ChopGrad: Pixel-Wise Losses for Latent Video Diffusion via Truncated Backpropagation

Dmitriy Rivkin[1], Parker Ewen[1], Lili Gao[1], Julian Ost[1,2], Stefanie Walz[1], Rasika Kangutkar[1], Mario Bijelic[1,2], Felix Heide[1,2]

[1]Torc Robotics, [2]Princeton University

Abstract

Recent video diffusion models achieve high-quality generation through recurrent frame processing where each frame generation depends on previous frames. However, this recurrent mechanism means that training such models in the pixel domain incurs prohibitive memory costs, as activations accumulate across the entire video sequence. This fundamental limitation also makes fine-tuning these models with pixel-wise losses computationally intractable for long or high-resolution videos. This paper introduces ChopGrad, a truncated backpropagation scheme for video decoding, limiting gradient computation to local frame windows while maintaining global consistency. We provide a theoretical analysis of this approximation and show that it enables efficient fine-tuning with frame-wise losses. ChopGrad reduces training memory from scaling linearly with the number of video frames (full backpropagation) to constant memory, and compares favorably to existing state-of-the-art video diffusion models across a suite of conditional video generation tasks with pixel-wise losses, including video super-resolution, video inpainting, video enhancement of neural-rendered scenes, and controlled driving video generation. Our project page is available at `https://light.princeton.edu/chopgrad`.

1. Introduction

Recent methods in latent video diffusion are capable of generating high-resolution videos over long time horizons [22, 51, 57, 66]. Similar to latent image diffusion models, latent video diffusion models rely on pre-trained autoencoders to compress videos into latent embeddings and then learn over these embeddings [4, 5, 19]. An enabling factor for recent video diffusion results is the use of temporal compression, where the autoencoder not only compresses video frames along spatial dimensions, but also along the temporal dimension [37, 76, 84].

Temporal compression groups multiple image frames into a single latent frame group. To incentivize temporal consistency between these frame groups causal caching has been introduced [63, 70, 74]. This technique appends embeddings from previous frame group encodings onto the beginning of subsequent frame groups at each layer of the video encoder and decoder. Notably, this approach introduces a recurrent structure into the autoencoder, where the dependency graph of video latents requires gradients to be propagated through all previous frame embeddings.

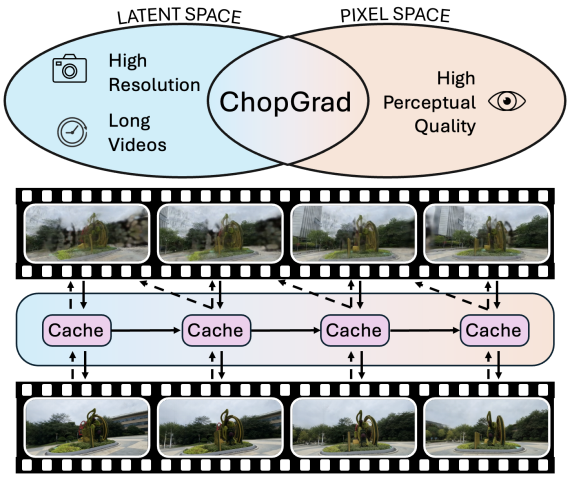

Figure 1. **ChopGrad Method**. *ChopGrad* unlocks pixel-wise losses for high resolution, long-duration video diffusion models. It leverages truncated backpropagation to eliminate recursive activation accumulation in video autoencoders with causal caching. Solid arrows indicate the flow of information in the decoder forward pass, dashed ones indicate the backward flow of gradients with *ChopGrad*. Adding *ChopGrad* to training procedures is easy and produces state of the art performance in a variety of applications that benefit from pixel-wise losses, such as video super-resolution, video inpainting, video enhancement of neural rendered scenes, and controlled driving video generation.

At the same time, most successful latent video diffusion models are trained within the latent space [2, 22, 51, 70], meaning gradients are not propagated through the encoder

or decoder during latent video diffusion training. As such, existing methods make *pixel-wise losses intractable* for long-duration videos as the gradients of these losses require the recurrent accumulation of activations through the decoder. These pixel-level perceptual losses are used extensively in finetuning image diffusion models and video models with *short-duration*, low-resolution videos in applications such as single-step model distillation [73], enhancement of neural rendered scenes [11, 61], image translation [41], video super-resolution [12], and controlled driving video generation [34, 55]. In work such as [34, 41, 55, 61], the decoder itself is finetuned, making support for pixel-wise losses a strict requirement for training these types of models.

To enable pixel-wise losses for high-resolution, long duration video diffusion, this work introduces *ChopGrad*, a truncated backpropagation scheme for video decoding (Fig. 1). Truncated backpropagation prevents activation accumulation over the full unrolled network by limiting the number of previous frames the gradients can propagate through. To validate this, we define latent temporal locality to demonstrate that the effect of prior video frames in the gradient error drops off at an exponential rate. We show that the proposed method enables efficient training using pixel-wise losses, such as the LPIPS [78] loss, across a variety of tasks and multiple video diffusion models. We evaluate our method on several applications, including video super-resolution, video inpainting, video enhancement of neural rendered scenes, and controlled driving video generation, outperforming existing latent video diffusion adaptation methods in terms of quantitative frame-wise and video performance metrics. These results are achieved with modest computational resources (training times of approximately 3 to 4 hours on 4 to 8 A100 GPUs). The contributions of this paper are:

- A mathematical derivation and error analysis of truncated backpropagation for causal video autoencoders,
- A memory-efficient, practical approach for implementing pixel-wise losses for fine-tuning latent video diffusion models that generalizes across multiple diffusion models,
- Validation of the method across several tasks requiring pixel-wise losses, including video super-resolution, video inpainting, video enhancement of neural-rendered scenes, and controlled driving video generation, comparing favorably to existing baselines in all experiments.

2. Related Work

Latent video diffusion has experienced rapid advancement in recent years thanks in part to novel video auto-encoding methods [5, 8, 37, 66]. In particular, temporal compression and causal caching have demonstrated significant improvements in video quality and temporal consistency.

Latent video diffusion models extend latent image diffusion methods to model temporally coherent video sequences by operating in a compressed latent space rather than pixel space [4, 25, 66, 75]. Operating in a latent space [3, 31, 47, 57] reduces per-frame dimensionality and enables tractable scaling to longer and higher-resolution clips while preserving perceptual fidelity [22, 24]. Early video diffusion formulations applied standard image-based diffusion techniques directly to short clips, jointly denoising fixed-length frame blocks and introducing conditioning strategies to extend temporal length [2, 13, 25, 47].

One of the most prevalent architectural advancements powering latent video diffusion is the use of temporal compression [9, 14, 18, 19, 80, 84] and causal caching to preserve latent integrity and temporal consistency when processing long sequences [2, 17, 74]. Causal caching has been used to maintain reconstruction fidelity and avoid temporal flicker while dramatically reducing memory and latency during encoding/decoding [32, 63]. Unfortunately, this causal caching mechanism for video encoding introduces a recurrent structure into the encoders and decoders used by latent video diffusion models, resulting in prohibitive memory consumption due to activation accumulation during training when pixel-wise losses are used.

A similar problem was encountered in early natural language processing with recurrent neural networks [42–44], where truncated backpropagation through time was used to mitigate this issue [1, 60]. To the best of our knowledge, this paradigm has not been investigated or applied for image or video models.

Diffusion models often require long inference times, as the model must be run many times to generate an output. Single and few-step distillation [6, 26, 36, 39, 45, 52, 69, 73] has been used to reduce the number of steps required. Single-step distillation has also been used to adapt diffusion models to image-to-image translation tasks like changing weather or generating images from sketches [10, 30, 41]. In applications where input/output pairs are readily available (such as super-resolution [12, 21, 53, 58] or 3D gaussian splatting post-processing [16, 34, 55, 61]), pretrained diffusion models [12, 16, 50, 55] or their one-step distilled counterparts [34, 61] have been finetuned for single-step inference on the given task. Many of these single-step distillation and finetuning approaches rely on pixel-wise perceptual losses, albeit at low resolution and video duration in the case of video models due to memory constraints. As such, these single-step diffusion applications can derive the most benefit from *ChopGrad*.

2.1. Preliminaries

Latent video diffusion models work by first mapping from the high-dimensional pixel space to a lower-dimensional latent space, down-sampling both the spatial and temporal dimensions via a pre-trained 3D VAE video encoder [3, 70].

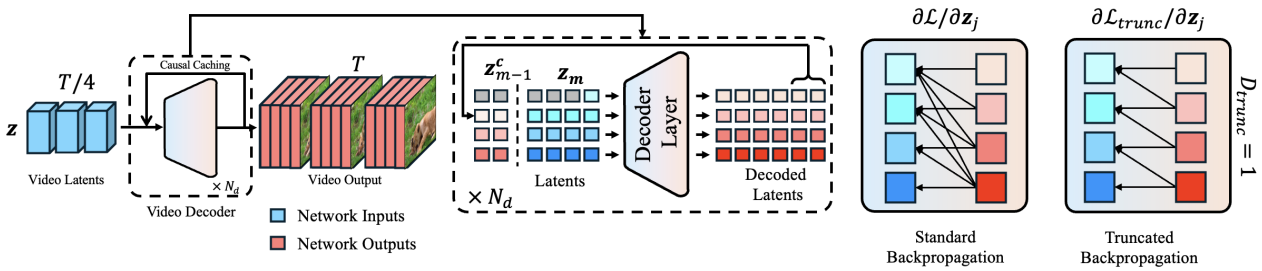

Figure 2. **ChopGrad Model Architecture**. Given the processed video frame latents, the video decoder iteratively applies causal caching at each layer, producing pixel outputs. Caching is performed by taking a subset of the layer outputs and appending these to the beginning of the layer inputs for the next frame group. While substantially reducing memory use at inference time compared to full 3D convolution over all frame groups, during training this process introduces recursive activation accumulation in the decoder, making backpropagation prohibitively expensive for high-resolution or long videos when using pixel-wise losses. Using truncated backpropagation, we only allow gradients to accumulate through a fixed number (D_{trunc}) of previous frame groups.

Once encoded, the video embeddings are then processed by the network backbone, often a transformer, which learns the temporal evolution of the video embeddings. Finally, the output embeddings are re-projected into pixel space via the pre-trained 3D VAE decoder.

The structure of such 3D VAE networks groups a set of frames into a single latent embedding. To retain temporal consistency these networks use what is called causal logic padding [63] or causal caching [32], where the trailing N outputs from the previous frame group are concatenated to the beginning of the subsequent frame group at each layer of the encoder and decoder [70, 74]. This results in a recurrent structure, where the gradients of pixel-wise losses on later frames propagate through all previous frame groups.

When training 3D VAEs, computational resources are dedicated solely to the VAE, and approaches such as sequence parallelism can be used to mitigate these issues, as described in [70]. In addition, 3D VAEs are also able to be trained at lower resolutions/durations with results generalizing to higher resolution/duration videos with no additional fine-tuning [70]. However, when training or fine-tuning latent video diffusion model transformers or U-nets, the majority of the memory budget is consumed by these backbones, prohibiting the allocation of significant memory resources to decoder backpropagation. The backbones must also be trained at high resolution/duration if they are to perform well for high-resolution/duration inference, further compounding these memory requirements, especially as adding pixel-wise losses also requires the decoders to perform inference at high resolution/duration, even if their own parameters are frozen.

3. ChopGrad

In order to enable training of video diffusion models on long, high-resolution videos with pixel-wise losses while maintaining modest memory requirements we present *ChopGrad*, a novel method for backpropagating through the video decoder. Sections 3.1 and 3.2 report that popular pre-trained video autoencoders with causal caching demonstrate temporal locality, where frame groups only affect other frame groups in close temporal proximity. Motivated by this insight, *ChopGrad* applies truncated backpropagation through time to the decoder cache to increase computational efficiency with minimal degradation in performance. With truncated backpropagation, gradients of each frame group are only able to accumulate to a portion of prior frame groups set by the truncation distance. This breaks the recursive loop present in popular video autoencoders and enables pixel-wise losses for long, high-resolution videos. In Section 3.3 we quantify temporal locality and truncation gradient error in the Wan2.1 decoder and transformer. Implementation details are provided in the Appendix.

3.1. Causal Caching in Temporal VAEs

The temporal VAE architecture with causal masking is first formalized. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ denote a video sequence of T frames, where each frame $\mathbf{x}_t \in \mathbb{R}^{H \times W \times C}$ has height H, width W, and C channels.

The 3D VAE encoder groups consecutive frames into non-overlapping segments. For a frame group of size G, the i-th frame group contains frames $\mathbf{X}_i = \{\mathbf{x}_{iG}, \mathbf{x}_{iG+1}, \ldots, \mathbf{x}_{iG+G-1}\}$ for $i = 0, 1, 2, \ldots, \lceil T/G \rceil$.

