HyP-NeRF: Learning Improved NeRF Priors using a HyperNetwork

Abstract

Neural Radiance Fields (NeRF) have become an increasingly popular representation to capture high-quality appearance and shape of scenes and objects. However, learning generalizable NeRF priors over categories of scenes or objects has been challenging due to the high dimensionality of network weight space. To address the limitations of existing work on generalization, multi-view consistency and to improve quality, we propose HyP-NeRF, a latent conditioning method for learning generalizable category-level NeRF priors using hypernetworks. Rather than using hypernetworks to estimate only the weights of a NeRF, we estimate both the weights and the multi-resolution hash encodings [35] resulting in significant quality gains.

To improve quality even further, we incorporate a denoise and finetune strategy that denoises images rendered from NeRFs estimated by the hypernetwork and finetunes it while retaining multiview consistency. These improvements enable us to use HyP-NeRF as a generalizable prior for multiple downstream tasks including NeRF reconstruction from single-view or cluttered scenes, and text-to-NeRF. We provide

*Equal authors (order decided by a coin flip)
†@research.iiit.ac.in

 qualitative comparisons and evaluate HyP-NeRF on three tasks: generalization, compression, and retrieval, demonstrating our state-of-the-art results.  

1 Introduction

Neural fields, also known as implicit neural representations (INRs), are neural networks that learn a continuous representation of physical quantities such as shape or radiance at any given space-time coordinate [70]. Recent developments in neural fields have enabled significant advances in applications such as 3D shape generation [77], novel view synthesis [32, 2], 3D reconstruction [72, 65, 38, 68], and robotics [52, 51]. In particular, we are interested in Neural Radiance Fields (NeRF) that learn the parameters of a neural network \( f(\sigma, c, x, \theta) \), where \( x \) and \( \theta \) are the location and viewing direction of a 3D point, respectively, and \( \sigma \) and \( c \) denote the density and color estimated by \( f \) at that point. Once fully trained, \( f \) can be used to render novel views of the 3D scene.

Despite their ability to model high-quality appearance, NeRFs cannot easily generalize to scenes or objects not seen during training thus limiting their broader application. Typically, achieving generalization involves learning a prior over a data source such as image, video, or point cloud distributions [21, 20, 58, 76, 29, 49], possibly belonging to a category of objects [65, 46]. However, NeRFs are continuous volumetric functions parameterized by tens of millions of parameters making it challenging to learn generalizable priors. Previous works try to address this challenge by relying on 2D image-based priors, 3D priors in voxelized space, or by using latent conditioning. Image-based priors re-use the information learned by 2D convolutional networks [75, 34] but may lack 3D knowledge resulting in representations that are not always multiview consistent. Methods that learn 3D priors in voxelized space [33] suffer from high compute costs and inherently lower quality due to voxelization limitations. Latent conditioning methods [19, 43] learn a joint network \( f(x, \theta, z) \) where \( z \) is the conditioning vector for a given object instance. These methods retain the advantages of native NeRF representations such as instance-level 3D and multiview consistency, but have limited capacity to model a diverse set of objects at high visual and geometric quality. InstantNGP [35] provides a way to improve quality and speed using instance-specific multi-resolution hash encodings (MRHE), however, this is limited to single instances.

We propose HyP-NeRF, a latent conditioning method for learning improved quality generalizable category-level NeRF priors using hypernetworks [15] (see Figure 1). We take inspiration from methods that use meta-learning to learn generalizable representations [55, 48], while retaining the quality of instance-specific methods [35]. Our hypernetwork is trained to generate the parameters—both the multi-resolution hash encodings (MRHE) and weights—of a NeRF model of a given category conditioned on an instance code \( z_n \). For each instance code \( z_n \) in the learned codebook, HyP-NeRF estimates \( h_n \) denoting the instance-specific MRHE along with \( \phi_n \) indicating the weights of an MLP. Our key insight is that estimating both the MRHEs and the weights results in a significant improvement in quality. To improve the quality even further, we denoise rendered views [42] from the estimated NeRF model, and finetune the NeRF with the denoised images to enforce multiview consistency. As shown in Figure 2 and the experiments section, this denoising and finetuning step significantly improves quality and fine details while retaining the original shape and appearance properties.

