Benchmarking Robustness in Neural Radiance Fields

Chen Wang[1] Angtian Wang[2] Junbo Li[3] Alan Yuille[2] Cihang Xie[3]
[1] University of Pennsylvania [2] Johns Hopkins University [3] UC Santa Cruz

Abstract

Neural Radiance Field (NeRF) has demonstrated excellent quality in novel view synthesis, thanks to its ability to model 3D object geometries in a concise formulation. However, current approaches to NeRF-based models rely on clean images with accurate camera calibration, which can be difficult to obtain in the real world, where data is often subject to corruption and distortion. In this work, we provide the first comprehensive analysis of the robustness of NeRF-based novel view synthesis algorithms in the presence of different types of corruptions.

We find that NeRF-based models are significantly degraded in the presence of corruption and are more sensitive to a different set of corruptions than image recognition models. Furthermore, we analyze the robustness of the feature encoder in generalizable methods, which synthesize images using neural features extracted via convolutional neural networks or transformers, and find that it only contributes marginally to robustness. Finally, we reveal that standard data augmentation techniques, which can significantly improve the robustness of recognition models, do not help the robustness of NeRF-based models. We hope our findings will attract more researchers to study the robustness of NeRF-based approaches and help improve their performance in the real world.

1. Introduction

Novel View Synthesis (NVS), a long-standing problem in computer vision and computer graphics research, aims to generate photo-realistic images at unseen viewpoints of a 3D scene given a set of posed images. Existing works, including those of image-based rendering, primarily rely on explicit geometry and hand-crafted heuristics [12, 38, 54], which require sophisticated design, extensive efforts for data collection and preprocessing and have difficulty in generalizing to new scenes and settings.

Recently, Neural Radiance Fields (NeRF) [26] have demonstrated as an effective implicit 3D scene representation over recent years and achieved state-of-the-art performance on NVS by leveraging end-to-end learnable components with 3D geometry context to reconstruct input images. NeRF encodes a scene into a continuous multi-layer perceptron (MLP), which regresses color and density given any 3D location and view direction. Novel views can thus be synthesized through differentiable volumetric rendering from arbitrary viewpoints.

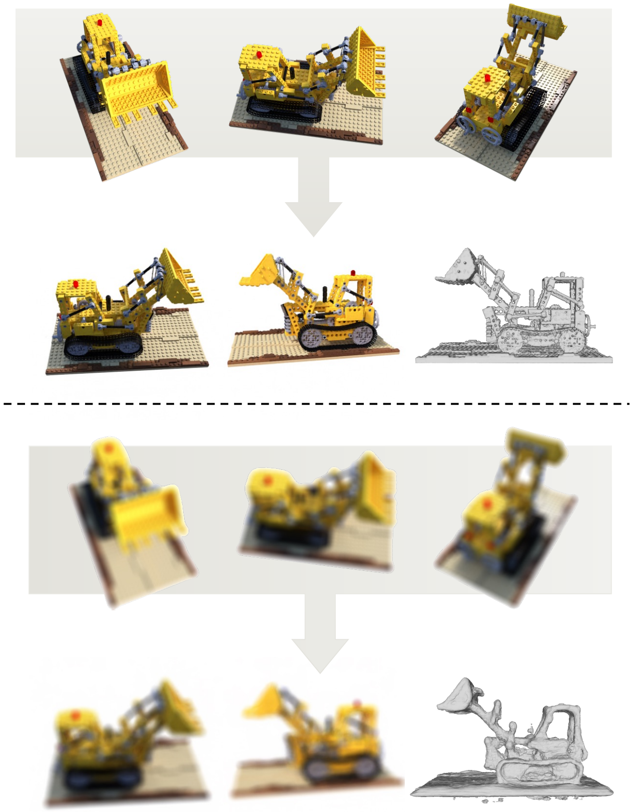

Figure 1. NeRF produces high quality novel view synthesis and accurate surface reconstruction when training on clean images (top). However, corruption on training images will significantly affect both the renderings and geometry of the objects (bottom). Meshes were obtained with marching cubes [21].

The emergence of NeRF has made NVS more usable in real-world scenarios by reducing the need for tedious preprocessing steps. In practice, we only have to take images with our phones and estimate camera poses with off-the-shelf structure-from-motion (SfM) techniques to train

NeRF-based approaches, which can achieve remarkable view synthesis results with accurately calibrated poses and clean images. However, to generate photo-realistic images at novel viewpoints, it is necessary to comprehensively model the 3D space, including the scene geometry, illumination, and reflections. This requires capturing local high-frequency details, which can make NeRF-based methods sensitive and fragile to perturbations in the inputs. In addition, NeRF-based methods are often optimized using a pixel-based L2 reconstruction loss, which can lead to overfitting the target scene without prior information. When the inputs are corrupted, such as being compressed in JPEG format or blurred due to motion, the reconstructed scenes may exhibit visible artifacts. Most importantly, corruptions often occur during the real-world capture and preprocessing stages. It may seem evident that they can lead to inaccurate reconstruction, but the more important and interesting question—*how different types and severities of corruption affect the robustness of NeRF*—remains open.

As a step forward, in this paper, we present the first benchmark to comprehensively evaluate the robustness of current NeRF-based methods. Firstly, we construct two benchmarking datasets: *LLFF-C* and *Blender-C*, both of which are the corrupted counterparts of the standard datasets used in NeRF-based methods, and the latter also contains 3D-aware corruptions. Using the metrics we propose, we show that modern NeRF-based models exhibit significant degradation across all types of corruptions, and that there is still a significant opportunity for improvement on both LLFF-C and Blender-C. In addition, difficulties for each corruption type exhibit a totally different nature in NeRF-based methods from recognition tasks. Especially, for generalizable methods that use image features to assist neural rendering, we find that scaling the feature encoder does not enhance robustness. We also show that fine-tuning a pretrained generalizable model leads the model to overfit the "incorrect" data, sometimes resulting in worse synthesis quality than using the pretrained model directly. For dealing with those corruptions, one might find using 2D restoration methods can help in certain aspects.