Let $\mathbf{z}_{i,m} \in \mathbb{R}^{d_m \times T' \times W' \times H'}$ be the video latent embedding of frame group i at encoder layer m, where H', W' are the down-sampled spatial dimensions, T' is the down-sampled temporal dimension, and d_m is the latent dimension for layer m.

The causal caching mechanism ensures that the decoder ($\mathcal{D}$) for frame group i receives context from the previous group. Specifically, let $\mathbf{z}^c_{i-1,m}$ denote the causal cache of size N of decoded features from group $i-1$ for the decoder layer m. The decoder then reconstructs the frames and constructs the cache

$$\mathbf{z}_{i,m+1}, \mathbf{z}^c_{i,m} = \mathcal{D}_m(\text{Concat}(\mathbf{z}^c_{i-1,m}, \mathbf{z}_{i,m})). \quad (1)$$

The causal structure creates a recurrent dependency

where the pixel-wise loss $\mathcal{L}_i^{\text{pix}}$ for group i depends on all previous groups through the concatenated context $\mathbf{z}_{i-1}^c$ at each decoder layer.

3.2. Truncated Backpropagation and Locality

Truncated backpropagation leverages temporal locality to enable efficient training while preserving the essential temporal dependencies. The following analysis focuses on causal caching within the decoder network.

Let $\mathbf{z}_i \in \mathbb{R}^d$ denote the unrolled latent, where the layer indices m are omitted for notational convenience. Let $D(i,j)$ be a distance metric such that $D(i,j) = 0$ if and only if i and j refer to latents belonging to the same frame group. This index-based distance formalism allows us to reason about temporal proximity and the influence of one latent on another.

Let $J_{i,j} = \partial \mathbf{z}_i / \partial \mathbf{z}_j \in \mathbb{R}^{d \times d}$ denote the Jacobian of latent i with respect to latent j. The scalar influence measure is then defined as

$$L_{i \leftarrow j} := \|J_{i,j}\|, \tag{2}$$

for a chosen matrix norm. This quantity captures the effect of latent j on latent i and is a vector-norm on a vector space.

Temporal locality is defined as the existence of constants $C, \alpha > 0$ such that the influence measure decays exponentially with distance

$$L_{i \leftarrow j} \leq C \cdot \exp(-\alpha D(i,j)). \tag{3}$$

Intuitively, this means that a latent only meaningfully affects nearby latents in time. Using the chain rule, the gradient of the overall loss $\mathcal{L}$ with respect to a latent $\mathbf{z}_i$ decomposes as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_j} = \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i} \frac{\partial \mathbf{z}_i}{\partial \mathbf{z}_j} = \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i} J_{i,j}. \tag{4}$$

Taking the norm of both sides and applying the triangle inequality,

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{z}_j} \right\| \leq \sum_i \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i} \right\| L_{i \leftarrow j}, \tag{5}$$

which shows that the loss gradient at $\mathbf{z}_i$ is dominated by contributions from latents in close temporal proximity assuming temporal locality holds. Our key insight is that the temporal locality enables effective truncated backpropagation in the 3D VAE decoder. When we truncate gradients to only flow through a limited number of previous frame groups, the exponential decay in the influence measure ensures that the approximation error is bounded.

Specifically, for truncated backpropagation at temporal distance D_{trunc}, the error in gradient computation is bounded by

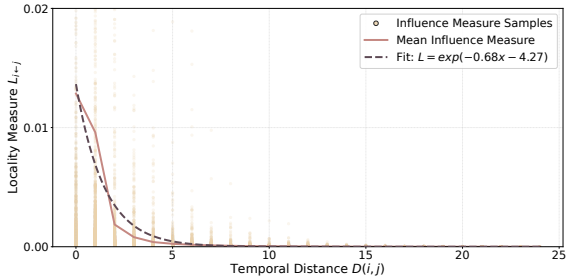

Figure 3. **Temporal Locality**. Influence measure samples (2) as a function of temporal distance between decoder inputs (i.e. latent embeddings) and outputs (i.e. pixels) alongside the mean and line of best fit. As temporal distance increases, the influence between embeddings decreases exponentially, resulting in minimal gradient contributions (5).

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{z}_j} - \frac{\partial \mathcal{L}_{\text{trunc}}}{\partial \mathbf{z}_j} \right\| \leq C \cdot \exp(-\alpha D_{\text{trunc}}) \sum_i \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i} \right\|, \tag{6}$$

where $\mathcal{L}_{\text{trunc}}$ denotes the loss computed with truncated backpropagation.

A truncation distance $D_{\text{trunc}} \geq \frac{1}{\alpha} \log(\frac{C}{\epsilon})$ can therefore be chosen to satisfy a desired error tolerance ϵ. In practice, the network still learns effectively with a small truncation distance as shown in Sections 3.3 and 4.

The integration of causal caching with truncated backpropagation creates a hybrid approach: the network backbone can still attend to all video latent embeddings for global temporal understanding, while the 3D VAE decoder operates with limited temporal context, reducing computational complexity. This design preserves essential temporal dependencies while making large-scale video diffusion model training using pixel-wise losses computationally tractable.

3.3. Analysis

Temporal Locality. We analyze the proposed method by first confirming that temporal locality holds in the popular WAN 2.1 video decoder [51]. The locality measure (3) is averaged across several videos, each with 97 frames and down-sampled to a resolution of 64×128 to prevent prohibitive memory requirements. Fig. 3 reports the mean of the influence measure (2) as a function of temporal distance, where a distance of 0 indicates pixel i is in the frame group of latent j. Notably, the locality measure decays at an exponential rate, meaning the influence of pixels on frame groups significantly decreases as the temporal distance increases. This property is demonstrated implicitly for other 3D VAEs by the results presented in Section 4.

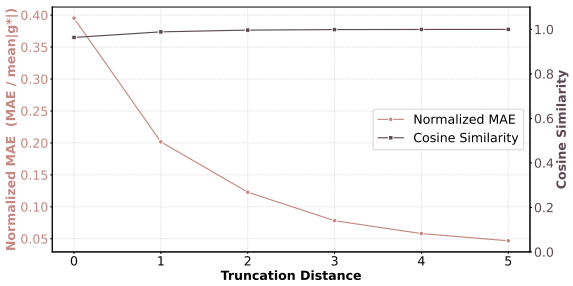

Figure 4. **Impact of Truncation Distance on Backbone Model Parameter Gradients**. Normalized MAE and cosine distance (computed by flattening all model parameters) are shown. Though error is significant at small truncation distances, the cosine similarity remains high across all distances, implying that the errors are primarily of magnitude, not direction.

Decoder Input Gradient Error. We likewise present the gradient error (6) between the full and truncated backpropagation algorithms as a function of truncation distance. Gradients are computed by backpropagating pixel-wise losses to each decoder input latent considering varying truncation distances. Reported results are the absolute and relative difference between the gradients for the truncated distance and the full backpropagation scheme. Differences are measured using the Frobenius matrix norm and these, along with relative differences, are presented in Fig. 5. From this plot we see that, even for low truncation distances, gradients approach those of full backpropagation, confirming that truncated backpropagation can be applied with minimal degradation in temporal consistency as the decoder network only considers small temporal neighborhoods.

Effect on Backbone Model Parameters. Next, we evaluate the effect of gradient truncation on the backbone model parameters during training by computing the average gradient of the parameters of the public Wan 2.1 1.3B transformer checkpoint over the entire training set of the DL3DV-benchmark dataset (see Section 4.2), around 100 videos. We perform this computation over a range of truncation distances and compare to the gradients of the full backwards pass, with results presented in Fig. 4. Reported is the normalized mean absolute error (MAE) and cosine similarity, computed by flattening all model parameters into a single vector. The error is large for small truncation distances, indicating that the errors introduced by truncation are not averaged out over the dataset, and are propagated to model parameters. However, the high cosine similarity indicates that the error is primarily one of magnitude, not direction, and since gradient magnitudes are scaled by optimizers, the impact on training is negligible. This is confirmed by the results in Table 2, where increasing truncation distance only modestly improves performance.

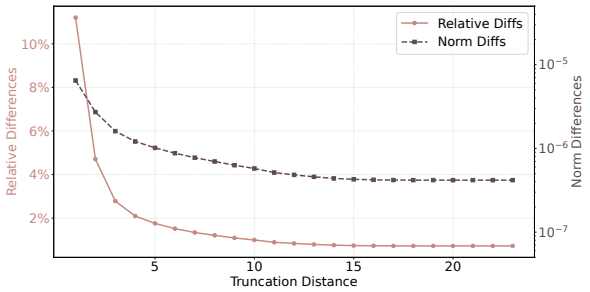

Figure 5. **Truncation Induced Gradient Error.** Mean gradient error (6) between the truncated and full backpropagation algorithms as a function of truncation distance.

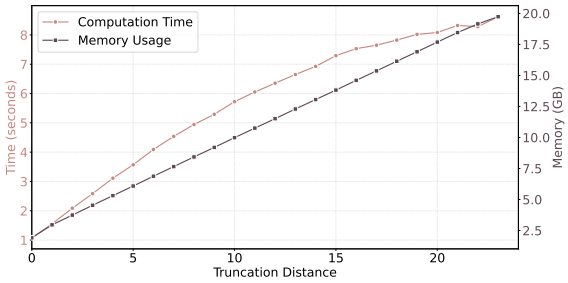

Figure 6. **Resource Utilization.** Computational time and memory requirements as a function of truncation distance.

Figure 7. **Spatial Locality in 3D VAEs**. The video frame on the left is decoded from the original latents, while on the right a section of latents is zeroed. The red line indicates the boundary between original and zeroed latents. The upper portion of the frame is entirely unaffected by the corruption of the bottom.

Runtime and Memory. Fig. 6 confirms that the proposed approach scales linearly with respect to truncation distance in terms of both computational time and memory. We reiterate that memory use is constant with respect to video length. To further save on memory, gradients are truncated spatially as well as temporally, such that gradients are computed over spatial chunks of the video separately. This spatial locality is illustrated in Fig. 7 and has been explored and leveraged by existing state-of-the-art video diffusion models [51, 70].

4. Applications

We validate the efficacy of *ChopGrad* in four applications across multiple diffusion models: video super-resolution (Sec. 4.1), novel view synthesis (Sec. 4.2), video inpainting (Sec. 4.3), and controlled driving video generation (Sec. 4.4).

4.1. Video Super-Resolution

We first show that adding *ChopGrad* to a state-of-the-art video super-resolution method yields significant improvements in perceptual losses by finetuning DOVE [12] using *ChopGrad*. DOVE finetunes CogVideoX [70], a DiT (Diffusion Transformer) model, for super-resolution. DOVE uses pixel-wise losses, including MSE and DISTS [15], but is forced to encode and decode each video frame separately during loss computation due to memory constraints, reducing inter-frame consistency and requiring the addition of a frame consistency loss to attempt to compensate for this. In contrast, for *ChopGrad*, we start with the publicly available DOVE checkpoint and perform full finetuning on the HQ-VSR dataset [12] for 500 steps using video lengths of 24 frames, omitting interframe consistency losses. We use frame-wise DISTS loss with a weight of 0.1 and pixel-wise MSE with a weight of 1. All other settings are consistent with the original DOVE Stage-2 implementation, except that in DOVE 80% of the batches are images, not videos, while we train on videos only. For the DOVE baseline, the publicly available DOVE checkpoint is used. As we found additional fine-tuning using the original DOVE method to result in equivalent performance, the results for the original model are presented.

Quantitative results for video super-resolution are presented in Table 1. The addition of the proposed truncated backpropagation scheme improves performance across the majority of datasets and metrics, and the improvements are more pronounced for perceptual metrics (LPIPS and DISTS). Selected frames from processed videos are shown in Fig. 8, where *ChopGrad* synthesizes fine-grained details such as fur, hair, and clouds better than the baseline approach.

