Grid4D: 4D Decomposed Hash Encoding for High-Fidelity Dynamic Gaussian Splatting

Jiawei Xu[1][†], **Zexin Fan**[1][†], **Jian Yang**[1][§][*], **Jin Xie**[2][3][‡][*]

[1]PCA Lab, VCIP, College of Computer Science, Nankai University
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[3]School of Intelligence Science and Technology, Nanjing University, Suzhou, China
[†]{jiaweixu, zexin_fan}@mail.nankai.edu.cn [‡]csjxie@nju.edu.cn [§]csjyang@nankai.edu.cn

Abstract

Recently, Gaussian splatting has received more and more attention in the field of static scene rendering. Due to the low computational overhead and inherent flexibility of explicit representations, plane-based explicit methods are popular ways to predict deformations for Gaussian-based dynamic scene rendering models. However, plane-based methods rely on the inappropriate low-rank assumption and excessively decompose the space-time 4D encoding, resulting in overmuch feature overlap and unsatisfactory rendering quality. To tackle these problems, we propose Grid4D, a dynamic scene rendering model based on Gaussian splatting and employing a novel explicit encoding method for the 4D input through the hash encoding. Different from plane-based explicit representations, we decompose the 4D encoding into one spatial and three temporal 3D hash encodings without the low-rank assumption. Additionally, we design a novel attention module that generates the attention scores in a directional range to aggregate the spatial and temporal features. The directional attention enables Grid4D to more accurately fit the diverse deformations across distinct scene components based on the spatial encoded features. Moreover, to mitigate the inherent lack of smoothness in explicit representation methods, we introduce a smooth regularization term that keeps our model from the chaos of deformation prediction. Our experiments demonstrate that Grid4D significantly outperforms the state-of-the-art models in visual quality and rendering speed. Project page: `https://jiaweixu8.github.io/Grid4D-web/`.

1 Introduction

Dynamic scene rendering aims to construct dynamic scenes from images with specific camera poses and timestamps, allowing rendering from arbitrary viewpoints and moments. Traditional methods use Neural Radiance Field (NeRF) [26] and deformation fields to reconstruct dynamic scenes for arbitrary rendering. However, these works rely on predicting deformations with the over-smooth full Multilayer Perceptron (MLP) [31, 40, 48, 10, 19, 28, 55, 47, 16, 5, 18, 1, 38, 23, 45, 21, 4, 53], resulting in slow training speeds and artifacts in rendering quality. To address these challenges, explicit representations such as planes [3] and hash encoding [27] have been introduced to enhance the rendering of dynamic scenes [33, 9, 2, 46, 8, 41, 11, 35, 36]. The explicit representations store the intermediate features generated by the partial forward propagation process in a grid-like format. This approach allows us to obtain intermediate features by directly interpolating the cached features based on the input, bypassing the need for the full forward propagation process. In addition to reducing computing resource consumption, the inherent flexibility of explicit representation offers advantages in rendering more complex scenes.

---

[*]Corresponding authors.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).
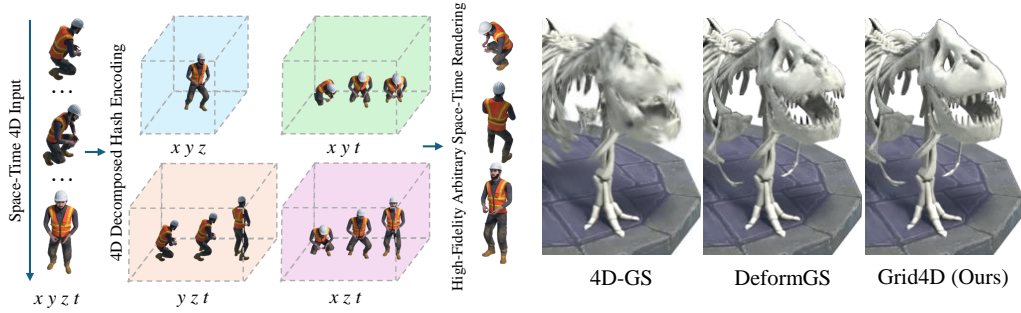
Figure 1: We propose a novel explicit representation method for dynamic scene rendering that decomposes the space-time 4D encoding into the 3D format without the unsuitable low-rank assumption. We achieve significant improvements over the state-of-the-art models [44, 50] in rendering quality.

Recently, Gaussian splatting [13] achieved fast and high-fidelity rendering of static scenes. Additionally, many works have employed Gaussian splatting for dynamic scene rendering by deforming Gaussians based on the timestamp [44, 50, 22, 17, 15, 7, 25, 49, 6, 20, 24, 12, 52, 37]. Deforming Gaussians through pre-defined functions is an effective way to reconstruct dynamic scenes with sufficient viewpoints [22, 17]. Additionally, implicit and explicit neural networks are more popular for deforming Gaussians in general cases [44, 50, 12]. However, fully MLP-based implicit neural networks have limited learning capacity because of their over-smooth inherent property, thereby struggling to render several complex scenes and details effectively. Hence, explicit representation might be an available method to address these problems. Prior works such as 4D-GS [44] use plane-based explicit representations to predict Gaussian deformations, decomposing the 4D space-time encoding into a format comprising six 2D planes, but the performance remains unsatisfactory. We consider that the plane-based methods for Gaussian deformation prediction are based on the low-rank assumption which assumes that the features for the deformations have a great deal of commonality and could be factorized into a very low-rank format [33, 9, 2, 44]. As shown in Figure 2, when facing Gaussians with massive overlapping coordinates, the over-decomposition makes the features have excessive overlap which limits their discriminability for deformation prediction. Therefore, such overlap might block the model from predicting different deformations, resulting in low rendering quality.

To address these problems, we present Grid4D, a novel model with high dynamic scene rendering quality based on Gaussian splatting [13]. Our approach leverages hash encoding [27] and proposes a new explicit representation method. Unlike the plane-based explicit representations relying on the unsuitable low-rank assumption, as shown in Figure 1, we decompose the 4D encoding into one spatial 3D hash encoding and three temporal 3D hash encodings. Figure 2 illustrates our proposed 4D decomposed hash encoding reduces overlap arising from the over-decomposed plane-based methods, resulting in more discriminative features. Notably, our encoder generates two types of features: spatial features, representing static information across the timeline, and temporal features, capturing dynamic information. For aggregation, we design a novel attention mechanism, directional attention, which leverages spatial features to generate attention scores in a directional range. This directional attention aligns with the observation that deformation consistency within each scene component often varies across different components, and the attention from the spatial features could better help the model fit such differences. However, like other explicit representation models, Grid4D often lacks smoothness. To address this issue, we propose a novel training strategy incorporating smooth regularization which mitigates chaotic deformation predictions to enhance rendering clarity.

We compare Grid4D with several state-of-the-art dynamic scene rendering models. Figure 1 and the experimental results show that Grid4D outperforms other models significantly in both visual quality and rendering speed. In general, the contribution of this paper can be summarized as the following.

- We propose a novel explicit representation method for dynamic scene rendering. By decomposing the 4D encoding into four 3D encodings, our 4D decomposed hash encoder effectively represents the features without relying on the low-rank assumption.
- We design a novel attention module for spatial and temporal feature aggregation. The directional attention module aligns with the variations in deformation consistency across different scene components, thereby enhancing deformation prediction accuracy.