In summary, the contributions of our paper include:

- We present the first framework for benchmarking and assessing the robustness of NeRF-based systems to visual corruptions.
- Our findings demonstrate that current NeRF-based methods perform poorly under corruptions, including those *generalizable* ones.
- We systematically analyze and discuss the robustness of NeRF-based methods under various corruptions, feature encoder design, etc.
- We find that standard image data-augmentation techniques do not improve the robustness of novel view synthesis that relies on cross-view consistency.

2. Related Work

Novel view synthesis with NeRF. Traditional image-based rendering methods generate novel views by warping and blending input frames [10, 20], and learning-based methods predict blending weights through neural networks or hand-crafted heuristics [11, 33, 34]. Different from these, geometry-based methods render images through an explicit 3D model, *e.g.,* Thies *et al.* [38] stored neural textures on 3D meshes and rendering with standard graphics pipeline. Other 3D proxies such as point clouds [1, 35], voxel grids [18, 29], multi-plane images [25, 36, 54] are also used. However, these approaches often require large amounts of data and memory to produce satisfying results.

Recently, neural fields [47] leverages an MLP to represent 3D shapes or scenes by encoding them into signed-distance, occupancy, or density fields. Among them, NeRF [26] demonstrated remarkable results for novel view synthesis with posed images. NeRF uses a coordinate-based MLP representation and obtains color by differentiable volumetric rendering. The optimization of NeRF can be simply done by minimizing the photometric loss at training camera viewpoints. Although NeRF has been investigated in several aspects, *e.g.,* generation [27], 3D recontruction [42, 45], 3D super-resolution [40], in this work, we still focus on the novel view synthesis task.

Robustness benchmarks. The robustness of deep learning models is crucial for real-world applications, and its assessment has received growing attention in recent years [14, 19, 31]. ImageNet-C [14] benchmark, as one of the pioneering works, evaluates image classifiers' robustness under simulated image corruptions such as motion blur and jpeg compression. Imagenet-V2 [31] creates new test sets for ImageNet and CIFAR-10 and evaluates the accuracy gap caused by natural distribution shift. Specifically for object recognition, ObjectNet [3] presents a real-world test set containing objects with random backgrounds, rotations, and imaging viewpoints. ImageNet-A [16] and ImageNet-R [13] further propose additional benchmarks for natural adversarial examples and abstract visual renditions like image style, camera operation, and geographic location etc.

Researchers have also examined benchmarks beyond image classification. RobustNav [5] quantifies the performance of embodied navigation agents when exposed to both visual and dynamic corruptions. Ren *et al.* [32] provide a taxonomy of common 3D corruptions and identify the atomic corruptions for point clouds for the first time, followed by an evaluation of existing point cloud classification models. More recently, Kar [19] proposes 3D common corruptions that resemble real-world ones by integrating geometry information *i.e.,* depth, into the corruption process. However, a comprehensive benchmark for evaluating the robustness of NeRF is still lacking.

Improving model robustness. Adversarial training [23] is a common method used to protect models from corruption. For example, Xie *et al.* [46] show that adversarial perturbations can be used as data augmentation and improve image classification accuracy. Similar conclusions are also reached in natural language processing [41].

On the other hand, data augmentation can significantly improve generalization performance, and many works augment input images to boost recognition. Mixup [52] and CutMix [50] mixe two images and enforce neural networks to favor linear behavior between training examples by creating convex interpolated samples. Augmix [15] creates compositions of multiple augmented samples that preserve original semantics and statistics. However, mixup [52] and cutmix [50] are not suitable for NeRF augmentation as they disturb the cross-view constraint required by NeRF, while the efficacy of augmix [15] remains to be studied.

To increase the robustness of NeRF, Aug-NeRF [7] firstly proposes a triple-level augmentation training pipeline that is robust to noisy inputs. Other works tackle specific corruption [22, 28]. Recently, NeRFool [9] finds that increased conditioning and adversarial perturbations on density can attack generalizable NeRF. Perez *et al.* [30] introduces image augmentations to enhance neural rendering methods. Azzarelli *et al.* [2] proposes a new framework to evaluate NeRF-based methods. Our work differs from them in that we aim to thoroughly evaluate the robustness of standard NeRF methods and seek ways to improve them.

3. Background

We present a benchmark for evaluating the robustness of rendering models that utilize NeRF [26]. This section provides an overview of NeRF and our classification of NeRF-based methods.

3.1. NeRF for Novel View Synthesis

NeRF represents a 3D scene as a continuous function, which takes as inputs a 5D vector containing 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\boldsymbol{\theta}, \boldsymbol{\phi})$, and outputs the corresponding radiance $\mathbf{c}(\mathbf{x}, \mathbf{d}) = (r, g, b)$ with volume density $\sigma(\mathbf{x})$. NeRF is typically parameterized as an MLP $f : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$.

NeRF is an emission-only model, *i.e.,* the color of a pixel only depends on the radiance along the viewing ray. Therefore, according to volume rendering [17], the color along the camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ that shots from the camera center $\mathbf{o}$ in direction $\mathbf{d}$ can be calculated via standard volume rendering, the discrete format is expressed as the following:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))c_i, \tag{1}$$

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \tag{2}$$

where N is the number of sampled points along the ray, $\delta_i = t_{i+1} - t_i$ is the distance between two adjacent samples, c_i and σ_i are the per-point radiance and density, and T_i denotes the accumulated transmittance.