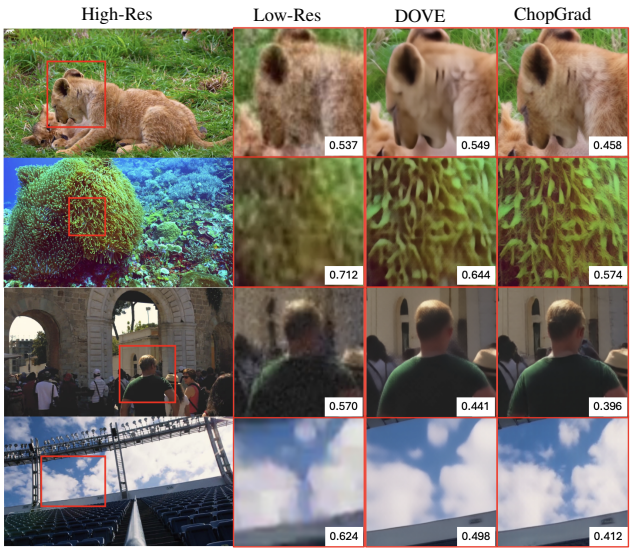

Figure 8. **Video Super-Resolution Comparison**. Shown from left to right: high-resolution, low-resolution input, DOVE [12], and the proposed approach, *ChopGrad*. *ChopGrad* synthesizes fine textures better and reduces motion blur, especially in regions with high-frequency details like fur, hair, cloth, and clouds. LPIPS scores for each frame are shown in the bottom right-hand corner, where a lower score indicates better perceptual quality. The associated videos can be found in the Appendix.

4.2. Artifact Removal in Novel View Synthesis

Next, we use *ChopGrad* for refining renders from imperfect neural rendering models [29, 38], which has recently become an established task [11, 61]. Renders of 3D Gaussian Splatting novel view synthesis methods [29] often contain artifacts such as "floaters" that a set of recent diffusion models mitigate. Specifically, MVSplat-360 [11] and Difix3D+ [61] are designed for this task. MVSplat-360 is trained to refine video sequences of 14 frames rendered from 3DGS models while Difix is trained to refine individual frames. As a result, MVSplat-360 operates at a lower resolution (448×256) with a small window of temporal consistency while Difix operates at a higher resolution (960×544) but has no capacity to enforce temporal consistency. While MVSplat-360 and Difix both leverage pixel-wise losses, they are unable to scale to long and high-resolution videos.

We generate a dataset using the DL3DV-Benchmark [33], a collection of 140 videos and camera trajectories. Gaussian splat models are generated using every 50th frame of each video and rendered videos are constructed along entire camera trajectories. For *ChopGrad*, we initialize the video diffusion model from a pre-trained Wan 2.1 14B [51] model and fine-tune the transformer backbone for 10 epochs. Difix is fine-tuned for 10000 steps on the same data. As MVSplat-360 is trained on the DL3DV dataset, no finetuning is applied. We found that using the MVSplat-360 refinement model on our rendered videos led to poor performance. Performance was significantly improved using the same number of sparse views for constructing the 3DGS model when using the views specified in the MVSplat-360 repository. As such, we opt to use these improved selections for computing MVSplat-360 metrics.

Fig. 9 depicts *ChopGrad* alongside the baseline methods for several scenes from the DL3DV-Benchmark test set and Table 2 presents quantitative results. *ChopGrad* outperforms the baselines across all metrics except temporal flickering where results are competitive with MVSplat-360. A user study, available in the Appendix, also found that 95.6% *of users preferred the videos generated by Chop-*

Table 1. **Quantitative Comparison for Video Super-Resolution**. The first, second, and third best results are highlighted with dark green, light green, and yellow, respectively. *ChopGrad* outperforms all baselines in the majority of metrics and datasets, and achieves competitive performance otherwise.

Dataset	Metrics	RealESRGAN [56]	ResShift [77]	RealBasicVSR [7]	Upscale-A-Video [82]	MGLD-VSR [67]	VEnhancer [20]	STAR [65]	DOVE [12]	*ChopGrad* (Ours)
UDM10	PSNR (↑)	24.04	23.65	24.13	21.72	24.23	21.32	23.47	26.48	26.70
	SSIM (↑)	0.7107	0.6016	0.6801	0.5913	0.6957	0.6811	0.6804	0.7827	0.7753
	LPIPS (↓)	0.3877	0.5537	0.3908	0.4116	0.3272	0.4344	0.4242	0.2696	0.2346
	DISTS (↓)	0.2184	0.2898	0.2067	0.2230	0.1677	0.2310	0.2156	0.1492	0.1143
SPMCS	PSNR (↑)	21.22	21.68	22.17	18.81	22.39	18.58	21.24	23.11	23.67
	SSIM (↑)	0.5613	0.5153	0.5638	0.4113	0.5896	0.4850	0.5441	0.6210	0.6274
	LPIPS (↓)	0.3721	0.4467	0.3662	0.4468	0.3263	0.5358	0.5257	0.2888	0.2647
	DISTS (↓)	0.2220	0.2697	0.2164	0.2452	0.1960	0.2669	0.2872	0.1713	0.1448
YouHQ40	PSNR (↑)	22.82	23.32	22.39	19.62	23.17	19.78	22.64	24.30	24.58
	SSIM (↑)	0.6337	0.6273	0.5895	0.4824	0.6194	0.5911	0.6323	0.6740	0.6760
	LPIPS (↓)	0.3571	0.4211	0.4091	0.4268	0.3608	0.4742	0.4600	0.2997	0.2581
	DISTS (↓)	0.1790	0.2159	0.1933	0.2012	0.1685	0.2140	0.2287	0.1477	0.1079
RealVSR	PSNR (↑)	20.85	20.81	22.12	20.29	22.02	15.75	17.43	22.32	22.43
	SSIM (↑)	0.7105	0.6277	0.7163	0.5945	0.6774	0.4002	0.5215	0.7301	0.7193
	LPIPS (↓)	0.2016	0.2312	0.1870	0.2671	0.2182	0.3784	0.2943	0.1851	0.1934
	DISTS (↓)	0.1279	0.1435	0.0983	0.1425	0.1169	0.1688	0.1599	0.0978	0.0944
MVSR4x	PSNR (↑)	22.47	21.58	21.80	20.42	22.77	20.50	22.42	22.42	22.55
	SSIM (↑)	0.7412	0.6473	0.7045	0.6117	0.7418	0.7117	0.7421	0.7523	0.7550
	LPIPS (↓)	0.4534	0.5945	0.4235	0.4717	0.3568	0.4471	0.4311	0.3476	0.3212
	DISTS (↓)	0.3021	0.3351	0.2498	0.2673	0.2245	0.2800	0.2714	0.2363	0.2071

Table 2. **Neural Novel View Synthesis Results**. **Top section:** *ChopGrad* out-performs all baselines across all metrics except temporal flickering, where it achieves competitive performance with MVSplat-360 [11]. Interestingly, while increasing the truncation distance noticeably increases training time memory, the metric differences are minimal. **Bottom section:** Ablation Results for *ChopGrad*. *ChopGrad** uses the same 1-step diffusion network, but is only trained using latent mean-squared error. *ChopGrad*† likewise uses latent mean-squared error for training but is trained twice as long. As such, both ablations do not propagate gradients through the video decoder. The performance of *ChopGrad* using various truncation distances is also presented.

Method	FID (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Dists (↓)	VBench Overall Quality (↑)	VBench Temporal Flickering (↑)	Inference Time [s/frame]	Train Time [H]
Difix [61]	16.637	17.213	0.561	0.407	0.122	0.766	0.898	0.37	2.0
MVSplat-360 [11]	38.203	15.502	0.492	0.532	0.231	0.743	0.926	2.89	-
ChopGrad	11.209	19.237	0.610	0.342	0.113	0.783	0.921	1.11	4.0
ChopGrad*	48.525	19.501	0.588	0.440	0.244	0.753	0.933	1.11	2.3
ChopGrad†	48.173	19.401	0.586	0.439	0.238	0.751	0.932	1.11	4.5
ChopGrad $D_{trunc} = 0$	11.775	19.231	0.605	0.345	0.115	0.782	0.920	1.11	3.5
ChopGrad $D_{trunc} = 1$	11.209	19.237	0.610	0.342	0.113	0.783	0.921	1.11	4.0
ChopGrad $D_{trunc} = 2$	11.742	19.308	0.609	0.343	0.115	0.782	0.922	1.11	4.5

Grad over those generated by MVSplat-360 or Difix. Notably, while MVSplat-360 requires 60K training iterations [11], *ChopGrad* requires a small number of fine-tuning iterations when starting with the WAN2.1 14B pre-trained model. This demonstrates that *ChopGrad* enables diffusion models to quickly generalize to unseen tasks by fine-tuning using pixel-space losses.

To demonstrate that the performance gains are a result of pixel-wise losses enabled by ChopGrad and not simply a more powerful backbone, we report ablation experiments in Table 2 (bottom section) and a qualitative comparison in Fig. 10, where *ChopGrad* is trained using only MSE loss in the latent space and using various truncation distances. While training only on the video latents is faster, the perceptual quality is worse and blurring is prevalent, especially in regions with fine details. As discussed in Section 3.3, truncation distance has a minor impact on result quality. Videos of the DL3DV-Benchmark for *ChopGrad* and baselines can be found in the Appendix.

4.3. Video Inpainting

We demonstrate that in video inpainting applications, *ChopGrad* allows for reducing inference time by $50\times$ while remaining on-par in terms of quality. We evaluate *ChopGrad* for video inpainting on three datasets: DL3DV-Benchmark [33], Waymo Open Dataset [48], and ROVI [62]. For DL3DV-Benchmark and Waymo, we mask a fixed central region covering half the height and width of each frame and use an uninformative prompt. With ROVI, we use the included object masks and text descriptions. For *ChopGrad*

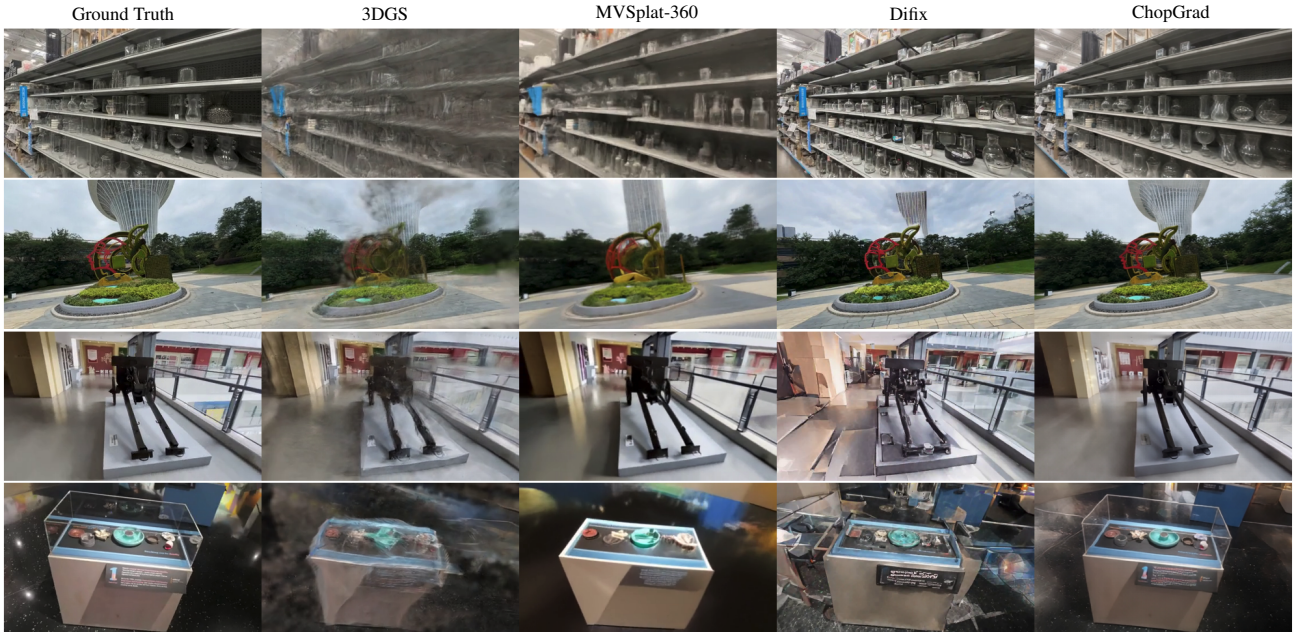

Figure 9. ***ChopGrad* vs Baselines for Neural Novel View Synthesis.** Ground truth video frames and 3D Gaussian Splat renders are shown on the left. Results for MVSplat-360 [11] and Difix [61] are presented alongside *ChopGrad*.

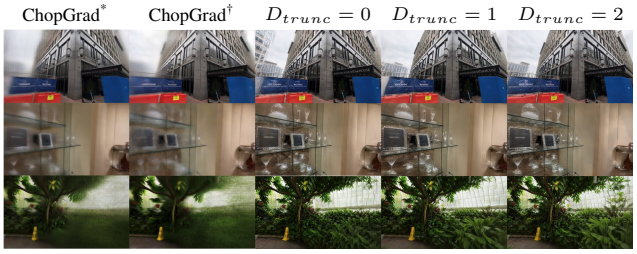

Figure 10. **Ablation Experiments for Neural Novel View Synthesis.** ChopGrad* and ChopGrad† are trained using only the MSE loss in the latent space. The D_{trunc} cases show *ChopGrad* results at various truncation distances.

we finetune a Wan 2.1 14B model using latent MSE and pixel LPIPS losses for single-step inference using a truncation distance of 1. The baseline is VACE [28] 14B, a control adapter for Wan 2.1 14B which is trained for a variety of tasks, including inpainting. VACE inference is performed using the default 50 steps from the VACE repository [28]. For all datasets, we train both *ChopGrad* and VACE the same number of steps. More training details are available in the Appendix.

