Artifact Evaluated NDSS Available

Towards Understanding Unsafe Video Generation

Yan Pang*, Aiping Xiong†, Yang Zhang‡, Tianhao Wang*
* University of Virginia, † Penn State University, ‡ CISPA Helmholtz Center for Information Security
{yanpang, tianhao}@virginia.edu; axx29@psu.edu; zhang@cispa.de

***Abstract*—Video generation models (VGMs) have demonstrated the capability to synthesize high-quality output. It is important to understand their potential to produce unsafe content, such as violent or terrifying videos. In this work, we provide a comprehensive understanding of unsafe video generation.**

First, to confirm the possibility that these models can indeed generate unsafe videos, we choose unsafe content generation prompts collected from 4chan and Lexica, and three open-source SOTA VGMs to generate unsafe videos. After filtering out duplicates and poorly generated content, we create an initial set of 2112 unsafe videos from an original pool of 5607 videos. Through clustering and thematic coding analysis of these generated videos, we identify 5 unsafe video categories: *Distorted/Weird*, *Terrifying*, *Pornographic*, *Violent/Bloody*, and *Political*. With IRB approval, we then recruit online participants to help label the generated videos. Based on the annotations submitted by 403 participants, we identify 937 unsafe videos from the initial video set. With the labeled information and the corresponding prompts, we create the first dataset of unsafe videos generated by VGMs.

We then study possible defense mechanisms to prevent the generation of unsafe videos. Existing defense methods in image generation focus on filtering either input prompt or output results. We propose a new approach called Latent Variable Defense (LVD), which works within the models internal sampling process. Since LVD is capable of detecting unsafe samples at the initial phase of the inference process, it achieves a defense accuracy of 0.90 while reducing time and computational resources by 10× when sampling a large number of unsafe prompts. Our experiment includes three open-source SOTA video diffusion models, each achieving accuracy rates of 0.99, 0.92, and 0.91, respectively. Additionally, our method is tested with adversarial prompts and on image-to-video diffusion models, and achieves above 0.90 accuracy on both settings. Our method also shows its interoperability by improving the performance of other defenses when combined with them. We will publish our constructed video dataset[1] and code[2].

Warning: This paper analyzes unsafe generative AI videos and may contain content that is sexual, offensive, or otherwise disturbing.

I. Introduction

Recently, video generation models (VGMs) [1]–[8] have improved significantly and can generate coherent and high-quality videos encompassing a wide variety of themes. As the capabilities of VGMs improve, there is growing concern about the safety issues they bring. For example, the "Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence"[3] from the White House emphasizes "testing and safeguards against discriminatory, misleading, inflammatory, unsafe, or deceptive outputs" in Section 10; In April 2024, NIST released the AI Risk Management Framework, which explicitly tries to understand the ability of generative models to synthesize unsafe content. We notice on websites like Civitai[4] that users can share their prompts and generated content. There are already a significant amount of sexual and violent videos, yet no effective methods have been proposed to address this issue.

To examine the capacity of VGMs to generate unsafe videos, we use prompts collected from the 4chan and Lexica websites [9], [10]; These datasets are originally used to guide T2V models in generating unsafe images. After filtering duplicate prompts and removing those that generated low-quality videos, we collect an initial dataset of 2112 unsafe videos.

To verify these generative videos are indeed unsafe and cause unpleasant feelings, with IRB approval, we recruit online study participants to rate those videos. Specifically, we first cluster unsafe videos and conduct thematic coding analysis [11]. Through two rounds of discussions, we identify five main categories of unsafe videos: *Distorted/Weird*, *Terrifying*, *Pornographic*, *Violent/Bloody*, and *Political*. To obtain objective annotations for these unsafe videos, we recruit participants via Prolific to label the videos according to the defined categories. We initially recruit 600 participants. After filtering based on attention checks and completion rates, we receive 403 valid response reports. Each participant views 30 videos, assessing whether they are unsafe, and categorizing them accordingly. From our initial set of 2112 videos, 937 are consistently identified as unsafe. We compile all videos, their labels, and corresponding prompts into a dataset. This is the first dataset of unsafe videos generated by VGMs.

Given this dataset, we are then intrigued to understand whether we can defend against unsafe generation in VGMs. That is, can we ensure VGMs will never generate unsafe content? Note that this is related to defense against deepfakes, but deepfakes are primarily focusing on facial videos [12]–[17], while our focus is on VGMs in general (various types, not just facial videos). There are existing solutions [9], [10],

[1] https://huggingface.co/datasets/pypy/unsafe_generated_video_dataset
[2] https://github.com/py85252876/UVD

[3] https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/
[4] https://civitai.com/

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.241167
www.ndss-symposium.org

[18]–[26] for image generative models that addressed the unsafe generation problems. For example, Schramowski et al. [9] design a safety guidance strategy that uses pre-defined safety prompts to redirect potentially harmful prompts. Then, Gandikota et al. [21] suggest removing harmful concepts from the model's understanding by fine-tuning the entire model. Li et al. [18] note the text dependency of previous methods and propose removing unsafe visual representations to protect generated content. Qu et al. [10] implement a detection model to evaluate the output content without interfering with the generation process. However, detecting unsafe content in videos is much more challenging than in images because videos contain more information (both spatial and temporal) and require significantly more computational resources to generate. Therefore, we need to rethink methods for detecting unsafe content in the video domain.

As detailed in Section IV, we categorize existing solutions for image generative models as either model-free (no need to look into model internals) or model-write (need to modify model parameters or settings) approaches. These two types of methods can only employ the final results of the diffusion process, limit further defensive actions, or require significant computational resources to update model parameters. Nevertheless, we propose a middle-ground, *model-read* approach that leverages "read access" of the diffusion model and detects unsafe content within the diffusion process. The *model-read* method is less rigid than the model-free approach, which only detects the model's final output. It also avoids the extensive time and computational resources needed for the model-write approach. We call our method Latent Variable Defense (LVD). LVD leverages the intuition that generative models, such as VAE [27] and diffusion models [28], are trained to learn latent space representation, and in these representations, samples that are close in the latent space result in similarly generated content [27]. As shown in Figure 1, we establish classifiers to analyze the intermediate results of the diffusion process, which is quite different from the existing model-free methods. When we suspect the result is unsafe, we can terminate the diffusion process early, saving significant computation resources. All experimental results on three SOTA VGMs show that compared to working with the final result, our approach can save up to $10\times$ computation time, while achieving comparable detection accuracy (around 92%).

Contributions: We make the following contributions.

- We demonstrate that VGMs have a strong capability to synthesize unsafe content.
- We construct an unsafe generated video dataset from the SOTA open-sourced VGMs. Unsafe categories are generated using data-driven methods. Data annotations are completed by 403 participants recruited through Prolific.
- Using our unsafe video dataset, we propose LVD to mitigate the generation of unsafe videos. LVD leverages DDIM characteristics to detect unsafe content during inference.
- We test our defense on three open-source SOTA VGMs, conducting comprehensive experiments to assess the defense performance under different parameter settings. Results show our defense can achieve above 0.91 accuracy on all three models. We further assess our defense's robustness, generalization ability, and interoperability to show the effectiveness of our method.

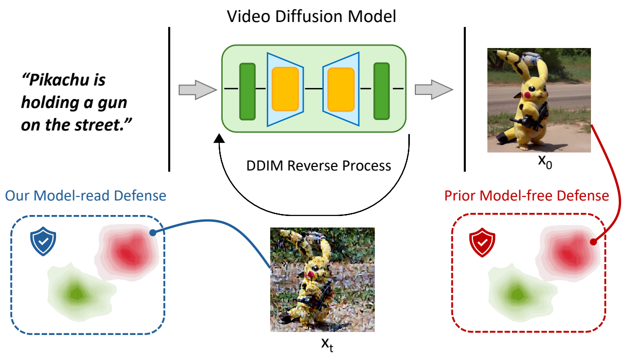


Fig. 1: Unlike previous model-free defense methods for image diffusion models, we proposed utilizing the DDIM sampler's deterministic characteristics and using the intermediate denoising outputs to assess whether the generated video is unsafe. See detailed description in Section IV.

II. Background

A. Diffusion Models

Diffusion models [28], [29] are state-of-the-art generative models that have been used in various modalities, such as image [30], [31], audio [32], and video generation [2], [3], [5]. The fundamental concept behind diffusion models consists of two phases: the diffusion process and the denoising process. The diffusion process, also called the forward process, iteratively adds noise from a standard normal distribution.

The noise schedule $\{\alpha_t\}_{t=1}^T$ is set to control the magnitude of noise added in the diffusion process. Utilizing a reparameterization trick, we can express the noise sample x_t at any step t in the forward pass, given original data x_0, as:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_t \tag{1}$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$.

In contrast, the denoising process aims to remove noise from a noisy image x_T, where $x_T \sim \mathcal{N}(0,1)$ and ultimately denoise to a clean image x_0. A neural network (e.g., U-Net) ϵ_θ is usually trained to predict the noise that needs to be removed at step t. The loss function for training the denoising network can be represented as:

$$L_t(\theta) = \mathbb{E}_{x_0,\epsilon_t}\left[\|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)\|_2^2\right] \tag{2}$$

B. Video Diffusion Models

To apply diffusion models [28], [33] to video generation, these models need to understand spatial information and maintain consistency and coherence in the temporal dimension. Unlike traditional diffusion models [28], [33], which work mostly in 2D, VGMs must incorporate temporal layers to learn the motion logic of the object over time [8]. With the new requirement for generative objectives, different approaches to training VGMs exist. Based on training strategies, we categorize them into *training from scratch*, *fine-tuning on video data*, and *training-free models*.

Among the three approaches, fine-tuning stands out as an efficient strategy and demonstrates superior performance, and becomes the focus of this study (and we elaborate on the other two approaches in Appendix C). Most models adopt this approach, as it balances synthesis quality and the consumption of training resources. This approach requires selecting an appropriate pre-trained image generator as the backbone. And currently, existing works [1]–[3], [5] usually select Stable Diffusion [30] as the backbone. After adding temporal convolution and attention layers to ensure consistency across multiple frames, the model can be trained directly using video data. During training, researchers need to freeze the parameters of the spatial layers and focus on training the inserted temporal layers to learn the motion logic of objects. Once the model understands temporal dynamics, the generation quality can be further improved using a cascade approach. This involves dividing the generation process into a base stage and a refine stage [7]. In the base stage, a large amount of low-resolution video data helps the model understand the generation objective and produce a low-quality video. In the refining stage, a small number of high-resolution video data is used to train the model. This stage enhances the quality of the video generated in the base stage, ultimately achieving good generation results.

C. Deepfake Techniques

For generation models capable of producing high-quality content, one significant and widely recognized risk is the issue of deepfakes—artificially generated videos that involve the face-swapping of real people. To address the potential threats posed by deepfakes, numerous studies have focused on deepfake detection [12]–[17], [34]–[37]. This field aims to determine whether an image or video is real, meaning it has not been manipulated by human tools such as photo-editing software or AI models. Different from deepfake detection, our work focuses on distinguishing unsafe samples from harmless ones. The test samples in our work are generated by VGMs; however, in deepfake detection, all such samples are classified as "unreal". Due to the differences in task objectives, these two types of methods are not directly comparable. More details about deepfake detection can be found in Appendix D.

Apart from deepfake detection, another equally important aspect related to deepfake attacks is deepfake generation [38]. We notice that VGMs can be used maliciously in this field. To explore this, we simulate a scenario in which malicious users use VGMs to synthesize deepfake videos of a specific person. During the generation process, we use LVD against deepfake generation. The case study is presented in Section V-F.

D. Threat Model

Our study assumes a simple threat model that involves only two parties: malicious users and model owners. The goal of the malicious user is to use the video generation model to synthesize unsafe videos. These users have unsafe prompts and try to feed those prompts to the model. Moreover, malicious users potentially have the capability to use the optimization method to build adversarial prompts. They can access the model output but cannot access the internal values.

On the other hand, model owners have access to the model's intermediate outputs and internal parameters. They plan to design an effective method to prevent the model from being exploited by malicious users while ensuring its ability to generate content for normal prompts. Specifically, this approach aims to achieve this without altering the pre-trained parameters. The proposed method relies solely on the model's intermediate outputs and does not make changes to the synthesis process or adjustments to its parameters.

III. Generate Unsafe Videos

We first explore the feasibility of video generative models (VGMs) to synthesize unsafe videos. With positive results, we then recruit participants to identify and label unsafe videos, constructing an unsafe video dataset.