NeRF is trained to minimize the mean-squared error (MSE) between the predicted renderings and the corresponding ground-truth color:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \|\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\|^2, \tag{3}$$

where $\mathcal{R}$ denotes the batch of rays randomly sampled from all training images or one specific image. $\hat{\mathbf{C}}(r)$ and $\mathbf{C}(r)$ are the ground truth and output color of ray r. This per-pixel optimization lacks holistic spatial understanding and might make NeRF sensitive to disturbance in pixel values.

After the emergence of NeRF, several other methods based on NeRF have been proposed for novel view synthesis. Although they differ in many aspects, *e.g.,* sampling strategy, positional encoding, network architecture etc, all of them aggregate colors and densities of discontinuous points along viewing rays via differentiable volumetric rendering to synthesize novel views, which are named *NeRF-based* methods. Non-NeRF-based neural rendering methods obtain pixel colors using explicit representations [35, 39], *e.g.,* point clouds or surface-based implicit representation [48] without volume densities.

3.2. Two genres of NeRF-based methods

We further divide NeRF-based methods into two categories: *scene-specific* and *generalizable*. Scene-specific methods such as [4, 26] optimize a single model from scratch with a set of training images of one scene, leading to different network parameters for each scene. In contrast, generalizable methods [6, 43, 49] firstly train a model on a dataset containing hundreds of 3D scenes, then directly infer or finetune a few steps on a single testing scene. Specifically, generalizable methods contain CNN or transformer encoders $\boldsymbol{F}$ to extract image features from inputs $\mathcal{I}_i, i = 1, 2, ...n$:

$$\boldsymbol{f}_i = \boldsymbol{F}(\mathcal{I}_i), \tag{4}$$

For a 3D point $\boldsymbol{p}$, image features $\boldsymbol{f}_i$ across input views are aggregated by the function $\mathcal{A}$. $\mathcal{A}$ is quite different in different methods, but often it consists of a series of operations: perspective projection $\boldsymbol{p}$ into each input image, image feature interpolation, and multi-view feature fusion (*e.g.,* cost volume, pooling or simple concatenation):

$$\boldsymbol{f_p} = \mathcal{A}(\{\boldsymbol{f}_i\}_{i=1}^{n}; \boldsymbol{p}), \tag{5}$$

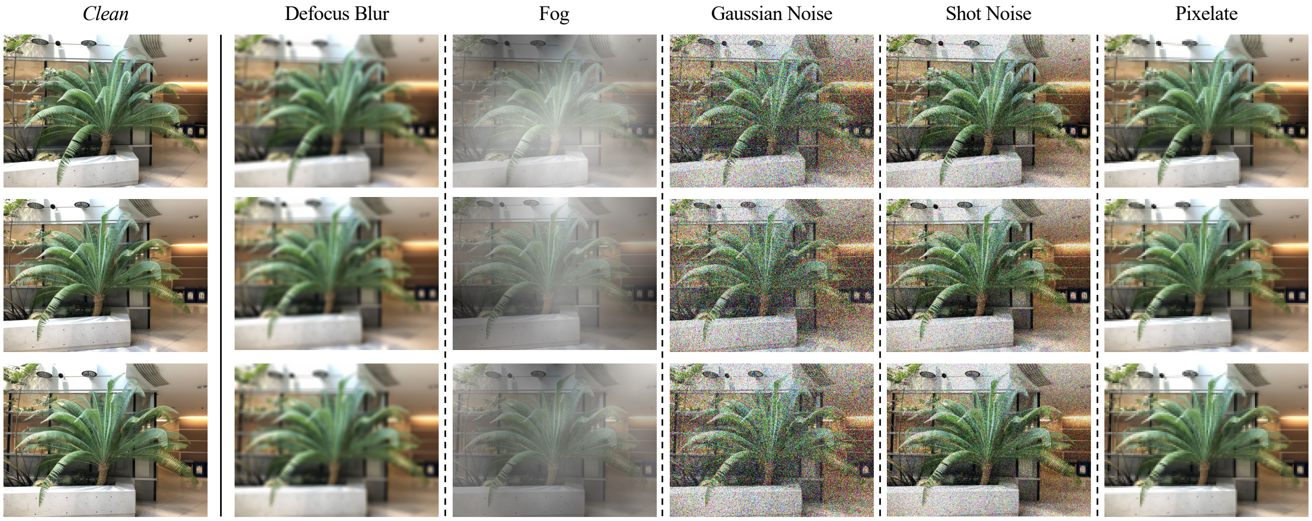

Figure 2. Examples of our proposed LLFF-C on the *fern* scene under five corruption types at the severity of 2.

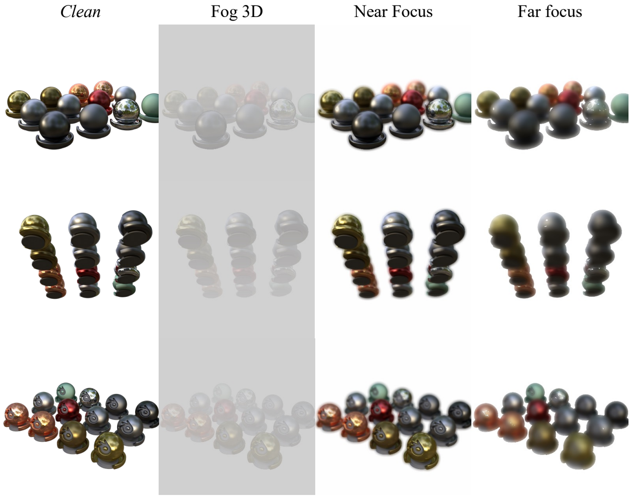

Figure 3. Examples of our proposed Blender-C (left) on the *materials* scene under 3D-aware corruptions, from which we can see that the same corruption has varying effects across scene depth.