2

- We employ a smooth training strategy to ensure the smoothness of our model. The smooth regularization effectively mitigates chaotic deformation predictions, resulting in high clarity in the rendered images produced by Grid4D.

2 Related Works

NeRF-Based Dynamic Scene Rendering. NeRF [26] reconstructs light fields of static scenes by implicit representations and achieves significant visual improvements. To extend NeRF capabilities for reconstructing dynamic scenes, applying implicit deformation fields to static models finds widespread use in dynamic scene rendering [31]. To model dynamic scenes more accurately, various studies segment a scene into components with different attributes for different modeling [10, 38]. Moreover, several works apply higher-dimensional latent codes for the network input [16, 28] and incorporate additional supervision such as flow supervision across frames [40, 11, 21, 5, 18, 48, 19, 4] and motion mask supervision [47]. Meanwhile, focusing on modeling rigid objects is important in improving accuracy because of their unique physical properties and prevalence in most scenes [38, 53]. Additionally, some research addresses the problems of dynamic scene models in several challenging scenes such as dynamic human modeling [1], specular objects [47] and the scenes without camera poses [23]. However, implicit representations based on full MLPs suffer from the over-smoothing inherent property and require time-consuming training processes. On the other hand, explicit representations, such as Triplanes [3] and Hash Encoding [27], enhance NeRF by improving both visual quality and training speed. A popular technique for plane-based explicit representations in dynamic scene rendering is decomposing 4D inputs into six 2D inputs [9, 2, 33, 46, 35]. Also, hash encoding and 3D grid explicit representations can assist MLPs in predicting deformations with faster speed and higher precision [8, 41, 11, 42, 29, 36].

Gaussian-Based Dynamic Scene Rendering. Recently, Gaussian splatting [13] models static scenes by Gaussian points, achieving both fast training and high visual quality. When it comes to dynamic scene rendering, using 4D Gaussians or deforming Gaussians with pre-defined functions perform well in the cases with sufficient viewpoints [49, 6, 22, 17, 25]. Alternatively, deforming the attributes of 3D Gaussians according to timestamps with neural networks has led to better outcomes in general dynamic scene rendering [50, 44, 15, 7, 20, 24, 12, 52, 37]. Fully MLP-based deformation fields achieve high rendering quality [50] but suffer from the over-smooth inherent property, resulting in the failure of some detail rendering and complex scenes. Explicit representation models, for example, 4D-GS [44], utilize the planes-based methods as the deformation field. Although plane-based representations are more flexible, they are based on the unsuitable low-rank assumption, leading to massive feature overlap and rendering artifacts. Our work mainly focuses on tackling the unsuitable low-rank assumption inherent in plane-based explicit representations to improve the rendering quality of Gaussian-based models.

3 Method

3.1 Prelimaries: Gaussian Splatting

Gaussian splatting [13] is a static scene rendering model, known for its high training speed and visual quality. This model assumes that the scene is composed of 3D Gaussian kernels with $\{\boldsymbol{\mu}, S, R, \sigma, \mathbf{c}\}$, corresponding to the position, scaling, rotation, opacity, and color. Notably, the color attribute is defined by the spherical harmonic coefficients (SH). To render the scene, by using a view transform matrix W and a projective Jacobian matrix J, Gaussians can be splatted onto camera planes [56, 51].

$$\Sigma' = JW\Sigma W^T J^T, \ \Sigma = RSS^T R^T \tag{1}$$

where Σ' is the covariance matrix in camera planes and Σ is the original Gaussian covariance which can be calculated by the scaling and rotation attributes. Finally, supposing that the pixel on the camera planes is $\mathbf{p}$, the splatted Gaussians can be rendered by the volume rendering equation,

$$\mathbf{C}(\mathbf{p}) = \sum_{i=1}^{N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \alpha_i = \sigma_i e^{-\frac{1}{2}(\mathbf{p}-\boldsymbol{\mu}_i^p)^T \Sigma_i'^{-1}(\mathbf{p}-\boldsymbol{\mu}_i^p)} \tag{2}$$

where $\boldsymbol{\mu}^p$ is the projected coordinates of the 3D Gaussians, and N is the number of overlapped Gaussians on the pixel.

3

(a) comparison of encoding overlapping coordinates   (b) t-SNE and deformation visualization
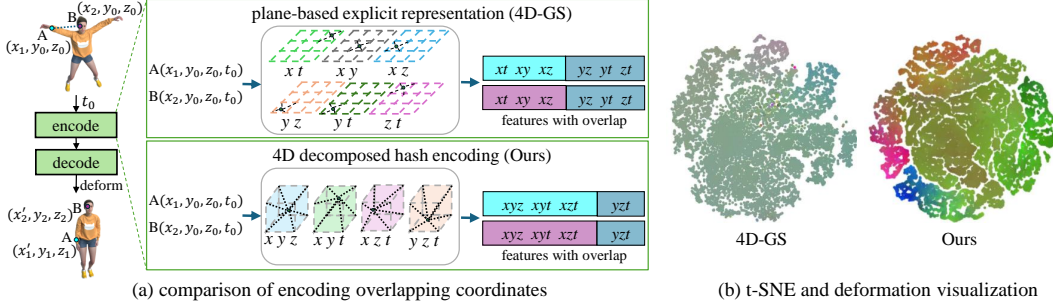
Figure 2: Comparison of our proposed 4D decomposed hash encoding with the plane-based explicit representation [44]. (a) Compared to the plane-based methods based on the low-rank assumption, our methods reduce the overlap ratio in the features from a half to a quarter when encoding points A and B with heavily overlapping coordinates. (b) is the t-SNE [39] visualization of all the features, and the colors denote the corresponding represented deformations. The diversity of colors demonstrates that the reduced overlap makes the features represent different deformations more effectively.

In optimization, adaptive density control is crucial for convergence. It involves pruning low-opacity Gaussians and densifying them based on the gradients and scaling. However, original Gaussian splatting cannot represent dynamic scenes and needs the help of deformation fields.

3.2   4D Decomposed Hash Encoding

Dynamic scene rendering involves deforming Gaussians according to a 4D coordinate (x, y, z, t) input, where t represents the timestamp and (x, y, z) means the position of a Gaussian. Instead of employing the over-smooth fully MLP-based implicit representations, we use explicit representation for Grid4D. However, existing plane-based explicit representation relies on the unsuitable low-rank assumption which overly decomposes the (x, y, z, t) encoding into $(x, y), (y, z), (x, z), (x, t), (y, t), (z, t)$ plane encodings [33, 9, 2, 44]. As shown in Figure 2(a), for instance, considering the Gaussians A and B with the same y and z coordinates. The plane-based method has the same encoded features in the $(y, z), (y, t), (z, t)$ planes. Such a high overlap ratio might lead to the low discriminability of the features and block the model from fitting different deformations accurately. To address this problem, directly removing the decomposition by simply adding the time dimension to the traditional 3D grid for the 4D hyper-grid hash encoding is a possible way. However, the 4D hyper-grid hash encoding leads to high collision rates due to the high space complexity $O(n^4)$ of the 4D hyper-grid [41]. Therefore, thoroughly eliminating the overlap might not be an available solution.