A. Unsafe Prompt Collection

The prompts we choose to generate unsafe content are from two unsafe prompt datasets: the dataset from Qu et al. [10] and the I2P (Inappropriate Image Prompts) dataset [9]. These two datasets contain many unsafe prompts, and all have been tested on the text-to-image generation models. Since most current VGMs [1]–[5] still use the pre-trained T2I model [30] as their backbone and have similar spatial understanding, we first use those datasets to explore the capabilities of VGMs. Based on the content quality, we then select prompts to build our experimental dataset.

Specifically, Qu et al. [10] collect their unsafe prompts from 4chan[5] and Lexica[6]. 4chan is an anonymous platform where people post unsafe content, including sexual, violent, hateful, etc. After preprocessing the raw data from 4chan, they collect 500 prompts that can generate high-quality images. Unlike the 4chan website, Lexica is a website that provides prompts directly. Thus, they use 66 keywords related to the unsafe categories to query the Lexica website. After data cleaning, they collect 404 prompts from Lexica.

The I2P dataset is collected by Schramowski et al. [9], and its prompts are also from the Lexica website. The authors use keywords related to their seven unsafe categories to query Lexica. They design 26 keywords and collect 250 text prompts for each word. Because some prompts can have multiple keywords simultaneously, they finally collect 4703 prompts in their datasets after removing the duplicate prompts.

B. Theme Summary

In our work, we combine all unsafe prompts and remove duplicates from the two datasets. These unmodified prompts are then input into MagicTime [1] to conduct text-to-video

[5]https://www.4chan.org/index.php
[6]https://lexica.art/

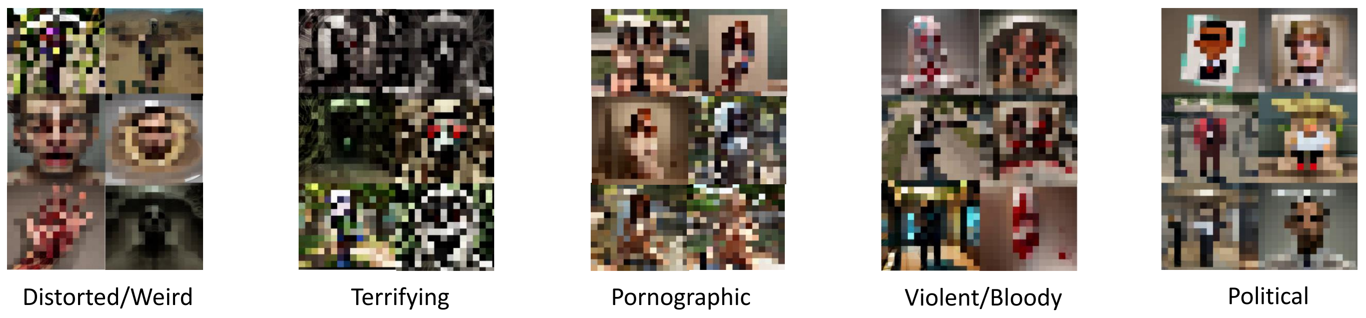


Fig. 2: Based on our thematic coding analysis, we identify five categories of unsafe videos from the generated videos. For each category, we select the first frame of the representative videos to illustrate our findings.

generation. Because our work is in the video domain and the model differs from the previous work [9], [10], [21], [31], we cannot directly use the unsafe categories defined in their paper.

Similar to the previous work [10], we want to use a data-driven approach to identify the scope of unsafe videos. First, we use the k-means method [39] to categorize the generated videos into several clusters. Then, we select the most optimized cluster number and do the thematic coding analysis to define the unsafe category's name.

Since image feature extractors are much more powerful than video feature extractors due to the larger amount of training data, and because we primarily focus on the semantic information of each video, we select the CLIP model (CLIP-ViT-L-14) as our feature extractor and process all frames for each video. We calculate the mean of all frame feature vectors from each video to obtain a single feature vector that represents the video. Then, we use the k-means clustering algorithm to get the video feature clusters and set the possible cluster number from 2 to 30. According to the elbow method [40], we find that when we set the cluster number to 23, the generated videos would have the best cluster performance.

The next step is to identify unsafe video categories. Since no unsafe video detector currently exists, we cannot remove unsafe videos before performing clustering. Some video clusters consist of normal videos that require manual inspection. We apply thematic coding analysis [11] in our work, which is usually used in social science, to conclude the theme by qualitatively analyzing data. To get better results, we collect 10 videos from each cluster, and three authors of our work write the text description for each cluster. This is the initial version of our code book, and then we calculate the initial Krippendorff's alpha [41] is 0.39, and Fleiss' kappa [42] is 0.56. Then, we discuss the code and try to refine our code book. We build an overall text description for each cluster and change some of our initial code based on that. For our second code book, Krippendorff's alpha achieves 0.83, and the score of Fleiss' Kappa is 0.94. Both of these scores represent that we have an agreement for almost every cluster. The final step is to group the clusters, as shown in Appendix I. Although some videos are from different clusters, they can be the same type of unsafe videos. After removing the harmless video cluster, we conclude five unsafe categories: *Distorted/Weird*, *Terrifying*, *Pornographic*, *Violent/Bloody*, and *Political*. We show the videos represented for each category in Figure 2. In our work, we use these five unsafe categories to classify unsafe videos and design our defense.

C. Data Collection

We design an online survey based on the five categories of unsafe videos identified through thematic coding analysis [11]. Our survey contains $2,112$ unsafe videos produced by MagicTime [1]. Unlike previous studies that use authors to label inappropriate content [10], we aim to reduce subjective bias and gather more authentic and neutral data. Initially, we obtain approval from the institute's IRB protocol and subsequently create the survey using Qualtrics website. Participants for annotating unsafe videos are recruited through Prolific platform, and compensation is provided. We report human subject's demographics in Appendix H.

Given the sensitive nature of our survey, we include a disclaimer on the first page of Qualtrics and the Prolific homepage, stating that the content contains unsafe information. Participants need to be over 18 years old and are informed that they could withdraw from the survey at any time if they feel uncomfortable without facing any penalties. Each participant is assigned 30 videos and one additional attention check. If participants consider a video to be unsafe, they are prompted to categorize it into one of five categories: *Distorted/Weird*, *Terrifying*, *Pornographic*, *Violent/Bloody*, and *Political*. An "Other" option is provided for any videos that do not fit these categories. Participants are compensated at a rate of $\$10.15/\text{hr}$.

We first conduct a pilot study to validate the design and logic of our Qualtrics survey. We recruit 20 participants to test our questionnaire. The goal of the pilot study is to verify the appropriateness of our time settings, the number of questions, and the attention check. After examining the responses submitted by each participant, we find no modifications are necessary for the questions or the attention check. Accordingly, we decide to recruit 600 participants for the main survey study. These 600 participants for the main survey do not overlap with those who participated in the pilot study.

From the initial pool of 600 participants, 197 individuals fail the attention check and are subsequently excluded from the study. For these participants, we still provide 10% of their compensation through bonus payments. As a result, a total of 403 participants successfully complete the labeling task for 30 generated videos. We first categorize all videos deemed "unsafe" according to participants' labels. Specifically, every video is labeled by at least two participants in our survey. Next, we clean the data by integrating our assessments of "unsafe video". For example, if a video is labeled as "unsafe" and categorized as *Pornographic*, it is retained in the *Pornographic* category only if more than half of the participants who mark it as "unsafe video" also identify it as *Pornographic*. After data cleaning, the videos are categorized based on participants' labels and our assessments. We get 590 videos as *Distorted/Weird*, 579 as *Terrifying*, 445 as *Pornographic*, 204 as *Violent/Bloody*, and 39 as *Political*.

We allow participants to suggest new unsafe categories during our online study. However, because there is not a sufficient number of participants who reach a consensus on any additional category, and because some of the suggested categories are similar to our defined ones (e.g., sexually explicit), we maintain the five unsafe categories generated through thematic coding analysis.

Potential Bias. In this work, we recruit participants to help identify and label unsafe videos to minimize subjective bias. However, we acknowledge that potential biases may still exist in the data collection and annotation phases, and we provide more details about this in Appendix H.

IV. Defense Methodology

Several studies have discussed potential safety issues within image generation models, and proposed defense methods. We group these existing defense methods into two clusters: model-write defense [9], [18], [21]–[26] that modifies part of the model weight or generation process, and model-free defense [10], [19], [20] that based on input and/or output filtering and does not require access to the model internal. Because of their different underlying design purposes, these two methods vary in the level of model access. The model-write method typically operates in a white-box scenario, while the model-free defense can function in a black-box setting.

Model-write defenses change or update the model's generation process or parameters, which might affect the model generation quality, and usually need to fine-tune the model, which takes time [18], [21], [23]–[26]. On the other hand, model-free defense trains classifiers/detectors that predict whether the input prompt and/or output result is harmless. Input prompt filtering is vulnerable to adversarial prompts and jailbreak attacks (described briefly in Section VII-B), while output filtering still needs model owners to first generate the results and also takes time (generating videos takes tens of times longer than generating images). We provide more details of existing defenses for image diffusion models in Section VII-A.

A. Overview of Our Method

Given the drawbacks of both model-write and model-free approaches, we propose a model-read approach, which sits between these two defense mechanisms and works in the gray-box scenario. At a high level, we only require "read access" to the diffusion model in order to detect unsafe content during the diffusion process. This "read access" allows LVD to gather additional information throughout the diffusion process. While LVD shares similarities with the model-free approach—since both rely on training detectors—it surpasses the model-free method by leveraging the extra information obtained from the diffusion model.

With this new detection design, the model-read defense is more robust than input filtration (because it is text-independent) and more efficient than output filtration (as it does not require waiting for the entire generation process to complete). Note that our model-read defense can also potentially collaborate with the other two defenses (as discussed in Section IV-C), providing a more comprehensive defense.

Our solution relies on the insight that generative models are designed to learn latent space representations. We conjecture that the same type of unsafe videos are generated by latent variables close to each other in the latent space.

B. Latent Variable Defense

In this section, we introduce our defense called Latent Variable Defense (LVD). Our method is based on the DDIM sampler [33] used in modern diffusion models for video/image generation, which can significantly enhance inference speed.

DDIM Foundation. Compared to the traditional DDPM's Markovian sampling process [28], DDIM is non-Markovian and deterministic. Different video diffusion models may vary in structure due to distinct design choices [1], [3], [4]. However, to efficiently generate samples, these models move the diffusion process to the latent space. The reverse process in these models can be represented as:

$$\begin{aligned} z_{t-1} =& \sqrt{\bar{\alpha}_{t-1}}\left(\frac{z_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(z_t,t)}{\sqrt{\bar{\alpha}_t}}\right) \\ &+\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(z_t,t)+\sigma_t\epsilon_t \end{aligned}$$

where σ_t denotes the hyper-parameter that controls the level of randomness in the forward process [33], and other symbols share the same notations as in Section II-A. When the $\sigma_t = 0$ for all t, the whole reverse/forward process is a deterministic trajectory. Song et al. [33] prove that when both reverse and forward processes are fixed, model sampling steps can be accelerated by defining the reverse process on a subset of the original T steps and still get the high-quality output. For a video/image diffusion model, the inference steps can be accelerated to only k steps $\tau = \{\tau_1, \tau_2, ..., \tau_k\}$ (these k steps evenly partition the original T steps). The accelerated DDIM sampler equation can be represented as:

$$\Pr\left[z_{\tau_i}|z_{\tau_{i+1}}, z_0\right] = \frac{\Pr\left[z_{\tau_{i+1}}|z_{\tau_i}, z_0\right]\Pr\left[z_{\tau_i}|z_0\right]}{\Pr\left[z_{\tau_{i+1}}|z_0\right]} \quad (3)$$

where $\tau_i < \tau_{i+1}-1$ and $\tau_k \leq T$; z refers to the latent variable. For each latent variable z_{τ_i} in the reverse process, it does not have randomness and is controllable. Given the specific latent variable z_{τ_i}, the final denoised output z_0 is also constant. Therefore, we want to use the DDIM deterministic denoising trajectory to design our defense method.

Unlike the previous model-free methods [10], [19], [20], which rigidly use the final synthesis images as input for detection, we want to train our detection model M using the intermediate latent variables at every step, from τ_k to τ_1. This approach helps build a more agile and integrable defense mechanism.

