MAGNITUDER LAYERS FOR IMPLICIT NEURAL REPRESENTATIONS IN 3D

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving the efficiency and performance of implicit neural representations in 3D, particularly Neural Radiance Fields (NeRF) and Signed Distance Fields (SDF) is crucial for enabling their use in real-time applications. These models, while capable of generating photo-realistic novel views and detailed 3D reconstructions, often suffer from high computational costs and slow inference times. To address this, we introduce a novel neural network layer called the "magnituder", designed to reduce the number of training parameters in these models without sacrificing their expressive power. By integrating magnituders into standard feed-forward layer stacks, we achieve improved inference speed and adaptability. Furthermore, our approach enables a zero-shot performance boost in trained implicit neural representation models through layer-wise knowledge transfer without backpropagation, leading to more efficient scene reconstruction in dynamic environments.

1 INTRODUCTION

In recent years, the integration of advanced machine learning techniques into robotics has transformed the field of robotic manipulation and trajectory planning. As robots increasingly operate in unstructured environments, the ability to perceive and interact with complex scenes becomes essential. Traditional robotic perception methods often rely on explicit geometric models, which can limit flexibility and adaptability. In contrast, implicit neural representations such as Neural Radiance Fields (NeRF) (Mildenhall et al., 2020) and incremental Signed Distance Fields (iSDF) (Ortiz et al., 2022) have emerged as powerful tools that offer robust ways to model 3D scenes with high fidelity.

NeRF, initially developed for novel view synthesis, uses multiple feed-forward layers (FFLs) to encode 3D scenes into a continuous volumetric representation, enabling photo-realistic rendering from arbitrary viewpoints. Its ability to learn intricate scene details from sparse data makes NeRF particularly appealing for robotics applications, especially in dynamic or poorly structured environments. On the other hand, iSDF provides a versatile framework for capturing the geometric properties of objects, facilitating real-time collision detection and path planning. By representing surfaces as implicit functions, iSDF seamlessly integrates shape information into robotic control systems.

Despite the exciting opportunities that NeRF and iSDF offer for enhancing robotic manipulation and trajectory planning, their practical deployment in real-world scenarios still requires improvements in training and inference speed. While iSDF provides real-time mapping, it relies on known camera poses, which adds computational overhead in real-world applications. Accelerating both training and inference can make the entire system faster, expanding the potential for real-time applications.

In real-world applications, robots often encounter novel objects and unpredictable conditions. Rapid adaptation to new scenarios is crucial for maintaining operational efficiency and safety. Quickly retraining models or adapting them on-the-fly can significantly improve a robot's performance in tasks such as grasping or navigating cluttered spaces. Slow training cycles may result in outdated models that fail to accurately represent the current environment, reducing reliability and effectiveness.

Moreover, inference speed is vital for real-time decision-making in robotic systems. Delays in processing visual and spatial data can negatively impact a robot's ability to perform timely actions, which is particularly important in fast-paced environments. For instance, in collaborative settings where robots work alongside humans, latency can create safety risks and hinder task efficiency.

Thus, optimizing inference times while maintaining high accuracy is essential for deploying these advanced representations in practical applications.

The main computational block for NeRF and iSDF are FFLs. To reduce the computational complexity, various methods have been proposed that reduce the burden of FFLs by exploiting explicit representations (Chen et al., 2022; Sun et al., 2022; Müller et al., 2022; Yu et al., 2021; Fridovich-Keil et al., 2022; Liu et al., 2020; Barron et al., 2023), reducing the number of network inferences through sample pruning with additional occupancy flags (Sun et al., 2022; Müller et al., 2022; Li et al., 2023a), or introducing a novel volume rendering procedure (Han et al., 2024).

In this work, we address this challenge by introducing a novel neural network layer, called *magnituder* that can be combined with standard FFLs in an implicit neural representation. The magnituders disentangle the processing of weights and inputs to the layer, only connecting them at the end through simple linear operations. The magnituders use structured transformations of the input to the layer that depend solely on the magnitude of the input (hence their name). This design allows for a reduction in the training parameters of the FFL blocks while maintaining the same level of expressivity as a standard FFL.

To summarize, our contributions in this work are the following :

- We introduce a novel random feature mechanism called *magnituders* that optimizes the computation of FFLs in implicit neural representations (Section 3).
- We demonstrate how this mechanism can approximate FFL with commonly used activation functions like ReLU and Softplus (Section 4.1).
- We showcase the versatility of our method across a wide range of scenes and a variety of implicit neural representations (Section 4.2).
- We distill existing MLP layers in an implicit neural representation with our magnituders which are trained analytically *without backpropagation*. Our method allows for zero-shot integration of our novel layer with pretrained models, improving inference speed without costly retraining. To the best of our knowledge, this is the first successful application of knowledge distillation without backpropagation for implicit neural representations, demonstrating its effectiveness in downstream tasks (Section 4.3).

Note that our method is orthogonal to the sparse sampling and the partitioning techniques and can be combined in such to further reduce the computational footprint.

2 Related Work

Neural Radiance Fields (NeRF) (Mildenhall et al., 2020) have transformed the synthesis of novel views from posed 2D images, offering impressive quality in rendering complex 3D scenes. However, both training and rendering processes remain time-consuming, prompting extensive research aimed at accelerating these tasks, by integrating explicit representations (Liu et al., 2020; Yu et al., 2021; Hu et al., 2022; Tancik et al., 2022; Takikawa et al., 2021; Chen et al., 2022; Sun et al., 2022; Müller et al., 2022; Fridovich-Keil et al., 2022; Xu et al., 2023; Barron et al., 2023; Takikawa et al., 2023; Hu et al., 2023), dividing a scene into smaller blocks (Reiser et al., 2021; Tancik et al., 2022; Turki et al., 2022; Xiangli et al., 2022; Turki et al., 2023), caching (baking) the implicit functions (Garbin et al., 2021; Hedman et al., 2021; Chen et al., 2023; Reiser et al., 2023), and devising some novel tweaks (Lindell et al., 2021; Han et al., 2024). Given their effectiveness, there has also been significant work extending NeRF to accommodate dynamic (Pumarola et al., 2021; Park et al., 2021a;b; Li et al., 2022; Weng et al., 2022; Fang et al., 2022; Park et al., 2023; Cao & Johnson, 2023; Fridovich-Keil et al., 2023; Li et al., 2023c; Wang et al., 2023d; Shao et al., 2023; Attal et al., 2023; Wang et al., 2023a; Lin et al., 2023; Xu et al., 2024), unbounded (Wang et al., 2021b; Barron et al., 2022; Choi et al., 2023; Barron et al., 2023; Wang et al., 2023c), and challenging appearances scenarios (Martin-Brualla et al., 2021; Ichnowski et al., 2021; Mildenhall et al., 2022; Bemana et al., 2022; Verbin et al., 2022; Guo et al., 2022; Fujitomi et al., 2022; Zhan et al., 2023; Warburg et al., 2023; Lee et al., 2023; Kim et al., 2023; Goli et al., 2024; Verbin et al., 2024; Ren et al., 2024).

Signed Distance Fields (SDF) are scalar fields that represent the distance to the nearest surface point, crucial in robotics for tasks such as environment mapping, collision avoidance, and trajectory optimization (Zucker et al., 2013; Toussaint, 2009; Schulman et al., 2014; Mukadam et al., 2018; Dong et al., 2018; 2016; Chaplot et al., 2021; Das & Yip, 2020; Verghese et al., 2022; Zhang et al., 2023; Johari et al., 2023; Deng et al., 2023; Yan et al., 2023; Zhu et al., 2024). In computer vision, Truncated Signed Distance Fields (TSDF) are employed in SLAM systems, as traditional SDF computations can be prohibitively expensive in real time (Newcombe et al., 2011; 2015). Several approaches have emerged to facilitate real-time SDF construction due to its importance in robotics (Ortiz et al., 2022; Han et al., 2019; Oleynikova et al., 2017), while others have addressed challenges in constructing SDF for complex environments (Finean et al., 2021; Reijgwart et al., 2020; Geng et al., 2023; Qiu et al., 2024). To get a photo-realistic quality of scene reconstruction, recent works replace the volume density in NeRF formulations with SDF, using well-designed SDF-to-volume density conversion formulas (Yariv et al., 2020; Wang et al., 2021a; Yariv et al., 2021). Further acceleration is achieved by combining explicit grid structures with implicit SDF (Yu et al., 2022; Wang et al., 2023e; Li et al., 2023b; Wang et al., 2023b; Rosu & Behnke, 2023), populating thin shells near the zero level set (Wang et al., 2023f), or baking SDF into a high-quality triangle mesh (Yariv et al., 2023).

A common challenge in training neural networks is that their space and time complexity scale quadratically with the size of their hidden layers. This challenge is also inherited by implicit neural representations. One of the ways to tackle this issue is via dimensionality reduction à la Johnson-Lindenstrauss Transform (or JLT) (Dasgupta & Gupta, 2003; Dasgupta et al., 2010; Ailon & Liberty, 2013) or kernel methods (Rahimi & Recht, 2007). Kernel methods have a rich history, spanning the linearization of 2-layer neural networks (Cho & Saul, 2009; 2011), Neural Tangent Kernels (NTK)(Jacot et al., 2018), and the linearization of attention mechanisms in Transformers (Choromanski et al., 2020). Kernel methods using random feature mechanisms have shown promise in reducing the computational complexity of neural networks (Choromanski et al., 2020; Rahimi & Recht, 2007). However, these approaches are not directly applicable to the MLP layers that are the backbone of NeRF and iSDF systems. To address the computational complexity of FFLs, Sehanobish et al. (2024) introduced the Universal Random Feature (URF), which disentangles weights and inputs in specific MLP layers. Their method relies on Fourier transforms of activation functions, limiting its applicability to commonly used activation functions in machine learning.

3 MAGNITUDERS

In this section we define our magnituder-layers (MAG-layers in short). Consider a layer $\mathcal{M} : \mathbb{R}^d \to \mathbb{R}^l$ taking as input a d-dimensional vector, outputting an l-dimensional vector and given as follows:

$$\mathcal{M}_{\mathbf{W},\theta}(\mathbf{x}) = \mathbb{E}\left[\mathbf{W}\mathbf{v}_\theta(\|\mathbf{x}\|_2)\right], \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{l\times d}$ is a learnable matrix and $\mathbf{v}_\theta : \mathbb{R} \to \mathbb{R}^d$ is a map operating on the magnitude (length) of the input $\mathbf{x}$. This transformation is potentially learnable (with a set of learnable parameters θ) or probabilistic (in the latter setting, θ encode the parameters of the distribution $\mathcal{D}_\theta$ used to sample the values of $\mathbf{v}_\theta(\mathbf{x})$). The expectation is with respect to $\mathcal{D}_\theta$. In practice, in the probabilistic setting, $\mathcal{M}_{\mathbf{W},\theta}(\mathbf{x})$ is not computed exactly, but unbiasedly approximated by sampling.

Magnituders disentangle the processing of the input $\mathbf{x}$ and weights $\mathbf{W}$, only to connect them via simple linear matrix-vector multiplication at the end. In magnituders, randomness is introduced via multiplications of $\mathbf{x}$ with a Gaussian matrix $\mathbf{G} \in \mathbb{R}^{m\times d}$ where the entries are drawn iid from $\mathcal{N}(0, 1)$, for some $m > 0$ which are subsequently followed by purely deterministic transformations. The parameter m is called the number of random features in a MAG-layer. Under this randomization strategy, taking a kernel perspective, magnituders can be interpreted as outputting a vector of approximate kernel values $\mathrm{K}(\mathbf{w}_i, \mathbf{x})$ for $i = 1, ..., l$, where kernel $\mathrm{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is defined as:

$$\mathrm{K}_{\mathrm{MAG}}(\mathbf{u}, \mathbf{v}) \overset{\mathrm{def}}{=} \sum_{i=1}^{d} u_i \mathbb{E}[f_{\theta_i}(\|\mathbf{v}\|_2 g)], \tag{2}$$

where $g \overset{\mathrm{iid}}{\sim} \mathcal{N}(0, 1)$ and functions $f_{\theta_1}, ..., f_{\theta_d} : \mathbb{R} \to \mathbb{R}$ are deterministic. This comes directly from the fact that for $\mathbf{g} \sim \mathcal{N}(0, \mathbf{I}_d)$, we have: $\mathbf{v}^\top \mathbf{g} \sim \|\mathbf{v}\|_2 \times \mathcal{N}(0, 1)$.