Tri-axial 4D Decomposed Grid. To address this problem, we propose a novel decomposition approach that decomposes the 4D encoding (x, y, z, t) into four 3D hash encodings $(x, y, z), (x, y, t), (y, z, t), (x, z, t)$. The decomposition allows us to work with fewer parameters, which reduces the space complexity from $O(n^4)$ to $O(n^3)$ without relying on the low-rank assumption. As shown in Figure 2(a), the tri-axial decomposition can effectively reduce the overlap ratio from a half to a quarter, thereby enhancing each feature to represent the corresponding deformation. Figure 2(b) demonstrates that the features encoded by our methods are more discriminative for deformation prediction than plane-based methods.

Multiresolution Hash Encoding. In the original hash encoding technique [27], the grid employed in the encoder has the same resolution across all dimensions. Consistent resolutions could be suitable for static scene rendering, where the isotropic sampling assumption holds in the 3D space. Nevertheless, the sampling of the 4D space is usually anisotropic, which is usually sparse in the time dimension. Therefore, in our implementation, the temporal 3D encodings $(x, y, t), (y, z, t), (x, z, t)$ have different resolutions in the t dimension to account for this sparsity. Following the InstantNGP [27], we set the multiple resolutions of each dimension in a geometric progression:

$$N_l = \lfloor N_{min} \cdot b \rfloor, \; b = \exp\left(\frac{\ln N_{max} - \ln N_{min}}{L - 1}\right) \tag{3}$$

where N_{min}, N_{max} is the coarsest and finest resolutions, l is level number, L is the max level, and N_l is the resolution we select. The grid voxel positions for the input $\mathbf{x}$ could be calculated by rounding

4
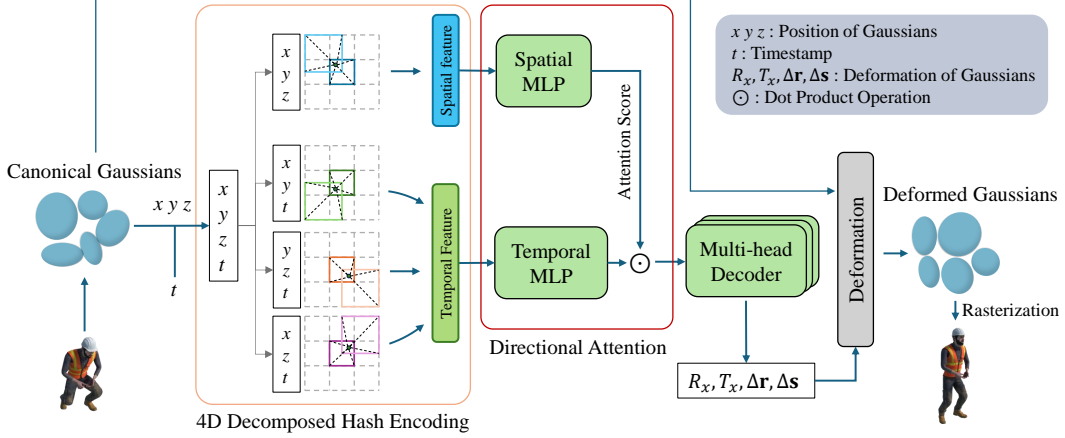
Figure 3: The overview of Grid4D. Given the canonical Gaussians and the timestamp, we first encode the decomposed input separately. Then we apply the directional attention scores generated by the spatial static features to the temporal dynamic features, and we decode the features with a tiny multi-head MLP. Finally, the Gaussians deformed by the predicted deformations are splatted by the differentiable rasterization operation [13] to render the images for supervision.

down and up in each level $\lfloor \mathbf{x}_l \rfloor = \lfloor \mathbf{x} \cdot N_l \rfloor, \lceil \mathbf{x}_l \rceil = \lceil \mathbf{x} \cdot N_l \rceil$. The voxels in each level could be obtained from the hash table by hashing the corresponding positions:

$$h_l(\mathbf{x}_l) = \left(\bigoplus_{i=1, x_i \in \mathbf{x}_l}^{d} x_i \pi_i \right) \mod T_l \tag{4}$$

where $\bigoplus$ is the bit-wise XOR operation, d is the dimension of the input, π_i are unique large prime numbers, T_l is the size of the level l hash table. Then the encoded features could be calculated by the trilinear interpolation of the grid voxel values. Generally, the encoded features of the 4D input (x, y, z, t) include the spatial and temporal features from the spatial grid hash encoder G_{xyz} and temporal grid hash encoders $G_{xyt}, G_{yzt}, G_{xzt}$ respectively.

3.3   Multi-head Directional Attention Decoder

Our 4D decomposed hash encoding generates two types of features: temporal features and spatial features. The temporal features represent the information related to the timestamp while the spatial features represent the common information across the timeline. The Gaussians representing different scene components often have various deformations in almost every timestamp. Therefore, the spatial features could be used to help the model fit such variations, and we design the directional attention module for the spatial and temporal feature aggregation.

Directional Attention. We infer the attention features from the spatial grid hash encoder G_{xyz} with a tiny spatial MLP f_s, and generate the score $\mathbf{a}$ through the following formula,

$$\mathbf{a} = 2 \cdot \Phi(\mathbf{h}_{xyz}) - 1, \ \mathbf{h}_{xyz} = f_s \circ G_{xyz}(x, y, z) \tag{5}$$

where Φ is the Sigmoid function. We consider that several components probably have entirely opposite deformations against the neighboring Gaussians. For example, the Gaussians for the shadows often have opposite motions relative to the objects. Therefore, different from the common range $(0, 1)$ of the attention score $\mathbf{a}$, we scale it to a directional range $(-1, 1)$ to represent neighboring deformations with opposite directions, thereby enhancing the representation ability of the attention mechanism.

Then we apply the attention score to the activated deformation features encoded by the three temporal grid hash encoders $G_{xyt}, G_{yzt}, G_{xzt}$ and a tiny temporal MLP f_t.

$$\mathbf{h} = \mathbf{a} \odot f_t(G_{xyt}(x, y, t), G_{yzt}(y, z, t), G_{xzt}(x, z, t)) \tag{6}$$

where $\odot$ is the dot product operation. Finally, we get the deformation features $\mathbf{h}$ with high representation ability. Our experiments demonstrate that our attention module outperforms the architecture

5

which either directly decodes the concatenation of the spatial and temporal features or uses the common range $(0, 1)$ of the attention score.

Multi-head Deformation Decoder. The decoder is required to decode the features $\mathbf{h}$ to get the Gaussian deformation. Different from the prior works [44, 50], we use a tiny multi-head MLP D to decode the features and predict the position deformation with a rotation matrix R_x and a translation matrix T_x as [12]. Finally, we deform the position, scaling, and rotation of the Gaussians in the canonical space with the predicted deformation.

$$\boldsymbol{\mu}' = R_x\boldsymbol{\mu} + T_x, \; S' = S + \Delta\mathbf{s}, \; R' = R + \Delta\mathbf{r}, \; D(\mathbf{h}) = \{R_x, T_x, \Delta\mathbf{r}, \Delta\mathbf{s}\}, \qquad (7)$$

We define the rotation matrix R_x with a quaternion for more accurate interpolation and stable optimization. Following Gaussian splatting [13], the deformed Gaussians could be rendered into images of specific timestamps via differentiable rasterization.