Defense Algorithm. The details of LVD is shown in Algorithm 1. According to Song et al. [33], setting the number of denoising steps to $k = 50$ achieves nearly the same quality as using $T = 1000$ steps, saving time and computing resources. Therefore, We also set the number of denoising steps to 50 in our experiment, resulting in our defense mechanism that involves 50 detection models, $M_1, M_2, ..., M_{50}$. We use the latent variable at i-th step to train each M_i. It is important to note that we do not directly feed the noised latent variable z_{τ_t} into the detection model. While Equation 1 demonstrates how to obtain the denoised sample during the denoising process at the pixel level (represented by variable x), we adapt this process to work in the latent space (represented by variable z). Given timestep τ_t and noised sample z_{τ_t}, we can calculate the denoised latent $z^0_{\tau_t}$[7] and use it as input for the detection model. The detection results at i-th step can be represented as $s_i = M_i(z^0_{\tau_i})$. For each data point, our mechanism can obtain a vector of scores $s_1, .., s_k$ that helps us leverage a voting strategy to determine whether the current video is unsafe.

LVD leverages detection models to determine at each step whether the intermediate result is unsafe and makes the final decision based on cumulative scores. We introduce two hyperparameters, λ and η. η improves LVD's efficiency by considering only the first $\eta < k$ steps, and λ represents the voting threshold. As displayed in Algorithm 1, we do not apply λ to each step (line 5) because doing so can lead to unstable and noisy predictions due to the inherent variability in each step's output. Instead, we use λ as a voting threshold applied to the cumulative count of positive (unsafe) detections. This strategy reduces the risk of outlier predictions and improves the robustness of the detection process.

To further improve efficiency, LVD can dynamically check the score vector for the generated sample. At the i-th step, the score for the generated video is given by score $= \sum_{j=i}^{k} M_j(z^0_{\tau_j})$. If the score is greater than or equal to $\lambda \cdot (k - i + 1)$, LVD can classify the current video as unsafe.

C. Interoperability with Existing Defenses

Since our method is applicable to all diffusion models, we want to briefly demonstrate how it can be combined with existing defense methods. We showcase this using both model-write [9] and model-free methods [10].

Interoperate with Model-free Defense. For model-free defense, integrating with LVD is straightforward. For example, in

[7] $z^0_{\tau_t}$ represents denoised sample at the t-th step as calculated by Equation 1, which differs from the final denoised output z_0.

Algorithm 1 Latent Variable Defense (LVD)

```
Input: Input prompt $c_p$, detection models across $k$ step $M_1, \dots, M_k$, a set of sampling steps $\tau_1, \dots, \tau_k$, and defense parameters $\lambda$, and $\eta$. $D$ is the decoder in the video generation model.
1: Sampling the initial latent variable $z_{\tau_{k+1}} \sim \mathcal{N}(0, 1)$
2: for $i \leftarrow k$ to $k - \eta + 1$ do                ▷ Perform $\eta$ steps.
3:     $z_{\tau_i} \leftarrow \epsilon_\theta(z_{\tau_{i+1}}, \tau_{i+1})$
4:     Get denoised $z^0_{\tau_i}$ from $z_{\tau_i}$          ▷ By Equation 1
5:     $s_i \leftarrow M_i(D(z^0_{\tau_i}))$                  ▷ 1: unsafe, 0: safe.
6: end for
Output: $\mathbb{1}\left(\sum_{j=k-\eta+1}^{k} s_j \geq \lambda \cdot \eta\right)$    ▷ 1: unsafe, 0: safe.
```

Unsafe Diffusion [10], with a diffusion model set to k denoising steps, the combined defense pipeline can be represented as:

$$\mathcal{M}_{combine}(G(c_p)) = \gamma \cdot \mathbb{1}\left(\sum_{i=1}^{k} s_i \geq \lambda \cdot k\right) + (1 - \gamma)\left(\mathcal{M}_u(D(z_0))\right)$$

Here, D denotes the decoder component of the VGM, c_p represents the input prompt, and γ is a hyperparameter used to balance the combined system. We denote the multi-headed safety classifier from Unsafe Diffusion as $\mathcal{M}_u$. $\mathcal{M}_u$ takes the output sample from the decoder, $D(z_0)$, as input for evaluation.

Interoperate with Model-write Defense. For the model-write defense methods, besides updating the model parameters [9], [18], [23]–[26], we can combine our approach with text-dependent methods [9], [22]. In this section, we briefly discuss how to integrate our method with SLD [9]. The primary goal of SLD is to shift unsafe concepts during the inference step. In their original work, they use the difference between noise from the safety prompt c_s and the input prompt c_p to determine whether to guide the generation direction in the opposite direction. However, to keep the changes minimal, they set a warmup step δ. This is because the noise difference is significant at the beginning of the inference process. In the early stages of inference, LVD can achieve high detection accuracy. Therefore, we can replace the calculation of noise difference $\mu(c_p, c_s)$ at each step with our LVD. At i-th step, $\bar{\epsilon}_\theta(z_{\tau_i}, e_p, e_s) =$

$$\begin{cases} \epsilon_\theta(z_{\tau_i}, e_p) + w(\epsilon_\theta(z_{\tau_i}, e_p) - \epsilon_\theta(z_{\tau_i}, e_s)), & \text{if } \mathcal{M}(z_{\tau_i}; \eta, \lambda) > \beta \\ \epsilon_\theta(z_{\tau_i}, e_p) + w(\epsilon_\theta(z_{\tau_i}, e_p) - \epsilon_\theta(z_{\tau_i})), & \text{otherwise} \end{cases} \tag{4}$$

where e_p and e_s are text embedding for c_p and c_s, β is the pre-defined threshold value, and w is the guidance scale. In Equation 4, when LVD detects the generated content is unsafe, the VGM can redirect the generation direction that is opposite to the safety concept.

D. Discussion

It is important to note that LVD performs detection based on intermediate samples during the generation process, making it a gray-box defense method. Furthermore, LVD does not modify the original model in any way. It simply detects unsafe content during the generation process and can interrupt it early if a sample is flagged as unsafe.

When using LVD alone, there is no difference under white-box or gray-box scenarios. LVD is a plug-in method that can be easily interoperated with other approaches. In white-box scenarios, with access to more information (model architecture and parameters), LVD can work with model-write methods to offer a stronger defense (as shown in Section VI-C). This combined mechanism ensures that even if model-write methods fail to block unsafe concepts, LVD can still prevent unsafe generation.

TABLE I: The number of different categories of unsafe videos generated by various models. One unsafe video can belong to multiple unsafe groups.

Model	Distorted or Weird	Terrify	Porn	Violent or Bloody	Political	Total
MagicTime [1]	590	579	445	204	39	937
VideoCrafter [3]	571	564	353	197	79	931
AnimateDiff [4]	586	577	391	204	75	945

V. Evaluation

A. Experiment Setup

Preparation. As mentioned in Section III-C, we recruit 600 participants via the Prolific platform to label $2,112$ videos generated by MagicTime [1]. Moreover, since AnimateDiff [4] and VideoCrafter [3] both use Stable Diffusion [30] as their backbone (MagicTime and AnimateDiff used SD v1-5, and VideoCrafter used SD 2.1; their semantic-level understanding of unsafe prompts is similar). After carefully reviewing the label information from participants, we annotate the videos generated by the other two models. The number of videos in different unsafe groups generated by the various video diffusion models is presented in Table I. In the evaluation process, we use 20% unsafe videos to build an evaluation dataset to test defense accuracy. We set $k = 50$ following existing literature [33], as $k = 50$ can already ensure high-quality generation while saving significant time. More details about the experimental preparation can be found in Appendix A.

Evaluation Metrics. In our experiments, we not only present the overall accuracy of the defense method but also emphasize the TPR (true positive rate, for correctly classifying unsafe videos) and TNR (true negative rate, for correctly classifying harmless videos). The main reason for considering these two values is to align with our objective of ensuring that the defense mechanism does not interfere with the generation of harmless videos. If detection's accuracy is high but the TNR is low, it indicates that many harmless videos are being incorrectly detected as unsafe.

Furthermore, to illustrate the relationship between TPR and FPR (false positive rate, for misclassifying safe videos), we use the Area Under the Receiver Operating Characteristic Curve (AUC ROC) to demonstrate the defense performance of LVD.

B. Impact of Inference Steps

We notice the efficiency of our method is significantly influenced by the hyperparameters λ and η, as shown in Algorithm 1. In our defense mechanism, λ controls the degree of trust in the detection accuracy at different denoising steps, while η determines how many steps the LVD takes before performing a detection analysis. LVD is designed based on the characteristics of DDIM [33]. Therefore, we first aim to explore the detection success rate of the detection model at different denoising steps in order to help us get the range of λ and η. We trained detection models on each type of unsafe video generated by the generative model separately. The experiment results for five groups of unsafe videos are in Appendix G. In this section, we trained 50 models for each unsafe category and totally trained 250 detection models for each generation model. The evaluation set is not utilized here because our goal is to explore the performance of each group of detection models across all denoising steps. The tests are conducted separately for each group. For the *Pornographic* group, only unsafe *Pornographic* videos are used to assess its performance.

According to the results illustrated in Figure 7, it can be observed that model is capable of achieving perfect detection with MagicTime [1] and AnimateDiff [4] at the initial phase of the synthesis process, but shows relatively poor detection accuracy with VideoCrafter [3].

C. Impact of η and λ

Based on the results observed in Section V-B, we can set the ranges of η and λ for the three models used in our experiment. For MagicTime [1], the detection accuracy remains high even during the initial stages of the denoising process. Thus, we believe that at a lower η, the detection accuracy is already perfect. This is because the model is able to reconstruct the outline of the denoised object approximately by the early step, enabling the detection model to make an accurate judgment based on this outline. Conversely, for VideoCrafter [3], the initial stages of the denoising process fail to generate reasonable object representations. Therefore, we think LVD for VideoCrafter might need a higher η to achieve the best detection accuracy. Similarly, due to VideoCrafter's inferior detection accuracy, we can impose looser conditions by setting a lower λ value to ensure that every potentially unsafe video is detected.

When conducting evaluations on LVD, we ensure the balance of the samples under testing. The number of class 0 (harmless video) and class 1 (unsafe video) samples is equal. Based on the result presented in Figure 7, we believe it is unnecessary to set the range of η from 0 to 50. We set η to range from 1 to 20 and assign λ values of 0.3, 0.6, and 1 in our work.

TABLE II: Illustrates the defense accuracy of LVD employed on MagicTime [1], AnimateDiff [4], and VideoCrafter [3] under varying settings of the hyper-parameters η and λ. We highlight the best detection performance (based on accuracy) for each η. In addition, we also present the TNR (0: harmless video) and TPR (1: unsafe video). We think this can provide insights into the method's performance in correctly identifying benign and unsafe instances, respectively.

Model	Evaluation Metrics	Latent Variable Defense															# unsafe samples
		$\eta = 1$			$\eta = 3$			$\eta = 5$			$\eta = 10$			$\eta = 20$			
		0.3	0.6	1.0	0.3	0.6	1.0	0.3	0.6	1.0	0.3	0.6	1.0	0.3	0.6	1.0	
MagicTime [1]	TNR		0.68		0.34	0.67	**0.90**	0.40	0.64	**0.95**	0.40	0.77	**0.99**	0.74	**0.98**	1.00	203
	TPR		0.95		0.98	0.95	**0.91**	0.99	0.97	**0.87**	0.99	0.99	**0.84**	0.99	**0.99**	0.81	
	Accuracy		0.81		0.66	0.81	**0.90**	0.70	0.81	**0.91**	0.70	0.88	**0.92**	0.87	**0.99**	0.90	
AnimateDiff [4]	TNR		0.73		0.45	0.73	**0.93**	0.54	0.72	**0.97**	0.51	0.74	**0.99**	0.59	**0.88**	1.00	297
	TPR		0.98		1.00	0.97	**0.89**	0.98	0.96	**0.85**	0.99	0.96	**0.81**	0.98	**0.95**	0.74	
	Accuracy		0.85		0.72	0.85	**0.91**	0.76	0.84	**0.91**	0.75	0.85	**0.90**	0.79	**0.92**	0.87	
VideoCrafter [3]	TNR		0.54		0.31	0.63	**0.87**	0.50	0.65	**0.93**	0.47	**0.71**	0.95	0.56	**0.87**	1.00	307
	TPR		0.89		0.99	0.95	**0.80**	0.98	0.96	**0.75**	0.99	**0.94**	0.69	0.98	**0.94**	0.66	
	Accuracy		0.72		0.65	0.79	**0.84**	0.74	0.81	**0.84**	0.73	**0.83**	0.82	0.77	**0.91**	0.83	

Table II demonstrates that the detection accuracy of LVD increases with higher η values. We highlight the best detection accuracy obtained at different η values.