Quantitative results are reported in Table 3, qualitative results in Fig. 11. *ChopGrad* outperforms VACE on reconstruction-based metrics and maintains similar video quality metrics (VBench overall quality score within 1% across all datasets) while reducing inference time compute budget by $50\times$. FVD (Fréchet Video Distance) is higher for *ChopGrad* on ROVI but lower for the other two datasets, likely stemming from the overall more extreme masking in D3LDV and ROVI. Qualitatively, we observe that the *ChopGrad* model adheres better to the scene and introduces fewer novel structures compared to VACE, occasionally at the cost of visual quality. In the more extreme masking regime of DL3DV and Waymo, VACE is penalized less for novel structures (relative to *ChopGrad*), as the unmasked region is less informative about the region inside the mask, resulting in smaller relative improvements in reconstruction-based losses.

4.4. Controlled Driving Video Generation

Visually realistic controlled driving video generation is essential for autonomous vehicle safety as it enables validation of vehicle behavior in rarely encountered scenarios. 3DGS [29] offers powerful scene reconstruction approaches, and recent neural driving simulators allow for manipulation of vehicles and reconstructed assets using scene graphs [35, 40, 81] of reconstructed splats to enable this kind of simulation. However, large manipulation of vehicles and assets in these simulators [35, 81] leads to myriad visual artifacts (see Naive Insertion columns of Fig. 12 for examples). Post-processing videos rendered from such neural scenes with single-step diffusion is a promising approach for overcoming these issues, but existing methods such as [34, 55] suffer from resolution / duration limitations.

Following [34, 55] we create a dataset based on Waymo Open Dataset [48] where 3DGS models are constructed,

Table 3. **Video Inpainting Evaluation**. *ChopGrad* results are output in a single step, a $50\times$ compute time improvement over VACE. VBench components are provided in the Appendix. Dark green is best, light green is second best.

Dataset	Method	FID ↓	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	VBench Overall ↑
DL3DV	VACE	45.060	574.441	20.678	0.757	0.236	0.083	0.792
	ChopGrad	40.948	583.581	21.699	0.765	0.221	0.077	0.792
Waymo	VACE	34.856	440.651	23.229	0.804	0.212	0.079	0.836
	ChopGrad	27.057	470.545	25.048	0.823	0.192	0.071	0.835
ROVI	VACE	30.491	201.610	22.618	0.834	0.223	0.112	0.752
	ChopGrad	27.547	188.546	25.200	0.859	0.199	0.095	0.747

Table 4. **Controlled Driving Video Generation Results**. *ChopGrad* was produced by initializing with Mirage followed by further finetuning Mirage's Harmonization stage for 1000 steps at high resolution (HR) / duration using *ChopGrad*. Dark green is best, light green is second best.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	FID ↓	FVD ↓
Mirage (HR)	27.30	0.8912	0.2067	0.0740	10.28	204.66
ChopGrad	29.49	0.9031	0.1719	0.0561	5.86	154.49

then assets are extracted and reinserted, producing the desired artifacts and input/output pairs to train and test with. Dataset construction details are presented in the Appendix.

We demonstrate *ChopGrad* for controlled driving video generation on Mirage [55] with our own Wan2.1-based implementation (details in the Appendix), as we were unable to acquire the original implementation even after contacting the authors. We train our implementation on 9-frame clips at a resolution of 480x832. After training Mirage we performed inference and evaluation at high resolution / duration (720x1280, 97 frames) as up-scaling training resolution outputs yielded poorer results. Subsequently, we finetuned Mirage's harmonization stage model using *ChopGrad* for 1000 steps at 720x1280 resolution, 49 frame duration, and performed inference on 49 frame segments. Results are reported in Table 4 and Fig. 12. Quantitative metrics are improved across all tests, while inspection of the qualitative results shows that finetuning Mirage with ChopGrad improves lighting fixing, artifact removal, and shadow insertion. Notably, the parameters of the decoder itself are finetuned in Mirage, confirming that *ChopGrad* can be used for decoder, as well as transformer, training.

5. Conclusion

We introduce *ChopGrad*, a truncated backpropagation approach that enables pixel-wise supervision at high resolutions and long durations in latent video diffusion models with causal caching. In architectures where the decoder is finetuned (e.g. [55]) this capability is required, while in others it leads to significantly improved results (bottom of Table 2). Applications of such models trained with pixel-wise losses are numerous, including single-step model distillation [73], enhancement of neural rendered scenes [11, 61], image translation [41], video super-resolution [12], and controlled driving video generation [34, 55].

Figure 11. **Video Inpainting**. We find that the recent VACE [28] tends to hallucinate (e.g., top section, top panel), while *ChopGrad* stays closer to the input but can also produce implausible results. *ChopGrad* results are output in a single step, a $50\times$ compute time improvement over VACE. Top:DL3DV, Middel: Waymo, Bottom: ROVI.

Figure 12. **Controlled Driving Video Generation**. Training with *ChopGrad* improves lighting, removes more artifacts, and produces better shadows.

By analyzing latent temporal locality, we demonstrate that long-range gradient dependencies in causal video autoencoders decay exponentially, allowing gradients to be truncated without compromising performance. This insight enables efficient fine-tuning of high-resolution, long-duration video diffusion models using perceptual losses that were previously intractable due to recursive activation accumulation.

Acknowledgements

Felix Heide was supported by an NSF CAREER Award (2047359), a Packard Foundation Fellowship, a Sloan Research Fellowship, a Sony Young Faculty Award, a Project X Innovation Award and a Amazon Science Research Award. Felix Heide is a co-founder of Algolux (now Torc Robotics), Head of AI at Torc Robotics, and a cofounder of Cephia AI.

References

[1] Aicher, C., Foti, N.J., Fox, E.B.: Adaptively truncating backpropagation through time to control gradient bias. In: Uncertainty in Artificial Intelligence. pp. 799–808. PMLR (2020)

[2] An, J., Zhang, S., Yang, H., Gupta, S., Huang, J.B., Luo, J., Yin, X.: Latent-Shift: Latent diffusion with temporal shift for efficient text-to-video generation. arXiv preprint arXiv:2304.08477 (2023)

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

[4] Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S.W., Fidler, S., Kreis, K.: Align Your Latents: High-resolution video synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 22563–22575 (2023)

[5] Ceylan, D., Huang, C.H.P., Mitra, N.J.: Pix2Video: Video editing using image diffusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 23206–23217 (2023)

[6] Chadebec, C., Tasar, O., Benaroche, E., Aubin, B.: Flash diffusion: Accelerating any conditional diffusion model for few steps image generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 15686–15695 (2025)

[7] Chan, K.C., Zhou, S., Xu, X., Loy, C.C.: Investigating tradeoffs in real-world video super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5962–5971 (2022)

[8] Chen, H., Zhang, Y., Cun, X., Xia, M., Wang, X., Weng, C., Shan, Y.: VideoCrafter2: Overcoming data limitations for high-quality video diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7310–7320 (2024)

[9] Chen, L., Li, Z., Lin, B., Zhu, B., Wang, Q., Yuan, S., Zhou, X., Cheng, X., Yuan, L.: OD-VAE: An omni-dimensional video compressor for improving latent video diffusion model. arXiv preprint arXiv:2409.01199 (2024)

[10] Chen, S., Ye, T., Lin, Y., Jin, Y., Yang, Y., Chen, H., Lai, J., Fei, S., Xing, Z., Tsung, F., et al.: Genhaze: Pioneering controllable one-step realistic haze generation for real-world dehazing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9194–9205 (2025)

[11] Chen, Y., Zheng, C., Xu, H., Zhuang, B., Vedaldi, A., Cham, T.J., Cai, J.: MVSplat360: Feed-forward 360 scene synthesis from sparse views. Advances in Neural Information Processing Systems **37**, 107064–107086 (2024)

[12] Chen, Z., Zou, Z., Zhang, K., Su, X., Yuan, X., Guo, Y., Zhang, Y.: DOVE: Efficient one-step diffusion model for real-world video super-resolution. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)

[13] Danier, D., Zhang, F., Bull, D.: LDMVFI: Video frame interpolation with latent diffusion models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 1472–1480 (2024)

[14] D'Avino, D., Cozzolino, D., Poggi, G., Verdoliva, L.: Autoencoder with recurrent neural networks for video forgery detection. arXiv preprint arXiv:1708.08754 (2017)

[15] Ding, K., Ma, K., Wang, S., Simoncelli, E.P.: Image quality assessment: Unifying structure and texture similarity. CoRR **abs/2004.07728** (2020), `https://arxiv.org/abs/2004.07728`

[16] Dong, Y., Zhang, Q., Jiang, M., Wu, Z., Fan, Q., Feng, Y., Zhang, H., Bao, H., Zhang, G.: One-shot refiner: Boosting feed-forward novel view synthesis via one-step diffusion. arXiv preprint arXiv:2601.14161 (2026)

[17] Gao, K., Shi, J., Zhang, H., Wang, C., Xiao, J., Chen, L.: Ca2-VDM: Efficient autoregressive video diffusion model with causal generation and cache sharing. arXiv preprint arXiv:2411.16375 (2024)

[18] Golinski, A., Pourreza, R., Yang, Y., Sautiere, G., Cohen, T.S.: Feedback recurrent autoencoder for video compression. In: Proceedings of the Asian Conference on Computer Vision (2020)

[19] HaCohen, Y., Chiprut, N., Brazowski, B., Shalem, D., Moshe, D., Richardson, E., Levin, E., Shiran, G., Zabari, N., Gordon, O., et al.: LTX-Video: Realtime video latent diffusion. arXiv preprint arXiv:2501.00103 (2024)

[20] He, J., Xue, T., Liu, D., Lin, X., Gao, P., Lin, D., Qiao, Y., Ouyang, W., Liu, Z.: VEnhancer: Generative space-time enhancement for video generation. arXiv preprint arXiv:2407.07667 (2024)

[21] He, X., Tang, H., Tu, Z., Zhang, J., Cheng, K., Chen, H., Guo, Y., Zhu, M., Wang, N., Gao, X., et al.: One step diffusion-based super-resolution with time-aware distillation. arXiv preprint arXiv:2408.07476 (2024)

[22] He, Y., Yang, T., Zhang, Y., Shan, Y., Chen, Q.: Latent video diffusion models for high-fidelity long video generation. arXiv preprint arXiv:2211.13221 (2022)

[23] Hess, G., Lindström, C., Fatemi, M., Petersson, C., Svensson, L.: Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 11982–11992 (2025)

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

[25] Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., Fleet, D.J.: Video diffusion models. Advances in neural information processing systems **35**, 8633–8646 (2022)

[26] Huang, X., Li, Z., He, G., Zhou, M., Shechtman, E.: Self forcing: Bridging the train-test gap in autoregressive video diffusion. arXiv preprint arXiv:2506.08009 (2025)

[27] Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., Liu, Z.: VBench: Comprehensive benchmark suite for video generative models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)

[28] Jiang, Z., Han, Z., Mao, C., Zhang, J., Pan, Y., Liu, Y.: Vace: All-in-one video creation and editing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17191–17202 (2025)

[29] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4) (2023)

[30] Lee, S., Kim, K., Ye, J.C.: Single-step bidirectional unpaired image translation using implicit bridge consistency distillation. arXiv preprint arXiv:2503.15056 (2025)

[31] Li, X., Zhang, Y., Ye, X.: DrivingDiffusion: Layout-guided multi-view driving scenarios video generation with latent diffusion model. In: European Conference on Computer Vision. pp. 469–485. Springer (2024)

[32] Li, Z., Lin, B., Ye, Y., Chen, L., Cheng, X., Yuan, S., Yuan, L.: WF-VAE: Enhancing video VAE by wavelet-driven energy flow for latent video diffusion model. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 17778–17788 (2025)

[33] Ling, L., Sheng, Y., Tu, Z., Zhao, W., Xin, C., Wan, K., Yu, L., Guo, Q., Yu, Z., Lu, Y., et al.: DL3DV-10k: A large-scale scene dataset for deep learning-based 3D vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)

[34] Ljungbergh, W., Taveira, B., Zheng, W., Tonderski, A., Peng, C., Kahl, F., Petersson, C., Felsberg, M., Keutzer, K., Tomizuka, M., et al.: R3d2: Realistic 3d asset insertion via diffusion for autonomous driving simulation. arXiv preprint arXiv:2506.07826 (2025)