3.4 Training with Smooth Regularization

Although the proposed model architecture could effectively predict the Gaussian deformation, the 4D decomposed hash encoder still suffers from the lack of smoothness, a common challenge in most explicit representation methods. We consider that the MLP decoder has the smooth inherent property and does not require additional smoothing. Therefore, we set our regularization in the feature space without involving the MLP decoder inference for higher efficiency. Generally, to regularize the hash encoder, we propose a novel smooth regularization loss.

$$\mathcal{L}_r = ||G_{xyzt}(x, y, z, t) - G_{xyzt}(x + \epsilon_x, y + \epsilon_y, z + \epsilon_z, t + \epsilon_t)||_2^2 \qquad (8)$$

where $(\epsilon_x, \epsilon_y, \epsilon_z, \epsilon_t)$ is the small random perturbation for the input (x, y, z, t) respectively, and G_{xyzt} is the concatenation of four grid hash encoders. This regularization enforces similarity among encoded features in neighboring regions, thereby making the nearby Gaussians have similar deformations. Due to the difference of spatial and temporal encoding, we use a different regularization setting for the spatial encoding for several cases. Notably, to improve the efficiency, we randomly select partial Gaussians for the regularization instead of using them all. Our experiments demonstrate that this smooth regularization effectively mitigates the deformation chaos, leading to significantly improved rendering clarity.

In general, similar to Gaussian splatting [13], our total loss function can be summarized as the weighted sum of L1 color loss, D-SSIM loss, and the proposed smooth regularization term.

$$\mathcal{L} = (1 - \lambda_c)\mathcal{L}_1 + \lambda_c\mathcal{L}_{D-SSIM} + \lambda_r\mathcal{L}_r \qquad (9)$$

where λ_c, λ_r are the hyperparameters to balance the losses. Following [50], we use the detached Gaussian positions for deformation prediction, which results in better performance. Also, similar to prior works [50, 44], we initialize the static canonical Gaussians without deformation at the beginning of the training process. Specifically for SfM [32] initialized Gaussians, we shorten or remove the static initialization process. We apply the same adaptive density controller and opacity resetting mechanism as Gaussian splatting [13]. The pipeline of Grid4D is illustrated by Figure 3.

4 Experiments

In this section, we introduce our experiments conducted on a single RTX 3090 GPU. We build our code mainly on PyTorch [30], while we implement our 4D decomposed hash encoder with CUDA/C++. More experimental results and analysis can be found in the supplementary.

4.1 Experimental Setup

Datasets. We evaluate Grid4D on two popular datasets. D-NeRF [31] dataset is a public monocular synthetic dataset that provides accurate and time-varying camera poses. HyperNeRF [28] dataset is a public real-world dataset captured by one or two moving cameras. Neu3D [16] dataset is a public dataset captured by multiple cameras with fixed poses. However, different from synthetic datasets, the camera poses of the HyperNeRF and Neu3D datasets are estimated by COLMAP [32], which is not accurate. We set the rendering resolutions of the D-NeRF, HyperNeRF and Neu3D datasets to 800×800, 536×900 and 1352×1024 respectively. Notably, we find several mistakes in the

6

| TiNeuVox | 4D-GS | DeformGS | Grid4D (Ours) | Ground Truth |

Figure 4: Qualitative comparisons on the synthetic D-NeRF dataset [31] with our baselines [8, 44, 50].

ground truth of the 'Lego' scene in the D-NeRF dataset, as shown in the last row of Figure 4, so we ignore this scene in all quantitative comparisons of rendering quality.

Baselines. We compare Grid4D with several state-of-the-art models [2, 8, 44, 50, 12]. HexPlane [2] and TiNeuVox [8] are NeRF-based dynamic scene rendering models, utilizing plane-based and 3D grid explicit representations respectively. 4D-GS [44] and DeformGS [50] are Gaussian-based models, employing plane-based explicit representation and fully MLP-based implicit representation for the deformation fields respectively. SC-GS [12] is a model built on DeformGS [50], and proposes to use sparse control points for better dynamic scene rendering and edit.

Hyperparameters. For all datasets, we configure the resolution of the spatial grid hash encoder to span from 16 to 2048 across 16 levels. Meanwhile, the max level number L of temporal grid hash encoders remains consistent at 32. We set λ_c and λ_r to 0.2 and 0.5 for common scenes and follow a similar learning rate schedule as DeformGS [50, 13].

4.2 Comparisons

Comparison of Visual Quality. We compare Grid4D with the state-of-the-art models on the synthetic D-NeRF [31] dataset (Table 1 and Figure 4), the real-world HyperNeRF [28] dataset (Table 2 and Figure 5) and the real-world Neu3D [16] dataset (Table 3 and Figure 6). The PSNR, SSIM [43], LPIPS [54](VGG [34]), and MS-SSIM are the metrics denoting visual quality. Notably, the DeformGS [50] model fails to construct several HyperNeRF scenes with large motions and imprecise camera poses, as mentioned in their paper. Several failed cases can be found in Section B of the supplementary, and we consider that this is also due to the over-smooth inherent property of fully MLP-based implicit representation.

Due to the inherent flexibility of the explicit representation, the results of the 'Hook' scene show that Grid4D has a stronger ability to reconstruct fine structures than DeformGS [50] which is based on the implicit representation. We also apply the sparse control points in SC-GS [12] to our model and build SC-GS on Grid4D rather than DeformGS for further evaluation. We refer to it as 'Grid4D + SC', and observe an improvement in comparison to Grid4D and SC-GS as list in the last three rows of Table 1. Thanks to our 4D decomposed hash encoding, when facing the scenes with complex motions

7

Table 1: Quantitative comparisons on the synthetic D-NeRF [31] dataset. The higher PSNR (↑), higher SSIM (↑) and lower LPIPS (↓) denote better rendering quality. The color of each cell shows the best and the second best.

Model	Bouncing Balls			Hell Warrior			Hook			Jumping Jacks		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
HexPlane [2]	40.36	0.992	0.031	24.30	0.944	0.073	28.26	0.955	0.052	31.74	0.974	0.036
TiNeuVox [8]	40.28	0.992	0.042	27.29	0.964	0.076	30.51	0.959	0.060	33.46	0.977	0.041
4D-GS [44]	40.77	0.994	0.015	28.80	0.974	0.037	32.95	0.977	0.027	35.50	0.986	0.020
DeformGS [50]	40.91	0.995	0.009	41.34	0.987	0.024	37.06	0.986	0.016	37.66	0.989	0.013
SC-GS [12]	41.59	0.995	0.009	42.19	0.989	0.019	38.79	0.990	0.011	39.34	0.992	0.008
Grid4D (Ours)	42.62	0.996	0.008	42.85	0.991	0.015	38.89	0.990	0.009	39.37	0.993	0.008
Grid4D + SC	42.17	0.995	0.008	42.81	0.990	0.017	40.26	0.992	0.008	39.58	0.993	0.008

Model	Mutant			Standup			Trex			Mean		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
HexPlane [2]	33.66	0.982	0.028	34.12	0.983	0.019	31.01	0.976	0.028	31.92	0.972	0.038
TiNeuVox [8]	32.07	0.961	0.048	34.46	0.980	0.033	31.43	0.967	0.047	32.78	0.972	0.050
4D-GS [44]	37.75	0.988	0.016	38.15	0.990	0.014	33.95	0.985	0.022	35.41	0.985	0.021
DeformGS [50]	42.47	0.995	0.005	44.14	0.995	0.007	37.56	0.993	0.010	40.16	0.991	0.012
SC-GS [12]	43.43	0.996	0.005	46.72	0.997	0.004	39.53	0.994	0.009	41.65	0.993	0.009
Grid4D (Ours)	43.94	0.996	0.004	46.28	0.997	0.004	40.01	0.995	0.008	42.00	0.994	0.008
Grid4D + SC	44.07	0.996	0.004	46.87	0.997	0.004	41.12	0.995	0.008	42.41	0.994	0.008

Table 2: Quantitative comparison on the validation rig part (Rig) and the interpolation part (Interpolation) of the real-world HyperNeRF [28] dataset. The higher PSNR (↑) and higher MS-SSIM (↑) denote better rendering quality. The color of each cell shows the best and the second best.