The above kernel can be efficiently and accurately approximated by *quasi Monte Carlo* methods (QMCs).

3.1 Magnituders and QMCs

A straightforward approach to estimating kernels from Equation 2 is through Monte Carlo methods, which is effectively implemented in versions of the magnituder layers that leverage Gaussian matrices $\mathbf{G}$.

To be more specific, the kernel from Eq. 2 is approximated as follows : For $\mathbf{g}_1, ..., \mathbf{g}_m \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbf{I}_d)$:

$$\mathrm{K}_{\mathrm{MAG}}(\mathbf{u}, \mathbf{v}) \approx \sum_{i=1}^{d} u_i f_{\theta_i}(\mathbf{v}^\top \mathbf{g}_i), \tag{3}$$

A standard technique to reduce the variance for approximators based on ensembles of Gaussian vectors is via the so-called *orthogonal random features* or ORFs (Yu et al., 2016). The ensembles of iid vectors are replaced by block-orthogonal ensembles, where within each d-element block vectors have marginal Gaussian distributions $\mathcal{N}(0, \mathbf{I}_d)$, yet different vectors in each block are exactly orthogonal. Blocks are constructed independently. ORFs are popular since they can be easily constructed. For magnituders we have the following variance reduction lemma:

Lemma 3.1. *Denote by* $\widehat{\mathrm{K}}^{\mathrm{iid}}_{\mathrm{MAG}}(\mathbf{u}, \mathbf{v})$ *and estimator from the RHS of Eq. 3 appying iid ensemble:* $\mathbf{g}^{\mathrm{iid}}_1, ..., \mathbf{g}^{\mathrm{iid}}_m$ *and by* $\widehat{\mathrm{K}}^{\mathrm{ort}}_{\mathrm{MAG}}(\mathbf{u}, \mathbf{v})$ *its variant leveraging block-orthogonal ensemble (ORFs):* $\mathbf{g}^{\mathrm{ort}}_1, ..., \mathbf{g}^{\mathrm{ort}}_m$. *If functions* f_{θ_i} *are analytic with non-negative Taylor coefficients and* $u_i \geq 0$ *for* $i = 1, ..., m$ *then:*

$$\mathrm{Var}(\widehat{\mathrm{K}}^{\mathrm{ort}}_{\mathrm{MAG}}(\mathbf{u}, \mathbf{v})) \leq \mathrm{Var}(\widehat{\mathrm{K}}^{\mathrm{iid}}_{\mathrm{MAG}}(\mathbf{u}, \mathbf{v})) \tag{4}$$

Proof. The proof follows directly from Lemma 4 and Theorem 6 in Choromanski et al. (2020). □

We observe that, empirically, even in cases where the conditions of Lemma 3.1 are not satisfied, the ORFs still provide quality gains (see Figure 19).

Finally, we would like to posit our magnituder layers in the broader context of linearizing neural networks via kernel methods. If f_{θ_i}'s are deterministic functions, the right hand side of Equation 2 represents a 2-layer NN where the input layer is fixed and only u_i is trainable. The linearization of such 2-layer NN is widely considered under various conditions (Cho & Saul, 2009; 2011; Ghorbani et al., 2020; Neal, 1996). The notable difference between the previous works and our work is that $\mathrm{K}_{\mathrm{MAG}}$ cannot be modeled as a standard neural network i.e. matrix-vector multiplications followed by point-wise nonlinear transformations, so such techniques can not be readily applied here.

3.2 Connections with SNNK

We now discuss our magnituder layer with another similar mechanism SNNK introduced by Sehanobish et al. (2024). Like magnituders, SNNKs also disentangle the weights and the inputs by leveraging the so-called Universal Random Features (URFs). However there are some key differences, which we detail below:

The SNNK-layers are defined as follows:

$$\mathrm{SNNK}_{\mathbf{W}}(\mathbf{x}) = \mathbb{E}[\Phi(\mathbf{W})\Psi(\mathbf{x})], \tag{5}$$

where: (1) probabilistic map $\Phi : \mathbb{R}^{l \times d}$ is obtained by applying probabilistic maps $\Phi^i : \mathbb{R}^d \to \mathbb{R}^d$ for $i = 1, ..., l$, independently on the verticalized rows $\mathbf{w}_1, ..., \mathbf{w}_l$ of $\mathbf{W}$ respectively and furthermore, (2) $\Psi : \mathbb{R}^d \to \mathbb{R}^d$ is another probabilistic map.

Thus the randomness is introduced in both the weight and the input tower. And the kernel that is associated to SNNK can be defined as :

$$\mathrm{K}_{\mathrm{SNNK}}(\mathbf{u}, \mathbf{v}) \overset{\mathrm{def}}{=} \mathbb{E}_{\mathbf{g} \in \mathcal{N}(0, \mathbf{I}_d)}[\Phi(\mathbf{u}^\top \mathbf{g})\Psi(\mathbf{v}^\top \mathbf{g})] \tag{6}$$

which can be estimated using QMC methods via the following approximation :

$$\mathrm{K}_{\mathrm{SNNK}}(\mathbf{u}, \mathbf{v}) \approx \frac{1}{m} \sum_{i=1}^{m} \Phi(\mathbf{u}^\top \mathbf{g}_i)\Psi(\mathbf{v}^\top \mathbf{g}_i) \tag{7}$$

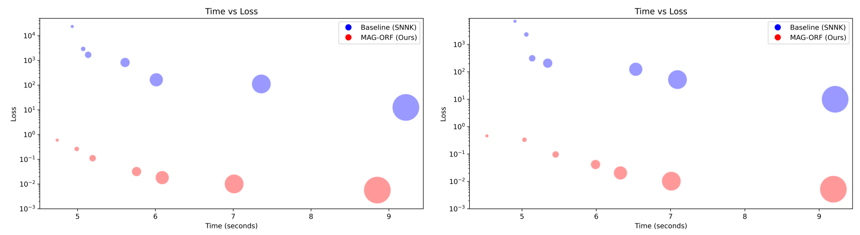


Figure 1: Approximation capability of our MAG layer. Here we compare with the SNNK layer as our baseline. The size of the dots represents the number of training parameters and for fairness, we made sure that the MAG layer and the SNNK has the same number of training parameters. (Left) : Approximating a ReLU-Linear layer. (Right) : Approximating a Softplus-Linear layer.

However, rigorous theoretical results regarding variance reduction from ORFs for the estimators defined in the RHS of Eq. 7 are known only for very specific nontrivial functions Φ and Ψ, i.e. the exponential map: $z \rightarrow \exp(c \times z)$ (Choromanski et al., 2020) and sgn-function $z \rightarrow \mathrm{sgn}(z)$ (Choromanski et al., 2017). Moreover, we believe that the explicit dependence on $\|\mathbf{x}\|_2$ in magnituders, enables them to outperform SNNKs in applications leveraging implicit neural-network-based representations, as we show in Sec. 4.1 (Figure 1).

4 Experiments

In this section, we present a comprehensive empirical evaluation of our MAG-layers. Specifically, we design experiments to answer the following research questions:

- (Q1) How accurate are MAGs in approximating some commonly used FFLs?
- (Q2) How can MAGs be used to speed up various existing INRs?
- (Q3) How can MAGs be used to improve inference speeds of an already trained INR?

In all our experiments, we use the MAG-layer of the form $\mathbf{M}_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}f(\mathbf{G}\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{d' \times m}$ is a trainable matrix, $\mathbf{G} \in \mathbb{R}^{m \times d}$ is a random matrix, d (resp. d') is the dimension of the input (resp. output), m is the number of random features, and $f = \mathrm{ReLU}$ applied point-wise. By choosing $m << d$, we can achieve a reduction in training parameters while maintaining comparable performance, as we will demonstrate in these experiments. Additional experiments including detailed ablation studies can be found in Appendix D.

4.1 Synthetic Experiments

First, we justify our choice of f by demonstrating the superiority of our magnituder (MAG) layers in approximating both ReLU-linear and SoftPlus-linear layers. These activations are selected because they are commonly used in downstream applications. Additionally, we show the performance boost over the existing random feature mechanism (SNNK) (Sehanobish et al., 2024). We replicate this experiment over 10 random seeds and present the average of these runs in Figure 1. More details can be found in Appendix C.1. Furthermore, we support our choice of using ORFs (i.e. $\mathbf{G}$ is an orthogonal random matrix) with the results presented in Figure 19.

Thus in the subsequent sections, we will only use ORFs in our MAG-layers.

4.2 MAG-Layers for Implicit Neural Representations

In this section, we show how MAG layers can be seamlessly injected into various implicit neural representation methods including NeRF (Mildenhall et al., 2020), Zip-NeRF Barron et al. (2023), D-NeRF (Pumarola et al., 2021) and iSDF (Ortiz et al., 2022), to accelerate existing methods. Additionally, we introduce novel design choices for these models to further enhance inference speeds.

Recent works have shown training the projection matrix can lead to accuracy gains (Chowdhury et al., 2022; Zhang et al., 2024). We call those variants MAG-layers with trainable $\mathbf{G}$.

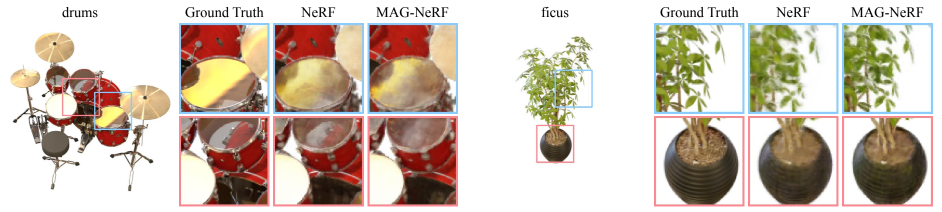
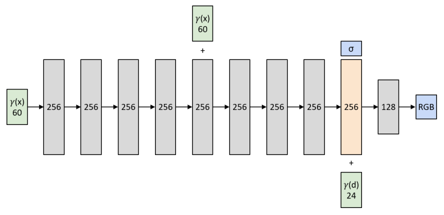


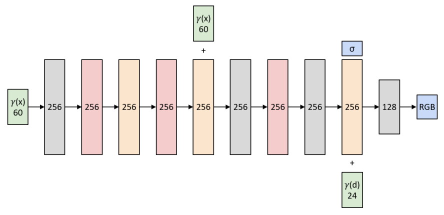
Figure 2: Results of rendering through NeRF versus MAG-NeRF: *drums* (left) and *ficus* (right). The Ground Truth column shows the reference image, followed by the rendered results.

Table 1: Comparison between NeRF and our MAG-NeRF on the Synthetic NeRF dataset (Mildenhall et al., 2020). We report the mean results for all 8 scenes. The inference time refers to the approximate time for forward passes of each model to render a single 400×400 image. Our MAG-NERF is only 1% behind the baseline while being 24% faster.