The features are further decoded to the color $\mathbf{c}_{\boldsymbol{p}}$ and density $\sigma_{\boldsymbol{p}}$ of $\boldsymbol{p}$:

$$\mathbf{c}_{\boldsymbol{p}} = \mathcal{D}_c(\boldsymbol{f}_{\boldsymbol{p}}; \boldsymbol{z}_c), \quad (6)$$

$$\sigma_{\boldsymbol{p}} = \mathcal{D}_\sigma(\boldsymbol{f}_{\boldsymbol{p}}; \boldsymbol{z}_\sigma), \quad (7)$$

where $\boldsymbol{z}_c$ and $\boldsymbol{z}_\sigma$ are optional auxiliary vectors (*e.g.,* visibility) to enhance decoding, $\mathcal{D}_c$ and $\mathcal{D}_\sigma$ denote the decoding networks (mostly MLPs) for color and density respectively ($\mathcal{D}_c$ and $\mathcal{D}_\sigma$ sometimes share the parameters). With colors and densities for queried 3D points, the final pixel color can be similarly obtained using Equation (1).

4. Corruptions and Test Suite

In this section, we present a detailed description of the structure of our benchmark. Our study focuses on the impact of corruption on NeRF-based models, and we primarily conduct experiments on NeRF-based novel view synthesis methods. However, our test suite can also be directly applied to recent non-NeRF-based methods, such as the one presented in [37], as described in Section 5.1. Additionally, the test suite can be easily adapted to other tasks that involve NeRF, *e.g.,* relighting, reconstruction, and video synthesis.

The goal of our work is to act as a foundation for building robust NeRF-based systems in the real world. Unlike previous benchmarks [14], each data chunk in ours is not a single image, but a 3D scene comprising multi-view images of one scene. *Scene-specific* methods are optimized directly on "corrupted" unseen scenes, while *generalizable* methods have already been pretrained on clean training scenes and then tested or finetuned on corrupted target scenes. To avoid confusion, in this paper, *training set* only refers to the 3D scenes used for pretraining generalizable methods, while *target training set* and *target testing set* refer to the training and testing images for a specific target scene or object. The corruptions are drawn from a predefined set which we will elaborate in the following.

4.1. Corruptions

Similar to images, scene corruptions are artifacts that degrade the quality of a system's RGB observations. The corruptions we include have the following characteristics: (1) the target training set and target testing set have the same lighting conditions; and (2) corrupted images and their clean counterparts are taken from the same camera poses, with only the content altered. We provide around ten types (the exact number depends on the dataset) of corruptions *e.g.,* gaussian noise and fog, mainly from those proposed in [14] and exclude those that do not meet the above requirements. Each corruption has three levels of severity ($1 \to 3$) indicating an increase in the degree of degradation.

4.2. Datasets

LLFF-C. LLFF [25] consists of 8 real-world scenes that contain mainly forward-facing images. Each of the eight images is held out as the target testing set. We corrupt the target training set in each scene with 9 corruptions to create the LLFF-C test suite (see Figure 2 for an example).

Blender-C. Blender-C is constructed based on the Blender dataset (also known as NeRF-Synthetic) [24] that contains 8 detailed synthetic objects with 100 images taken from virtual cameras arranged on a hemisphere pointed inward. Inspired by [19], we specifically create 3D-aware fog, near focus, and far focus on replacing their 2D counterparts for Blender-C (see Figure 3). For example, in 3D-aware fog, pixels far from the camera are occluded to a higher extent. The RGB images and the corresponding depth maps are rendered with the official blend files for 3D-aware corruptions. We use 100 images as the target training set and 25 images as the target testing set for each scene.

4.3. Task and Metrics

The task of our benchmark is novel view synthesis learned from images of a 3D scene. The standard procedure for evaluating the performance of novel view synthesis methods is to compare the ground truth images and predicted images at testing viewpoints with the three mostly used metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [44] and LPIPS [53]. For scene-specific models, we directly provide a corrupted target training set for each model. Generalizable methods are trained on clean training scenes but perform inference or finetuning on corrupted target scenes.

Inspired by the 2D common corruptions on images [14] and point clouds [32], the first step for evaluation is to optimize (inference or finetune for generalizable methods) a model f with a set of clean images and compute the relevant metrics $m^f_{\text{clean}}, m \in M$ at the target testing set. Next, the same process will be repeated, but with corrupted target training set for each corruption type c and severity s. We then calculated the metrics again, which are denoted by $m^f_{\text{c,s}}, m \in M$. Thus, the corruptions metric (CM) is defined as the mean metric over severities:

$$\text{CM}_{c,m} = \frac{1}{3}\sum_{s=1}^{3} m_{c,s}, \tag{8}$$

Unlike classification benchmarks, we do not use another *baseline* model as the denominator in Equation (8) because it washes out the models' absolute performance on the corrupted data. Mean CM is thus the average of CM over all the corruption types:

$$\text{mCM}_m = \frac{1}{N}\sum_{c} \text{CM}_{c,m}, \tag{9}$$

where N is the number of corruption types.

While CM and mCM measure the absolute robustness of NeRF-based models, we are also interested in their relative performance drop, *i.e.,* the amount a model degrades from clean inputs to corrupted ones, defined by Relative mCM as the following:

$$\text{RCM}_{c,m} = \frac{1}{3}\sum_{s=1}^{3} \frac{|m_{\text{clean}} - m_{c,s}|}{m_{\text{clean}}}, \tag{10}$$

$$\text{RmCM}_m = \frac{1}{N}\sum_{c} \text{RCM}_{c,m}, \tag{11}$$

where $m \in M$. We use absolute $|m_{\text{clean}} - m_{c,s}|$ in $\text{RCM}_{c,m}$ considering that PSNR and SSIM are higher the better, while LPIPS is lower the better.