Except when η equals 1, the value of λ does not affect the results. We observe four different η values and note an interesting phenomenon. When η is low, such as η = 3 or η = 5, better detection accuracy is usually achieved when λ equals 1. However, as η increases, the most accurate detection results are often obtained when λ is 0.6. This trend is consistent with the detection results for three VGMs.

We think this occurs because, with a high η, setting λ to 1 makes the model's detection very stringent. In other words, LVD must consistently identify a sample as unsafe at every denoising step to finally classify it as an unsafe video. When η is low, LVD needs to be more stringent at each denoising step due to insufficient data for each sample. However, as η increases, a high λ value can cause LVD to misclassify some samples because a few denoising steps might indicate safety, affecting the overall judgment. To validate our hypothesis, we fix λ at 0.3, 0.6, and 1.0. Then we present the TPR, TNR, and accuracy for η ranging from 1 to 50 from MagicTime [1] in Figure 5. It is evident that when λ is 1, and η is low, the accuracy is higher than when λ is set to 0.3 or 0.6. However, as η increases, accuracy decreases. When λ is 1, the TPR value also rapidly decreases as η increases.

Therefore, when η is set to 20, the best detection performance is achieved with λ equal 0.6. Under this parameter setting, the TNR value remains at 0.98, ensuring efficient detection of unsafe videos while minimizing the impact on harmless video generation. More details on how η and λ affect detection accuracy can be found in Appendix E.

D. Impact of Different Unsafe Categories

Additionally, we want to examine whether the detection performance of LVD is affected by targeting different unsafe categories. First of all, we classify the data samples in the evaluation dataset according to their unsafe categories. Then, using the optimal λ values obtained from Table II, we evaluate the model and present the results in Figure 3.

In Figure 3, we show the detection accuracy for unsafe categories under different η settings. We find that for all video generation models, when η is set to a low value (i.e., using the initial intermediate outputs of the model's denoising process), the detection accuracy for *Political* samples is lower than that for other unsafe samples. We believe this is because the sample size in the *Political* category is much smaller than in other categories. As a result, the detection model is prone to overfitting the training data and does not effectively learn the common features of this category during training. When η is small, relying on a limited number of detection results cannot achieve accurate detection. However, the results show that when $\eta = 20$, LVD can achieve nearly 0.95 detection accuracy for *Political* unsafe samples generated by three VGMs. It can be clearly observed that the detection accuracy for the other unsafe video categories is balanced and improves with higher values of η. This phenomenon is aligned with Table II and demonstrates that LVD can effectively detect unsafe video samples of all categories.

TABLE III: Compared the optimal accuracy of our defense mechanism for MagicTime [1] under different η values with existing model-free works [10].

Evaluation Metrics	Latent Variable Defense				Unsafe Diffusion [10]
	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	
TNR	0.90	0.95	0.99	**0.98**	0.56
TPR	0.91	0.87	0.84	**0.99**	0.98
Accuracy	0.90	0.91	0.92	**0.99**	0.77

E. Comparison with Existing Methods

In this section, we aim to compare our approach with existing methods. Since ours is the first work to focus on unsafe synthesis in the video domain, we only compare against defense methods designed for image generation models. However, many model-write defense methods are designed to change the output object and require adjustments to the model itself or modifications to the model's attention matrix [9], [18], [21], [23]–[26]. Given that the use of attention mechanisms in VGMs differs from that in image generators [30]. Additionally, MagicTime [1] employs a more complex DiT-based architecture [43]. We primarily compare our method with Unsafe Diffusion [10] and Safe Latent Diffusion [9] (SLD).

Comparison with Model-Free Methods. In Table III, we

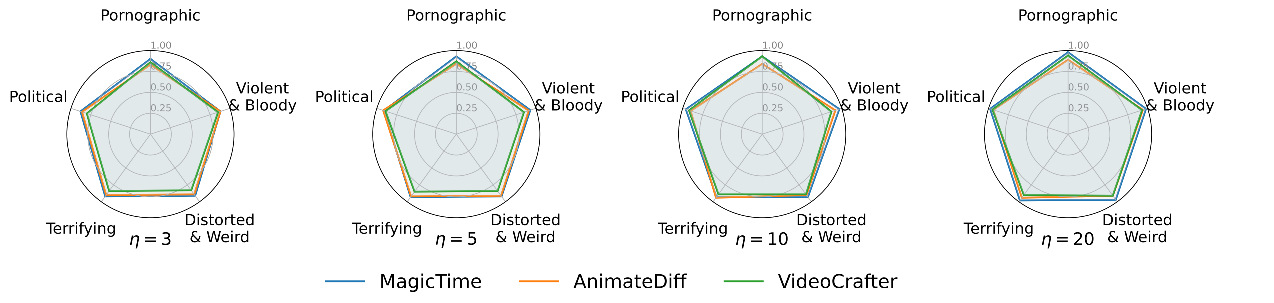


Fig. 3: Detection accuracy for different unsafe categories for MagicTime [1], AnimateDiff [4], and VideoCrafter [3].

TABLE IV: Comparison of our method with Safe Latent Diffusion [9] for defending against content generated from NSFW-200. For each η, we set λ to align with the value that achieves optimal detection accuracy in Table II. The evaluation metric for SLD is the nudity removal rate, and all parameter configurations follow the original paper. "# Unsafe Samples" represents the actual number of unsafe samples generated by each model.

Model	Latent Variable Defense				Safe Latent Diffusion				# Unsafe Samples
	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	Weak	Medium	Strong	Max	
MagicTime [1]	0.87	0.88	0.91	0.97	0.52	0.63	0.73	0.86	172
AnimateDiff [4]	0.90	0.91	0.93	0.96	0.43	0.62	0.75	0.78	188
VideoCrafter [3]	0.84	0.88	0.89	0.91	0.67	0.71	0.73	0.75	181

compare the optimal detection accuracy of our defense mechanism under different η values with the detection performance of Unsafe Diffusion [10]. To ensure a fair comparison, we keep the number of training samples and other parameters consistent. Although the original work use an image classifier as the detection model, our current study deals with video content. To maintain fairness in the comparison, we also use VideoMAE as the detection model for Unsafe Diffusion.

The results show that our defense mechanism significantly outperforms Unsafe Diffusion in terms of detection accuracy. Although Unsafe Diffusion [10] achieves TPR value of 0.99 for unsafe videos, it correctly identifies only 0.56 of harmless videos. In contrast, our defense mechanism attains TNR values of 0.90, 0.95, 0.99, and 0.98, respectively. This indicates that Unsafe Diffusion is likely to misclassify a large number of safe samples, thereby affecting normal user experience.

Furthermore, we observe that sample generation time varies across models, with MagicTime [1] taking the longest at 85.4 ± 1.1 seconds per sample. When employing a model-free safety filter [10], [19], [20], users must wait model to complete the entire sampling before using a feature extractor to detect unsafe content. In contrast, LVD achieves higher accuracy at $\eta = 5$ (one-tenth of the full generation time), demonstrating a $10\times$ improvement in computational efficiency.

More comparisons between our defense mechanism and Unsafe Diffusion [10] on AnimateDiff [4] and VideoCrafter [3] can be found in Appendix F.

Comparison with Model-Write Methods. We also compare our method with model-write approaches. However, model-write methods aim to prevent unsafe content from being generated rather than detecting it. It is implausible to directly compare model-write methods with LVD. Following ESD [21], we use the nudity removal rate (NRR) to quantify the defense performance of model-write methods. A high NRR indicates that the method effectively disrupts most of the unsafe generation process and converts the unsafe sample into a harmless one. Similarly, when our method detects unsafe samples, it can serve as a trigger to block the generation of these samples. Thus, high detection accuracy also reflects the ability to interrupt most unsafe generation processes. In this section, we compare our method with four different configurations of SLD [9]. The prompts used to generate unsafe videos are the same for both methods.

In Table IV, we present the defense success rate and noticed despite using the "Max" configuration (threshold set to 1, affecting normal sample generation), SLD [9] still fails to fully prevent unsafe sample generation. We believe this is because SLD relies on modifying classifier-free guidance during the inference phase to avoid generating unsafe concepts. However, when the unsafe concepts are stubborn, and the inference steps are limited, this significantly impacts the effectiveness of SLD. In comparison, LVD achieves over 0.90 accuracy using the intermediate outputs from the initial 10 denoising steps.

F. Case Study Against Deepfake Generation

As mentioned in Section II-C, although LVD is not inherently designed to address deepfake attacks, it can be leveraged to defend against the generation of deepfake samples. According to Han et al. [44], deepfake attacks are typically targeted at famous persons, generating pornographic content. Since this type of content also falls within the scope of our detection, we design this case study to evaluate LVD's performance against the generation of pornographic deepfake videos.

In the experiment, we collect prompts from our malicious prompts set that are classified as pornographic. These prompts

TABLE V: Accuracy of LVD against four different adversarial prompt settings (Brute Force, Beam, Greedy, Reinforcement Learning.). For each η, the value of λ matches the highlighted parameters in Table II.

Model	MagicTime				AnimateDiff				VideoCrafter				# Adversaria Prompts
	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	
Brute Force Search	0.79	0.86	0.87	0.92	0.90	0.89	0.96	0.94	0.79	0.83	0.95	0.93	79
Beam Search	0.78	0.83	0.85	0.93	0.91	0.90	0.86	0.94	0.87	0.88	0.94	0.93	74
Greedy Search	0.84	0.88	0.89	0.91	0.85	0.82	0.93	0.92	0.83	0.81	0.94	0.93	76
Reinforcement Learning	0.86	0.91	0.91	0.99	0.91	0.92	0.87	0.97	0.90	0.91	0.93	0.96	156

TABLE VI: TPR scores of LVD in defending against deepfake generation targeting 5 well-known persons.

Deepfake Target	MagicTime	AnimateDiff	VideoCrafter
Donald Trump	0.80	0.92	0.94
Vladimir Putin	0.78	0.98	0.98
Elon Musk	0.98	0.98	0.96
Margot Robbie	0.98	0.94	0.96
Taylor Swift	0.92	0.92	0.98

are then modified by replacing words such as “man” and “woman” with famous people’s names while maintaining other components. As a result, we create a set of 50 generalizable pornographic prompts for each famous person. Using these prompts, we query VGMs to generate the deepfake samples.

Since we treat unsafe samples as the positive class and focus on them, we only present the detection TPRs as the evaluation metric in Table VI. Despite using different target persons for deepfake generation, the detection TPRs on all three VGMs are mostly above 0.80. For deepfake samples of Elon Musk, the TPR reaches up to 0.98. These results demonstrate that LVD can effectively serve as a defense solution against generating unsafe deepfake samples.

Takeaways: In this section, we observe the detection accuracy of models trained with different denoising steps to initially determine the experimental range for η and λ. Next, we test the effectiveness of our defense mechanism on three VGMs using the evaluation set. Because of the design objective for our mechanism, we focus on TNR, TPR, and accuracy. The experimental results show that our defense mechanism provides effective protection for all VGMs in our study. Then, we compare our method with existing defense methods for text-to-image models, demonstrating that our defense mechanism offers more efficient protection. Finally, we conduct a case study to demonstrate that LVD can also be applied to counter deepfake generation.

VI. Ablation Study

A. Evaluation with Adversarial Prompts

According to Yang et al. [45], and Qu et al. [10], normal prompts can query models to generate unsafe content. Specifically, Yang et al. conduct jailbreak experiments on text-to-image generation models. In their study, they first categorize model-free defense methods into three types: text-based safety filters, image-based safety filters, and text-image-based safety filters. In their work, they design SneakyPrompt to avoid these safety filters. They use beam search, greedy search, brute force search, and reinforcement learning to build their SneakyPrompt algorithm. Then, they define an evaluation metric called the bypass rate, which measures the number of adversarial prompts that successfully bypass the safety filter. Their adversarial prompts achieve a 100% bypass rate against the text-only safety filter that built-in Stable Diffusion [30]. Furthermore, the bypass rate of their adversarial prompts exceeds that of the manually crafted prompts by Rando et al. [20] and Qu et al. [10].

Given that the core idea of LVD is to use a detection model at each denoising step, it can be considered a type of safety filter operating in the inference process. Using the currently most potent adversarial prompt algorithm, SneakyPrompt [45], we build adversarial datasets to evaluate the robustness of our defense. We set different configurations for SneakyPrompt and employ them to test LVD on three VGMs. All datasets are built by applying SneakyPrompt to the original NSFW-200 dataset.

Due to differences in the models, the amount of unsafe content generated by adversarial prompts varies among the three models. For each model, we filter out the harmless videos before conducting the experiments.