Metric	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Inference time (↓) (s)	Model Size (↓) (MB)
NeRF	30.45	0.950	0.032	2.36	3.27
MAG-NeRF (ours)	30.10	0.945	0.035	1.79	2.27



(a) Baseline NeRF architecture



(b) MAG-NeRF architecture

Figure 3: Network architecture of the baseline NeRF and our MAG-NeRF model. **Black** denotes Linear ReLU layer, **Orange** is Linear without activation, **Red** is the MAG layer, + is concatenation, and the activation for the RGB output is Softmax. Note that MAG layer followed by the Linear layer can be compressed for more efficient network inference.

4.2.1 NeRF Experiments

In this section, we focus on integrating MAG-layers into NeRF models (Mildenhall et al., 2020). While NeRF has demonstrated impressive capabilities in generating photo-realistic scenes, it suffers from long training and inference times. Thus, to accelerate inference times and reduce model sizes, we replace three MLP layers in the baseline NeRF architecture with our random feature mechanism, which we refer to as MAG-NeRF (see Figure 3). Our design choice enables the "bundling process", allowing subsets of layers to be grouped together to create a lean, efficient model without sacrificing accuracy (see Appendix A).

We evaluate both baseline NeRF and MAG-NeRF using the PyTorch implementation of NeRF (Yen-Chen, 2020) on the Synthetic NeRF dataset (Mildenhall et al., 2020). We train MAG-NeRF in the same manner as the baseline NeRF by running 200k iterations and rendering the test images at half resolution. During inference, we apply a bundling process to the MAG-NeRF network by collapsing the MAG layer with any subsequent Linear layers before nonlinear activation. When concatenation is involved, we decompose the weight matrix into two parts and bundle them together. For instance, in Figure 3 (b), we bundle three consecutive layers (4^{th}-6^{th} arrows: Red, Orange, and Black) by decomposing the Linear-ReLU layer (black arrow) into two parts: 60×256 and 256×256.

In Table 1 and Figure 2, we present the quantitative and qualitative results of our MAG-NeRF. We report the rendering quality metrics with PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018), along with the network inference time required to render an image. MAG-NeRF significantly reduces both inference time and model size by approximately **24**% and **30**%, respectively, with virtually no loss in reconstruction quality. Detailed results on each scene is presented in Tables 6, 7 and 8. Other rendered images are in Figures 10 and detailed ablation studies can be found in Figure 17 as well as in Table 14.

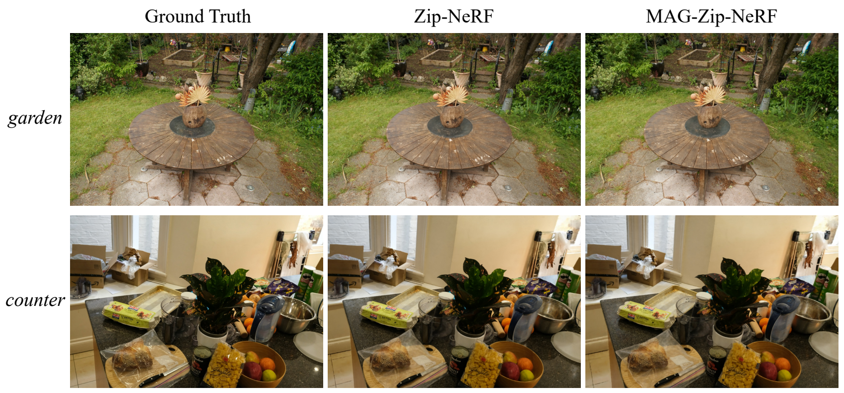


Figure 4: Rendered images from the Mip-NeRF 360 dataset (Barron et al., 2022). For each scene, the images from left to right show the Ground Truth, Zip-NeRF, and MAG-Zip-NeRF.

Table 2: Comparison between Zip-NeRF (Barron et al., 2023) and MAG-Zip-NeRF on the *360_v2* dataset (Barron et al., 2022). We report the mean results for all 7 scenes. Our trainable **G** variant is only .9% behind the baseline while being 6% faster.

Metric	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Inference time (↓) (s)
Zip-NeRF	28.18	0.823	0.193	4.03
MAG-Zip-NeRF (ours)	27.81	0.815	0.198	3.78
MAG-Zip-NeRF w/ trainable **G** (ours)	27.94	0.818	0.196	3.78

4.2.2 Zip-NeRF experiments

Prior works have accelerated NeRF by integrating explicit grid structures with MLPs (Chen et al., 2022; Sun et al., 2022; Müller et al., 2022; Barron et al., 2023). By incorporating these explicit structures, networks can encode spatial information more efficiently, reducing the computational burden on MLPs. Despite these improvements, there remains room for further acceleration within the MLP architectures themselves. We address this shortcoming using our MAG layers.

We specifically focus on Zip-NeRF (Barron et al., 2023), which has demonstrated exceptional performance in unbounded scenes and relies on shallow MLPs to decode grid-based features. We use the Pytorch implementation of Zip-NeRF (Gu, 2023) for both the baseline and our modification. Similar to our approach with the baseline NeRF architecture, we identify two consecutive linear layers with ReLU activations within Zip-NeRF and replace the first of these with our MAG layer, resulting in a variant we refer to as MAG-Zip-NeRF. Since this replacement occurs in only a single place, this experiment serves as an extreme case study for the effectiveness of our method.

Figure 4 and Table 2 present the qualitative and quantitative results. Despite this minimal modification, we maintain similar rendering quality while improving the rendering speed by **6**%. Figure 11 shows the rendered images for other scenes.

4.2.3 Dynamic NeRF experiments

For our next NeRF experiment, we focus on D-NeRF (Pumarola et al., 2021), which can synthesize novel views of dynamic scenes with complex non-rigid geometries. It represents a dynamic scene using an implicit neural network by modeling the dynamic movement, named deformation network. We choose Pytorch implementation of D-NeRF (Tang, 2022) which further integrates the hash-grid (Müller et al., 2022) to accelerate the training and rendering. We inject our MAG layer in the deformation network, thereby accelerating the rendering of dynamic NeRF (see Appendix C.4 and Figure 12 for more details).

We present the quantitative results in Table 3 with the rendered images in Figure 13. We achieve a **7**% speedup in inference while maintaining a similar reconstruction quality.

Table 3: Comparison between D-NeRF and our MAG-D-NeRF on the Dynamic NeRF dataset (Pumarola et al., 2021).As an average over 8 scenes, our trainable $\mathbf{G}$ variant is only 1% behind the baseline while being 7% faster.

Metric	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Inference time (↓) (s)
D-NeRF	30.70	0.954	0.033	1.269
MAG-D-NeRF (ours)	30.26	0.948	0.038	1.177
MAG-D-NeRF w/ trainable $\mathbf{G}$ (ours)	30.32	0.948	0.038	1.177

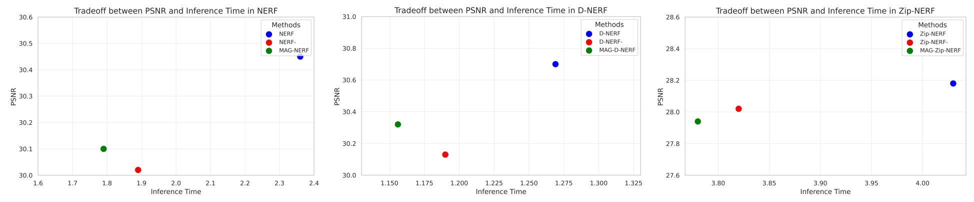


Figure 5: Comparison of MAG models with baseline models with same number of trainable parameters. In all cases, our MAG-models achieves better or similar reconstruction quality than the base models with same number of parameters while being faster. Model- refers to the baseline models with the same number of parameters as the MAG-model. Left : NERF. Middle : D-NERF. Right : Zip-NERF.

4.2.4 Mip-NeRF 360 Experiments

For our final NeRF experiment, we focus on Mip-NeRF 360 (Barron et al., 2022) as it is a SOTA architecture for novel view synthesis. Due to the computational expense of this experiment, we exclusively use the trainable $\mathbf{G}$ variant, as it is our most performant configuration. We observe that our model is only 2% off from the baseline while being 18% faster during inference (see Table 4).

4.2.5 iSDF experiments

In our final application, we explore iSDF (Ortiz et al., 2022), a real-time module designed to reconstruct SDF from depth sequences. This application highlights the versatility of our method, showcasing its effectiveness in real-time scenarios. The network architecture consists of a 7-layer MLP (See Fig. 14). We replace Softplus with ReLU for initial speedup and then evaluate whether further acceleration can be achieved with MAG layers (See Appendix D and Fig. 16 for justification).

We modify the original iSDF network using a similar strategy as in our NeRF experiments, resulting in our new variant, MAG-iSDF (details are provided in Appendix C.6). We evaluate the accuracy of our MAG-iSDF in ReplicaCAD (Szot et al., 2021) sequences using three metrics: average SDF error ($\mathrm{L1}_{\mathrm{avg}}$) over the entire scene, surface SDF error ($\mathrm{L1}_{\mathrm{surf}}$) on GT mesh surface, and the one-way Chamfer Distance (CD) from the reconstructed to the GT mesh. We also report the training time per step (t_{train}) including sample generation and forward-backward passes, and inference time (t_{infer}) by measuring 1000 random rays' forward pass.

In Table 5, we compare the original iSDF with MAG-iSDF. Our method is only 2% behind the baseline while being $\mathbf{23}\%$ faster. We show detailed ablation results with different number of random features in Table 12. The results reveal a trade-off between computational speed and reconstruction quality (Fig. 7 (Right)). Moreover, when we visualize slices of the reconstructed SDFs on the xy-plane at $z = 70$ cm, the reconstruction quality remains quite similar (Fig. 6).

Reducing hidden dimensions is a straightforward method to accelerate training. For fair comparison, we reduce the number of hidden channels in the last layer to 32 and replace the last hidden layer with a MAG layer containing 32 random features. We evaluate this on the longest sequence, *apt_2_nav*, from the Replica dataset in iSDF. In Figure 7, we plot the inference time and Chamfer Distance. While hidden dimension reduction (DR) reduces inference time at the expense of quality, our MAG layer achieves a similar speed-up while maintaining reconstruction quality. The bundling process combines the last two layers into a single operation, leading to additional speed-up without compromising quality (Figure 7).

Table 4: Comparison between Mip-NeRF 360 (Barron et al., 2022) and MAG-Mip-NeRF 360 on the *360_v2* dataset. As an average over 7 scenes, our model is only 2% behind the baseline while being 18% faster.

Metric	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Inference time (↓) (s)
Mip-NeRF 360	27.27	0.733	0.335	33.15
MAG-Mip-NeRF 360 (ours)	26.81	0.710	0.363	27.30

Table 5: Quantitative results of MAG-iSDF. $\mathrm{L1}_{\mathrm{avg}}$ is calculated across all SDF levels, and $\mathrm{L1}_{\mathrm{surf}}$ is for ground truth surface points. t_{train} refers to iteration time (including backward pass), and t_{infer} measures forward pass inference time. Results are averaged across 6 ReplicaCAD (Szot et al., 2021) sequences with 3 random seeds. Our MAG-iSDF model achieves less than similar quality as the baseline while being 23% faster.

		Distance			Time	
Method	Trainable **G**	$\mathrm{L1}_{\mathrm{avg}}$ (↓) [cm]	$\mathrm{L1}_{\mathrm{surf}}$ (↓) [cm]	CD (↓) [cm]	t_{train} (↓) [ms]	t_{infer} (↓) [ms]
iSDF	NA	8.99	4.08	2.02	8.24	1.37
MAG-iSDF (ours)	No	9.62	4.28	2.12	7.74	1.06
	Yes	9.37	4.17	2.10	7.81	1.06

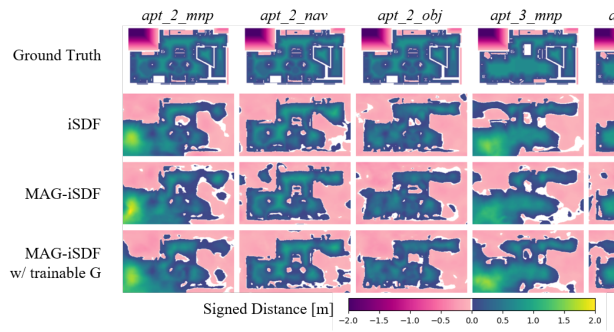


Figure 6: Visualization of reconstructed SDFs in iSDF for 6 ReplicaCAD dataset. Each figure shows the reconstructed SDF, following the colormap at the bottom.