5. Experiment

5.1. Setup

We benchmark a total of seven methods, all of which are representative of recent novel view synthesis works. For scene-specific methods, we include NeRF [26], MipNeRF [4], and the network-free method Plenoxels [8]. For generalizable methods, we experiment with IBRNet [43] and MVSNeRF [6]. We also include Generalizable Patch-Based Neural Rendering (GPNR) [37], the latest pure transformer-based architecture, as we were interested in its robustness compared to other methods. We omit PixelNeRF [49] due to its poor performance on datasets other than its testing set, DTU.

For all of the methods, we use publicly available codebases and checkpoints, re-training some as necessary. For IBRNet [43] and MVSNeRF [6], we present results for both direct inference and fine-tuning. The details of dataset processing, training, etc., for each method are included in the supplementary material.

5.2. Results and Findings

We report the CM values of PSNR and LPIPS on LLFF-C and Blender-C at Table 1 and Table 2. Figure 4 shows part of the qualitative results. Other results, including CM values of SSIM and detailed RmCM can be found in the supplementary materials.

Generally, all state-of-the-art methods suffer a performance drop from the clean setting across all corruptions. As seen in Figure 5, methods that achieve better quality on

		Noise			Blur			Weather	Digital	
	Clean	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Fog	Pixel	JPEG
NeRF	27.68/0.151	24.32/0.297	23.74/0.300	24.04/0.297	20.84/0.501	22.26/0.372	19.20/0.483	11.87/0.528	24.28/0.302	25.29/0.288
MipNeRF	27.69/0.159	23.96/0.340	23.34/0.345	23.69/0.341	20.86/0.510	22.24/0.380	19.10/0.507	12.03/0.545	24.22/0.316	22.13/0.257
Plenoxel	27.45/0.100	20.72/0.480	20.31/0.486	17.71/0.494	20.45/0.509	21.91/0.373	19.20/0.463	12.44/0.660	23.81/0.289	24.78/0.300
MVSNeRF	17.03/0.409	16.78/0.545	16.50/0.551	16.72/0.550	17.13/0.576	17.33/0.500	16.30/0.558	12.97/0.567	17.65/0.448	17.37/0.465
MVSNeRF (ft)	23.94/0.244	21.51/0.442	21.17/0.443	21.41/0.441	20.58/0.517	21.62/0.408	19.42/0.490	13.12/0.606	22.70/0.353	23.15/0.349
IBRNet	25.71/0.158	21.88/0.452	21.36/0.455	22.05/0.434	20.54/0.514	21.77/0.392	19.39/0.453	13.74/0.453	23.35/0.311	23.91/0.338
IBRNet (ft)	27.80/0.112	24.03/0.321	21.33/0.451	23.83/0.322	19.67/0.517	21.84/0.395	19.45/0.456	13.33/0.501	22.82/0.341	24.67/0.311
GPNR	24.58/0.210	21.56/0.481	21.11/0.482	21.43/0.476	20.31/0.530	21.38/0.418	19.13/0.483	12.83/0.569	22.82/0.341	23.34/0.346

Table 1. PSNR↑ / LPIPS↓ results for clean and corrupted data on LLFF-C, ft indicates results after fine-tuning for generalizable methods.

		Noise			Blur				Weather	Digital	
	Clean	Gauss.	Shot	Impulse	Near Focus	Far Focus	Glass	Motion	Fog	Pixel	JPEG
NeRF	30.98/0.071	22.02/0.163	19.21/0.208	23.75/0.167	26.99/0.127	24.81/0.158	22.48/0.210	20.73/0.233	15.44/0.200	26.81/0.138	28.34/0.117
MipNeRF	33.34/0.061	22.68/0.149	19.59/0.211	24.94/0.178	27.92/0.099	25.93/0.121	22.49/0.208	21.04/0.211	15.52/0.184	27.66/0.118	29.74/0.098
Plenoxel	32.94/0.035	20.53/0.556	17.68/0.603	22.41/0.461	27.68/0.100	25.73/0.116	22.28/0.210	21.10/0.225	15.62/0.477	26.56/0.130	28.58/0.115
MVSNeRF	19.56/0.288	15.68/0.559	14.26/0.603	15.36/0.599	19.17/0.310	18.98/0.322	18.24/0.357	17.21/0.364	14.11/0.467	19.28/0.316	19.26/0.314
MVSNeRF (ft)	23.28/0.199	17.72/0.512	16.12/0.561	18.25/0.538	18.25/0.538	22.87/0.217	22.58/0.208	21.38/0.255	14.33/0.406	22.87/0.220	23.03/0.217
IBRNet	27.15/0.143	20.45/0.489	18.16/0.535	21.49/0.499	24.98/0.142	23.80/0.212	21.74/0.270	20.64/0.254	15.28/0.237	24.87/0.120	25.64/0.120
IBRNet (ft)	29.91/0.081	20.97/0.483	19.97/0.496	22.36/0.489	24.88/0.126	22.72/0.197	21.42/0.203	19.95/0.210	15.17/0.256	23.94/0.177	23.81/0.186

Table 2. PSNR↑ / LPIPS↓ results for clean and corrupted data on Blender-C, ft indicates results after fine-tuning for generalizable methods.

clean data mostly excel in mCM, with scene-specific models more robust than generalizable ones. However, it seems that the corruption robustness is mainly explained by the model's original ability for scene representation since, by looking at RmCE, no models have demonstrated a remarkable ability to resist corruption.