According to Table V, our defense mechanism successfully detects unsafe content across various settings of adversarial prompts on VGMs. When η is set to 20, LVD achieves an accuracy exceeding 0.90 on videos generated from all adversarial prompt sets. We attribute the success of our defense mechanism to its focus on detecting unsafe content during the inference steps. In contrast, adversarial prompts are typically designed to ensure that the final generated unsafe samples can evade safety filters.

B. Evaluation with Image-to-Video Models

After testing with the adversarial prompt dataset, a natural idea arises for our method. our defense mechanism uses detection models to identify denoised latent variables z_0 during the inference process of VGMs. In text-to-video models, at the t-th step, z_0 can be represented as

$$z_0 = \frac{z_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(z_t, t, c_p)}{\sqrt{\bar{\alpha}_t}}$$

where c_p represents the input prompt guidance and other notations aligned with the previous sections. However, in image-to-video tasks [2], [5], [46], the primary difference is

TABLE VII: We used 200 selected images generated from Unsafe Diffusion [10] as inputs for AnimateDiff [4] and VideoCrafter [3]. Our primary focus is on demonstrating the unsafe video detection performance in the image-to-video tasks of these two models. Therefore, we mainly focus on the changes in True Positive Rate. We highlight cases where detection accuracy from I2V tasks is much lower than that of T2V tasks due to the generation tasks changed. The generation process for I2V is different from T2V.

Task	Latent Variable Defense												# unsafe samples
	$\eta = 3$			$\eta = 5$			$\eta = 10$			$\eta = 20$			
	0.3	0.6	1.0	0.3	0.6	1.0	0.3	0.6	1.0	0.3	0.6	1.0	
VideoCrafter-(Text-to-Image)	0.99	0.95	0.80	0.98	0.96	0.75	0.99	0.94	0.69	0.98	0.94	0.66	307
VideoCrafter-(Image-to-Image)	1.00	0.99	**0.86**	1.00	1.00	**0.78**	1.00	0.98	**0.65**	0.99	0.97	**0.58**	180
AnimateDiff-(Text-to-Image)	1.00	0.97	0.89	0.98	0.96	0.85	0.99	0.96	0.81	0.98	0.95	0.74	297
AnimateDiff-(Image-to-Image)	0.99	0.89	**0.61**	0.97	0.92	**0.48**	0.99	0.86	**0.24**	0.98	0.87	**0.21**	180

that the conditional input c_p is converted from a prompt to an image. The rest of the generation process remains unchanged. Consequently, to explore the generalization ability of our method, we aim to use unsafe images to query an image-to-video diffusion model [3]–[5], [46] and thereby confirm the versatility of our approach.

It is noteworthy that, during model selection, both AnimateDiff [4] and VideoCrafter [3] are capable of performing image-to-video generation tasks. However, MagicTime can only perform text-to-video generation and cannot be used in this section. To query the model with unsafe images to generate unsafe videos, we select 200 images they deemed most unsafe from those generated using unsafe prompts by Qu et al. [10] from Stable Diffusion [30].

After data cleaning, we compile a dataset of 200 unsafe images to query the video generation model. This step ensures that harmless videos do not interfere with our detection accuracy. We remove 20 poorly generated videos for each model and those without harmful content. Then, we proceed to detect harmful content in the remaining videos generated by the respective models.

In this section, we do not retrain detection models. Instead, we use detection models trained on unsafe videos generated by the same model's text-to-video task. We ensure the pre-trained parameters are consistent with those used in the text-to-video generation task. For example, for AnimateDiff [4], we use the same version of Stable Diffusion v1.5 [30] parameters for the image-to-video task as for the text-to-video task. Additionally, the versions and types of the LoRA module and motion module remain consistent; only the modality of the input data has changed.

From the results in Table VII, we can see that LVD still maintains a high detection success rate when detecting unsafe content generated by different tasks of the same model. In Table VII, we choose to display only the TPR scores. This is because the negative samples in the detection tests are harmless videos generated by the same model's text-to-video task. These samples have already shown detection results in Table II. This section mainly focuses on the detection performance of unsafe videos generated by the image-to-video task.

It can be seen that when λ is set to 0.3 and 0.6, both models achieve around 0.90 TPR across all values of η for the image-to-video task. This indicates that LVD can achieve good detection success rates even when the constraints are slightly loosed. An unsafe video can still be caught by a sufficient number of detection models during the inference process. However, as we continue to increase λ, we observe a significant drop in TPRs for the image-to-video task. This phenomenon is particularly notable in AnimateDiff [4]. When η equals 20, the TPR drops to only 0.21. This is much lower than the 0.74 TPR for detecting unsafe videos generated by the text-to-video task.

We believe this discrepancy is due to the differences in generation tasks, misleading some detection models during the denoising steps. These models incorrectly classify the videos as harmless, leading to substantially decreased TPRs when LVD requirements are stricter. However, if we relax the constraints slightly (set λ to 0.6) when η is high, our defense mechanism can still achieve over 95% detection success rate for the image-to-video task. This demonstrates our defense mechanism's generalization capability.

C. Interoperability Evaluation

LVD can serve as a plug-in model-read defense mechanism, allowing for easy integration with other defense strategies to provide more effective protection. In this subsection, we test the combination of LVD with the classic model-free method, Unsafe Diffusion [10], as well as the model-write approach, Safe Latent Diffusion (SLD) [9].

Integrate with Model-free Methods. We examine the detection performance of LVD combined with Unsafe Diffusion using different η and λ settings (γ is set to 0.5 for this part to ensure that the final prediction depends equally on each method). As shown in Table VIII, combining with LVD at $\eta = 3$ significantly improves overall defensive accuracy with original Unsafe Diffusion. Results from MagicTime [1] indicate that the TNR increased from 0.56 to 0.95, and the accuracy rose from 0.77 to 0.92. We notice that combining Unsafe Diffusion with LVD effectively addresses the poor detection of negative cases by Unsafe Diffusion. The detection results from all models support this point. Additionally, the combined defense enhances accuracy in certain η settings compared to using LVD alone. For instance, at $\eta = 10$, the TNR, TPR, and accuracy for LVD alone are 0.99, 0.84, and 0.92, respectively, while the combined defense achieves 0.96, 0.93,

TABLE VIII: Testing the combination of Unsafe Diffusion [10] with different η settings LVD on MagicTime [1], AnimateDiff [4], and VideoCrafter [3]. The λ values are set to align with the best performance in Table II.

Method	MagicTime			AnimateDiff			VideoCrafter		
	TNR	TPR	Accuracy	TNR	TPR	Accuracy	TNR	TPR	Accuracy
Unsafe Diffusion	0.56	0.98	0.77	0.68	0.95	0.82	0.65	0.95	0.80
UD + LVD ($\eta = 3$)	0.95	0.90	0.92	0.95	0.88	0.91	0.90	0.78	0.84
UD + LVD ($\eta = 5$)	0.91	0.92	0.92	0.97	0.85	0.91	0.73	0.93	0.83
UD + LVD ($\eta = 10$)	0.96	0.93	0.94	0.99	0.81	0.90	0.89	0.89	0.89
UD + LVD ($\eta = 20$)	0.98	0.98	0.98	0.91	0.93	0.92	0.89	0.92	0.91

and 0.94. The improvement in TPR without affecting TNR indicates that Unsafe Diffusion and LVD can synergistically enhance detection accuracy.

Integrate with Model-write Methods. With SLD [9], our method can replace the calculation of distances between safety concept embeddings and prompt embeddings during detection. Additionally, it dynamically adjusts the momentum based on the detection model's confidence that the sample is unsafe.

To evaluate the effectiveness of SLD on VGMs and the performance improvement when combined with LVD, we use NSFW-200 [45] for testing. Since the primary goal of SLD is to eliminate unsafe concepts encountered during the generation process, the objective of our experiment is to evaluate whether introducing LVD can improve the nudity removal rate [21]. Given that the ultimate goal is to eliminate unsafe concepts and obtain the generated samples, we set η to align with the current step in the denoising process and set λ to 0.6.

From the demonstration results,[8] we find that combining LVD with SLD [9] more effectively removes unsafe concepts. We observe that, with the original SLD configurations, 'LVD + SLD (weak)' effectively removes unsafe concepts in samples where SLD (medium) fails. The videos generated with 'LVD + SLD' can identify specific areas that need to be covered with clothing or other items. Therefore, even when only the weak SLD configuration is used for defense, it still provides excellent protection. Using 'LVD + SLD (medium)' provides stronger and more reasonable defenses. For example, as shown in the fifth row, petals are generated as task objects in videos while preserving the background similarity to the original video. More results of 'LVD + SLD' on three video generation models are in Appendix B.

Takeaways: In this section, we further examine our defense mechanism's robustness, generalization ability, and interoperability. Firstly, we use adversarial prompts to generate videos and tested the detection capability of our defense mechanism. The results show that adversarial prompts do not reduce detection performance across the three models, demonstrating the robustness of our method. Then, we test our defense mechanism on different generation tasks within the same model. We find that the detection models trained on text-to-video tasks still effectively detect unsafe content in image-to-video tasks, maintaining a TPR close to 1.00. The experimental results show that our model is task-agnostic and has strong generalization ability. Finally, we combine LVD with existing model-write and model-free methods. The results show that our method can enhance the performance with other methods.

[8]Due to content restrictions, we do not show the results in the main paper but have uploaded them to https://github.com/py85252876/UVD/tree/main/SLD_with_LVD

VII. Related Work

A. Existing Defenses for Image Diffusion Models

Model-write Defense. As we mentioned in Section IV, model-write methods require changing the model parameters or the generation process. Schramowski et al. [9] first propose a safety guidance strategy to prevent models from generating inappropriate content. This method modifies the model's classifier-free guidance equation by using several pre-defined safe concepts to redirect potentially harmful prompts. Subsequently, Gandikota et al. [21] argue that harmful concepts could be erased from the model's understanding. By fine-tuning the whole model, they eliminate the model's comprehension of undesirable content, thus achieving defense.

The problem with model-write defense is that this type of method needs to change or update the model's generation process or parameters, which might affect the model generation quality [9], [18], [21], [23]–[26]. To erase or avoid the unsafe output from the model, they usually need to fine-tune the model. These methods require substantial time for fine-tuning generative models in the image domain, and implementing them on more complex video diffusion models demands even greater computational resources and time. Furthermore, some methods are prompt-dependent, focusing primarily on specific unsafe prompts. However, as Qu et al. [10] discover, using normal prompts can still generate inappropriate outputs. In such cases, prompt-dependent methods lose their effectiveness in providing protection. Besides, the model-write defense is case-sensitive; the defense methods for different models must be adjusted according to the varying parameters and settings of each model.

Model-free Defense. The typical feature of model-free defense is the defense process does not interact with the generation model [10], [19], [20]. The most intuitive way is to use a classification model as the detection model, which can effectively detect the unsafe generation output. Rando et al.

show the safety filter in Stable Diffusion [30] determines the safety of generated images by extracting features with CLIP and comparing them to 17 unsafe concepts. After that, Qu et al. [10] use linear probing with a pre-trained CLIP model to create a detection model. When training with unsafe images, they only update the parameters of the linear layer while keeping the pre-trained CLIP model frozen. Because they only detect the output from the generator and can ignore the internal mechanism [10], [19], [20], model-free methods have better generalization than model-write defenses. However, they lack flexibility and are relatively rigid. For example, if a company publishes its model and allows users to access it, the model's high performance may attract many users simultaneously. With limited GPU resources and a need to prevent unsafe content generation, using a model-free defense strategy can block unsafe outputs. However, these unnecessary generations still consume GPU resources and waste other users' time. Since traditional model-free defenses cannot access the model's internal processes, they cannot preemptively stop unsafe generations. Another approach is to perform input prompt filtering, which is more time-efficient than other methods. However, this method is vulnerable to adversarial prompts and jailbreak attacks (described briefly in Section VII-B).

B. Jailbreak Attacks on Generative Models

As discussed in Section VII-A, numerous safety filters have been proposed to prevent the misuse of generative model capabilities and ensure the security of generated content. The unsafe generation issue is not limited to image [30], [47] and video generators [1], [3], [4] but extends to large language [48] and audio [49] models. Various jailbreak techniques have been investigated to evaluate safety filters' robustness.

For example, in the text domain (aka, in large language models), Liu et al. [50] analyze different types of jailbreak prompts, including pretending, attention shifting, and privilege escalation. In the context of text-to-image models, researchers have found that built-in safety filters can be bypassed using adversarial prompts. As discussed in our paper, Rando et al. [20] initially bypass filters by manually adding extraneous, irrelevant information to prompts. Qu et al. [10] identify that normal prompts could query the model to generate unsafe images and manually collect these prompts to create a structured jailbreak prompt dataset. Believing the previous methods are inefficient, Yang et al. [45] employ various search strategies and reinforcement learning techniques to develop a highly effective adversarial prompt-building technique and collect a dataset. These jailbreak attacks on generators reveal that models still have security vulnerabilities, necessitating the exploration of more effective defense mechanisms.