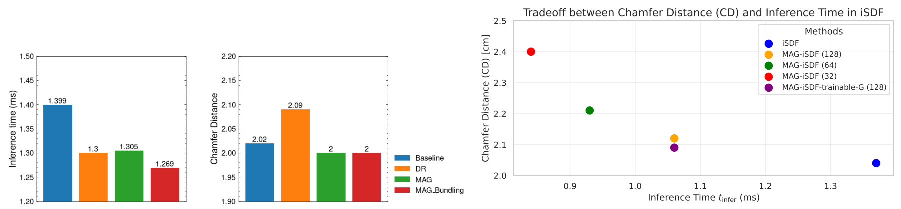


Figure 7: (Left) : Comparison of two acceleration methods: DR and MAG. The left figure shows network inference time and the right figure shows the Chamfer distance of the reconstructed mesh. MAG accelerates the network more efficiently than DR while preserving the reconstruction quality. (Right) : Trade-off between reconstruction quality and speed for various MAG models with 32, 64, 128 random features.

4.3 Knowledge Distillation with MagnitudeR

In previous sections, we demonstrate that our MAG layer can significantly accelerate the inference time of implicit neural representations. However, this acceleration requires retraining the model from scratch, a process that is both computationally intensive and time-consuming, especially for complex models like NeRF and Zip-NeRF.

To address this limitation, we introduce a layer distillation mechanism that eliminates the need for retraining on a per-scene basis. This method allows us to accelerate existing, pretrained models without having to rebuild them from the ground up. By storing the inputs and outputs of the target hidden layer in trained implicit neural representations, and then optimizing the MAG layer using a mean squared error objective, we can efficiently replicate the behavior of the original model. This optimization can be performed without the need for *backpropagation* due to the existence of a closed-form solution (Additional details in Section B). We present additional details on distillation experiments with trainable $\mathbf{G}$ in Appendix D.5).

First, we distill a pretrained Zip-NeRF model with a MAG layer and test the distillation in *bicycle* dataset (additional details in Appendix C.3). We set the number of random features to 64, with ablations provided in Appendix D.3 (Figure 18 and Table 15). The qualitative results of the distillation are shown in Figure 9. Notably, this distillation process is lightning fast, taking only $\mathbf{0.12}$ seconds on a single RTX 4090, thanks to the closed-form solution for the input-output pairs. Since we use the same number of random features as in Section 4.2.2, we achieve the same $\mathbf{6}\%$ rendering acceleration without needing to train from scratch. We show that we can recover the quality of the original NERF model while reducing the inference speeds by up to $\mathbf{42}\%$ (Table 13 and Fig 8 (right)).

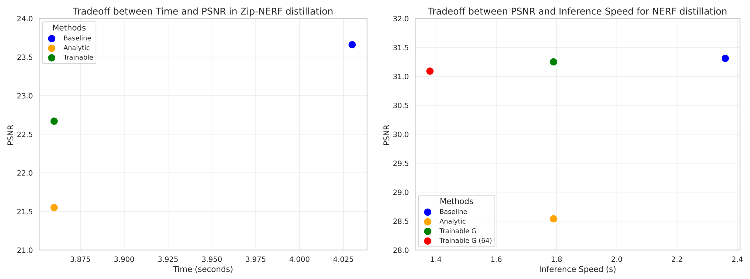
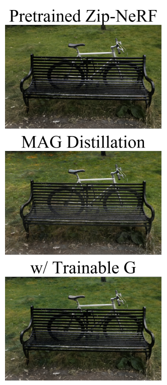
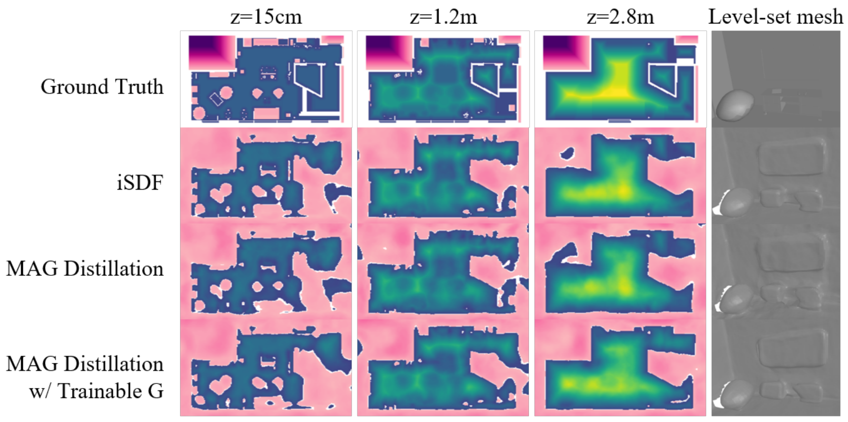


Figure 8: Distillation results comparing the analytic and trainable variants with the baseline. The analytic and trainable use 128 random features. Left : Zip-NERF, Right : NERF. Trainable $\mathbf{G}(64)$ uses 64 random features.

We also test the distillation in the last hidden layer of iSDF (Ortiz et al., 2022). In Figure 9, we visualize the reconstructed SDF and the extracted zero level-set mesh from *apt_2_nav* dataset. Our results indicate that the reconstruction quality of the distilled MAG-iSDF is comparable to that of the baseline iSDF, with a $\mathbf{9}\%$ improvement in inference speed.



(a) Zip-NeRF



(b) iSDF

Figure 9: Visualization of reconstructed SDFs with MAG distillation. The last column shows the zoom-in rendering of zero level-set mesh.

5 Conclusion

We introduce a novel type of neural network layer, termed 'magnituders', which can efficiently approximate the computations of ReLU and Softplus linear layers. By integrating magnituders into Neural Radiance Fields (NeRF) and Signed Distance Fields (SDF) models, we achieve a reduction in training parameters while preserving expressivity. Our approach enhances inference speed and adaptability for real-time robotic applications. Additionally, the straightforward design of magnituders facilitates layer-wise knowledge distillation without requiring backpropagation. We demonstrate that this distillation process leads to efficient inference.

Ethics Statement

This paper focuses mostly on developing novel neural network layers that can lower the computation footprint of certain feedforward layers. The experiments with NeRF and iSDF illustrate how this method can lower the inference speed while remaining competitive with the baselines. It should be noted though that even though these models are widely used, these models have considerable computational footprint and thus the corresponding carbon footprint.

Reproducibility Statement

Details and the code pointers to reproduce each experiment are provided in Appendix C.

References

Nir Ailon and Edo Liberty. An almost optimal unrestricted fast johnson-lindenstrauss transform. *ACM Trans. Algorithms*, 9(3), jun 2013. ISSN 1549-6325. doi: 10.1145/2483699.2483701. URL `https://doi.org/10.1145/2483699.2483701`.

Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16610–16620, 2023.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19697–19705, 2023.

Mojtaba Bemana, Karol Myszkowski, Jeppe Revall Frisvad, Hans-Peter Seidel, and Tobias Ritschel. Eikonal fields for refractive novel-view synthesis. In *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–9, 2022.

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*, pp. 130–141, 2023.

Devendra Singh Chaplot, Deepak Pathak, and Jitendra Malik. Differentiable spatial planning using transformers, 2021. URL `https://arxiv.org/abs/2112.01010`.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper_files/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf`.

Youngmin Cho and Lawrence K. Saul. Analysis and extension of arc-cosine kernels for large margin classification, 2011. URL `https://arxiv.org/abs/1112.3712`.

Changwoon Choi, Sang Min Kim, and Young Min Kim. Balanced spherical grid for egocentric view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16590–16599, 2023.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

Krzysztof Marcin Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 219–228, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/bf8229696f7a3bb4700cfddef19fa23f-Abstract.html`.

Sankalan Pal Chowdhury, Adamos Solomou, Kumar Avinava Dubey, and Mrinmaya Sachan. Learning the transformer kernel. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL `https://openreview.net/forum?id=tLIBAEYjcv`.

Nikhil Das and Michael Yip. Learning-based proxy collision detection for robot motion planning applications. *IEEE Transactions on Robotics*, 36(4):1096–1114, 2020. doi: 10.1109/TRO.2020.2974094.

Anirban Dasgupta, Ravi Kumar, and Tamás Sarlos. A sparse johnson: Lindenstrauss transform. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, pp. 341–350, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300506. doi: 10.1145/1806689.1806737. URL `https://doi.org/10.1145/1806689.1806737`.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. doi: https://doi.org/10.1002/rsa.10073. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.10073`.

Junyuan Deng, Qi Wu, Xieyuanli Chen, Songpengcheng Xia, Zhen Sun, Guoqing Liu, Wenxian Yu, and Ling Pei. Nerf-loam: Neural implicit representation for large-scale incremental lidar odometry and mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8218–8227, 2023.

Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Proceedings of Robotics: Science and Systems (RSS)*, 2016.

Jing Dong, Mustafa Mukadam, Byron Boots, and Frank Dellaert. Sparse Gaussian processes on matrix Lie groups: A unified framework for optimizing continuous-time trajectories. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6497–6504. IEEE, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

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*, pp. 1–9, 2022.

Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. Predicted composite signed-distance fields for real-time motion planning in dynamic environments, 2021. URL `https://arxiv.org/abs/2008.00969`.

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*, pp. 5501–5510, 2022.

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*, pp. 12479–12488, 2023.

Taku Fujitomi, Ken Sakurada, Ryuhei Hamaguchi, Hidehiko Shishido, Masaki Onishi, and Yoshinari Kameda. Lb-nerf: light bending neural radiance fields for transparent medium. In *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 2142–2146. IEEE, 2022.

Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

Shuang Geng, Qianhao Wang, Lei Xie, Chao Xu, Yanjun Cao, and Fei Gao. Robo-centric esdf: A fast and accurate whole-body collision evaluation tool for any-shape robotic planning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 290–297, 2023. doi: 10.1109/IROS55552.2023.10342074.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension, 2020. URL `https://arxiv.org/abs/1904.12191`.

Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20061–20070, 2024.

Chun Gu. zipnerf-pytorch. `https://github.com/SuLvXiangXin/zipnerf-pytorch/`, 2023.

Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. Nerfren: Neural radiance fields with reflections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18409–18418, 2022.

Kang Han, Wei Xiang, and Lu Yu. Volume feature rendering for fast neural radiance field reconstruction. *Advances in Neural Information Processing Systems*, 36, 2024.

Luxin Han, Fei Gao, Boyu Zhou, and Shaojie Shen. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4423–4430, 2019. doi: 10.1109/IROS40897.2019.8968199.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5875–5884, 2021.

Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12902–12911, 2022.

Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19774–19783, 2023.

Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.

Yoonwoo Jeong, Seungjoo Shin, and Kibaek Park. Nerf-factory: An awesome pytorch nerf collection, 2022. URL `https://github.com/kakaobrain/NeRF-Factory/`.

Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17408–17419, 2023.

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

Sang Min Kim, Changwoon Choi, Hyeongjun Heo, and Young Min Kim. Robust novel view synthesis with color transform module. In *Computer Graphics Forum*, volume 42, pp. e14931. Wiley Online Library, 2023.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Junho Lee, Sang Min Kim, Yonghyeon Lee, and Young Min Kim. Nfl: Normal field learning for 6-dof grasping of transparent objects. *IEEE Robotics and Automation Letters*, 2023.

Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966*, 2023a.

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*, pp. 5521–5531, 2022.

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8456–8465, 2023b.

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*, pp. 4273–4284, 2023c.

Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. High-fidelity and real-time novel view synthesis for dynamic scenes. In *SIGGRAPH Asia 2023 Conference Papers*, pp. 1–9, 2023.

David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14556–14565, 2021.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

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 *ECCV*, 2020.

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*, pp. 16190–16199, 2022.

Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time Gaussian process motion planning via probabilistic inference. In *The International Journal of Robotics Research (IJRR)*, volume 37, pp. 1319–1340, 2018.

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.

Radford M. Neal. Priors for infinite networks, 1996. URL `https://api.semanticscholar.org/CorpusID:118117602`.

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011. doi: 10.1109/ISMAR.2011.6092378.

Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Robotics: Science and Systems*, 2022.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5865–5874, 2021a.

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*, 2021b.

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*, pp. 4212–4221, 2023.

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*, pp. 10318–10327, 2021.

Ri-Zhao Qiu, Yafei Hu, Ge Yang, Yuchen Song, Yang Fu, Jianglong Ye, Jiteng Mu, Ruihan Yang, Nikolay Atanasov, Sebastian Scherer, et al. Learning generalizable feature fields for mobile manipulation. *arXiv preprint arXiv:2403.07563*, 2024.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pp. 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234, January 2020. ISSN 2377-3774. doi: 10.1109/lra.2019.2953859. URL `http://dx.doi.org/10.1109/LRA.2019.2953859`.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021. URL `https://arxiv.org/abs/2103.13744`.

Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023.

Weining Ren, Zihan Zhu, Boyang Sun, Jiaqi Chen, Marc Pollefeys, and Songyou Peng. Nerf on-the-go: Exploiting uncertainty for distractor-free nerfs in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8931–8940, 2024.

Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8466–8475, 2023.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *Int. J. Rob. Res.*, 33(9):1251–1270, aug 2014. ISSN 0278-3649. doi: 10.1177/0278364914528132. URL `https://doi.org/10.1177/0278364914528132`.

Arijit Sehanobish, Krzysztof Marcin Choromanski, Yunfan Zhao, Kumar Avinava Dubey, and Valerii Likhosherstov. Scalable neural network kernels. In *The Twelfth International Conference on Learning Representations*, 2024.

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*, pp. 16632–16642, 2023.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5459–5469, 2022.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11358–11367, 2021.

Towaki Takikawa, Thomas Müller, Merlin Nimier-David, Alex Evans, Sanja Fidler, Alec Jacobson, and Alexander Keller. Compact neural graphics primitives with learned hash probing. In *SIGGRAPH Asia 2023 Conference Papers*, pp. 1–10, 2023.

Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. *arXiv*, 2022.

Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1049–1056, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553508. URL `https://doi.org/10.1145/1553374.1553508`.

Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12922–12931, June 2022.

Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12375–12385, 2023.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5481–5490. IEEE, 2022.

Dor Verbin, Pratul P Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T Barron. Nerf-casting: Improved view-dependent appearance with consistent reflections. *arXiv preprint arXiv:2405.14871*, 2024.

Mrinal Verghese, Nikhil Das, Yuheng Zhi, and Michael Yip. Configuration space decomposition for scalable proxy collision checking in robot planning and control. *IEEE Robotics and Automation Letters*, 7(2):3811–3818, 2022. doi: 10.1109/LRA.2022.3147458.

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*, pp. 21128–21137, 2023a.

Fangjinhua Wang, Marie-Julie Rakotosaona, Michael Niemeyer, Richard Szeliski, Marc Pollefeys, and Federico Tombari. Unisdf: Unifying neural representations for high-fidelity 3d reconstruction of complex scenes with reflections. *arXiv preprint arXiv:2312.13285*, 2023b.

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*, 2021a.

Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4150–4159, 2023c.

Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19795–19806, 2023d.

Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3295–3306, 2023e.

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.

Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *arXiv preprint arXiv:2311.10091*, 2023f.

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021b.

Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18120–18130, 2023.

Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, pp. 16210–16220, 2022.

Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, pp. 106–122. Springer, 2022.

Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8296–8306, 2023.

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*, pp. 20029–20040, 2024.

Zike Yan, Haoxiang Yang, and Hongbin Zha. Active neural mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10981–10992, 2023.

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.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 1–9, 2023.

Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5752–5761, 2021.

Felix X. Yu, Ananda Theertha Suresh, Krzysztof Marcin Choromanski, Daniel N. Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1975–1983, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/53adaf494dc89ef7196d73636eb2451b-Abstract.html.

Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in neural information processing systems*, 35:25018–25032, 2022.

Yifan Zhan, Shohei Nobuhara, Ko Nishino, and Yinqiang Zheng. Nerfrac: Neural radiance fields through refractive surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18402–18412, 2023.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4g02l2N2Nx.

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*, pp. 586–595, 2018.

Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3727–3737, 2023.

Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pp. 42–52. IEEE, 2024.

Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10): 1164–1193, 2013. doi: 10.1177/0278364913488805. URL https://doi.org/10.1177/0278364913488805.

A COMPRESSING NETWORKS BY COMBINING LAYERS

For training NERF, we use a RF layer followed by a linear layer. In this section, we explain how this choice of architecture allows for more efficient inference. In this section, we use the PyTorch convention of describing a linear layer.

Recall the RF layer be given by the following equation :

$$\mathbf{Y}_1 = \phi_2(\mathbf{X_1})\mathbf{W_1}^\top \tag{8}$$

If the RF layer is followed by the linear layer with weight $\mathbf{W}_2$ and bias $\mathbf{b}_2$, then

$$\begin{aligned}\mathbf{Y}_2 &= \mathbf{Y_1}\mathbf{W_2}^\top + \mathbf{b_2}\\ &= \phi_2(\mathbf{X_1})\mathbf{W_1}^\top\mathbf{W_2}^\top + \mathbf{b_2}\\ &= \phi_2(\mathbf{X_1})\hat{\mathbf{W}}_2^\top + \mathbf{b_2}\end{aligned}$$

where $\hat{\mathbf{W}}_2 := \mathbf{W_2}\mathbf{W_1}$.

Note that the above compression works when a non-linearity is applied on the weights (as in the case of (Sehanobish et al., 2024)) or an optional bias in Equation 8.

This simple compression technique allows us to collapse the final 2 layers in our iSDF experiments allowing for more efficient inference.

B KNOWLEDGE DISTILLATION MADE EASY

The simple form of the RF layer makes knowledge distillation particularly simple.

To be precise, we want to train a MAG layer $\mathcal{M}$ to match the outputs of a trained FFL using MSE as the metric. If $\mathbf{x}$ (resp. $\mathbf{y}$) is the input (resp. output of a trained FFL, we want to find the weights $\hat{\mathbf{W}}$, which minimizes the following error :

$$||\mathcal{M}_{\mathbf{W}}(\mathbf{x}) - \mathbf{y}||_2$$

The optimal $\hat{\mathbf{W}}$ has a closed form :

$$\hat{\mathbf{W}} = (\mathbf{x}'^\top\mathbf{x}')^{-1}\mathbf{x}'^\top\mathbf{y} \tag{9}$$

where $\mathbf{x}' := \text{ReLU}(\mathbf{G}\mathbf{x})$, $\mathbf{G}$ is an (orthogonal) random matrix.

The above solution is clearly unique iff $\mathbf{x}'^\top\mathbf{x}'$ is invertible. In the case where $\mathbf{x}'^\top\mathbf{x}'$ is not invertible, one can replace the inverse by the Moore-Penrose pseudoinverse.

Thus the problem of distilling specific input-output pairs in the MAG layer can be solved *without backpropagation*.

C EXPERIMENTAL DETAILS

In this section, we provide additional details for the experiments in our main paper. Our code for the MAG layer implementation can be found `https://anonymous.4open.science/r/SNNK-NERF-2FF7`, and other experiment codes are available at `https://anonymous.4open.science/r/iSDF_MAG-2242`, `https://anonymous.4open.science/r/dnerf_MAG-69D0`, `https://anonymous.4open.science/r/zipnerf_MAG-22FF`, and `https://anonymous.4open.science/r/nerf-pytorch_MAG-425F`. Except for the core MAG-layer, all the other repos are forks of the open source repos of the respective projects.

C.1 SYNTHETIC EXPERIMENTS

In this subsection, we provide additional experimental details for the synthetic experiments. For these experiments, 10,000 samples are drawn randomly from the range $(0, 1)$ with a dimension of

Table 6: PSNR of rendered images for NeRF reconstruction in Synthetic NeRF Mildenhall et al. (2020) dataset. Higher is better.

Scene	chair	drums	ficus	hotdog	lego	materials	mic	ship
NeRF	33.70	25.46	26.40	35.81	31.31	29.00	33.11	28.79
MAG-NeRF	32.76	24.92	28.11	35.22	30.14	28.62	32.42	28.61

Table 7: SSIM of rendered images for NeRF reconstruction in Synthetic NeRF Mildenhall et al. (2020) dataset. Higher is better.

Scene	chair	drums	ficus	hotdog	lego	materials	mic	ship
NeRF	0.978	0.930	0.939	0.979	0.965	0.956	0.979	0.870
MAG-NeRF	0.972	0.922	0.958	0.975	0.952	0.951	0.975	0.857

Table 8: LPIPS of rendered images for NeRF reconstruction in Synthetic NeRF Mildenhall et al. (2020) dataset. Lower is better.

Scene	chair	drums	ficus	hotdog	lego	materials	mic	ship
NeRF	0.013	0.049	0.048	0.012	0.017	0.021	0.020	0.074
MAG-NeRF	0.019	0.056	0.025	0.017	0.025	0.025	0.026	0.084

512, and are then passed through a Linear layer of shape $(512, 512)$. Non-linearity of ReLU (resp. Softplus) is applied on the outputs of the linear layer. Our aim is to show that our MAG-layer can accurately approximate these outputs.

We use the SNNK layers (Sehanobish et al., 2024) as a baseline. We train MAG and SNNK layers for 1k epochs with $8, 16, 32, 64, 128, 256, 512$ random features using the Adam optimizer (Kingma, 2014). We repeat this experiment 10 times and report the average time vs the MSE loss across the replicates. This experiment is run on free Google Colab equipped with a T4 GPU with 12Gb of RAM.

We show that our new MAG layers can accurately approximate the outputs of these linear layers while SNNKs struggle to fit the data.

C.2 NeRF Experiments

In NeRF experiments, we use the default configuration of the Pytorch implementation of NeRF (Yen-Chen, 2020). The inference time is computed using a single NVIDIA RTX 4090 and an AMD Ryzen 7 7700 8-Core Processor. All tests are conducted on the Synthetic NeRF dataset (Mildenhall et al., 2020), and the details of our architecture choices are illustrated in Figure 3. In our experiments, we set the number of random features to 256, with additional results for the different number of random features provided in Section D.2.

We report the detailed results for each scene in Tables 6, 7 and 8 and the rendering results are in Figure 10.