Model	Rig(4 scenes)		Interpolation(6 scenes)	
	PSNR	MS-SSIM	PSNR	MS-SSIM
TiNeuVox [8]	24.20	0.836	27.08	0.922
4D-GS [44]	24.99	0.838	27.54	0.912
Grid4D (Ours)	25.50	0.856	28.56	0.933

| TiNeuVox | 4D-GS | DeformGS | Grid4D (Ours) | Ground Truth |
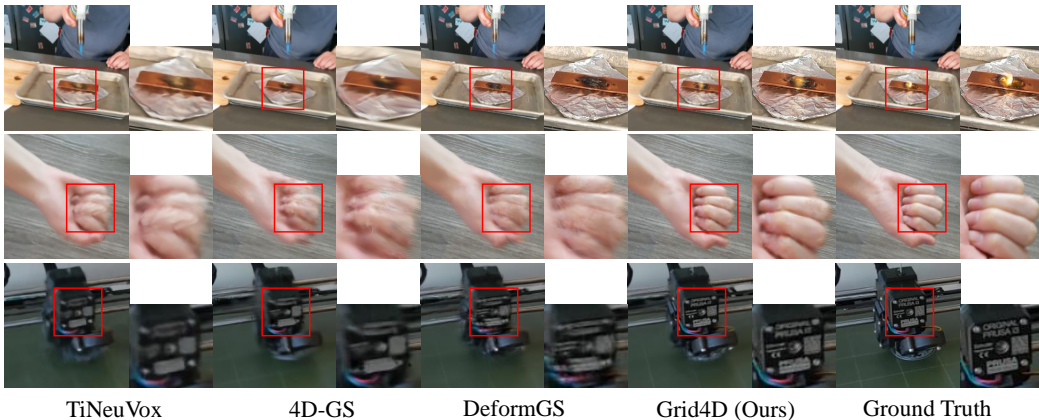
Figure 5: Qualitative comparisons on the real-world HyperNeRF [28] dataset.

and Gaussians with heavily overlapping coordinates, such as 'JumpingJacks', Grid4D predicts the deformations much more accurately than 4D-GS [44] which is built on the planed-based explicit representation relying on the unsuitable low-rank assumption.

Comparison of Rendering Speed. Comparing Frames Per Second (FPS) directly might not be a fair experiment because the number of Gaussians is quite different among different models. Therefore, we list both the FPS and the corresponding Gaussian count in Table 4. Despite the acceleration provided by the CUDA/C++ implementation in Grid4D, our proposed explicit representation makes

8

Table 3: Quantitative comparison on the real-world Neu3D [16] dataset. The higher PSNR (↑) and higher SSIM (↑) denote better rendering quality. The color of each cell shows the <mark>best</mark>.

| Model | Coffee Martini | | Cook Spinach | | Cut Beef | | Flame Salmon | | Flame Steak | | Sear Steak | |
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
4D-GS [44]	27.34	0.898	32.50	0.942	32.26	0.942	27.99	0.902	32.54	0.951	33.44	0.954
Grid4D (Ours)	28.30	0.898	32.58	0.948	33.22	0.950	29.12	0.908	32.56	0.955	33.16	0.957

| 4D-GS | Grid4D (Ours) | Ground Truth |

Figure 6: Qualitative comparisons on the real-world Neu3D [16] dataset.

Table 4: Rendering speed comparison on the synthetic D-NeRF [31] dataset. We report the FPS based on the number of Gaussian points. Compared to other models, our model still achieves high rendering speed and real-time rendering when facing a much larger amount of Gaussians.

FPS / Num(k)	Balls	Warrior	Hook	Jumping	Lego	Mutant	Standup	Trex
4D-GS [44]	182 / 28	168 / 40	91 / 39	207 / 24	104 / 93	173 / 38	201 / 27	151 / 68
DeformGS [50]	37 / 180	161 / 37	43 / 150	71 / 90	30 / 289	49 / 169	77 / 81	30 / 217
Grid4D (Ours)	91 / 192	334 / 46	79 / 210	241 / 68	64 / 302	157 / 126	170 / 100	86 / 254

Grid4D exhibit significantly faster rendering performance. When facing a huge number of Gaussians, Grid4D maintains high rendering speed and achieves real-time rendering. However, Grid4D has no improvement in the training speed compared to DeformGS [50]. Although we do not use all Gaussians for the regularization, the smooth regularization requires Grid4D to encode the input twice, which slows down the training process. Meanwhile, we find that Grid4D's accurate deformation predictions often lead to an increase in the number of Gaussians, contributing to time overhead. Nevertheless, Grid4D needs less GPU memory for training than DeformGS [50], and the extra computational cost has little influence on training in comparison to the significant improvements in rendering quality. More details can be found in Section D of our supplementary.

4.3 Ablation Study and Analysis

To mitigate the distraction of imprecise camera poses, we mainly conduct our ablation studies on the synthetic D-NeRF [31] dataset. Table 5 and Figure 7 show the results of our ablations.

Ablation of 4D Decomposed Hash Encoding. The proposed 4D decomposed hash encoding splits the 4D input into four 3D inputs, encoding them separately without the unsuitable low-rank assumption and high space complexity. To demonstrate the advantages of our encoding method, we employ the simple 4D hyper-grid hash encoding in Grid4D w/o dec. The chaos in Figure 7(a) illustrates the rendering degradation caused by the high hash collision rate.

Ablation of Directional Attention. The directional attention helps Grid4D accurately predict the different deformations across different scene components. We scale the attention score to $(0, 1)$ for Grid4D w/o dir to demonstrate the advantage of the directional range $(-1, 1)$. We also compare our attention module with the simple architecture Grid4D w/o att which directly decodes the concatenation of the spatial and temporal features. In Figure 7(b), the shadow has obvious different deformations

9

Table 5: Quantitative ablation results on the synthetic D-NeRF [31] dataset. The color of each cell shows the <mark>best</mark> and the <mark>second best</mark>.