[35] Ljungbergh, W., Tonderski, A., Johnander, J., Caesar, H., Åström, K., Felsberg, M., Petersson, C.: Neuroncap: Photorealistic closed-loop safety testing for autonomous driving. In: European Conference on Computer Vision. pp. 161–177. Springer (2024)

[36] Mao, X., Jiang, Z., Wang, F.Y., Zhang, J., Chen, H., Chi, M., Wang, Y., Luo, W.: Osv: One step is enough for high-quality image to video generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 12585–12594 (2025)

[37] Melnik, A., Ljubljanac, M., Lu, C., Yan, Q., Ren, W., Ritter, H.: Video diffusion models: A survey. arXiv preprint arXiv:2405.03150 (2024)

[38] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1) (2021)

[39] Noroozi, M., Hadji, I., Martinez, B., Bulat, A., Tzimiropoulos, G.: You only need one step: Fast super-resolution with stable diffusion via scale distillation. In: European Conference on Computer Vision. pp. 145–161. Springer (2024)

[40] Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2856–2865 (2021)

[41] Parmar, G., Park, T., Narasimhan, S., Zhu, J.Y.: One-step image translation with text-to-image models. arXiv preprint arXiv:2403.12036 (2024)

[42] Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International conference on machine learning. pp. 1310–1318. Pmlr (2013)

[43] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep., Institute of Cognitive Science (1985)

[44] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., Valaee, S.: Recent advances in recurrent neural networks. arXiv preprint arXiv:1801.01078 (2017)

[45] Sauer, A., Boesel, F., Dockhorn, T., Blattmann, A., Esser, P., Rombach, R.: Fast high-resolution image synthesis with latent adversarial diffusion distillation. In: SIGGRAPH Asia 2024 Conference Papers. pp. 1–11 (2024)

[46] Sauer, A., Lorenz, D., Blattmann, A., Rombach, R.: Adversarial diffusion distillation. In: European Conference on Computer Vision. Springer (2024)

[47] Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., et al.: Make-A-Video: Text-to-video generation without text-video data. arXiv preprint arXiv:2209.14792 (2022)

[48] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2446–2454 (2020)

[49] Tao, X., Gao, H., Liao, R., Wang, J., Jia, J.: Detail-revealing deep video super-resolution. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)

[50] Teng, S., Gao, G., Danier, D., Jiang, Y., Zhang, F., Davis, T., Liu, Z., Bull, D.: Gfix: Perceptually enhanced gaussian splatting video compression. arXiv preprint arXiv:2511.06953 (2025)

[51] Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.W., Chen, D., Yu, F., Zhao, H., Yang, J., et al.: WAN: Open and advanced large-scale video generative models. arXiv preprint arXiv:2503.20314 (2025)

[52] Wang, H., Liu, F., Chi, J., Duan, Y.: Videoscene: Distilling video diffusion model to generate 3d scenes in one step. In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16475–16485. IEEE (2025)

[53] Wang, J., Lin, S., Lin, Z., Ren, Y., Wei, M., Yue, Z., Zhou, S., Chen, H., Zhao, Y., Yang, C., et al.: Seedvr2: One-step video restoration via diffusion adversarial post-training. arXiv preprint arXiv:2506.05301 (2025)

[54] Wang, R., Liu, X., Zhang, Z., Wu, X., Feng, C.M., Zhang, L., Zuo, W.: Benchmark dataset and effective inter-frame alignment for real-world video super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2023)

[55] Wang, S., Sun, H., Wang, B., Ye, H., Yu, X.: Mirage: One-step video diffusion for photorealistic and coherent asset editing in driving scenes. arXiv preprint arXiv:2512.24227 (2025)

[56] Wang, X., Xie, L., Dong, C., Shan, Y.: Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1905–1914 (2021)

[57] Wang, Y., Chen, X., Ma, X., Zhou, S., Huang, Z., Wang, Y., Yang, C., He, Y., Yu, J., Yang, P., et al.: LAVIE: High-quality video generation with cascaded latent diffusion models. International Journal of Computer Vision **133**(5), 3059–3078 (2025)

[58] Wang, Y., Yang, W., Chen, X., Wang, Y., Guo, L., Chau, L.P., Liu, Z., Qiao, Y., Kot, A.C., Wen, B.: Sinsr: diffusion-based image super-resolution in a single step. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 25796–25805 (2024)

[59] Wang, Z., Zhang, Z., Pang, T., Du, C., Zhao, H., Zhao, Z.: Orient anything: Learning robust object orientation estimation from rendering 3d models. arXiv preprint arXiv:2412.18605 (2024)

[60] Williams, R.J., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Backpropagation, pp. 433–486. Psychology Press (2013)

[61] Wu, J.Z., Zhang, Y., Turki, H., Ren, X., Gao, J., Shou, M.Z., Fidler, S., Gojcic, Z., Ling, H.: Difix3D+: Improving 3D reconstructions with single-step diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2025)

[62] Wu, J., Li, X., Si, C., Zhou, S., Yang, J., Zhang, J., Li, Y., Chen, K., Tong, Y., Liu, Z., et al.: Towards language-driven video inpainting via multimodal large language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12501–12511 (2024)

[63] Wu, P., Zhu, K., Liu, Y., Zhao, L., Zhai, W., Cao, Y., Zha, Z.J.: Improved video VAE for latent video diffusion model. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 18124–18133 (2025)

[64] Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 21469–21480 (2025)

[65] Xie, R., Liu, Y., Zhou, P., Zhao, C., Zhou, J., Zhang, K., Zhang, Z., Yang, J., Yang, Z., Tai, Y.: STAR: Spatial-temporal augmentation with text-to-video models for real-world video super-resolution. arXiv preprint arXiv:2501.02976 (2025)

[66] Xing, Z., Feng, Q., Chen, H., Dai, Q., Hu, H., Xu, H., Wu, Z., Jiang, Y.G.: A survey on video diffusion models. ACM Computing Surveys **57**(2), 1–42 (2024)

[67] Yang, X., He, C., Ma, J., Zhang, L.: Motion-Guided latent diffusion for temporally consistent real-world video super-resolution. In: European conference on computer vision. pp. 224–242. Springer (2024)

[68] YANG, X., Xiang, W., Zeng, H., Zhang, L.: Real-world video super-resolution: A benchmark dataset and a decomposition based learning scheme. ICCV (2021)

[69] Yang, Y., Huang, H., Peng, X., Hu, X., Luo, D., Zhang, J., Wang, C., Wu, Y.: Towards one-step causal video generation via adversarial self-distillation. arXiv preprint arXiv:2511.01419 (2025)

[70] Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., et al.: CogVideoX: Text-to-video diffusion models with an expert transformer. arXiv preprint arXiv:2408.06072 (2024)

[71] Ye, V., Li, R., Kerr, J., Turkulainen, M., Yi, B., Pan, Z., Seiskari, O., Ye, J., Hu, J., Tancik, M., et al.: gsplat: An open-source library for gaussian splatting. Journal of Machine Learning Research **26**(34) (2025)

[72] Yi, P., Wang, Z., Jiang, K., Shao, Z., Ma, J.: Multi-temporal ultra dense memory network for video super-resolution. IEEE Transactions on Circuits and Systems for Video Technology **30**(8) (2019)

[73] Yin, T., Gharbi, M., Park, T., Zhang, R., Shechtman, E., Durand, F., Freeman, B.: Improved distribution matching distillation for fast image synthesis. Advances in neural information processing systems **37**, 47455–47487 (2024)

[74] Yu, L., Lezama, J., Gundavarapu, N.B., Versari, L., Sohn, K., Minnen, D., Cheng, Y., Birodkar, V., Gupta, A., Gu, X., et al.: Language model beats diffusion–Tokenizer is key to visual generation. arXiv preprint arXiv:2310.05737 (2023)

[75] Yu, S., Sohn, K., Kim, S., Shin, J.: Video probabilistic diffusion models in projected latent space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 18456–18466 (2023)

[76] Yu, W., Xing, J., Yuan, L., Hu, W., Li, X., Huang, Z., Gao, X., Wong, T.T., Shan, Y., Tian, Y.: ViewCrafter: Taming video diffusion models for high-fidelity novel view synthesis. arXiv preprint arXiv:2409.02048 (2024)

[77] Yue, Z., Wang, J., Loy, C.C.: ResShift: Efficient diffusion model for image super-resolution by residual shifting. Advances in Neural Information Processing Systems **36**, 13294–13307 (2023)

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

[79] Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., et al.: PyTorch FSDP: Experiences on scaling fully sharded data parallel. arXiv preprint arXiv:2304.11277 (2023)

[80] Zheng, Z., Peng, X., Yang, T., Shen, C., Li, S., Liu, H., Zhou, Y., Li, T., You, Y.: Open-Sora: Democratizing efficient video production for all. arXiv preprint arXiv:2412.20404 (2024)

[81] Zhou, H., Lin, L., Wang, J., Lu, Y., Bai, D., Liu, B., Wang, Y., Geiger, A., Liao, Y.: Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving. IEEE Transactions on Pattern Analysis and Machine Intelligence (2025)

[82] Zhou, S., Yang, P., Wang, J., Luo, Y., Loy, C.C.: Upscale-A-Video: Temporal-consistent diffusion model for real-world video super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2535–2545 (2024)

[83] Zhou, S., Yang, P., Wang, J., Luo, Y., Loy, C.C.: Upscale-A-Video: Temporal-consistent diffusion model for real-world video super-resolution. In: CVPR (2024)

[84] Zhou, Y., Wang, Q., Cai, Y., Yang, H.: Allegro: Open the black box of commercial-level video generation model. arXiv preprint arXiv:2410.15458 (2024)

Appendix

Section A provides additional implementation information for the proposed *ChopGrad* architecture using the WAN 2.1 [51] and CogVideoX [70] video autoencoders. Next, Section B reports additional details regarding evaluation setups and baseline implementations for all applications, while Section C provides details about model architectures, training setups, and inference schemes. Next, Section D describes additional algorithmic optimizations for *ChopGrad* to minimize computation times when long truncation distances are used. Finally, Sections F and E report additional quantitative and qualitative results, respectively.

A. Implementation Details

ChopGrad is formalized in Algorithm 1 and illustrated in Fig.2 of the primary document. The latent cache is first either initialized as empty or detached from the previous decoder pass (lines 2 - 3). Critically, the cache is detached prior to running the forward pass, so the gradients do not propagate backwards through the full video. The pixel-wise loss is computed using the decoded frames (line 4) and the gradients backpropagated to the latents and cache (lines 5-6). Truncated backpropagation is then run using the specified truncation distance (lines 7-9), and the compute graph for latent $\mathbf{z}_{i-D_{trunc}}$ is subsequently released. Cache gradients are zeroed after each backpropagation. Notably, maintaining the compute graph in memory requires storing activations, resulting in memory use that scales linearly with D_{trunc}, as does compute time as shown in Fig.6.

Algorithm 1 *ChopGrad*.

Require: Video latents $\{\mathbf{z}_i\}_{i=1}^{\lceil T/G \rceil}$, D_{trunc}, $\mathcal{L}_{pix}$.
Ensure: Gradients $\{\nabla_{\mathbf{z}_i}\mathcal{L}\}$ for all latent frame groups

1: **for** i from 1 to $\lceil T/G \rceil$ **do**
2: $\quad\mathbf{z}^c_{i-1} \leftarrow \text{detach}(\mathbf{z}^c_{i-1})$
3: $\quad\hat{\mathbf{X}}_i, \mathbf{z}^c_i = \mathcal{D}(\text{Concat}(\mathbf{z}^c_{i-1}, \mathbf{z}_i))$
4: $\quad\mathcal{L}_i = \frac{1}{T}\sum_t \mathcal{L}^{pix}_i(\hat{\mathbf{X}}_{i,t}, \mathbf{X}_{i,t})$
5: $\quad\nabla_{\mathbf{z}_i} \leftarrow \frac{\partial \mathcal{L}_i}{\partial \mathbf{z}_i}$
6: $\quad\nabla_{\mathbf{z}^c_{i-1}} \leftarrow \frac{\partial \mathcal{L}_i}{\partial \mathbf{z}^c_{i-1}}$
7: $\quad$**for** $k = 1$ to $\min(D_{\text{trunc}}, i)$ **do**
8: $\quad\quad\nabla_{\mathbf{z}_{(i-k)}} += \frac{\partial \mathbf{z}^c_{(i-k)}}{\partial \mathbf{z}_{(i-k)}}\nabla_{\mathbf{z}^c_{(i-k)}}$
9: $\quad\quad\nabla_{\mathbf{z}^c_{(i-k-1)}} \leftarrow \frac{\partial \mathbf{z}^c_{(i-k)}}{\partial \mathbf{z}^c_{(i-k-1)}}\nabla_{\mathbf{z}^c_{(i-k)}}$
10: $\quad$**end for**
11: $\quad$**if** $i \geq D_{\text{trunc}}$ **then**
12: $\quad\quad$Release compute graph for $\mathbf{z}_{(i-D_{\text{trunc}})}$
13: $\quad$**end if**
14: **end for**

B. Additional Evaluation and Baseline Details

This section provides additional evaluation and baseline details. Sec. B.1 presents details for Video Super-Resolution, Sec. B.2 for Artifact Removal in Novel View Synthesis, Sec. B.3 for Video Inpainting, and Sec. B.4 for Controlled Driving Video Generation.