C.3 Zip-NeRF Experiments

Zip-NeRF is an extension of Mip-NeRF 360 (Barron et al., 2022), integrating explicit hash-grid for acceleration (Müller et al., 2022). The architecture consists of two shallow proposal MLPs, which help determine where samples should be placed, and a NeRF MLP, responsible for generating the final color output. We find that accelerating the proposal MLPs was challenging due to their shallow architecture and low-dimensional mapping ($6 \to 64 \to 1$). So we choose NeRF MLP as our

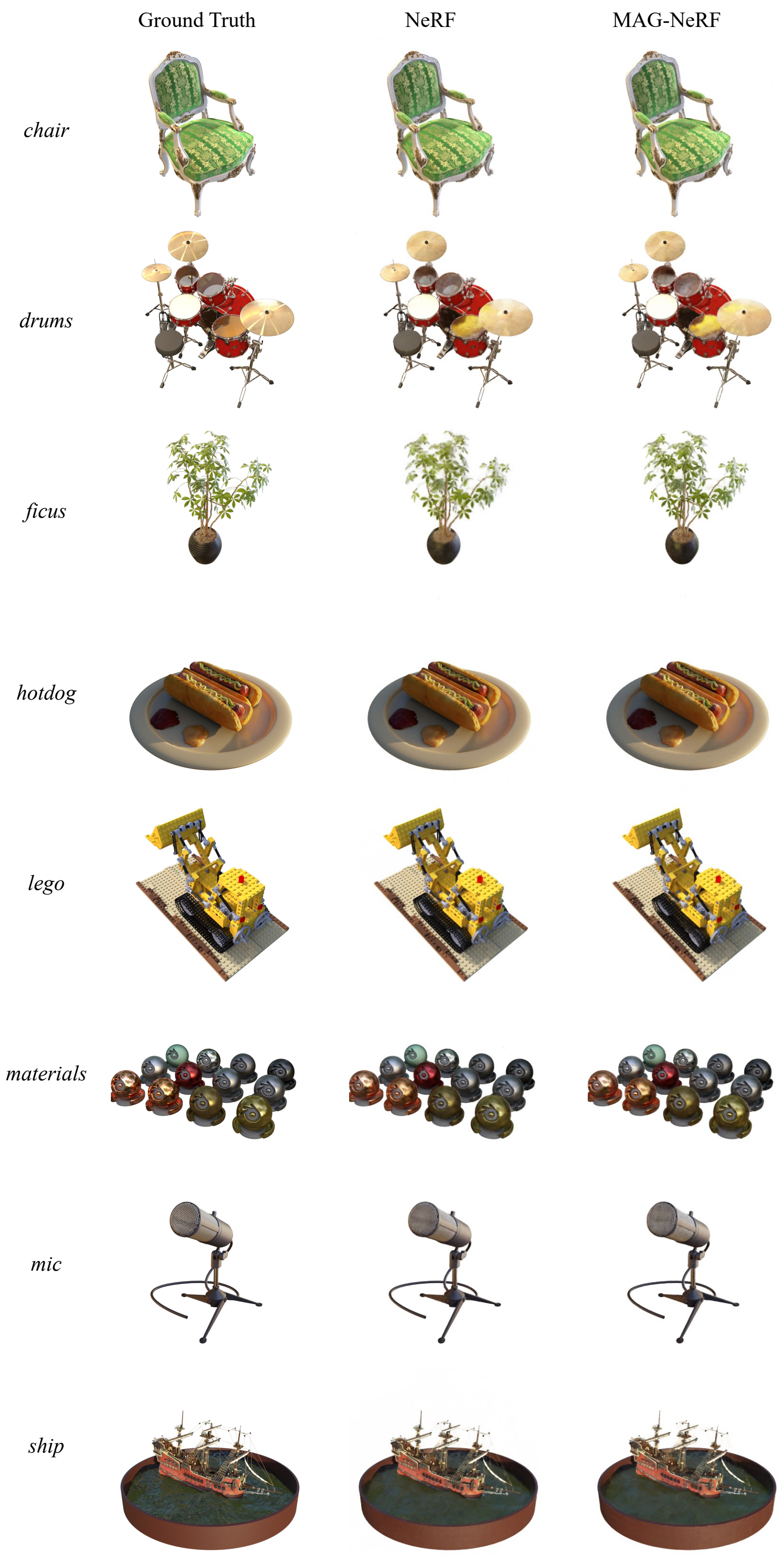


Figure 10: Rendered images from the Synthetic NeRF dataset (Mildenhall et al., 2020). For each scene, the images from left to right show the Ground Truth, NeRF (Mildenhall et al., 2020), and MAG-NeRF.

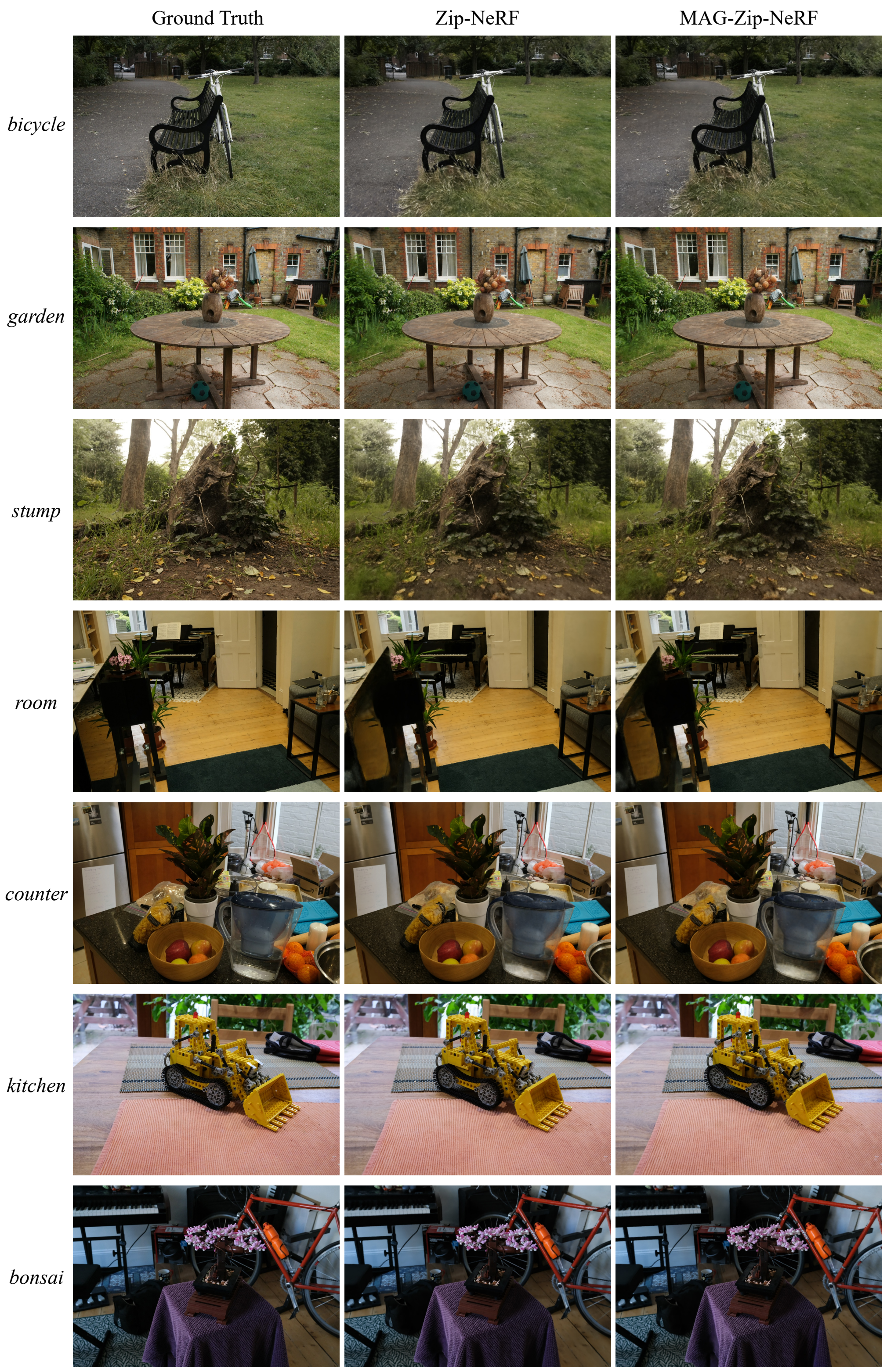


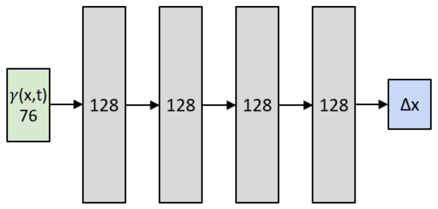
Figure 11: Rendered images from the Mip-NeRF 360 dataset (Barron et al., 2022). For each scene, the images from left to right show the Ground Truth, Zip-NeRF (Barron et al., 2023), and MAG-Zip-NeRF.

acceleration target, and we select the view-dependent part of MLP, which consists of two consecutive linear ReLU layers.

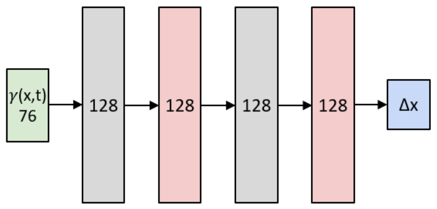
In all experiments, we use the default configuration of the PyTorch implementation of Zip-NeRF (Gu, 2023) on Mip-NeRF 360 dataset (Barron et al., 2022). The computational resources used for calculating rendering time are consistent with those in the NeRF experiments. However, due to hardware limitations, we reduce the ray batch size from 65,536 to 4,096, preventing out-of-memory issues. We set the number of random features to 64 for the MAG layer. All rendered images from Mip-NeRF 360 dataset can be shown in Figure 11.

For the distillation process, we need to get the input-output pairs from the pre-trained models. However, rendering a single image through the volume rendering process (Kajiya & Von Herzen, 1984) requires a large number of samples, which causes some memory issues if we use all of the samples for distillation. To address this, we reduce the sample size by selecting 1/16 of the images from the dataset and rendering them at 1/8 of the original training resolution. Additionally, we minimize the sample size by randomly subsampling 10% of the input-output pairs, ensuring the distillation process fits within our hardware limits and works efficiently.

C.4 D-NeRF Experiments



(a) D-NeRF architecture



(b) MAG-D-NeRF architecture

Figure 12: (a) Architecture of the deformation network in D-NeRF (Pumarola et al., 2021; Tang, 2022). (b) Our modified MAG-D-NeRF architecture. **Black** denotes Linear ReLU layer, **Red** is the MAG layer, and there is no nonlinear activation (ReLU) for Δx.

D-NeRF (Pumarola et al., 2021) extends NeRF to represent dynamic scenes using a deformation network. When rendering an image, the deformation network takes the position and time as inputs, producing a 3-dimensional offset that captures the current positional shift. This offset allows the ray to bend, simulating how the scene moves over time.

We use Pytorch implementation (Tang, 2022) for D-NeRF, which integrates the hash-grid (Müller et al., 2022) for further acceleration. Since it exploits the explicit grid structure, the network is shallower than the original D-NeRF. To accelerate this model, we target the deformation network, by replacing two of its layers with our MAG layers (See Figure 12 for more details).

For a fair apple-to-apple comparison between the baseline and our MAG variants, we minimize randomness by disabling certain acceleration methods in the implementation. Specifically, we use the `preload` configuration and remove other acceleration options to ensure consistency across experiments.

C.5 Mip-NeRF 360 Experiments

Mip-NeRF 360 Barron et al. (2022) is one of the state-of-the-art (SOTA) models for novel view synthesis. We utilize the PyTorch implementation from the NeRF-Factory (Jeong et al., 2022) codebase and reduce the *batch_size* from 4096 to 2048 to fit the available training memory. Inference times are measured using a single NVIDIA RTX 4090 GPU and an AMD Ryzen 7 7700 8-Core Processor. All experiments are conducted on the *360_v2* dataset Barron et al. (2022), with three layers replaced by our MAG layers. In our setup, the number of random features is fixed at 1024. We report the detailed results for each scene in the Tables below.