Model	w/o dec	w/o reg	w/o att	w/o dir	Grid4D (Ours)
PSNR	28.45	39.47	41.37	41.32	42.00
SSIM	0.949	0.991	0.993	0.993	0.994
LPIPS	0.055	0.012	0.009	0.009	0.008

w/o dec   w/o reg   Grid4D (Ours)   w/o att   w/o dir   Grid4D (Ours) Ground Truth

(a)   (b)
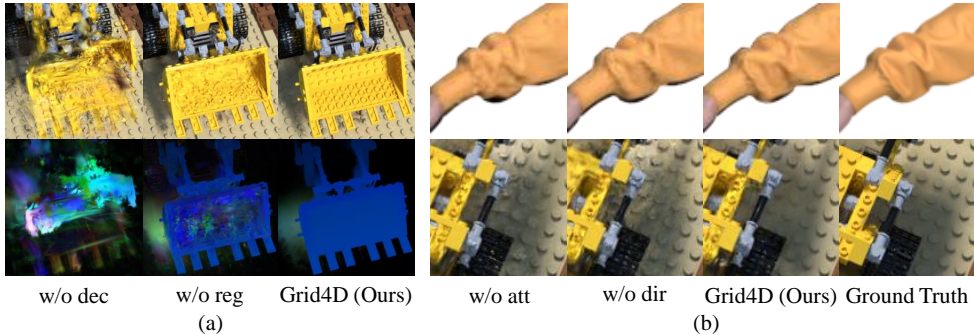
Figure 7: Qualitative results of our ablation studies. (a) is the ablation of the 4D decomposed hash encoding and the smooth regularization. The first row is the rendering results, and the second row is the visualization of the deformation. The similar colors in the deformation map mean similar deformation sizes on each axis. (b) is the ablation of the directional attention.

from the neighboring parts across the timeline. Our directional attention achieves high clarity in rendering the portion with the shadow, emphasizing its effectiveness in capturing such variations.

Ablation of Smooth Regularization. The proposed smooth regularization aims at mitigating the chaos of deformation prediction. We train Grid4D without the smooth regularization in Equation 8 and refer to the model as Grid4D w/o reg. The results in Figure 7(a) show that the regularization reduces the deformation artifacts caused by the lack of smoothness.

We conduct more ablation studies for the architecture and smooth regularization. We also visualize the intermediate results of our model. More results can be found in Section C of our supplementary.

5 Conclusion

In this paper, we have introduced Grid4D, a novel model for high-fidelity dynamic scene rendering. Grid4D utilizes the proposed 4D decomposed hash encoding without the unsuitable low-rank assumption and high space complexity. Additionally, the novel directional attention module effectively aggregates the spatial and temporal features for more accurate deformation prediction across different scene components. Moreover, we employed smooth regularization to mitigate chaos in deformation prediction, resulting in high rendering quality. Our experiments demonstrate that Grid4D achieves state-of-the-art performance and delivers high rendering speed for dynamic scene rendering. However, Grid4D has no improvement in training speed, and like the other dynamic scene rendering models, Grid4D might have artifacts when facing several dynamic scenes with complex and large motions. Addressing these challenges remains an area for future research.

Acknowledgements

The authors would like to thank the anonymous reviewers for their critical and constructive comments and suggestions. The authors also thank The Supercomputing Center of Nankai University (NKSC) for computation. This work was supported by the National Science Fund of China under Grant No. 62361166670 and Grant No. 62276144.

10

References

[1] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular RGB-D camera. *Advances in Neural Information Processing Systems*, 35:967–981, 2022.

[2] Ang Cao and Justin Johnson. HexPlane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.

[3] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021.

[4] Jaesung Choe, Christopher Choy, Jaesik Park, In So Kweon, and Anima Anandkumar. Spacetime surface regularization for neural dynamic scene reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17871–17881, 2023.

[5] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4D view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314. IEEE Computer Society, 2021.

[6] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4D Gaussian Splatting: Towards efficient novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2402.03307*, 2024.

[7] Bardienus P Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. MD-Splatting: Learning metric deformation from 4D Gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*, 2023.

[8] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[9] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[10] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.

[11] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16022–16033, 2023.

[12] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. SC-GS: Sparse-controlled Gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023.

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

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

[15] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. DynMF: Neural motion factorization for real-time dynamic view synthesis with 3D Gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023.

[16] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3D video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022.

[17] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian feature splatting for real-time dynamic view synthesis. *arXiv preprint arXiv:2312.16812*, 2023.

[18] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

11

[19] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4273–4284, 2023.

[20] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. GauFRe: Gaussian deformation fields for real-time dynamic novel view synthesis. *arXiv preprint arXiv:2312.11458*, 2023.

[21] Yiqing Liang, Eliot Laidlaw, Alexander Meyerowitz, Srinath Sridhar, and James Tompkin. Semantic attention flow fields for monocular dynamic scene decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21797–21806, 2023.

[22] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D reconstruction with dynamic 3D Gaussian particle. *arXiv preprint arXiv:2312.03431*, 2023.

[23] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13–23, 2023.

[24] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3D geometry-aware deformable Gaussian splatting for dynamic view synthesis. *arXiv preprint arXiv:2404.06270*, 2024.

[25] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[26] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[28] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[29] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4212–4221, 2023.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[31] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[32] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-Motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[33] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4D: Efficient neural 4D decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[35] Nagabhushan Somraj, Kapil Choudhary, Sai Harsha Mupparaju, and Rajiv Soundararajan. Factorized motion fields for fast sparse input dynamic view synthesis. *arXiv preprint arXiv:2404.11669*, 2024.

[36] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. NeRFPlayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023.

[37] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3DGStream: On-the-fly training of 3D Gaussians for efficient streaming of photo-realistic free-viewpoint videos. *arXiv preprint arXiv:2403.01444*, 2024.

12

[38] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-Rigid Neural Radiance Fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021.

[39] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[40] Chaoyang Wang, Lachlan Ewen MacDonald, Laszlo A Jeni, and Simon Lucey. Flow supervision for deformable NeRF. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21128–21137, 2023.

[41] Feng Wang, Zilong Chen, Guokang Wang, Yafei Song, and Huaping Liu. Masked space-time hash encoding for efficient dynamic scene reconstruction. *Advances in Neural Information Processing Systems*, 2023.

[42] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19706–19716, 2023.

[43] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[44] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[45] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-NeRF: Neural radiance fields for street views. *arXiv preprint arXiv:2303.00749*, 2023.

[46] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4K4D: Real-time 4D view synthesis at 4K resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20029–20040, 2024.

[47] Zhiwen Yan, Chen Li, and Gim Hee Lee. NeRF-DS: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023.

[48] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. BANMo: Building animatable 3D neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022.

[49] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023.

[50] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023.

[51] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[52] Heng Yu, Joel Julin, Zoltán Á Milacski, Koichiro Niinuma, and László A Jeni. CoGS: Controllable Gaussian splatting. *arXiv preprint arXiv:2312.05664*, 2023.

[53] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. STaR: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13144–13152, 2021.

[54] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[55] Ruiqi Zhang and Jie Chen. NDF: Neural deformable fields for dynamic human modelling. In *European Conference on Computer Vision*, pages 37–52. Springer, 2022.

[56] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001.

13

– Supplementary –

A Details of Experimental Setup

Network Architecture of Grid4D. The upper limit of the hash table size is 2^{19} for both spatial and temporal 3D grids. The feature dimension of each voxel is 2 for all grids. We set the spatial dimension resolutions of temporal grids according to the scale of the scene. We usually set the time dimension resolution to a value between a half and a quarter of the time samples. The architecture of our multi-head directional attention decoder is illustrated by Figure 8. The spatial and temporal MLPs only have one fully connected layer and one activation layer. For the multi-head deformation decoder, we set the depth to two for the HyperNeRF [28] dataset and one for the D-NeRF [31] dataset, including the output layer, and set all the widths to 256.
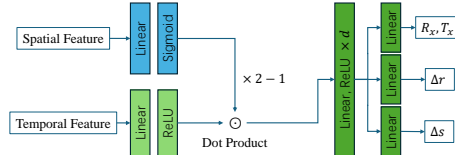
Figure 8: Architecture of our multi-head directional attention decoder.