In terms of relative robustness, Plenoxel [8] and IBRNet (ft) [43] have the highest RmCM on LLFF-C ($25.5\%/331\%$ and $25.7\%/271\%$ for PSNR/LPIPS) and Blender-C ($31.7\%/751\%$ and $28.1\%/248\%$ for PSNR/LPIPS). Moreover, as a voxel-based approach, Plenoxel [8] sometimes consumes much higher GPU memory on corrupted data than usual due to incorrect density distributions in the empty space of optimized 3D scenes. Also, from Figure 4, we can find it struggles with Gaussian Noise and Fog. This reveals that explicit representation might not be suitable for highly corrupted situations. MVSNeRF [6] has the lowest RmCM on both datasets because its m_{clean} is fairly low and leaves little room for degradation. Generalizable methods without finetuning are more relatively robust, maybe due to their prior knowledge learned from massive training data.

Not all corruptions are equally severe. For example, *Pixelate* and *JPEG Compression* only lead to an absolute drop of PSNR in less than 15%. However, *Fog* is the hardest of all methods, resulting in a nearly 50% absolute drop in PSNR for all methods except MVSNeRF [6]. The main reason for this huge discrepancy is that the *Pixelate* and *JPEG* have limited influence on original inputs, while *Fog* corruption leads to a drastic change in images, *i.e.,* occludes the objects with fog (see Figure 2). The difficulties of recognition tasks and 3D reconstruction can also vary. From [14], for image classification, *Fog* is the easiest corruption type, while *Glass Blur* is the hardest.

With regard to generalizable methods, fine-tuning on clean data significantly boosts models' performance (See the first column in Table 1 and Table 2). However, this does not hold for corrupted data. Although MVSNeRF [6] improves on all corruptions because of its generalization performance, IBRNet [43] drops on both datasets across several corruptions, with the largest degradation occurring at severity levels > 1. The main reason is that highly corrupted scenes disrupt the multi-view consistency that NeRF training relies on, and fine-tuning on such data causes the pretrained network to overfit incorrect geometries. An example can be found in Figure 4.

6. Discussion

In this section, we discuss various design and training details in NeRF-based systems and explore how they affect model robustness.

Feature encoding. As mentioned in Section 3.2, generalizable methods include an encoder $\boldsymbol{F}$ for image feature extraction. However, different existing methods have different encoder designs. For example, IBRNet [43] uses a U-Net structure containing several downsampling residual blocks and upsampling layers, resulting in 7.96 GFlops and 8.92M parameters, while MVSNeRF [6] only has a few convolutional downsampling layers with 484.5 MFlops and 42.26k parameters. It is important to understand whether the choice of the encoder impacts both the quality of novel view synthesis and the robustness of generalizable methods. The number of parameters also has a direct impact on total training time, as volume rendering is already quite time-intensive.

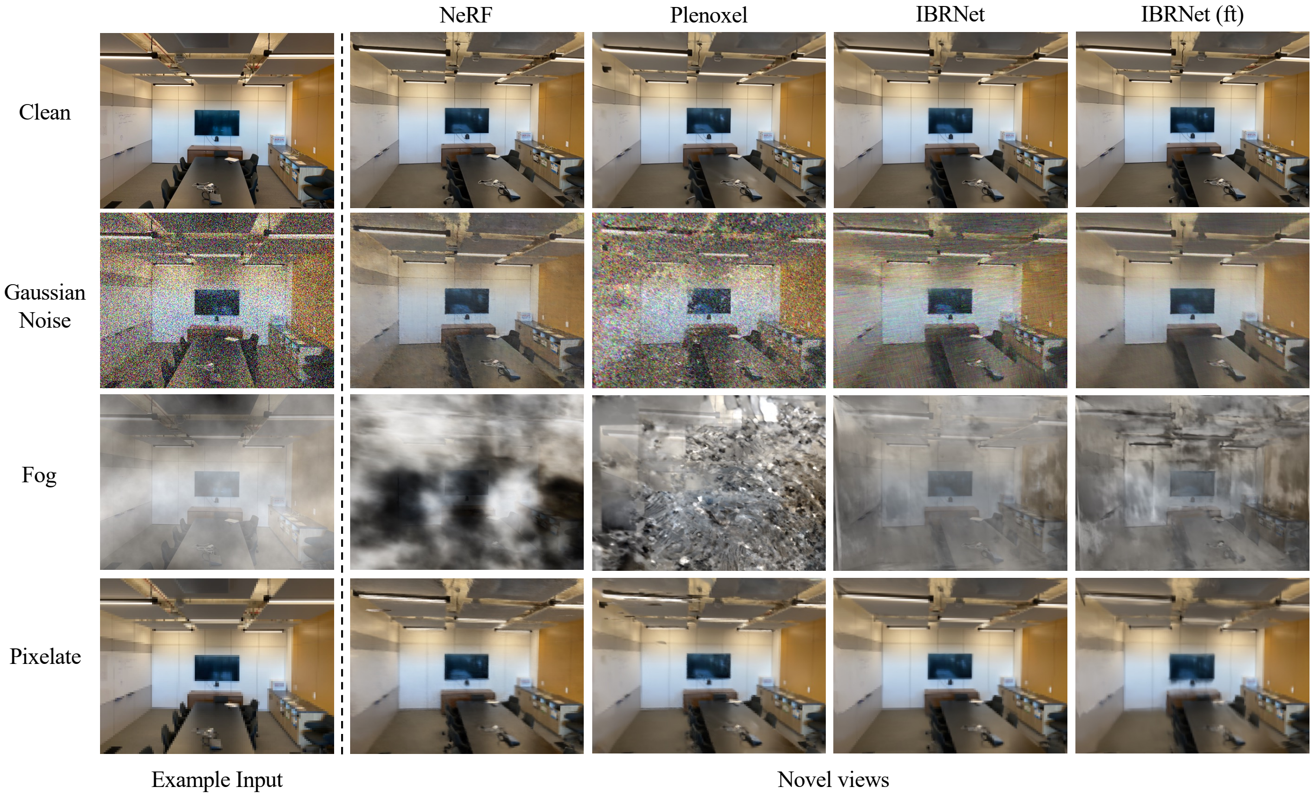

Figure 4. Qualitative results across corruptions (severity = 3) and methods on the *room* scene. Note we hereby use multiple input images for the scene and only show one example. We can see that each method behaves differently under these corruptions. We encourage the readers to zoom in for a better inspection.