VIII. Conclusion and Discussion

Limitations. In addition to the potential biases in the dataset discussed in Section III-C, there may also be incomplete coverage of unsafe categories. Although we have made every effort to search for appropriate benchmark prompts, the actual number of unsafe categories could exceed five in reality. Due to the limited number of prompts, we have only identified a portion of these categories. As a result, our defense mechanism may fail to detect new types of unsafe videos, though it could be extended to incorporate them.

Secondly, although our defense mechanism has achieved nearly perfect detection accuracy, one major issue is the high cost of training the model. Even when η is smaller, our model can still accurately identify most diffusion model generation tasks, achieving over 0.90 in TPR, TNR, and accuracy. However, more complex models in the future may require larger η values for better detection. This, in turn, will significantly increase the number of detection models needed. For instance, when η equals 30, at least $30 \times n$ detection models must be trained for n unsafe categories.

Potential Extensions. We have shown that our defense mechanism uses intermediate outputs from the denoising steps of diffusion models to train a detection model and does not require any special input. Therefore, it can form a general and robust defense mechanism. This approach can be applied to various types of diffusion models, including text-to-image models [30]. Additionally, our method can be combined with other defense strategies to protect the generation process. For example, when we set η to the total number of denoising steps, our method can work alongside external safety filters [10], [20]. When η is less than the number of denoising steps, it can collaborate with internal defense methods [9], [18], [21] to ensure that unsafe concepts are successfully removed from the generated outputs.

Conclusion. Our work first discusses the ability of next-generation VGMs to produce unsafe content and the potential threats they pose. We find that, with specific prompts input, VGMs can create various high-resolution unsafe videos. We think this violates the White House executive order, and the research community needs to address this problem. To thoroughly understand the models' capability, we collect unsafe prompt data from 4chan and Lexica. After cleaning and filtering the data, we obtain an initial dataset of 2112 prompts capable of guiding VGMs to produce unsafe videos.

We then use data-driven methods, including k-means and thematic coding analysis, to identify unsafe categories for the generated videos. Participants are recruited to label the videos based on these categories. From the initial 2112 generated unsafe videos, participants identify 937 that are universally recognized as unsafe and classify each one. Using the annotations and corresponding prompts, we construct the first unsafe video dataset specifically for VGMs.

Based on this dataset, we design the Latent Variable Defense (LVD), the first defense method to prevent unsafe generation processes in VGMs. Our experiment results indicate that LVD provides reliable defense across three types of VGMs included in our experiments, achieving 0.99, 0.92, and 0.91 detection accuracy. Furthermore, it maintains 95% effectiveness when tested against adversarial prompts and different generation tasks within the same model.

Acknowledgment

We thank the reviewers for their valuable comments and suggestions. This work is supported by NSF 1931441, 2340788, and 2217071.

References

[1] S. Yuan, J. Huang, Y. Shi, Y. Xu, R. Zhu, B. Lin, X. Cheng, L. Yuan, and J. Luo, "Magictime: Time-lapse video generation models as metamorphic simulators," *arXiv preprint arXiv:2404.05014*, 2024.

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

[3] H. Chen, Y. Zhang, X. Cun, M. Xia, X. Wang, C. Weng, and Y. Shan, "Videocrafter2: Overcoming data limitations for high-quality video diffusion models," *arXiv preprint arXiv:2401.09047*, 2024.

[4] Y. Guo, C. Yang, A. Rao, Y. Wang, Y. Qiao, D. Lin, and B. Dai, "Animatediff: Animate your personalized text-to-image diffusion models without specific tuning," *arXiv preprint arXiv:2307.04725*, 2023.

[5] S. Zhang, J. Wang, Y. Zhang, K. Zhao, H. Yuan, Z. Qin, X. Wang, D. Zhao, and J. Zhou, "I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models," *arXiv preprint arXiv:2311.04145*, 2023.

[6] D. J. Zhang, J. Z. Wu, J.-W. Liu, R. Zhao, L. Ran, Y. Gu, D. Gao, and M. Z. Shou, "Show-1: Marrying pixel and latent diffusion models for text-to-video generation," *arXiv preprint arXiv:2309.15818*, 2023.

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

[8] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8633–8646, 2022.

[9] P. Schramowski, M. Brack, B. Deiseroth, and K. Kersting, "Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 522–22 531.

[10] Y. Qu, X. Shen, X. He, M. Backes, S. Zannettou, and Y. Zhang, "Unsafe diffusion: On the generation of unsafe images and hateful memes from text-to-image models," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 3403–3417.

[11] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[12] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in *2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[13] D. Wodajo, S. Atnafu, and Z. Akhtar, "Deepfake video detection using generative convolutional vision transformer," 2023.

[14] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. Jain, "On the detection of digital face manipulation," 2020.

[15] A. Gandhi and S. Jain, "Adversarial perturbations fool deepfake detectors," 2020.

[16] Y. He, B. Gan, S. Chen, Y. Zhou, G. Yin, L. Song, L. Sheng, J. Shao, and Z. Liu, "Forgerynet: A versatile benchmark for comprehensive forgery analysis," 2021.

[17] R. Wang, F. Juefei-Xu, L. Ma, X. Xie, Y. Huang, J. Wang, and Y. Liu, "Fakespotter: A simple yet robust baseline for spotting ai-synthesized fake faces," 2020.

[18] X. Li, Y. Yang, J. Deng, C. Yan, Y. Chen, X. Ji, and W. Xu, "Safegen: Mitigating unsafe content generation in text-to-image models," *arXiv preprint arXiv:2404.06666*, 2024.

[19] M. Li, "Nsfw text classifier on hugging face," 2022.

[20] J. Rando, D. Paleka, D. Lindner, L. Heim, and F. Tramèr, "Red-teaming the stable diffusion safety filter," *arXiv preprint arXiv:2210.04610*, 2022.

[21] R. Gandikota, J. Materzynska, J. Fiotto-Kaufman, and D. Bau, "Erasing concepts from diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2426–2436.

[22] M. Brack, F. Friedrich, D. Hintersdorf, L. Struppek, P. Schramowski, and K. Kersting, "Sega: Instructing text-to-image models using semantic guidance," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[23] N. Kumari, B. Zhang, S.-Y. Wang, E. Shechtman, R. Zhang, and J.-Y. Zhu, "Ablating concepts in text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 691–22 702.

[24] S. Kim, S. Jung, B. Kim, M. Choi, J. Shin, and J. Lee, "Towards safe self-distillation of internet-scale text-to-image diffusion models," *arXiv preprint arXiv:2307.05977*, 2023.

[25] M. Lyu, Y. Yang, H. Hong, H. Chen, X. Jin, Y. He, H. Xue, J. Han, and G. Ding, "One-dimensional adapter to rule them all: Concepts diffusion models and erasing applications," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 7559–7568.

[26] R. Gandikota, H. Orgad, Y. Belinkov, J. Materzyńska, and D. Bau, "Unified concept editing in diffusion models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 5111–5120.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[28] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[29] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[30] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[31] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[32] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, "Audioldm: Text-to-audio generation with latent diffusion models," *arXiv preprint arXiv:2301.12503*, 2023.

[33] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[34] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," 2020.

[35] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and simulating artifacts in gan fake images," 2019.

[36] M. Zhu, H. Chen, Q. Yan, X. Huang, G. Lin, W. Li, Z. Tu, H. Hu, J. Hu, and Y. Wang, "Genimage: A million-scale benchmark for detecting ai-generated image," 2023.

[37] D. Cozzolino, K. Nagano, L. Thomaz, A. Majumdar, and L. Verdoliva, "Synthetic image detection: Highlights from the ieee video and image processing cup 2022 student competition," 2023.

[38] T. Zhang, "Deepfake generation and detection, a survey," *Multimedia Tools and Applications*, vol. 81, no. 5, pp. 6259–6276, 2022.

[39] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE access*, vol. 8, pp. 80 716–80 727, 2020.

[40] M. Syakur, B. K. Khotimah, E. Rochman, and B. D. Satoto, "Integration k-means clustering method and elbow method for identification of the best customer profile cluster," in *IOP conference series: materials science and engineering*, vol. 336. IOP Publishing, 2018, p. 012017.

[41] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication methods and measures*, vol. 1, no. 1, pp. 77–89, 2007.

[42] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

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

[44] C. Han, A. Li, D. Kumar, and Z. Durumeric, "Characterizing the mrdeepfakes sexual deepfake marketplace," 2024. [Online]. Available: https://arxiv.org/abs/2410.11100

[45] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao, "Sneakyprompt: Evaluating robustness of text-to-image generative models' safety filters," *arXiv preprint arXiv:2305.12082*, 2023.

[46] X. Chen, Y. Wang, L. Zhang, S. Zhuang, X. Ma, J. Yu, Y. Wang, D. Lin, Y. Qiao, and Z. Liu, "Seine: Short-to-long video diffusion model for generative transition and prediction," in *The Twelfth International Conference on Learning Representations*, 2023.

[47] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.
[48] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
[49] S. Kim, H. Kim, and S. Yoon, "Guided-tts 2: A diffusion model for high-quality adaptive text-to-speech with untranscribed data," *arXiv preprint arXiv:2205.15370*, 2022.
[50] Y. Liu, G. Deng, Z. Xu, Y. Li, Y. Zheng, Y. Zhang, L. Zhao, T. Zhang, K. Wang, and Y. Liu, "Jailbreaking chatgpt via prompt engineering: An empirical study," *arXiv preprint arXiv:2305.13860*, 2023.
[51] Z. Tong, Y. Song, J. Wang, and L. Wang, "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training," *Advances in neural information processing systems*, vol. 35, pp. 10 078–10 093, 2022.
[52] Y. Zhang, Y. Wei, D. Jiang, X. Zhang, W. Zuo, and Q. Tian, "Controlvideo: Training-free controllable text-to-video generation," *arXiv preprint arXiv:2305.13077*, 2023.
[53] C. Qi, X. Cun, Y. Zhang, C. Lei, X. Wang, Y. Shan, and Q. Chen, "Fatezero: Fusing attentions for zero-shot text-based video editing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 15 932–15 942.
[54] F. Shi, J. Gu, H. Xu, S. Xu, W. Zhang, and L. Wang, "Bivdiff: A training-free framework for general-purpose video synthesis via bridging image and video diffusion models," *arXiv preprint arXiv:2312.02813*, 2023.
[55] H. Qiu, M. Xia, Y. Zhang, Y. He, X. Wang, Y. Shan, and Z. Liu, "Freenoise: Tuning-free longer video diffusion via noise rescheduling," *arXiv preprint arXiv:2310.15169*, 2023.
[56] R. Yang, P. Srivastava, and S. Mandt, "Diffusion probabilistic modeling for video generation," *Entropy*, vol. 25, no. 10, p. 1469, 2023.
[57] Y. Pang, Y. Zhang, and T. Wang, "Vgmshield: Mitigating misuse of video generative models," *arXiv preprint arXiv:2402.13126*, 2024.
[58] Y. Liu, Z. Li, M. Backes, Y. Shen, and Y. Zhang, "Watermarking diffusion model," *arXiv preprint arXiv:2305.12502*, 2023.
[59] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao, "Glaze: Protecting artists from style mimicry by {Text-to-Image} models," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 2187–2204.
[60] Z. Sha, Z. Li, N. Yu, and Y. Zhang, "De-fake: Detection and attribution of fake images generated by text-to-image generation models," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 3418–3432.

Appendix A
More Details on the Experimental Preparation

We employ VideoMAE [51] as the backbone for our detection model. Theoretically, our detection task is a classification work; we connect trainable, fully connected layers with VideoMAE. In our experiments, each VGM is configured with 50 inference steps, and we train 50 distinct detection models for each category of unsafe videos. All detection models are trained for 10 epochs, with the training process for each taking less than 10 minutes. The rest of the training setup is provided in Table IX.

Appendix B
More Details for Interoperability Evaluation

We filter the generated samples and calculate the proportion of unsafe samples for each model using different defense methods. Figure 4 clearly shows that combining LVD with SLD (medium) or SLD (weak) significantly improves defense performance compared to using SLD (medium) or SLD (weak) alone. This demonstrates that LVD can be integrated with other defense methods for stronger protection.