Optimization. The scheduler of the learning rate primarily follows DeformGS [50, 13]. The loss weight λ_c and λ_r in Equation 9 is set to 0.2 and 0.5 for common scenes. Notably, the learning rate of the MLP decoder is determined based on the scale of the scene. Additionally, the learning rate of the grid hash encoders is set to 10~50 times larger than the MLP decoder. We use Adam [14] optimizer with $\beta = (0.9, 0.999)$ for training and set the background to black. Due to the differences between the spatial and temporal grids, we set different smooth regularization parameters for the (x, y, z) grid in several scenes. For the scenes in the HyperNeRF [28] and Neu3D [16] dataset, we use the SfM [32] points to initialize Gaussians.

Deformation Map. The formula for deformation maps is

$$\Delta x = \frac{f(x, t + \tau) - f(x, t)}{\tau} \tag{10}$$

where x is the canonical Gaussian position, and f is the deformation field. For all experiments, We set τ to 0.05, and limit the absolute value of Δx for color. This map shows the situation of predicted deformation: the similar colors of two parts mean similar deformation sizes on each axis.

B Additional Comparisons

Additional Results on D-NeRF [31] Dataset. We visualize more experimental results on the D-NeRF [31] dataset in Figure 13. We observe that Grid4D exhibits superior rendering quality compared to the state-of-the-art models.

Per Scene Results on HyperNeRF [28] Dataset. We provide the per-scene results for the experiments on the real-world HyperNeRF [28] dataset. Table 6, Figure 10 and Figure 12 illustrate the comparisons. While the quantitative results for Grid4D do not surpass those of other models in several scenes, it is noteworthy that our model exhibits significantly improved clarity in rendering, as demonstrated in Figure 10 and Figure 12. We also find reconstruction failures of DeformGS [50] in the 'Teapot' and 'Broom' scenes (the second and last line of Figure 10), as mentioned in their paper.

C Additional Ablations

Additional Architecture Ablations. We also conduct more ablation studies for Grid4D. We change the depth d of the multi-head decoder to one and two for the D-NeRF dataset, and the max level number L of the temporal grid hash encoder to 8 and 16. Additionally, we apply the simple position deformation method to Grid4D as Grid4D w/o RT, which directly adds the deformation to the position and is used in the prior works [44, 50]. The results can be found in the Table 7. We can find that

14

Table 6: Per scene comparisons on the real world HyperNeRF [28] 'vrig' and 'interp' dataset. The higher PSNR (↑) and higher MS-SSIM ↑ denote better rendering quality. The color of each cell shows the best and the second best. We set all rendering resolutions to 536×900.

Model	3D Printer		Broom		Chicken (vrig)		Peel Banana		Teapot	
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
TiNeuVox [8]	22.77	0.839	21.27	0.683	28.27	0.948	24.50	0.874	24.15	0.893
4D-GS [44]	22.00	0.807	21.80	0.684	28.55	0.927	27.62	0.934	26.99	0.941
Grid4D (Ours)	22.35	0.825	21.86	0.710	29.26	0.942	28.53	0.946	26.53	0.932
Model	Chicken (interp)		Cut Lemon		Hand		Slice Banana		Chocolate	
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
TiNeuVox [8]	27.69	0.951	28.54	0.955	27.44	0.872	27.67	0.916	26.97	0.948
4D-GS [44]	26.91	0.912	30.26	0.936	29.87	0.939	25.18	0.812	26.05	0.933
Grid4D (Ours)	27.31	0.924	32.18	0.967	31.31	0.958	25.79	0.853	28.23	0.965

Table 7: Additional ablation results of model architecture on the synthetic D-NeRF [31] dataset. The color of each cell shows the best and the second best.

Model	$L=8, d=0$	$L=16, d=0$	$L=32, d=0$ (Ours)	$L=32, d=1$	$L=32, d=2$	w/o RT
PSNR (↑)	41.34	41.69	42.00	41.96	41.75	41.80

Table 8: Comparison of average training computational cost on the D-NeRF [31] dataset with a single RTX 3090 GPU.

Model	4D-GS [44]	DeformGS [50]	SC-GS [12]	Grid4D (Ours)
Time	20min	33min	75min	55min
GPU Memory	1GB	4.5GB	3.1GB	4.0GB
PSNR	35.41	40.16	41.65	42.00

when the model becomes deeper, the performance might become worse, and we consider that the reason might be the training difficulties of deep MLPs.

Additional Smooth Regularization Ablations. We conduct a smooth regularization for the both grid hash encoder and MLP decoder in Grid4D w both by adding the following loss,

$$L_d = ||R_x - R_{x+\epsilon}||_2^2 + ||T_x - T_{x+\epsilon}||_2^2 \tag{11}$$

where R_x, T_x are the predicted position deformation of the 4D input x, and $R_{x+\epsilon}, T_{x+\epsilon}$ are the deformation of the perturbed 4D input $x + \epsilon$. The results are shown in the left part of Table 9, and we can find that the smooth regularization of the MLP decoder does not make sense. We consider that this is because of the smooth inherent property of MLPs.

Visualization of Feature, Deformation, and Depth Maps. We also visualize the feature maps by projecting the L2 norm of the features encoding the Gaussian positions from our 4D decomposed hash encoder. We set the RGB color of the temporal feature map to the L2 norm of $(x, y, t), (y, z, t), (x, z, t)$ grid features, and the feature map is rendered by the rasterization of Gaussians with the specified color. The results in Figure 11 denote that the proposed encoding method effectively represents the deformation features in both temporal and spatial spaces. Also, the depth maps show that we have a precise depth prediction.

D Limitation

Although our model achieves state-of-the-art performance with our proposed explicit representation, Grid4D has no improvement in training speed. However, compared to DeformGS [50], Grid4D has less memory overhead. As shown in Table 8, the computational cost has little influence on model training. Figure 9 displays several artifacts when Grid4D and the state-of-the-art models [44, 50] render several scenes with large and complex motions.

15

Table 9: Additional ablation results of adding MLP decoder regularization on the D-NeRF [31] dataset.

Model	Grid4D w both	Grid4D (Ours)
PSNR (↑)	41.92	42.00

4D-GS   DeformGS   Grid4D (Ours)   Ground Truth   4D-GS   DeformGS   Grid4D (Ours)   Ground Truth
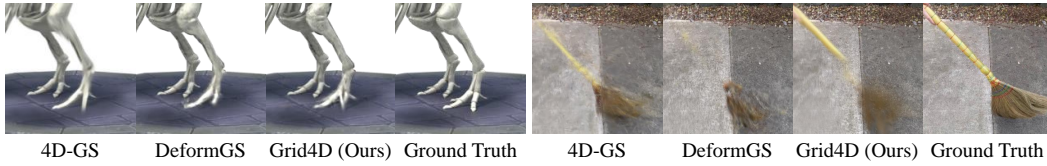
Figure 9: Artifacts of Grid4D and other state-of-the-art models [44, 50].