Therefore, we present additional control studies on IBRNet [43] and MVSNeRF [6] with encoders of different design choices by decreasing the channels of IBRNet's encoder or residual blocks (Please found the architecture details in the supplementary). We re-trained both methods on the IBRNet Collected and LLFF released scenes. Since the direct inference performance of MVSNeRF [6] remains poor, we continue fine-tuning each target testing scene. The results are reported on Table 3.

		IBRNet			MVSNeRF (ft)		
Encoder	Flops	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
ResUNet	7.96G	20.82	0.593	0.421	20.55	0.702	0.453
ResNet	2.62G	20.97	0.593	0.422	20.53	0.701	0.454
ResUNet-Small	2.22G	20.81	0.593	0.422	20.50	0.700	0.456
ResUNet-Tiny	1.09G	20.84	0.594	0.420	20.52	0.700	0.454
MVSNeRF	0.48G	20.76	0.586	0.424	20.58	0.702	0.453

Table 3. Robustness results with different encoder designs.

According to the results, surprisingly, the choices of encoder contribute marginally both to clean data and corruption data. However, training time does vary, *e.g.,* in IBRNet, the total time for training ResUNet-Tiny compared with ResUNet-Big is reduced by more than 60%. This suggests that only low-level features are needed for generalizable models in current frameworks, and the results might be more related to the feature aggregation part.

Improving Model Robustness To deal with the corruption in input images, we first use pretrained image restoration models to first restore a clean version of inputs and then train NeRF-based models. Specifically, we chose Restormer [51] for image denoising and deblurring. The results are shown in Table 4. We can see that robustness in *Gaussian Noise* has seen an apparent increase in both NeRF and IBRNet. This is because the restoration works well for *Gaussian Noise*. However, the denoising fails to deal with *Shot Noise* and *Impulse Noise*, even causing the images to be more blurry and deviate from clean ones. As for the blur types, *Defocus Blur* was successfully improved. For *Motion Blur*, LPIPS gets better but PSNR drops. The main reason is that part of the image was not fully deblurred, causing inconsistency between input images and higher pixel-level error (poor PSNR). Since the reconstructed scene is still cleaner than without deblurring, LPIPS has a substantial improvement.

We also benchmarked methods specifically designed for certain corruptions, *i.e.,* Deblur-NeRF [22]. PSNR/SSIM

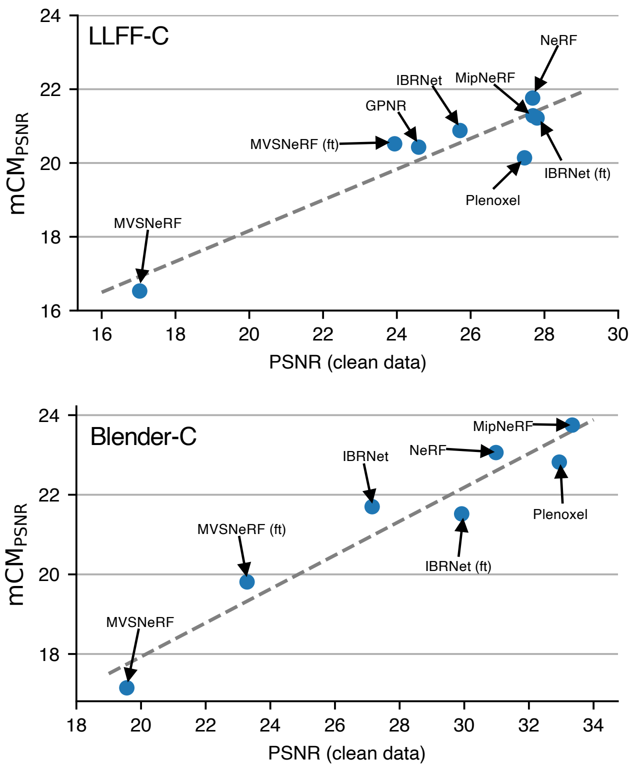

Figure 5. Robustness (mCM) of PSNR values on LLFF-C and Blender-C.

results for *Defocus Blur* and *Motion Blur* are 20.69/0.589 and 23.10/0.289 respectively. *Motion Blur* is handled perfectly with its deformable sparse kernel. However, it fails to deblur *Defocus Blur*, presumably because the blur is view consistent across images, thus the method cannot decompose the blur pattern by aggregating the information from different views.

	Noise			Blur	
	Gauss.	Shot	Impulse	Defocus	Motion
NeRF	24.54/0.259	20.49/0.444	20.90/0.436	22.44/0.347	18.77/0.445
IBRNet	23.23/0.289	19.84/0.491	20.13/0.473	21.91/0.367	19.00/0.340
IBRNet (ft)	24.38/0.259	20.38/0.462	20.76/0.449	22.42/0.351	19.08/0.411

Table 4. PSNR↑ / LPIPS↓ results for LLFF-C after image restoration, ft indicates results after fine-tuning.