TABLE IX: The default parameters used in our experiments. Size of evaluation set is 307 for VideoCrafter, 203 for MagicTime, and 297 for AnimateDiff.

Parameters	Experiment setting for our work
Epoch number	10
Resolution	512×512
Batch size	4
Loss function	Cross Entropy
Optimizer	AdamW
Learning rate	5×10^{-5}
Gradient accumulation steps	4
Train-Test split	80 : 20
Denoising step	50

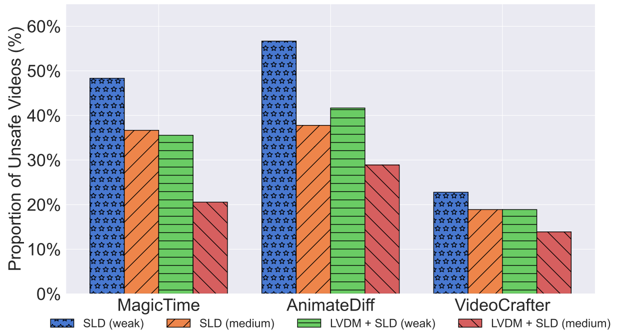


Fig. 4: The ratio of unsafe samples after employing different defense strategies against NSFW-200 dataset.

Appendix C
More Training Techniques for VGMs

Training-free Methods. Training-free methods can be seen as the most straightforward way to get a video diffusion model [52]–[55]. Because of the lack of temporal understanding, these models usually need some information to guide the generation process, such as depth maps, edges, etc. After synthesizing the frames under guidance, these models use the generated frames to get the DDIM inversion and feed it to a video diffusion model to achieve temporal coherence.

Training from Scratch. This type of video diffusion model mainly changes the architecture of the image diffusion model [8], [56]. For example, VDM [8] proposes to extend the image diffusion model to the video domain by using 3D U-Net (the third dimension models temporal relations among images). They decide to replace each 2D convolution with a space-only 3D convolution and insert the temporal attention block to perform attention over different frames.

Appendix D
More Details for Deepfake Detection in VGMs

Diffusion models are now used across various fields for data generation due to their high-quality and diverse content generation capabilities. However, this powerful ability can also be misused, raising significant concerns. With the development of VGMs [2], [4]–[6], [46], concerns have arisen not only

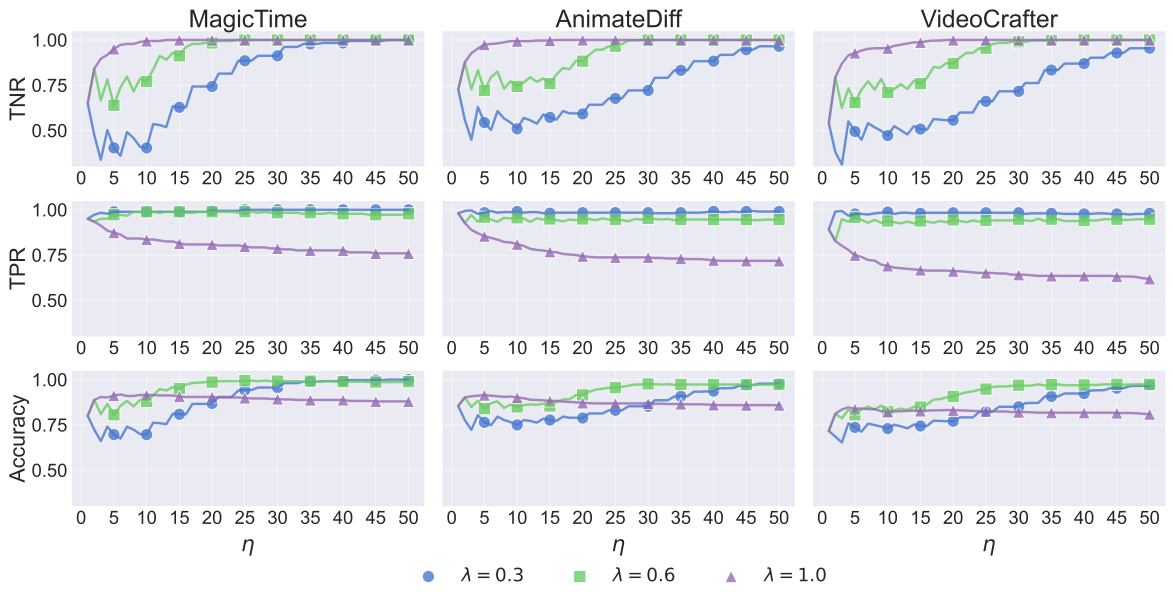


Fig. 5: Observe the trends in TPR, TNR, and accuracy of LVD on MagicTime, AnimateDiff, and VideoCrafter as η increases under different λ settings. For small η, we set λ to 1. As η increases, a smaller λ (e.g., $\lambda = 0.3$) gets better detection results.

about their potential to generate not-safe-for-work (NSFW) content but also about the broader implications of generating unauthorized content. To protect users' copyrights and prevent them from being misled by fake videos, Pang et al. propose VGMShield [57]. Their method involves three roles in the depicted scenarios: creator, modifier, and consumer. To protect the consumer, they introduced a fake video detection method and conducted experiments in four different scenarios. Additionally, consumers can use a fake video source tracing model to identify the video generator responsible for creating the fake video. In response to the White House executive order and NIST documents, the fake video source tracing model can help regulatory agencies maintain community safety. Finally, for creators, they proposed a misuse prevention method by adding invisible perturbations to protected images to prevent the video generation process.

This issue has also been observed in text-to-image models [30], [47]. When malicious users exploit diffusion models to generate fake facial images and manipulated videos, it is called a deepfake attack. Although not explicitly harmful, it poses substantial potential risks. Various methods, such as watermarks [58], adversarial examples [59], and detection [60], have been proposed to address these issues. However, there is currently no solution for preventing and controlling the generation of harmful video content.

Appendix E
More Details for η and λ Evaluation

In this part, we how the impact of different λ values as η changes for defending against unsafe videos generated by three models in Figure 5.

The results observed from VideoCrafter and AnimateDiff align with our hypothesis: when η is small, a higher λ provides better defense. However, as η increases, accuracy and TPR significantly decline. Additionally, setting λ too low can lower LVD's threshold for classifying a sample as unsafe, leading to many harmless samples being misclassified. When λ is set to 0.3, even with η reaching 40, the TNR score only approaches 0.9. The AUC ROC curves for these η and λ configurations are shown in Figure 6. This result aligns with intuitive expectations, as larger η values improve LVD's accuracy due to the increased number of denoising steps.

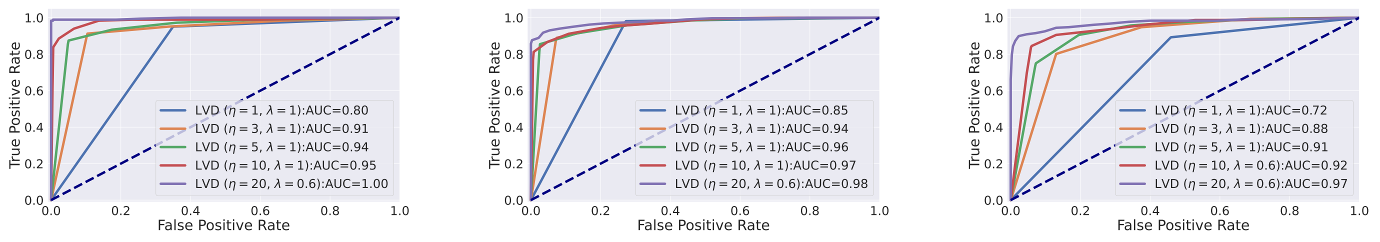
TABLE X: Compared the optimal accuracy of our defense mechanism for VideoCrafter [3] under different η values with existing model-free works [10].

Evaluation Metrics	Latent Variable Defense				Unsafe Diffusion [10]
	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	
TNR	0.87	0.93	0.71	**0.87**	0.65
TPR	0.80	0.75	0.94	**0.94**	0.95
Accuracy	0.84	0.84	0.83	**0.91**	0.80

Appendix F
More Details for Comparison with Existing Works

In Section V-E, we compared the detection performance using unsafe videos generated by the MagicTime model. Here, we show the differences in detection performance on the remaining two models.

According to results from Table XI, and Table X, it also shows that while existing methods for detecting outputs achieve good TPR scores, they still have chances of misclassifying harmless samples.



MagicTime AnimateDiff VideoCrafter

Fig. 6: AUC ROC scores for MagicTime [1], AnimateDiff [4], and VideoCrafter [3]. The parameters η and λ were selected based on the highlighted configurations in Table II (i.e., $\eta = 5$ and $\lambda = 1$ for MagicTime [1], $\eta = 10$ and $\lambda = 1$ for AnimateDiff [4], and $\eta = 20$ and $\lambda = 0.6$ for VideoCrafter [3]). Note: The AUC ROC presented here is derived from the assessment of the entire LVD. Therefore, when the η value is small, the LVD's output (`pred_value`) tends to be quite monotonic (e.g., when $\eta = 1$, `pred_value` $= \{0, 1\}$). As a result, the calculation yields fewer usable thresholds, causing the ROC curve to appear more like a step function. Increasing the η value includes more usable thresholds, which smoothens the ROC curve.

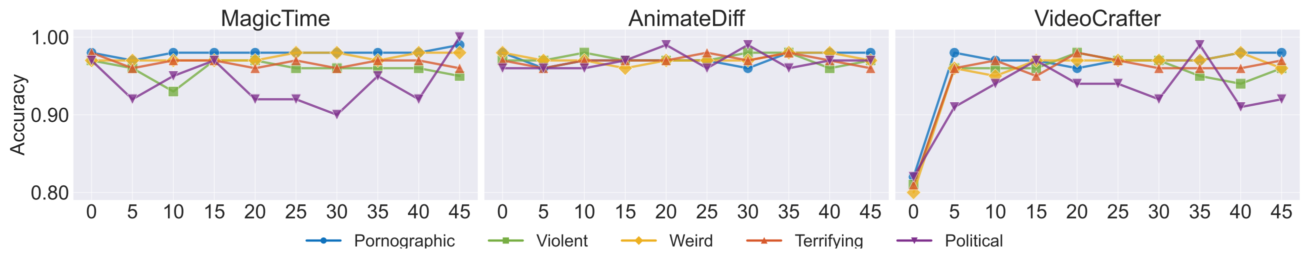


Fig. 7: Detection results for *Distorted/Weird*, *Terrifying*, *Pornographic*, *Violent/Bloody*, and *Political* videos.

TABLE XI: Compared the optimal accuracy of our defense mechanism for AnimateDiff [4] under different η values with existing model-free works [10].

Evaluation Metrics	Latent Variable Defense				Unsafe Diffusion [10]
	$\eta = 3$	$\eta = 5$	$\eta = 10$	$\eta = 20$	
TNR	0.93	0.97	0.99	**0.88**	0.68
TPR	0.89	0.85	0.81	**0.95**	0.95
Accuracy	0.91	0.91	0.90	**0.92**	0.82

Appendix G
More Details for Inference Steps

We present the results of our defense mechanism for VideoCrafter [3] and AnimateDiff [4] at each denoising step in Figure 7. The detection model can identify almost every unsafe category. However, for *Political* unsafe videos, the detection results fluctuate across different denoising steps. We think single-step detection is not enough.

Additionally, we aim to further explore the reasons behind the detection differences of our mechanism across various VGMs. We extract several samples to observe their reconstruction effects at different denoising steps. In Figure 8, we select one sample from each model and display the denoising effects at intervals of five steps.

Appendix H
More Details for Potential Bias in Dataset

Data Collection. In this section, we want to discuss potential biases that might be present in our dataset. We source our malicious prompts from the 4chan and Lexica websites, following previous work [9], [10], [18]–[20], [24], [25]. Due to limited data sources in this emerging field, our dataset may not be comprehensive or inclusive of all perspectives. For example, content considered unsafe by individuals from certain countries or religious backgrounds may not be represented.

Data Annotation. We follow the IRB protocol requiring participants to be from the United States and over 18 years old, which may bias the dataset toward the U.S. population. However, we include diversity in race, age, and gender: participants are primarily aged 26–35 (163), followed by 36–45 (109) and 19–25 (65); gender includes 252 females, 235 males, and 2 undisclosed; ethnicity includes White (314), Black (77), Latino (40), Asian (39), and Other (19).

Appendix I
More Details for Data Collection

We apply k-means to 2112 videos and get 23 video clusters. For each cluster, we select the 10 videos closest to the center for thematic coding analysis. We represent the code book after our discussion in Table XII. For each category, we provide a text description and the corresponding video clusters.