B.1. Video Super-Resolution

Video super-resolution is evaluated across the following datasets: UDM10 [72], SPMCS [49], YouHQ40 [83], RealVSR [68], and MVSR4x [54]. Metric evaluations are performed using the publicly available DOVE evaluation script and reference baseline metrics are taken from those reported by DOVE. Notably, the evaluation metrics computed using this script using the DOVE checkpoint match those originally reported.

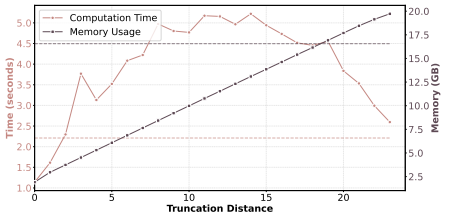

Figure 13. Computational time and memory requirements as a function of truncation distance for the modified *ChopGrad* algorithm. The full video length is 24 frame groups. The dashed horizontal lines indicate time and memory requirements of the original backpropagation scheme.

B.1.1. DOVE.

The publicly available DOVE code is used for inference and evaluation. DOVE contains a 2-stage training scheme, where the first stage uses video sequences and the second stage uses a combination of video sequences and individual frames to train the network.

Additional Stage 2 fine-tuning was performed for 500 iterations, but no performance improvement was observed, indicating that the public checkpoint was already converged and as such did not benefit from additional training. As such, the original DOVE checkpoint is used in the evaluations.

B.2. Artifact Removal in Novel View Synthesis

We evaluate our neural-rendering enhancements on the DL3DV-Benchmark [33], which we split into 95 training scenes, 40 testing scenes, and 5 validation scenes. Each video in the dataset is approximately 300 frames long and a 3DGS model is trained using GSplat [71] using only every 50th frame for each scene. Training of the 3DGS models follows the standard gsplat implementation, where the first 500 iterations do not modify the number of Gaussians,

then the next 14.5k iterations are tailored for adaptive density control and the remaining iterations are for Gaussian parameter optimization.

The 3DGS model is trained using 960×544 resolution images for a total of 30k iterations, requiring approximately 5 minutes of training for each scene using an A100 GPU.

In addition, we conduct an anonymized study for determining user preference for novel view synthesis video generation. Participants are presented with three side-by-side videos from ten scenes randomly selected from the test set. Each of the 3 videos is generated by one of the baselines, MVSplat [11], Difix [61], and the proposed method, ChopGrad ($D_{trunc} = 1$), and users are asked to mark the video which they preferred. The order of video appearance for each scene was randomized. A total of 34 users participated in the survey. The percentage of users who preferred ChopGrad is computed for each scene and then averaged across all 10 scenes, yielding an overall preference rate of 95.6% for the proposed method.

B.2.1. Difix.

Difix utilizes an image diffusion model backbone to process individual frames from input video sequences. This diffusion network is initialized from the SD-Turbo [46] checkpoint and is fine-tuned for neural render enhancement.

For a fair comparison, additional fine-tuning was performed on the DL3DV dataset using the training settings provided in the paper. For each training iteration, video frames are randomly sampled from the dataset. Difix is fine-tuned for 10k iterations on 4 A100 GPUs, taking approximately 2 hours.

B.2.2. MVSplat360.

The MVSplat-360 baseline approach takes as input a sparse set of views and uses a pre-trained feed-forward network to estimate a 3DGS model. Next, a video sequence is generated along a camera trajectory and this video is refined using a fine-tuned video diffusion network. The video diffusion network is initialized using the publicly available Stable Video Diffusion (SVD) checkpoint [3]. The fine-tuned video diffusion checkpoint is provided by the MVSplat-360 authors and used in the experiments.

The SVD model groups together 14 video frames during the diffusion process. Self attention is used to enable frame groups to attend to one another, but this resulted in prohibitive memory requirements when evaluating videos with 294 frames, even at low resolution. As such, this feature was disabled and frame groups were not able to attend to each other.

As MVSplat-360 was trained on the DL3DV dataset, no additional fine-tuning was performed.

B.3. Video Inpainting

We evaluate inpainting on 3 datasets: D3LDV-benchmark[33], Waymo[48], and ROVI[62]. For D3LDV-Benchmark we use the same dataset and train/test split as was described in Sec. B.2. For Waymo, we use the dataset as described in Sec. B.4. For both of these cases we mask the ground truth videos as described in Section 4.2, with a fixed rectangular mask having half the height and half the width of the video. We also use a fixed prompt for both. We also include a new experiment where vehicle bounding boxes are masked in Waymo. Specifically, we randomly select 50% of the labeled vehicles in the scene, and mask out the vehicle throughout the entire video. We refer to this setup as "Waymo-Bbox." For ROVI we use the train/test split from the original dataset, as well as prompts provided by this dataset.

B.3.1. VACE.

For all experiments, we train the VACE baseline for the same number of steps as *ChopGrad*, with the same video duration, and resolution. All other settings are also the same, unless otherwise indicated in this section. VACE is trained using the standard latent MSE velocity objective (as Wan 2.1 and thus VACE are flow models), with the denoising timestep sampled using a timestep shift of 5, the same shift being used during sampling. The model is sampled over 50 steps following default settings in the VACE implementation[1]. Masks are provided to the model during training and inference.

B.4. Controlled Driving Video Generation

We evaluate on the Waymo Open Dataset [48], reconstructing 300 sequences, 230 for training and 70 for evaluation, using SplatAD [23], a dynamic 3D Gaussian Splatting-based method. SplatAD [23] decomposes each scene into static background and dynamic actors represented as Gaussian primitives, which allows us to remove selected actors and replace them with generated 3D vehicles at the original pose. Our vehicle generation pipeline consists of vehicle extraction followed by vehicle alignment. We extract object-centric image patches from the curated Waymo object vehicle set using camera detections and LIDAR track IDs to assemble multi-view crops. Using instance and segmentation masks, we remove background pixels, and the resulting patches are used as input to the TRELLIS image→3D pipeline [64] to produce 3D Gaussian representations of the vehicles. Since TRELLIS produces reconstructions in an unanchored coordinate frame, the generated vehicles can be arbitrarily rotated. To ensure consistent forward-facing poses, we render each vehicle from angles of 0 °, 90°, 180°, 270° and estimate its yaw with Orient-Anything [59]. vehicles with inconsistent cyclic orientation patterns or orien-

[1] https://github.com/ali-vilab/VACE

tation confidence below a threshold are discarded, and the original 3DGS scene models are used.

B.4.1. Mirage.

Mirage code is not publicly available, and we were not able to gain access by emailing the authors, so we re-implement it based on the description of the method provided in the paper. We choose to use Wan 2.1 14B instead of CogVideoX as this is a more modern and capable diffusion model, but uses a similar VAE architecture. For fair comparison, for *ChopGrad* we keep the same model architecture, losses, etc. and only modify the training duration and resolution, since use of *ChopGrad* allows us to greatly increase these.

The main architectural modifications to the diffusion model proposed by Mirage are the addition of skip connections to the decoder, and the addition of several LoRAs. The skip connections are extracted from the encoder in "2D" mode, i.e., where each frame is encoded separately and no temporal compression is used. They are extracted from the output of the layer immediately preceding the first spatiotemporal downsample step, and are fused into the decoder immediately following the final spatio-temporal upsample step. To perform fusion, we concatenate the skip connections with decoder features along the channel dimension, then perform convolution with a 3×3 kernel to compute the fused features. We use LoRA rank and alpha values of 16 in the VAE, while for the transformer we set rank to 128 and alpha to 64.

We replicate Mirage's two training stages – Reconstruction and Harmonization. We train each for $10k$ steps, with separate LoRAs for each stage. The Reconstruction phase trains the fusion blocks and a decoder reconstruction LoRA, while Harmonization trains a transformer LoRA and another decoder LoRA and keeps the fusion blocks frozen. For both stages learning rate is 2×10^{-4}, batch size is 8, clip length is 9 frames and clip resolution is 480×832. We updated the resolution slightly from the original paper to match Wan 2.1's training resolution. We trained the reconstruction phase with LPIPS and MSE losses, with the LPIPS component scaled by a factor of 0.1.

The harmonization phase was trained with a fixed timestep (200). To be better aligned with Wan 2.1's velocity prediction training scheme, we trained the model to predict the difference between output and input. A combination of LPIPS and Gram losses was used, with Gram loss being scaled by 0.1. The AdamW optimizer was used with a weight decay of 0.01, betas of 0.9 and 0.99. All trainings were performed on a node with 8 80GB A100 GPUs. Though the original paper trained on H200s with a smaller base model, we were able to fit our implementation on A100s by fully sharding the model and optimizer parameters with FSDP[2], and using transformer and decoder activation checkpointing.

[2]https://docs.pytorch.org/docs/stable/fsdp.html

C. Architecture, Training, and Inference Details

This section provides additional details on architectures, training, and inference. Sec. C.1 presents details for Video Super-Resolution, Sec. C.2 for Artifact Removal in Novel View Synthesis, Sec. C.3 for Video Inpainting, and Sec. C.4 for Controlled Driving Video Generation.

C.1. Video Super-Resolution

Network Architecture. For video super-resolution, the DOVE [12] checkpoint is used to initialize *ChopGrad*. DOVE uses the CogVideoX [70] autoencoder. Similar to the WAN 2.1 [51] video autoencoder, the CogVideoX autoencoder compresses multiple video frames into frame groups. This temporal compression differs from the WAN 2.1 autoencoder however, in that the number of decoded frames changes depending on how many frame groups are used. When an odd number of frame groups are passed to the decoder, the first frame group is decoded into a single frame and the remaining frame groups are decoded into 4 frames. When an even number of frame groups are passed to the decoder, all frame groups are decoded into 4 frames. This behavior necessitates that 2 frame groups must be decoded together for each decoding step. Although this behavior could be addressed through minor modifications to the implementation, no changes were made to preserve compatibility with the original network.

Training. The default DOVE configuration performs training at a resolution of 640×320. Given this low resolution, no spatial chunking was used. Fine-tuning is conducted using videos with lengths of 24 frames. The second stage training implementation from DOVE is adopted with original hyperparameters and initialization is performed from the provided checkpoint. Fine-tuning is performed for 500 iterations on 4 A100 GPUs, requiring approximately 8 hours. In contrast to the original DOVE Stage 2 procedure, only videos are used, no images are trained on.

C.2. Artifact Removal in Novel View Synthesis

Network Architecture. *ChopGrad* and ablations are initialized using the pretrained Wan 2.1 14B [51] diffusion transformer model. This model is then fine-tuned using the latent embeddings of the 3DGS renders as inputs and the ground-truth images as targets. Notably, WAN is trained to output velocity $v = \hat{\boldsymbol{z}} - \boldsymbol{z}$, where $\hat{\boldsymbol{z}}$ is the latent embedding of the rendered video and $\boldsymbol{z}$ is the latent embedding of the ground truth video. In order to better align with the original training objective of Wan 2.1 [51], we leverage the same training scheme for fine-tuning. A fixed text caption is

used to condition the refinement and the diffusion timestep is fixed to 200. No modifications to the video encoder have been made and the pre-trained WAN network has not been pre-distilled for single-step inference.

The WAN 2.1 video autoencoder has a temporal compression factor of 4, meaning there are 4 video frames per frame group latent. Notably, the network pads the first video frame with 3 empty frames, meaning the total length of the video processed by WAN 2.1 is $4N + 1$, where N is the number of frame groups. This temporal compression factor corresponds to $G = 4$ from Section 3.4. *ChopGrad* decodes latents to pixels using a spatial chunk size of $H/2 \times W/2$, resulting in 4 chunks. This preserves the aspect ratio of the video and enables parallel processing for each chunk.

Training. *ChopGrad* is trained using both a latent MSE loss and a pixel-space LPIPS [78] loss with VGG features. An LPIPS weight of 100 was used while the latent MSE weight was set to 1 for all experiments, including ablations.