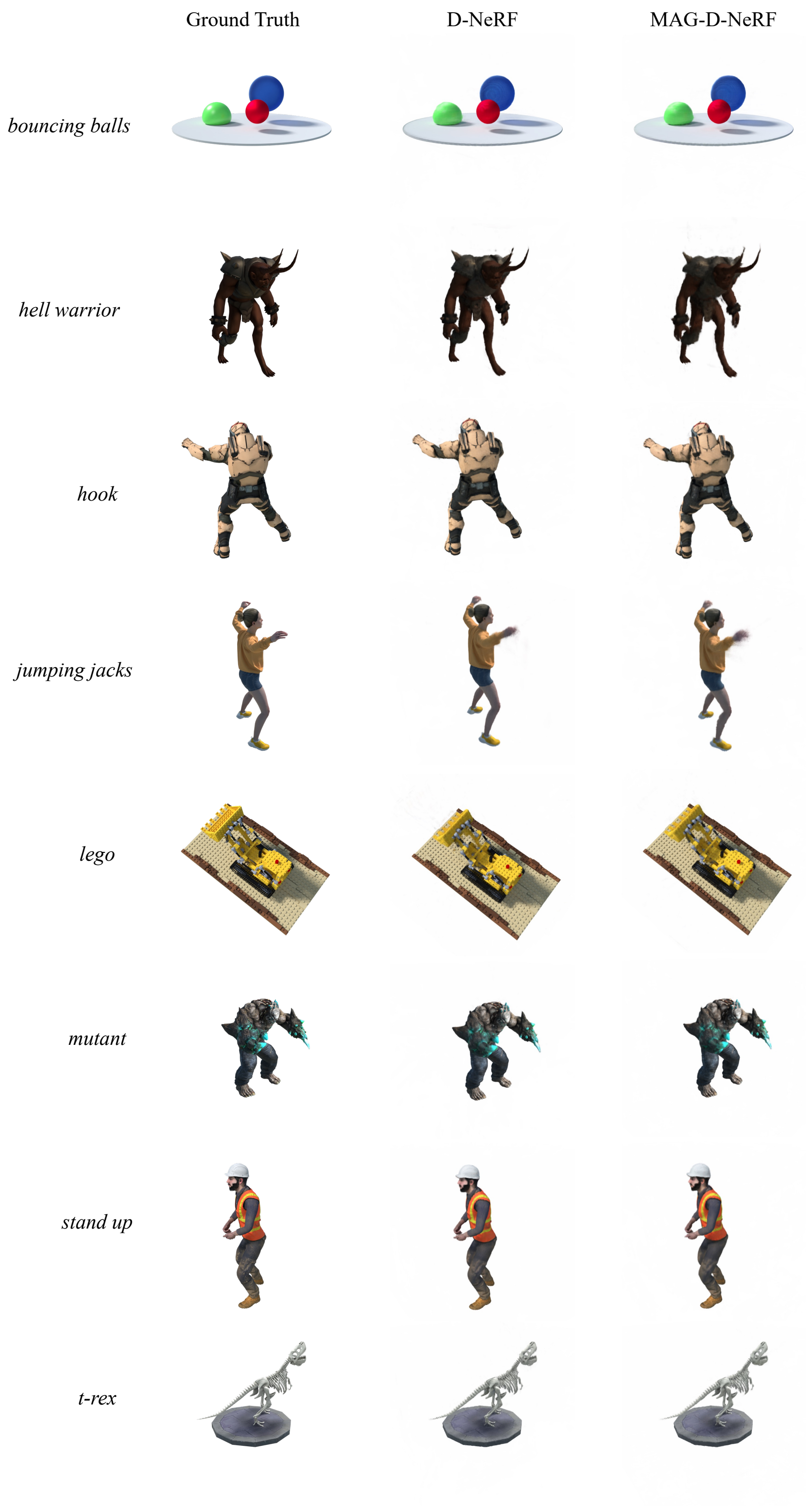


Figure 13: Rendered images from the D-NeRF dataset (Pumarola et al., 2021). For each scene, the images from left to right show the Ground Truth, D-NeRF (Pumarola et al., 2021; Tang, 2022), and MAG-D-NeRF.

Table 9: PSNR of rendered images for Mip-NeRF 360 in *360_v2* dataset. Higher is better.

Scene	bicycle	bonsai	counter	garden	kitchen	room	stump
Mip-NeRF 360	22.31	30.59	27.77	24.65	29.87	31.18	24.49
MAG-Mip-NeRF 360	22.1	29.99	27.29	24.30	29.34	30.6	24.04

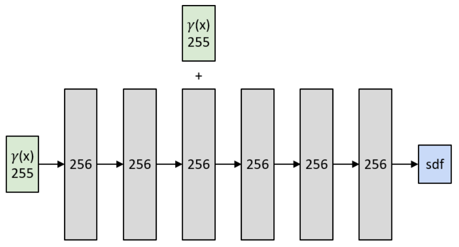
Table 10: SSIM of rendered images for Mip-NeRF 360 in *360_v2* dataset. Higher is better.

Scene	bicycle	bonsai	counter	garden	kitchen	room	stump
Mip-NeRF 360	0.466	0.897	0.809	0.626	0.879	0.885	0.569
MAG-Mip-NeRF 360	0.435	0.882	0.793	0.593	0.859	0.869	0.538

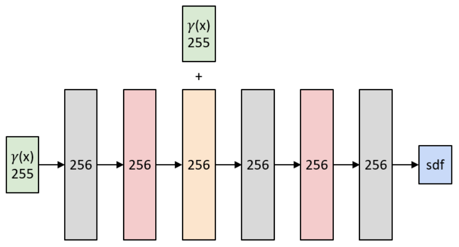
Table 11: LPIPS of rendered images for Mip-NeRF 360 in *360_v2* dataset. Lower is better.

Scene	bicycle	bonsai	counter	garden	kitchen	room	stump
Mip-NeRF 360	0.529	0.230	0.301	0.364	0.169	0.262	0.488
MAG-Mip-NeRF 360	0.557	0.263	0.328	0.389	0.196	0.295	0.516

C.6 iSDF Experiments



(a) Baseline iSDF architecture



(b) MAG-iSDF architecture

Figure 14: (a) Network architecture of the SDF prediction network in iSDF. (b) Our modified MAG-iSDF architecture. **Black** denotes Linear ReLU layer, **Orange** is Linear without activation, **Red** is the MAG layer, + is concatenation.

Figure 14 shows the baseline iSDF (Ortiz et al., 2022) and the modified MAG-iSDF architectures. We use six ReplicaCAD (Szot et al., 2021) scenes, following original iSDF (Ortiz et al., 2022) paper. In all experiments, we use the default configuration of public available iSDF code while replacing only the SDF network with our MAG-iSDF.

To evaluate efficiency, we report two time-related metrics: training time and inference time. However, it's important to note that, as iSDF is an incrementtal mapping module, the number of keyframes can vary between different experiment runs. Because the sample size is proportional to the number of keyframes, training time may not serve as a perfect measure of the network's efficiency. To account for this, we also report inference time, which is measured by sampling 1000 random rays and recording the time taken for a single forward pass through the network. Below we report detailed iSDF metrics and the qualitative comparison for different number of random features : 32, 64 and 128.

C.7 Distillation Results

We show the complete distillation results for NeRF in the table below.

Table 12: Quantitative results of MAG-iSDF. RF is the number of random features, $L1_{\text{avg}}$ is calculated across all SDF levels, and $L1_{\text{surf}}$ is for ground truth surface points. t_{train} refers to iteration time (including backward pass), and t_{infer} measures forward pass inference time. Results are averaged across 6 ReplicaCAD (Szot et al., 2021) sequences with 3 random seeds. Our MAG-iSDF model achieves less than similar quality as the baseline while being 23% faster.

		Distance			Time	
Method	RF **G**	$L1_{\text{avg}}$ (↓) [cm]	$L1_{\text{surf}}$ (↓) [cm]	CD (↓) [cm]	t_{train} (↓) [ms]	t_{infer} (↓) [ms]
iSDF	NA	8.99	4.08	2.02	8.24	1.37
MAG-iSDF (ours)	128	9.62	4.28	2.12	7.74	1.06
	64	9.39	4.46	2.21	7.40	0.93
	32	9.96	4.38	2.40	7.13	0.84

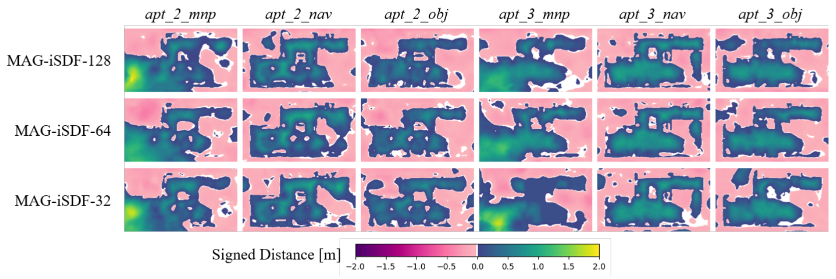


Figure 15: Visualization of reconstructed SDFs in iSDF for 6 ReplicaCAD dataset with various number of random features. Each figure shows the reconstructed SDF, following the colormap at the bottom.

Table 13: Distillation results for NeRF in *lego* dataset. Trainable **G** produces a leaner efficient model while being almost **42**% faster. Both analytic and trainable uses 128 random features except the last row which uses 64 random features.

Method	**PSNR**	**SSIM**	**LPIPS**	**Inference Speed (s)**
Baseline	31.31	0.965	0.017	2.36
Analytic	28.54	0.948	0.027	1.79
Trainable **G**	31.25	0.965	0.017	1.79
Trainable **G** (64)	31.09	0.964	0.017	1.38

D ADDITIONAL EXPERIMENTS

In this section, we present additional experiments on (a) justifying the use of ReLU over Softplus (Section D.1), (b) ablation studies (Section D.2, D.3), and (c) justify the choice of using ORF in our MAG layers.

D.1 RELU OVER SOFTPLUS IN ISDF

As our goal is to explore how far we can accelerate using MAG layers, we first investigate potential speed improvements in the baseline model before making modifications. We found that replacing Softplus with ReLU results in faster inference with minimal quality degradation. Specifically, this change causes a slight performance decrease for distances $s \geq 100\,\text{cm}$ (see Figure 16). However, since surface information is critical for robotics tasks such as manipulation or navigation, we exploit ReLU activation in our iSDF experiments both in baseline (iSDF) and MAG-iSDF to prioritize network speed.

D.2 EFFECT OF NUMBER OF RANDOM FEATURES IN MAG-NERF

In this experiment, we investigate how the number of random features affects the quality of MAG-NeRF. We evaluate three scenes: *hotdog*, *materials*, and *drums*, which represent easy, medium, and

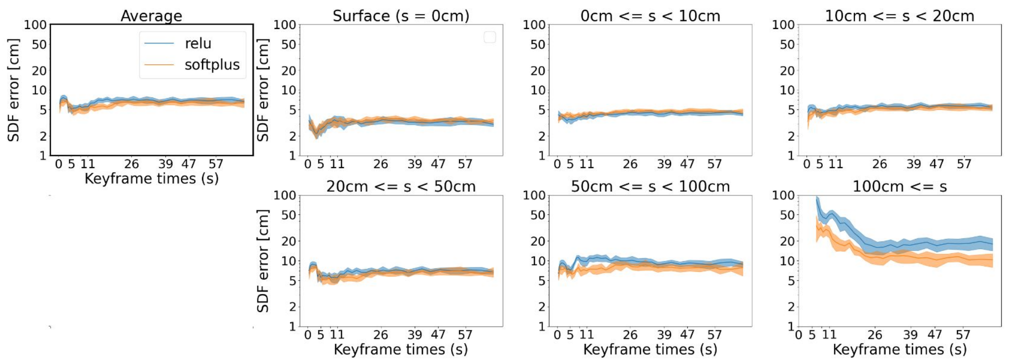


Figure 16: Error visualization comparing ground truth and reconstructed SDF using various activation functions. The first column displays the average error. The subsequent rows illustrate errors across different ranges of ground truth SDF values. In robotics scenarios such as grasping or manipulation, achieving more precise reconstruction, particularly for lower SDF values as shown in the first row, is crucial.

