Z.; Foh, C. H.; Xiao, P.; and Bennis, M. 2024. Latency-aware generative semantic communications with pre-trained diffusion models. *IEEE wireless communications letters*.

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

Rippel, O.; Nair, S.; Lew, C.; Branson, S.; Anderson, A. G.; and Bourdev, L. 2019. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3454–3463.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Ruiz, N.; Li, Y.; Jampani, V.; Pritch, Y.; Rubinstein, M.; and Aberman, K. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22500–22510.

Sheng, X.; Li, L.; Liu, D.; and Li, H. 2024. Prediction and Reference Quality Adaptation for Learned Video Compression. *arXiv preprint arXiv:2406.14118*.

Sivaraman, V.; Karimi, P.; Venkatapathy, V.; Khani, M.; Fouladi, S.; Alizadeh, M.; Durand, F.; and Sze, V. 2024. Gemino: Practical and robust neural compression for video conferencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 569–590.

Siyao, L.; Li, Y.; Li, B.; Dong, C.; Liu, Z.; and Loy, C. C. 2022. Animerun: 2d animation visual correspondence from open source 3d movies. *Advances in Neural Information Processing Systems*, 35: 18996–19007.

Stability.ai. 2024a. SD-Turbo. https://huggingface.co/stabilityai/sd-turbo [Accessed: December 1, 2025].

Stability.ai. 2024b. Stable Diffusion. https://stability.ai/ [Accessed: December 1, 2025].

Tang, Z.; Yang, Z.; Zhu, C.; Zeng, M.; and Bansal, M. 2024. Any-to-any generation via composable diffusion. *Advances in Neural Information Processing Systems*, 36.

Wang, T.-C.; Mallya, A.; and Liu, M.-Y. 2021. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10039–10049.

Wieckowski, A.; Brandenburg, J.; Hinz, T.; Bartnik, C.; George, V.; Hege, G.; Helmrich, C.; Henkel, A.; Lehmann, C.; Stoffers, C.; Zupancic, I.; Bross, B.; and Marpe, D. 2021. VVenC: An Open And Optimized VVC Encoder Implementation. In *Proc. IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 1–2.

x265. 2024. H.265. https://www.itu.int/rec/T-REC-H.265 [Accessed: December 1, 2025].

Yang, A.; Nagrani, A.; Seo, P. H.; Miech, A.; Pont-Tuset, J.; Laptev, I.; Sivic, J.; and Schmid, C. 2023. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10714–10726.

Yang, X.; Xiang, W.; Zeng, H.; and Zhang, L. 2021. Real-world video super-resolution: A benchmark dataset and a decomposition based learning scheme. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4781–4790.

Yeo, H.; Chong, C. J.; Jung, Y.; Ye, J.; and Han, D. 2020. Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 1–14.

Yeo, H.; Do, S.; and Han, D. 2017. How will deep learning change internet video delivery? In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 57–64.

Yeo, H.; Jung, Y.; Kim, J.; Shin, J.; and Han, D. 2018. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 645–661.

Yeo, H.; Lim, H.; Kim, J.; Jung, Y.; Ye, J.; and Han, D. 2022. Neuroscaler: Neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM 2022 Conference*, 795–811.

Yu, L.; Lezama, J.; Gundavarapu, N. B.; Versari, L.; Sohn, K.; Minnen, D.; Cheng, Y.; Gupta, A.; Gu, X.; Hauptmann, A. G.; et al. 2023. Language Model Beats Diffusion–Tokenizer is Key to Visual Generation. *arXiv preprint arXiv:2310.05737*.

Zafrir, O.; Boudoukh, G.; Izsak, P.; and Wasserblat, M. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, 36–39. IEEE.

Zhang, A.; Wang, C.; Han, B.; and Qian, F. 2022. {YuZu}:{Neural-Enhanced} volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 137–154.

Zhang, J.; Xu, C.; Liu, L.; Wang, M.; Wu, X.; Liu, Y.; and Jiang, Y. 2020. Dtvnet: Dynamic time-lapse video generation via single still image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, 300–315. Springer.

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

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

Zhou, Q.; Li, R.; Guo, S.; Dong, P.; Liu, Y.; Guo, J.; and Xu, Z. 2022. Cadm: Codec-aware diffusion modeling for neural-enhanced video streaming. *arXiv preprint arXiv:2211.08428*.