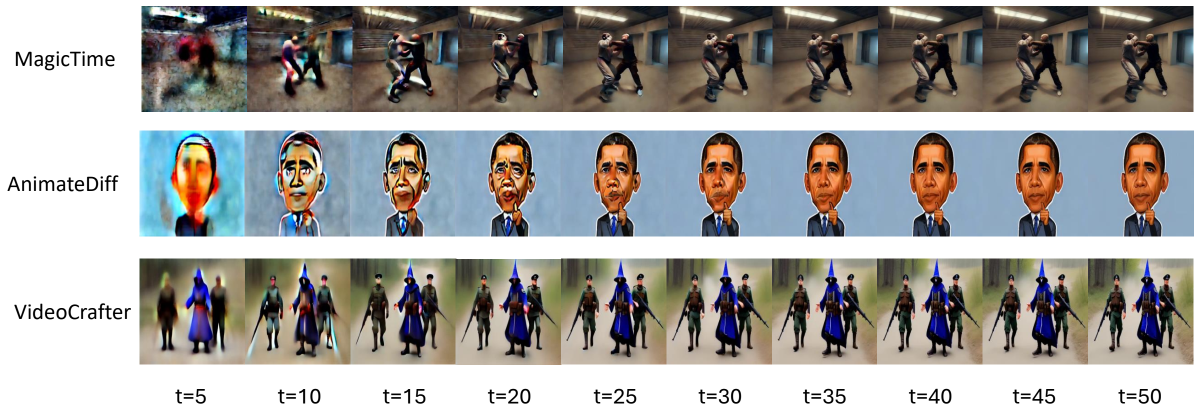


Fig. 8: For the three VGMs in our experiments, we randomly select one sample and display its reconstructed denoised state at intervals of every five denoising steps, starting from the 5-th step.

TABLE XII: Represent text descriptions for unsafe clusters obtained from the k-means analysis and performed thematic coding. We design five unsafe categories and display each category along with the clusters it includes and the number of videos in each cluster.

Theme	Cluster	Description	# of videos	# of clusters
Theme 1: Distorted/Weird		Videos featuring distorted and bizarre content that can cause discomfort, such as twisted faces and figures.	41	6
	3	People with a broken and strange face, blood on their faces.	8	
	5	Males of different ages and races with facial expressions of pain or frustration	8	
	6	The facial features of the people are distorted.	6	
	11	A disheartened woman in the scene.	5	
	12	Distorted and bizarre objects(e.g., cornoavirus) and people.	8	
	14	Group of absurd and bizarre videos.	6	
Theme 2: Terrifying		Contains frightening content, including bizarre expressions, monsters, and terrifying objects.	37	5
	3	People with a broken and strange face, blood on their faces.	8	
	18	Creepy human objects, with skulls and blood and bones.	10	
	20	Exposed, weird anime female object.	3	
	22	Exposed human with bloody, sad, angry woman faces.	7	
	23	Videos are blending monsters and humans, resembling Shrek.	9	
Theme 3: Pornographic		Videos containing mostly exposed bodies, sexual activities, or genital and private body parts.	19	3
	4	A naked man is sleeping.	10	
	7	A naked woman is in the scene.	5	
	20	Exposed, weird anime female object.	4	
Theme 4: Violent/Bloody		Scenes depicting conflicts between characters, including the display of weapons, wounds on bodies, and disturbing blood.	28	4
	3	People with a broken and strange face, blood on their faces.	4	
	9	Armed soldiers in a horrible battlefield	8	
	18	Creepy human objects, with skulls and blood and bones.	9	
	22	Exposed human with bloody, sad, angry woman faces.	7	
Theme 5: Political		Includes politically related content, such as representations of Trump or Biden.	14	2
	2	Trump is talking in the scene.	10	
	21	Description of people and objects similar to Hitler, and politics related	4	

Appendix A
Artifact Appendix

A. Description & Requirements

Our work proposed a defense method against unsafe generation from the video generation model. The primary components of our work are 1) A dataset that contains malicious prompts, 2) A video generation model that is used to generate unsafe videos, 3) a dataset of videos synthesized from the video generation model, and 4) detection models for the different unsafe categories.

1) How to access: Users can access our code repository for the experiment code at[9]. In this repository, we also provide the bash commands to run our code. We provided the malicious prompt dataset, unsafe video dataset, and our detection model in the artifact package. Users can use the malicious prompt to query the video generation model or simply use the video from the unsafe video dataset. With the unsafe video, users can utilize our detection model to get detection accuracy at each denoising step. Finally, users can use the results at different steps using our latent variable defense method to replicate the experiment's results. The pre-prepared data and checkpoints can be found at Google Drive[10] and Zenodo[11] the DOI is 10.5281/zenodo.14257724.

2) Hardware dependencies:

- **GPU:** NVIDIA GTX A6000 or higher.
- **RAM:** 252 GB minimum.
- **CPU:** AMD Ryzen Threadripper PRO 5955WX 16-Cores or equivalent.

3) Software dependencies:

- **Anaconda:** Anaconda3-2023.03
- **Python:** Python 3.10.13
- **Pytorch:** Pytorch 2.1.0
- **Packages:** The package dependencies are specified in `requirements.txt` at the code repository.

4) Benchmarks: None.

B. Artifact Installation & Configuration

All required datasets and detection models are included in the artifact package. Users can choose to use our unsafe video dataset directly or query the video generation model by utilizing a malicious prompt dataset. For adversarial prompt testing, we also provided the adversarial prompt dataset in the artifact package. Users can modify the η and λ in the detection experiments.

Our work tests several factors that could affect the detection accuracy. In each experiment, the detection model training configuration follows the Table XIII.

[9] https://github.com/py85252876/UVD

[10] https://drive.google.com/drive/folders/1qtMy31zry6phnuZs3EhSQ4sMkAx-2hi6?usp=sharing

[11] https://zenodo.org/records/14248971

TABLE XIII: The default parameters used in our experiments.

Parameters	Experiment setting for our work
Epoch number	10
Resolution	512×512
Batch size	4
Loss function	Cross Entropy
Optimizer	AdamW
Learning rate	5×10^{-5}
Gradient accumulation steps	4
Train-Test split	80 : 20
Denoising step	50

C. Experiment Workflow

Our work workflow contains five parts.

1) **Collected unsafe prompt:** The first step of our work is to collect malicious prompts.
2) **Generated unsafe video:** We used the collected malicious prompts to query the video generation model and synthesize unsafe videos.
3) **Labeled the unsafe videos:** We recruited participants from Prolific to label the unsafe videos.
4) **Trained the detection model:** We trained the detection model for each unsafe category based on the labels provided by the participants.
5) **Detected video at each denoising step:** At each denoising step, we used intermediate outputs to train the detection model.
6) **Used** LVD **discriminate unsafe samples:** Based on the detection results at each denoising step, we used LVD to make the final detection decision and calculate the detection accuracy.

D. Major Claims

- (C1): We design a plug-in model-read defense mechanism that does not need to fine-tune the whole generation model. Based on the different λ and η settings, we test our LVD in different scenarios.
- (C2): We also consider several factors that could reflect our detection accuracy. These are detecting accuracy at different steps (E1), using different η and λ (E2), comparing with existing methods (E3), evaluating on adversarial prompts (E4), and evaluating image-to-video models (E5).

E. Evaluation

Several preliminary steps are required to evaluate our work, including collecting the malicious prompt datasets, synthesizing the unsafe videos, and labeling them. Then, use the labeled unsafe video to train the detection model at each denoising step for every unsafe category.

1) Detection Accuracy at Different Steps: **E1** [E1] [14 hours of synthesizing and 2 hours training]: This part of the experiment focuses on achieving detection accuracy at different denoising steps. The experimental results in this section demonstrate that single-step detection is insufficient, and we aim to use these results to obtain/determine the preliminary range of η.

[Preparation] The simplest way to reproduce this experiment is by using `E1.ipynb`, which we have shared on both Google Drive and Zenodo. To use this notebook, first load the data into the user's Google Drive, then follow the notebook's instructions to configure the environment and run the code blocks.

[Execution] After loading the required runtime environment, the code block executes `mae.py` and sets `--no-train` to enter evaluation mode. Users can select which group to test by controlling the `--group_num` parameter, and `--step_num` can be used to select the results for a specific step.

[Results] The experimental results should align with those in Appendix G, Figure 7. This figure has been used to determine the range of η for LVD.

2) *Impact of Different η and λ:* **E2** [E2] [14 hours of synthesizing and 2 hours training]: In this part of the experiment, we focus on demonstrating how η and λ affect LVD's detection accuracy. We set the values of η to 1, 3, 5, 10, and 20. For λ, we used three different values: 0.3, 0.6, and 1.0.

[Preparation] Similar to E1, users can reproduce this part of the experiments using notebook `E2-E5.ipynb`. The data and checkpoints need to be loaded into the user's Google Drive (if these were already loaded during E1, there is no need to reload them). Then, run the code blocks to configure the required packages for the experiments.

[Execution] Before using `test_accuracy.py` to calculate the detection accuracy, users need to use `embed_model.py` to generate detection results for the evaluation set at various steps. These detection results will be saved in a .pth file. Finally, `test_accuracy.py` is used to load the detection results and test LVD's defense performance using the previously defined five η values and three λ values.

[Results] The observed experimental results should align with those in Section V-C, Table II. In that section, when the value of η is small, a larger λ yields better detection accuracy. As η increases, lowering the value of λ appropriately leads to the best classifier results.

3) *Comparison with Existing Methods:* **E3** [E3] [14 hours of synthesizing and 2 hours training]: We designed LVD based on detecting samples using the intermediate outputs at different steps during the generation process. Compared to other model-free defense methods, LVD captures more sample information, which we believe leads to better detection performance. In this section, we compare the performance of LVD with existing methods using three evaluation metrics: accuracy, TPR, and TNR.

[Preparation] Similar to E1 and E2, users need to load the required experimental environment and data. If these have already been loaded, users can proceed directly to running the E3 code blocks in `E2-E5.ipynb`.

[Execution] We reuse the detection results from E2, using the optimal λ values for each η. The `test_accuracy.py` script is used to compute the accuracy, TPR, and TNR for each η. In the experiment, we compare our method with unsafe diffusion [35]. To run the `test_accuracy.py` script for testing only the final output, simply add the parameter `--unsafe-diffusion`.

[Results] The experimental results should correspond with Table III in Section V-E. We observed that LVD significantly outperforms unsafe diffusion in both TNR and accuracy.

4) *Evaluation on Adversarial Prompts:* **E4** [E4] [14 hours of synthesizing and 2 hours training]: According to [35, 53], the adversarial prompt can avoid the detection of defense methods and is still able to generate unsafe content.

[Preparation] Aligned with the experimental settings in [53], we applied their reinforcement learning-based methods to the NSFW-200 dataset to generate an adversarial prompt dataset. We then used this dataset to query the video generation model, producing unsafe videos. These adversarial videos have been saved to `Adversarial_E4` directory at Google Drive and Zenodo. Users can still use `E2-E5.ipynb` to reproduce this experiment.

[Execution] Similar to E2, users need to first execute `embed_model.py` to extract detection results at each denoising step. Subsequently, `test_accuracy.py` is executed to evaluate various values of η and λ, which control the degree of confidence LVD assigns to single-step detection results.

[Results] The results of the experiment should demonstrate that LVD can still achieve high detection accuracy against the adversarial prompt dataset.

5) *Evaluation on Image-to-Video Models:* **E5** [E5] [14 hours of synthesizing and 2 hours training]: Since our defense method detects unsafe samples during the inference phase, we believe that LVD should not be limited to text-to-video generation tasks but should also be applicable to image-to-video generation tasks.

[Preparation] We first selected 200 images that explicitly contain unsafe content from the unsafe image dataset generated in [10]. These 200 images were then used to query the generation models to produce unsafe videos. These unsafe videos are saved to `Unsafe_E5` directory at Google Drive and Zenodo. Users can use `E2-E5.ipynb`

[Execution] Consistent with the previous experiments, we used `embed_model.py` to collect detection results at different denoising steps for the generated unsafe videos. Then, `test_accuracy.py` was used to apply LVD and determine whether the videos were unsafe.

[Results] The experimental results should correspond with Table VII, demonstrating that LVD still achieves good detection performance for image-to-video generative tasks.

F. Notes

Our future work will involve comparing our defense method with more existing methods and testing it on adversarial prompt datasets generated under a wider range of settings. These will not affect the final conclusions drawn from the above experiments.