A scene is randomly chosen from the dataset and a random 81-frame sequence from this scene is used for each training iteration. Videos have a resolution of 832×480. Notably, larger resolution videos may be used for training (i.e. 1280×720), but lower resolution videos were used in experiments for fair baseline comparisons.

Training is conducted for approximately 3-4 hours on 8 A100 GPUs. PyTorch's Fully Sharded Data Parallel architecture [79] is leveraged to shard the model parameters and optimizer states of the WAN diffusion transformer. To minimize memory, 8-way sequence parallelism is used.

The AdamW optimizer is used with a learning rate of $1e^{-5}$, a weight decay of 0.1, betas of 0.9 and 0.99, and a batch size of 1. *ChopGrad* and all ablations, excluding *ChopGrad*†, are trained for 880 training iterations. *ChopGrad*† is trained for 1760 training iterations. Training times are presented in Table 2.

Inference. Inference times in Table 1 are measured by processing the entire video and dividing by the number of total video frames and finally multiplied by the number of GPUs to account for sequence parallelism.

This ensures a fair comparison with the baseline methods which both utilize only a single GPU. These inference times also include pre-processing as well as the full network pass (i.e., video encoding, decoding, and transformer forward pass).

Inference is performed on the first 297 frames of the video as this corresponds to the temporal compression of the WAN 2.1 video autoencoder using a total of $N = 75$ frame groups. The evaluation is conducted on the first 294 frames of the video to maintain compatibility and a fair comparison with the baseline methods as MVSplat-360 is only able to process multiples of 14 frames.

C.3. Video Inpainting

Network Architecture. We use the same network setup as described in Section C.2. *ChopGrad* is initialized using the pretrained Wan 2.1 14B [51] diffusion transformer model. This model is then fine-tuned using the latent embeddings of the masked videos as inputs and the ground-truth videos as targets. Notably, WAN is trained to output velocity $v = \hat{\boldsymbol{z}} - \boldsymbol{z}$, where $\hat{\boldsymbol{z}}$ is the latent embedding of the rendered video and $\boldsymbol{z}$ is the latent embedding of the ground truth video. In order to better align with the original training objective of Wan 2.1 [51], we leverage the same training scheme for fine-tuning. A fixed text caption (except for the ROVI case) is used to condition the refinement and the diffusion timestep is fixed to 200. No modifications to the video encoder have been made and the pre-trained WAN network has not been pre-distilled for single-step inference.

The WAN 2.1 video autoencoder has a temporal compression factor of 4, meaning there are 4 video frames per frame group latent. Notably, the network pads the first video frame with 3 empty frames, meaning the total length of the video processed by WAN 2.1 is $4N + 1$, where N is the number of frame groups.This temporal compression factor corresponds to $G = 4$ from Section 3.4. *ChopGrad* decodes latents to pixels using a spatial chunk size of $H/2 \times W/2$, resulting in 4 chunks. This preserves the aspect ratio of the video and enables parallel processing for each chunk.

Training. Training is done using the same settings as described in Section C.2, except for differences in duration and number of training steps, which vary across the datasets. For DL3DV-benchmark we trained for 5 epochs at 49 frames, for Waymo (and Waymo-Bbox) 5 epochs at 49 frames, for ROVI $10k$ steps at 29 frames. The train times were 1.5, 3, and 18 hours, respectively. Train time for Waymo-Bbox was the same as for Waymo. The increased train steps for ROVI reflect the fact that it is a much larger dataset than the other two (5172 videos in the training set).

Inference. We perform inference at the same duration as training, evaluating on the first N frames of each video, where N is the training/inference duration.

C.4. Controlled Driving Video Generation

Network Architecture. For *ChopGrad*, we keep the same network as for Mirage (described in detail in Supplemental Section B.4.1).

Training. We initialize with the Mirage harmonization checkpoint which had been trained for $10k$ steps. The validation loss plateaued around $5k$ steps so we are confident the model had converged. We then train it for a subsequent $1k$ steps using *ChopGrad* at a resolution of 720×1280 and a

duration of 49 frames. We use 16 spatial chunks ($4h \times 4w$) and a truncation distance of 1. During this training batch size was set to 1 and the learning rate is maintained at 2×10^{-4}. The AdamW optimizer is used with a weight decay of 0.1, betas of 0.9 and 0.99. The training process took approximately 10 hours on 8 A100 GPUs.

D. Algorithmic Optimizations for Truncated Backpropagation

The time complexity of *ChopGrad* as described in Algorithm 1 scales linearly with respect to D_{trunc}. This complexity stems from the need to backpropagate over each frame group D_{trunc} times. This is not the case with the standard backpropagation scheme, which only needs to make one backward pass, having accumulated gradients from all frame groups $i{+}1...N$ prior to computing the gradient for frame group i. This section examines in more detail the origin of this difference in time complexity and introduces a minor modification to Algorithm 1 that ensures, as the truncation distance approaches the full video length, the overall time complexity converges to that of the full backpropagation scheme.

ChopGrad requires multiple backward passes over each frame group because in order to compute the gradient for frame group i, gradients for frame groups $i+1...i+D_{trunc}$ must be available. Furthermore, the compute graph for frame group i cannot be released from memory until all D_{trunc} future gradients have backpropagated through it. In order to release frame group i as soon as possible, i.e. once the algorithm reaches frame group $i + D_{trunc}$, it is imperative to backpropagate all the way from $i + D_{trunc}$ to i as soon $i + D_{trunc}$ is decoded and the loss computed. This necessitates performing a backward pass through all intermediate frame groups as well. Since backpropagation is performed through D_{trunc} frame groups each time the compute graph from frame group i is released, the compute must scale linearly with D_{trunc}. There is a time-memory tradeoff present – graphs could be released less often, for example every s steps instead of every single step, reducing the time complexity by a factor of s but increasing memory consumption accordingly, since $D_{trunc}+s$ frames worth of activations need to be stored in memory.

In Algorithm 1, backpropagation is performed all the way back through the previous D_{trunc} steps as soon as a new frame group is decoded. If this is delayed so that the backpropagation only occurs when it is time to evict the compute graph for frame group $i - D_{trunc}$, a performance improvement can be gained in regimes where D_{trunc} is close to the full video length, T. This is because the total number of backpropagation steps through individual frame groups (and as such the time complexity) becomes equal to $D_{trunc} * n_{evict}$, where n_{evict} is the number of cache evictions, and $n_{evict} = T - D_{trunc}$. Empirical complexity results for this modification are shown in Figure 13. Note that a low resolution video (128x64) is used to make computation of the vanilla backpropagation method and *ChopGrad* with high truncation distances tractable.

In Figure 13 *ChopGrad* has slightly worse time and memory performance that vanilla backpropagation at $D_{trunc} = T$ as some overhead is introduced by fragmenting the compute graph and maintaining detached versions of the cache.

It is not recommended to use *ChopGrad* at truncation distances greater than 2, as the results from Section 4 demonstrate that causal VAEs exhibit strong spatial locality, and terms from faraway latents do not contribute significantly to the gradient. In regimes with small values of D_{trunc}, the modified and unmodified algorithms exhibit practically equivalent performance characteristics. This section is included for completeness, but should not have much impact on real-world uses of *ChopGrad*.

E. Additional Quantitative Results

Supplemental Table 5 presents individual VBench [27] component scores used to compile the VBench Overall Quality metric reported for the Video Inpainting application in main document Table 3. Higher is better for all scores. In addition to a $50\times$ reduction in compute time, *ChopGrad* delivers modest improvements in motion smoothness and temporal flickering across all datasets, while VACE consistently has slightly higher imaging quality and subject consistency, with the remaining scores having mixed results across datasets. This pattern is consistent with our observation that VACE is more prone to hallucination – conforming less closely to the input video allows it to generate slightly higher quality images (despite being included in VBench, aesthetic quality is an image-based metric) and more consistent subjects. The *ChopGrad* trained single-step model hallucinates less, retains higher reconstruction metrics (as evidenced in main document Table 3), and its improved motion smoothness and temporal flickering can be attributed to it staying closer to the original video, as real videos often have smooth motion and less flicker than generated videos.

Supplemental Table 6 presents the same metrics as main document Table 3 for the new Waymo-Bbox task. Results are consistent with other tasks, with *ChopGrad* training resulting in improved reconstruction metrics and similar video quality metrics, while achieving a $50\times$ inference time reduction.

F. Additional Qualitative Results

Additional qualitative results for Video Super-Resolution, Artifact Removal in Novel View Synthesis, and Controlled Driving Video Generation experiments are presented in Figures 14, 15, and 20 respectively. Qualitative results for

Table 5. **Breakdown of VBench Scores for Video Inpainting**. Full VBench component scores corresponding to the VBench Overall values reported in main document Table 3. In addition to a $50\times$ reduction in compute time, *ChopGrad* delivers modest improvements in motion smoothness and temporal flickering across all datasets, while VACE consistently has slightly higher imaging quality and subject consistency, with the remaining scores having mixed results across datasets.

Dataset	Method	Aesthetic Quality	Background Consistency	Dynamic Degree	Imaging Quality	Motion Smoothness	Subject Consistency	Temporal Flickering	VBench Overall Quality
DL3DV	VACE	0.531	0.917	0.950	0.731	0.957	0.912	0.912	0.792
	ChopGrad	0.519	0.912	0.950	0.729	0.961	0.903	0.919	0.792
Waymo	VACE	0.512	0.956	0.882	0.716	0.987	0.953	0.969	0.836
	ChopGrad	0.517	0.960	0.894	0.695	0.988	0.945	0.972	0.835
Waymo-Bbox	VACE	0.519	0.957	0.859	0.704	0.987	0.950	0.970	0.834
	ChopGrad	0.525	0.962	0.847	0.696	0.988	0.955	0.972	0.835
ROVI	VACE	0.472	0.926	0.816	0.538	0.959	0.885	0.941	0.752
	ChopGrad	0.457	0.924	0.749	0.521	0.964	0.884	0.947	0.747

Table 6. **Waymo-Bbox Video Inpainting Results**. Quantitative comparison between VACE and *ChopGrad* on the Waymo-Bbox setting. Results are consistent with other tasks, with *ChopGrad* training resulting in improved reconstruction metrics and similar video quality metrics, while achieving a $50\times$ inference time reduction.

Method	FID	FVD	PSNR	SSIM	LPIPS	DISTS	VBench Overall
VACE	**12.084**	253.710	27.661	0.873	0.154	0.061	0.834
ChopGrad	12.358	**252.599**	**28.838**	**0.875**	**0.146**	**0.057**	**0.835**

inpainting are presented in four separate figures based on dataset: DL3DV in Fig. 16, Waymo in Fig. 17, Waymo-Bbox in Fig. 18, and ROVI in Fig. 19. These additional results report a number of the benefits of training at increased resolution / duration with *ChopGrad*. In the case of Video Super-Resolution (Fig. 14), *ChopGrad* enabled back-propagating through the decoder for entire videos (rather than individual frames), allowing the transformer to properly account for temporal compression and resulting in improved visual quality, especially for fine details such as fur, cloth, and clouds. In the case of Artifact Removal in Novel View Synthesis (Fig. 15), *ChopGrad* trained models have access to more views of the scene simultaneously as a result of extended video duration, leading to enhanced artifact removal capabilities. In Video Inpainting (Figs. 16, 17, 18, and 19), *ChopGrad* is used to produce a significantly faster model (single-step vs 50 steps for VACE) while also reducing hallucinations. Finally, Fig. 20 shows that for Controlled Driving Video Generation, tuning with *ChopGrad* (as compared to training at low resolution/duration and performing high resolution/duration inference) leads to a stronger model that is able to make larger changes to input images, improving lighting and shadows and removing more Gaussian Splat artifacts. Collectively, these results illustrate a variety of ways in which state-of-the-art methods may be improved further by using truncated backpropagation to enable pixel-wise perceptual losses.