Patch-based sampling. Recall that NeRF-based methods randomly sample a batch of rays in each training iteration and compare their predicted color with ground truth, which ensures ray diversity while training. Here, we explore another patch-based sampling strategy in which we sample m numbers of $n \times n$ image patches instead, and $m \times n \times n$ equals the number of rays per iteration. We experiment with $n = 2, 4$ on NeRF [24] and MVSNeRF [6], and find that it drops performance on clean data, but improves the absolute robustness on Fog (0.1 and 0.2 PSNR increase respectively). With $n = 2$, they are also more relatively robust.

Data augmentation. We study if data-augmentation techniques help to improve NeRF-based methods' resistance to visual corruption. We sought image data augmentation at generalizable methods originally designed for classification to help the encoders extract more robust features. However, most augmentations disturb the pixel values and even corrupt the semantic information, making it impossible to train with NeRF-based methods. We test on Augmix [15] that offer minimal deviation on the original image. We augment each image of a training scene with the same set of hyperparameters and train IBRNet [43], MVSNeRF [6] and GPNR [37], and fail to observe improvements upon these methods, for example, IBRNet trained with augmentation have a 0.35 and 0.13 drop in PSNR for clean and corrupted inputs. The main reason those image-based data augmentation offers limited help is that the augmented scenes fail to correspond to a real physical 3D scene, so NeRF-based methods cannot find enough cross-view information to be trained with. We also removed the random crop and random flip operation originally in IBRNet, and found a performance drop in both clean and corrupted data (PSNR decreases by 0.17 and 0.15, respectively).

7. Conclusion

In this paper, we present the first benchmark for evaluating the robustness of NeRF-based methods, which was made possible by introducing two new datasets containing corrupted 3D scenes of several severity types and levels. To succeed in this benchmark, a system must have the ability to recover the physical world despite encountering different types of unseen corruption. Our results show that existing methods suffer significant performance drops in a manner different than recognition models, and standard image-based augmentation offers limited improvements. For generalizable methods, the feature encoder for current architectures contributes little to the robustness. For combating those corruptions, using 2D image restorations might in some way helps but it largely depends on the restoration quality. Our findings offer valuable insights into the robustness of NeRF-based models. We hope this will inspire future research toward developing more robust NeRF systems for real-world applications.

Acknowledgement

This work is supported by a gift from Open Philanthropy, TPU Research Cloud (TRC) program, and Google Cloud Research Credits program.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *European Conference on Computer Vision*, pages 696–712. Springer, 2020. 2

[2] Adrian Azzarelli, Nantheera Anantrasirichai, and David R Bull. Towards a robust framework for nerf evaluation. *arXiv preprint arXiv:2305.18079*, 2023. 3

[3] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019. 2

[4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 3, 5

[5] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Aniruddha Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15691–15700, 2021. 2

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 3, 5, 6, 7, 8

[7] Tianlong Chen, Peihao Wang, Zhiwen Fan, and Zhangyang Wang. Aug-nerf: Training stronger neural radiance fields with triple-level physically-grounded augmentations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15191–15202, 2022. 3

[8] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 5, 6

[9] Yonggan Fu, Ye Yuan, Souvik Kundu, Shang Wu, Shunyao Zhang, and Yingyan Lin. Nerfool: Uncovering the vulnerability of generalizable neural radiance fields against adversarial perturbations. *arXiv preprint arXiv:2306.06359*, 2023. 3

[10] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996. 2

[11] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 2

[12] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016. 1

[13] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021. 2

[14] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 2, 4, 5, 6

[15] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019. 3, 8

[16] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021. 2

[17] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 3

[18] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016. 2

[19] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18963–18974, 2022. 2, 5

[20] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. 2

[21] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 1

[22] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-nerf: Neural radiance fields from blurry images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12861–12870, 2022. 3, 7

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 3

[24] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022. 5, 8

[25] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2, 5

[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. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 3, 5

[27] Michael Niemeyer and Andreas Geiger. Giraffe: Represent-

ing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 2

[28] Naama Pearl, Tali Treibitz, and Simon Korman. Nan: Noise-aware nerfs for burst-denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12672–12681, 2022. 3

[29] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 2

[30] Juan C Pérez, Sara Rojas, Jesus Zarzar, and Bernard Ghanem. Enhancing neural rendering methods with image augmentations. *arXiv preprint arXiv:2306.08904*, 2023. 3

[31] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. 2

[32] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. *arXiv preprint arXiv:2202.03377*, 2022. 2, 5

[33] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020. 2

[34] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12216–12225, 2021. 2

[35] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022. 2, 3

[36] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019. 2

[37] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. *arXiv preprint arXiv:2207.10662*, 2022. 4, 5, 8

[38] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 1, 2

[39] Angtian Wang, Peng Wang, Jian Sun, Adam Kortylewski, and Alan Yuille. Voge: A differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. *arXiv preprint arXiv:2205.15401*, 2022. 3

[40] Chen Wang, Xian Wu, Yuan-Chen Guo, Song-Hai Zhang, Yu-Wing Tai, and Shi-Min Hu. Nerf-sr: High quality neural radiance fields using supersampling. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6445–6454, 2022. 2

[41] Dilin Wang, Chengyue Gong, and Qiang Liu. Improving neural language modeling via adversarial training. In *International Conference on Machine Learning*, pages 6555–6565. PMLR, 2019. 3

[42] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 2

[43] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 3, 5, 6, 7, 8

[44] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 5

[45] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619, 2021. 2

[46] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020. 3

[47] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022. 2

[48] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020. 3

[49] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 3, 5

[50] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 3

[51] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5728–5739, 2022. 7

[52] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3

[53] 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. 5

[54] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning

view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1, 2