Table 14: Comparison between NeRF and our random feature mechanism on the *drums* dataset. The inference time refers to the approximate time for forward passes of each model to render a single 400x400 image.

		Rendering Quality			Efficiency	
Method	RF	PSNR (↑)	SSIM (↑)	LPIPS (↓)	t_{infer} (↓) [s]	Model Size (↓) [MB]
NeRF	-	25.46	0.930	0.049	2.36	3.27
MAG-NeRF	256	24.92	0.922	0.056	1.79	2.27
	128	24.64	0.918	0.059	1.47	1.39
	64	24.39	0.913	0.064	1.32	0.96
	32	23.90	0.903	0.076	1.22	0.74
	16	23.17	0.886	0.098	1.19	0.63

Table 15: PSNR of rendered images and inference time for pretrained Zip-NeRF and MAG layer distillation using various number of random features on the *bicycle* dataset (Barron et al., 2022).

	Pretrained	32	64	128	256	512
PSNR (↑)	23.66	20.72	21.06	21.55	21.72	22.06
Inference time (↓) (s)	4.03	3.76	3.78	3.86	4.06	4.43

hard cases, respectively, based on rendering quality in Table 1. We test models with 16, 32, 64, 128, and 256 random features.

We present the qualitative and quantitative results of the number of random features in MAG-NeRF in Figure 17 and Table 14. Using a small number of random features, like 16, results in approximately a twofold acceleration compared to the baseline model.

D.3 Effect of Number of Random Features in Zip-NeRF Distillation

When distilling a linear ReLU layer into our MAG layer, the number of random features must to be set. This choice impacts the quality and acceleration trade-off. In Figure 18 and Table 15, we present an ablation study to illustrate how the number of random features affects the distillation process.

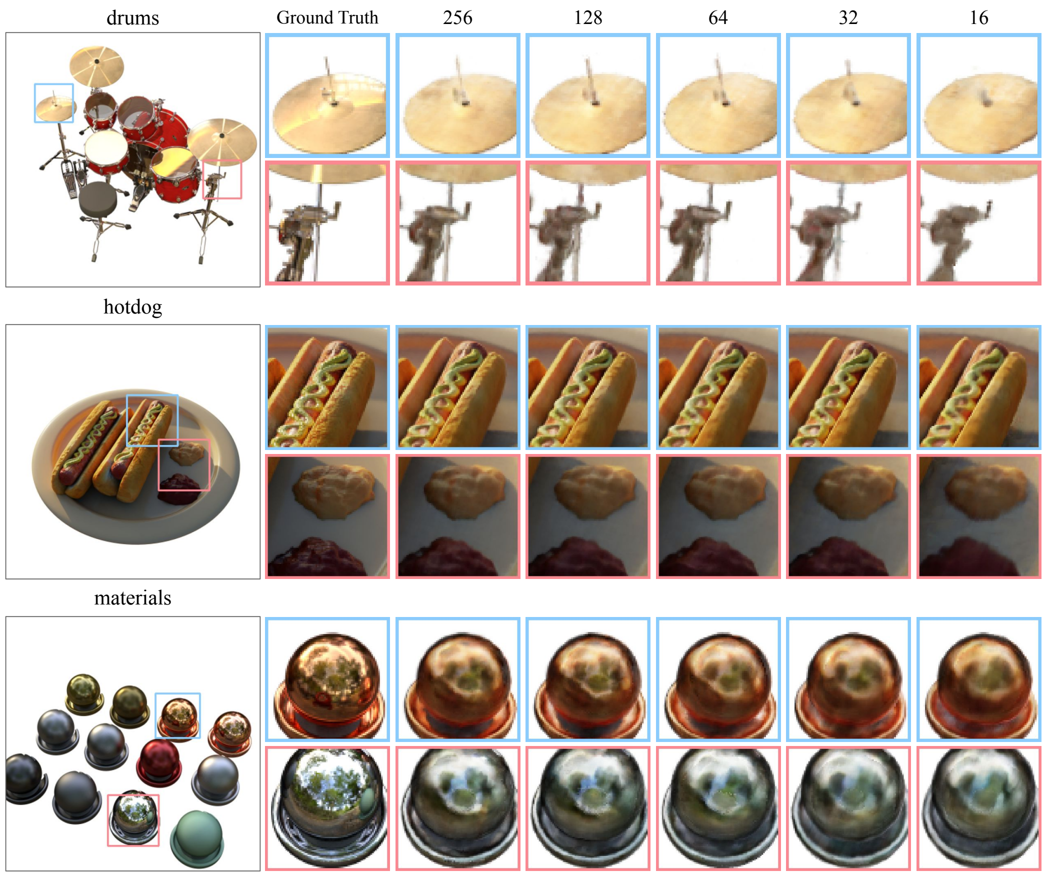


Figure 17: Comparison of rendering results with varying numbers of random features in MAG-NeRF. The numbers at the top indicate the number of random features used for each specific MAG-NeRF model. The ground truth image is on the left, while the right shows zoomed-in views of each reconstruction.

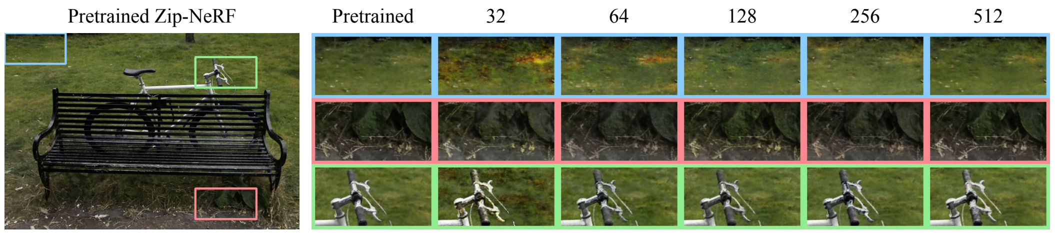


Figure 18: Comparison of rendering results with varying numbers of random features in Zip-NeRF distillation. The numbers at the top indicate the number of random features. Rendering from the target pretrained model is on the left, with zoomed-in views of each reconstruction on the right.

D.4 Orthogonal Random Features in MAG

In this subsection, we show that ORFs in MAG produce better approximation quality than random projections.

D.5 Distillation Experiments with Trainable Projection Matrix

Recent works have shown that training the projection matrix $\mathbf{G}$ instead of using a fixed probability distribution can lead to improved results (Chowdhury et al., 2022; Zhang et al., 2024). We validate

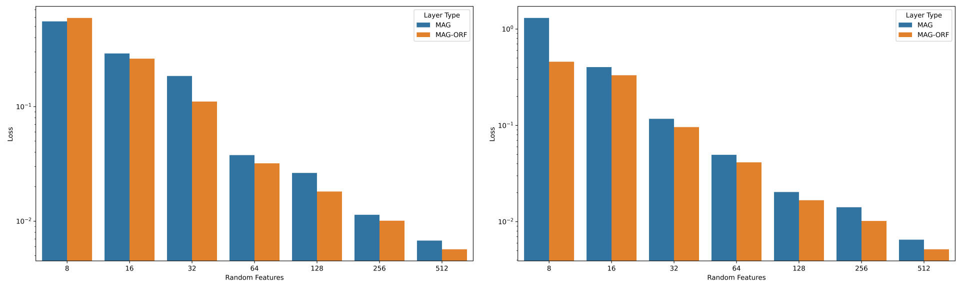


Figure 19: Comparing the approximation capability of our MAG layer with orthogonal random features vs random features. (Left) : Approximating a ReLU-Linear layer. (Right) : Approximating a Softplus-Linear layer.

this hypothesis in the context of our distillation experiments. However, optimizing the MAG layer to mimic the outputs of a FFL in a pretrained INR can not be done analytically in this case. So we use gradient descent to train this MAG layer. Table 16 shows the results for various loss functions that one can use to compare the output of MAG layer with the target FFL. We observe that the Huber loss performs better than MSE as it is robust to outliers. Adding a trainable bias improves performance, but does not add any extra latency during inference. That is why the trainable **G** with bias and Huber loss is the preferred choice for the distillation experiments.

Table 16: Performance comparison of methods based on PSNR, SSIM, and LPIPS.

Method	**Loss Function**	**PSNR**	**SSIM**	**LPIPS**
Baseline	NA	31.31	0.965	0.017
Analytic (no backprop)	MSE	28.54	0.948	0.027
Trainable **G**	MSE	31.07	0.964	0.017
Trainable **G**	Huber loss	31.11	0.964	0.017
Trainable **G** + bias	MSE	31.21	0.965	0.017
Trainable **G** + bias	Huber loss	31.25	0.965	0.017

E IMAGE CLASSIFICATION EXPERIMENTS

In this section, we showcase the usefulness of our MAG layers in image classification tasks. More specifically, we consider the problem of uptraining ViT (Dosovitskiy et al., 2021) on various downstream image classification datasets. In this setting, we replace a part of the Feed-Forward Network (FFN) block in Transformers with the MAG layer, namely the expansion layer with the GeLU activation. Note that : the MAG layer can then be combined with the following linear layer to create a single linear layer of size $[r, 768]$, where r is the number of random features. This allows for a significant compression of the ViT model. In all our experiments $r = 16$. We present detailed analysis of flops in Table 17 and report the accuracy in Fig. 20 as we successively replace MLP layers starting from the top layer. To summarize, by replacing top-6 layer's MLP blocks with MAG reduces the size of the ViT model from 346 Mb to 196.85 Mb. This speeds up inference by almost 35% and has minimal impact on accuracy.

F A PYTORCH STYLE PSEUDO-CODE

For convenience of the readers, we present a PyTorch style pseudo-code for our MAG layer.

Table 17: Detailed analysis of parameters and flops of MAG-ViT. The number k means MAG layers are inserted up to the kth layer starting from the top. The flops are computed for one 224×224 image.

	Full	12	11	10	9	8	7
# Inference parameters (millions)	86	82	77	72	67	62	57
# Inference Flops (billions)	35	33	31	29	27	25	23

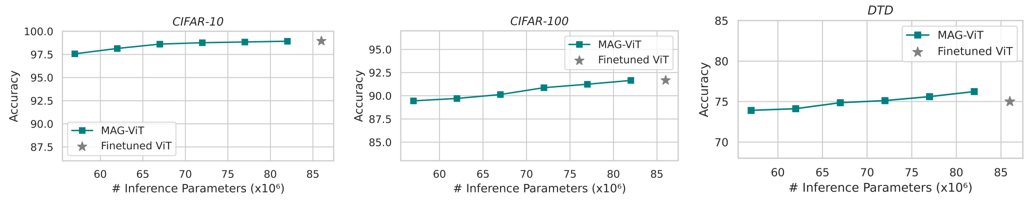


Figure 20: Accuracy vs Inference parameters trade-off plot for uptraining ViT by successively replacing MLP layers by the MAG layers from the bottom. (Left) : Accuracy vs Inference Parameters for CiFAR-10. (Middle) : Accuracy vs Inference Parameters for CiFAR-100. (Right) : Accuracy vs Inference Parameters for DTD.

Algorithm 1: PyTorch-style pseudo-code for MAG layer

```
class MAG(nn.Module):
  def __init__(self, num_rf, in_dim, out_dim):
    self.projection_matrix = nn.Parameter(
        torch.randn(self.in_dim, self.num_rf),
        requires_grad=False)
    self.weight = nn.Parameter(torch.randn(out_dim, num_rf))

  def forward(self, inputs):
    # Ignoring some normalization constant
    inputs = nn.ReLU(inputs @ self.projection_matrix) # shape =
     [*, num_rf]
    outputs = inputs @ self.weight.t() # shape = [*, out_dim]
    return outputs
```