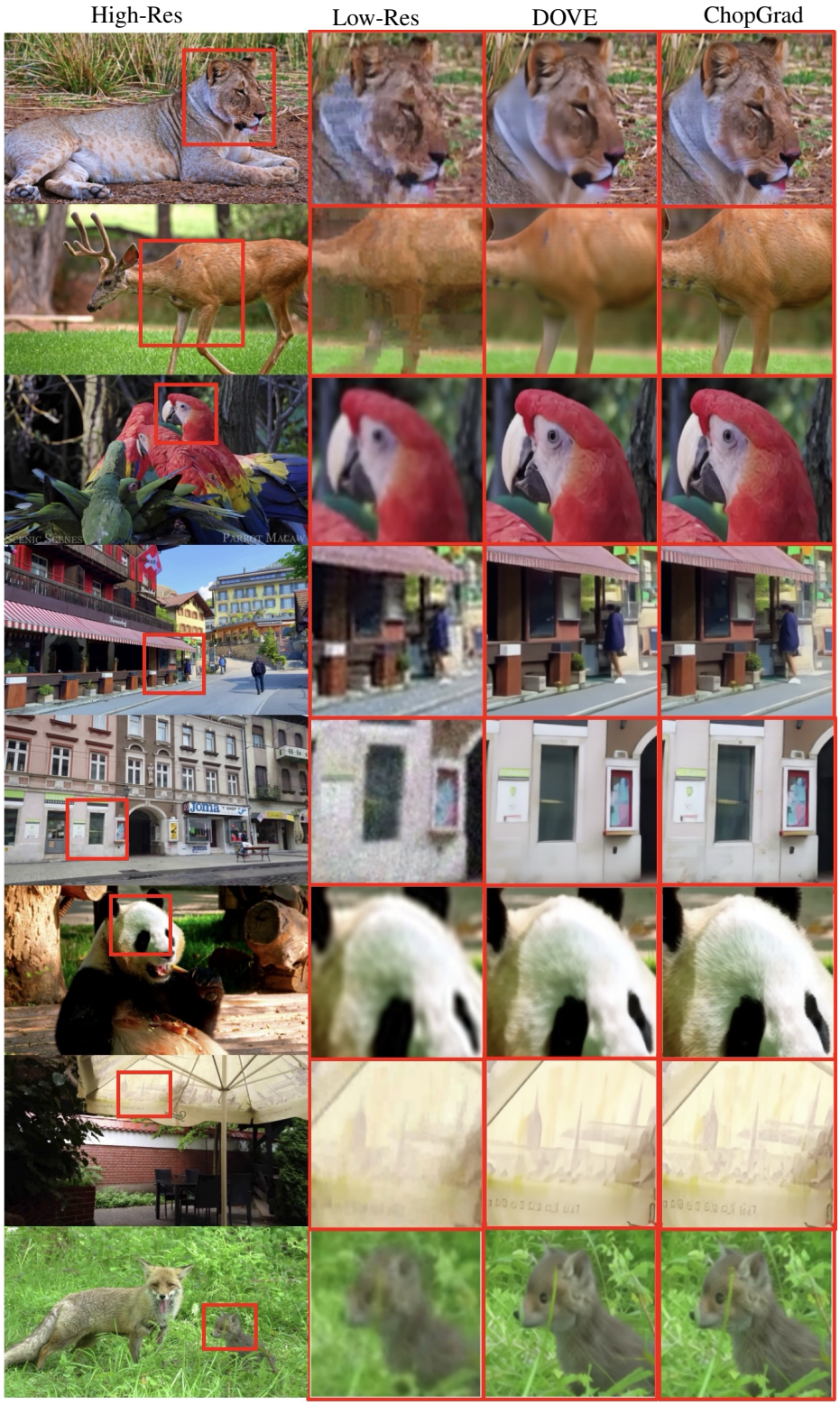

Figure 14. Additional Video Super-Resolution Comparison. Shown from left to right: high-resolution, low-resolution input, DOVE [12], and the proposed approach, *ChopGrad*. *ChopGrad* synthesizes fine textures better and reduces motion blur, especially in regions with high-frequency details like fur, hair, cloth, and clouds.

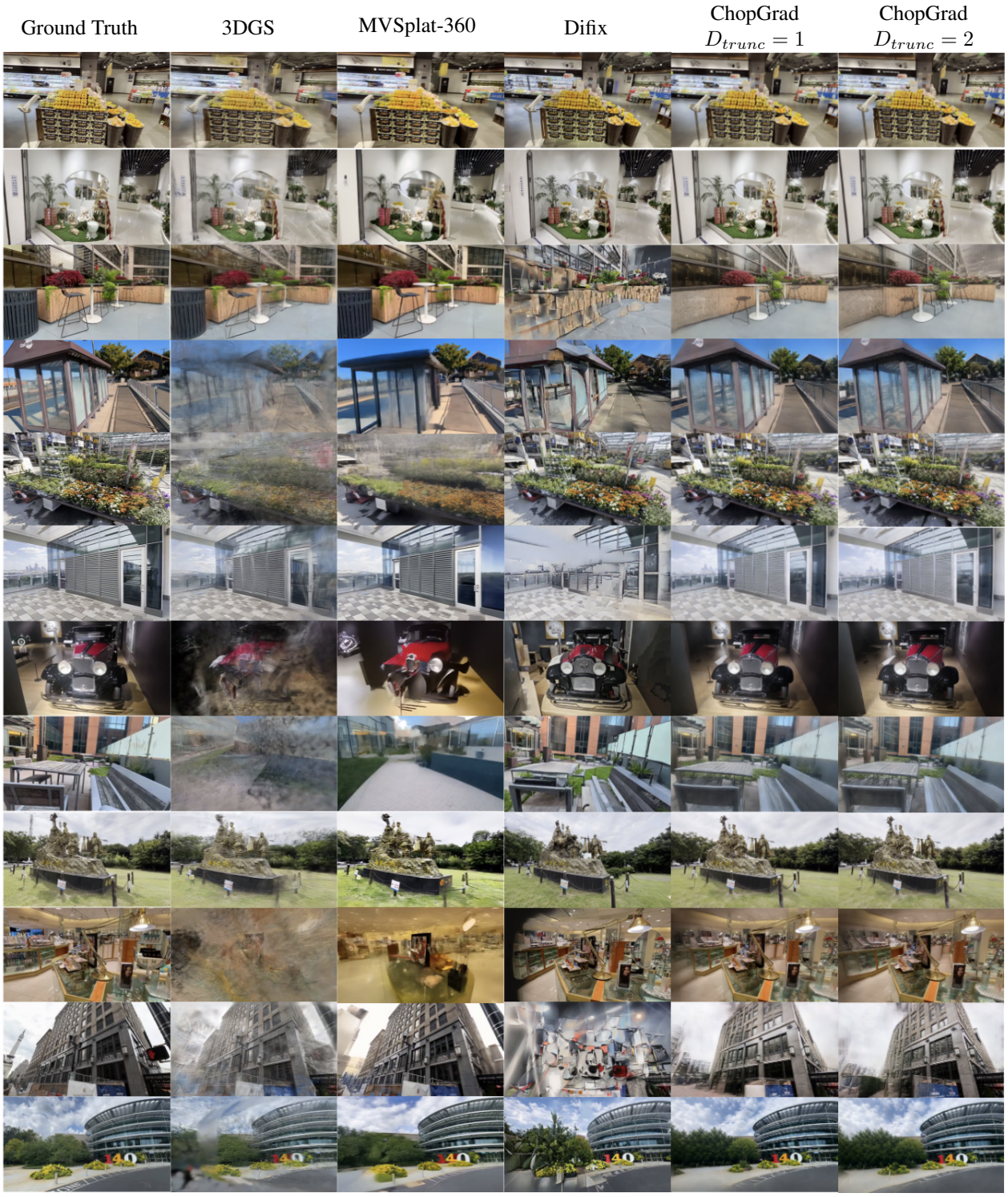

Figure 15. Additional Qualitative Results for Artifact Removal in Novel View Synthesis on the DL3DV-Benchmark Dataset [33]. Ground truth video frames and 3DGS model renders are shown on the left. Results for MVSplat-360 [11] and Difix [61] are presented alongside the *ChopGrad* with a truncation distance of 1 and 2. *ChopGrad* corrects significantly more artifacts than other methods (e.g., fourth row from the top) with fewer hallucinations (e.g., 5th row from the bottom), and maintains temporal consistency over the entire video sequence.

Figure 16. Additional Video Inpainting Comparison on D3LDV Dataset. Shown from left to right: VACE, *ChopGrad*, Ground Truth. Training with *ChopGrad* reduces hallucinations despite $50\times$ lower inference budget.

Figure 17. Additional Video Inpainting Comparison on Waymo Dataset. Shown from left to right: VACE, *ChopGrad*, Ground Truth. Training with *ChopGrad* reduces hallucinations despite $50\times$ lower inference budget.

Figure 18. Additional Video Inpainting Comparison on Waymo-Bbox Task. In this task, 50% of the vehicles are randomly selected for masking. Shown from left to right: VACE, *ChopGrad*, Ground Truth. Training with *ChopGrad* reduces hallucinations despite $50\times$ lower inference budget.

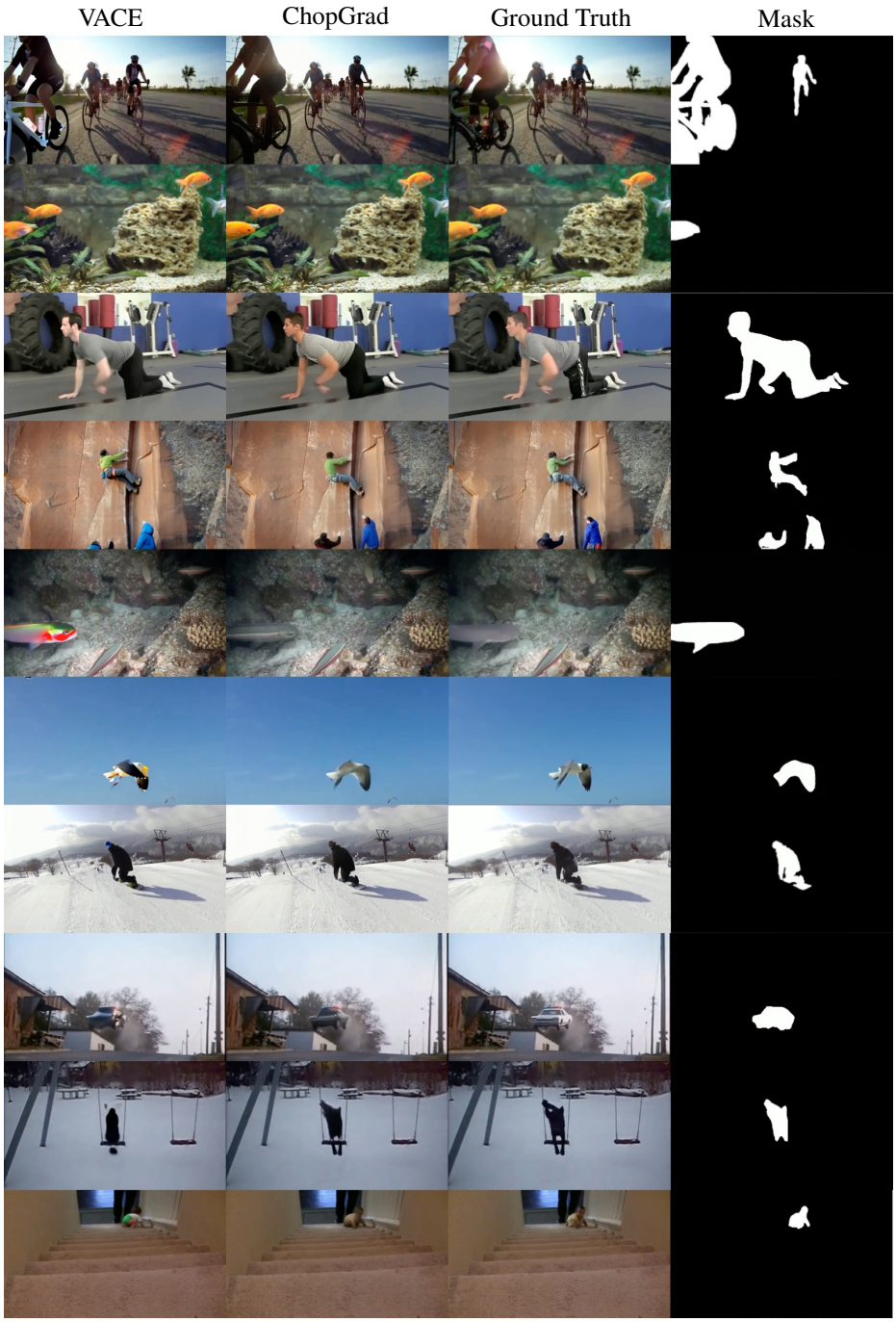

Figure 19. Additional Video Inpainting Comparison on ROVI Dataset. Shown from left to right: VACE, *ChopGrad*, Ground Truth. Training with *ChopGrad* reduces hallucinations despite $50\times$ lower inference budget.

Figure 20. Additional Controlled Driving Video Generation Comparison. Shown from left to right: Naive Insertion, Mirage [55], *ChopGrad*, and Ground Truth. The top six rows demonstrate that training with *ChopGrad* increases realism by improving lighting and shadows, and removing more Gaussian Splat artifacts. The bottom two rows, which have been cropped, demonstrate that training with *ChopGrad* enables the model to make stronger collections in the presence of very poor vehicle model quality – note that these very poor vehicle models are relatively rare in the dataset.