TiNeuVox   4D-GS   DeformGS   Grid4D (Ours)   Ground Truth

Figure 10: Additional qualitative comparisons on the real-world HyperNeRF [28] dataset.

16
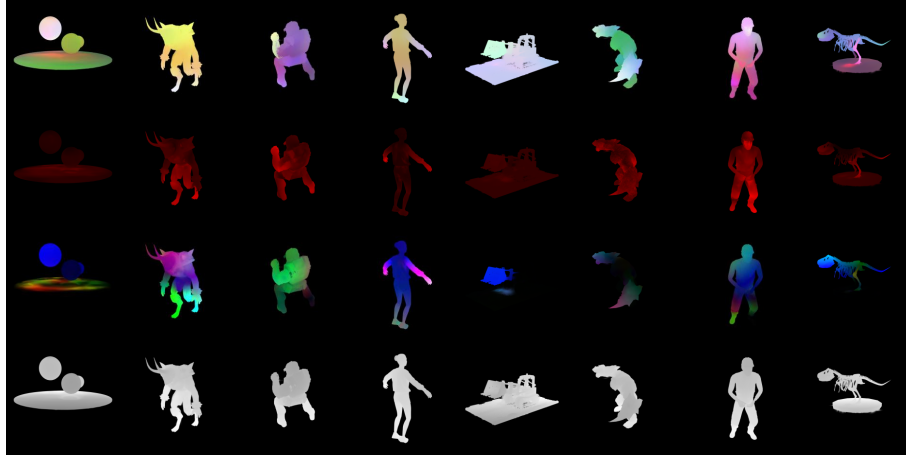
Figure 11: Additional analysis of Grid4D. Each line in turn shows the temporal feature maps, spatial feature maps, deformation maps, and depth maps. The lighter parts in the feature maps denote the stronger activation of the corresponding features.

| TiNeuVox | 4D-GS | DeformGS | Grid4D (Ours) | Ground Truth |

Figure 12: Additional qualitative comparisons on the real-world HyperNeRF [28] dataset.

17

TiNeuVox   4D-GS   DeformGS   Grid4D (Ours)   Ground Truth
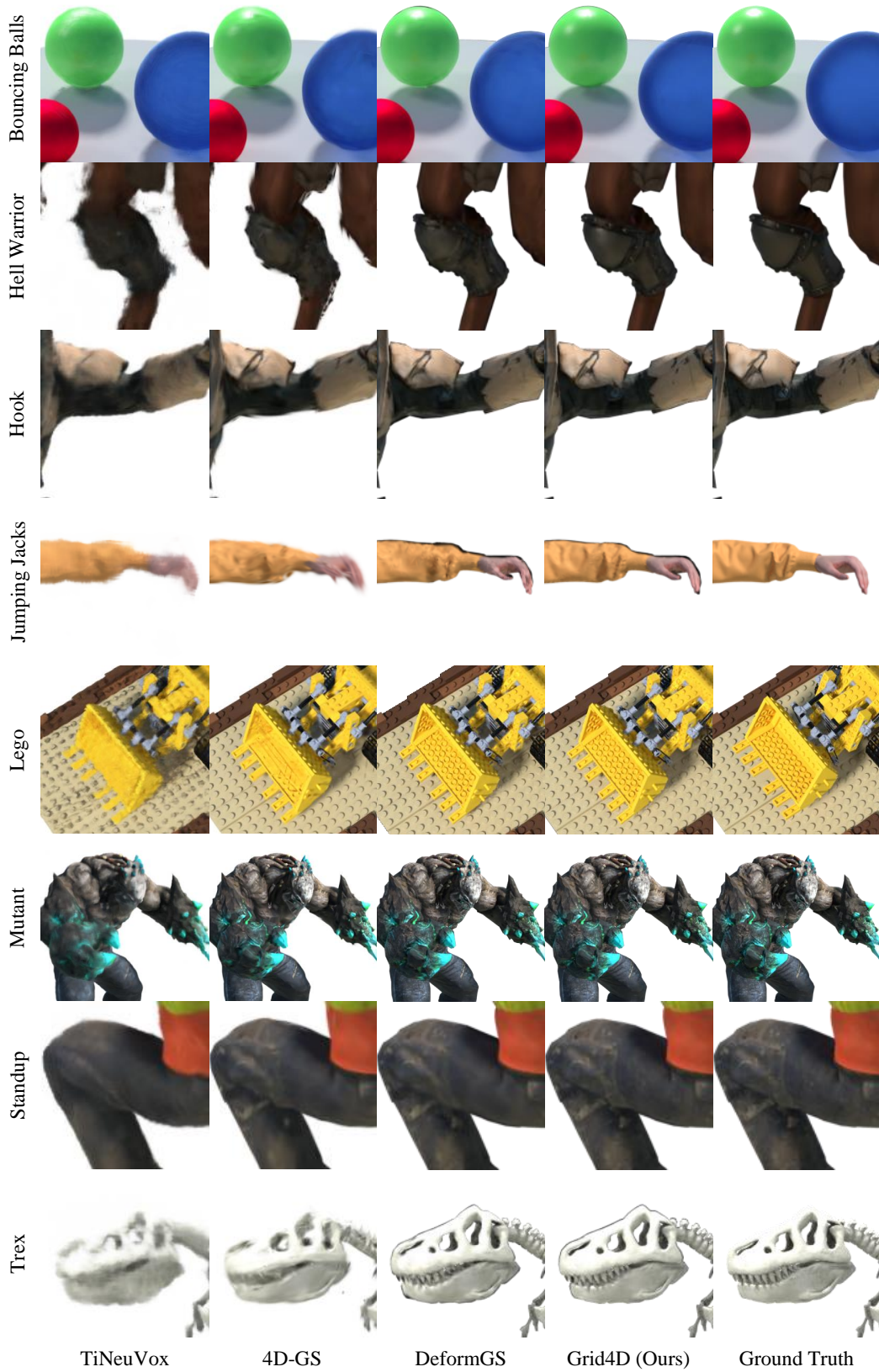
Figure 13: Additional qualitative comparisons on the synthetic D-NeRF [31] dataset.

18

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We accurately claim the contributions and scope of this paper in the abstract and Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the insignificant limitation of training efficiency and complex motion fitting in Section 5 and Section D of the supplementary.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

19

Answer: [NA]

Justification: The proposed methods are mainly experimental, and we demonstrate the effectiveness of our methods through abundant experiments.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe our methods clearly in Section 3 and Section 4.1. More details can be found in the supplementary, and we will release our codes and checkpoints if the paper is accepted.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

20

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release all our codes on the GitHub platform, and provide the URL in the camera-ready version. We will offer the instructions to reproduce the results in detail.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide important settings and an overview of the experimental setup in Section 4.1. More details can be found in Section A of the supplementary or our code released.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The experiments do not have obvious experimental statistical significance. However, we directly provide the detailed experimental results in every scenes.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

21

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide the GPU used in our experiments. We also describe the detailed compute resources of our methods in Section D of the supplementary and Section 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The research is strictly conducted with the NeurIPS Code of Ethics in every respect.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This work does not have obvious societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

22

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper does not have such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use creditable assets and properly respect them in Section 4.1.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

23

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We describe the used public datasets in Section 4.1, and we will release our code for academic usage.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The research is not with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The research is not with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

24

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

25