Once HyP-NeRF is trained, it can be used as a NeRF prior in a variety of different applications such as NeRF reconstruction from a single view posed or unposed images, single pass text-to-NeRF,
or even the ability to reconstruct real-world objects in cluttered scene images (see Figure 2). We show qualitative results on applications and quantitatively evaluate HyP-NeRF’s performance and suitability as a NeRF prior on the ABO dataset [11] across three tasks: generalization, compression, and retrieval. To sum up our contributions:

1. We introduce HyP-NeRF, a method for learning improved quality NeRF priors using a hypernetwork that estimates instance-specific hash encodings and MLP weights of a NeRF.
2. We propose a denoise and finetune strategy to further improve the quality while preserving the multiview consistency of the generated NeRF.
3. We demonstrate how our NeRF priors can be used in multiple downstream tasks including single-view NeRF reconstruction, text-to-NeRF, and reconstruction from cluttered scenes.

2 Related Work

Neural Radiance Fields [32] (NeRFs) are neural networks that capture a specific 3D scene or object given sufficient views from known poses. Numerous follow-up work (see [62] [70] for a more comprehensive review) has investigated improving quality and speed, relaxing assumptions, and building generalizable priors. Strategies for improving quality or speed include better sampling [2], supporting unbounded scenes [4], extensions to larger scenes [69] [60], using hybrid representations [55] [74], using learned initializations [61] [65], or discarding neural networks completely [73] [59]. Other work relaxes assumption of known poses [67] [31] [27] [25] [9] [53], or reduce the number of views [75] [43] [41] [34] [44] [71] [28] [57]. Specifically, PixelNeRF [75] uses convolution-based image features to learn priors enabling NeRF reconstruction from as few as a single image. VisionNeRF [28] extends PixelNeRF by augmenting the 2D priors with 3D representations learned using a transformer. Unlike these methods, we depend purely on priors learned by meta-learning, specifically by hypernetworks [15]. AutoRF [34] and LolNeRF [43] are related works that assume only a single view for each instance at the training time. FWD [5] optimizes NeRFs from sparse views in real-time and SRT [45] aims to generate NeRFs in a single forward pass. These methods produce NeRFS of lower quality and are not designed to be used as priors for various downstream tasks. In contrast, our focus is to generate high-quality multiview consistent NeRFs that capture fine shapes and textures details. HyP-NeRF can be used as a category-level prior for multiple downstream tasks including NeRF reconstruction from one or more posed or unposed images, text-to-NeRF (similar to [40] [18]), or reconstruction from cluttered scene images. Additionally, HyP-NeRF can estimate the NeRFs in a single forward pass with only a few iterations needed to improve the quality. Concurrent to our work, NerfDiff [13] and SSDNeRF [8] achieve high quality novel view synthesis by using diffusion models.

Learning 3D Priors. To learn category-level priors, methods like CodeNeRF [19] and LolNeRF [43] use a conditional NeRF on instance vectors $z$ given as $f(x, \theta, z)$, where different $z$s result in different NeRFs. PixelNeRF [75] depends on 2D priors learned by 2D convolutional networks which could result in multi-view inconsistency. DiffRF [33] uses diffusion to learn a prior over voxelized radiance field. Like us, DiffRF can generate radiance fields from queries like text or images. However, it cannot be directly used for downstream tasks easily.

Our approach closely follows the line of work that aims to learn a prior over a 3D data distribution like signed distance fields [39], light field [55], and videos [48]. We use meta-learning, specifically hypernetworks [15], to learn a prior over the MRHEs and MLP weights of a fixed NeRF architecture. LearnedInit [61] also employs standard meta-learning algorithms for getting a good initialization of the NeRF parameters. However, unlike us, they do not use a hypernetwork, and use the meta-learning algorithms only for initializing a NeRF, which is further finetuned on the multiview images. Methods like GRAF [47], $\pi$-GAN [6], CIPS-3D [29], EG3D [7], and Pix2NeRF [4] use adversarial training setups with 2D discriminators resulting in 3D and multiview inconsistency. [40] [63] [16] tightly couple text and NeRF priors to generate and edit NeRFs based on text inputs. We, on the other hand, train a 3D prior on NeRFs and separately train a mapping network that maps text to HyP-NeRF’s prior, decoupling the two.

3 HyP-NeRF: Learning Improved NeRF prior using a Hypernetwork

Our goal is to learn a generalizable NeRF prior for a category of objects while maintaining visual and geometric quality, and multiview consistency. We also want to demonstrate how this prior can be
used to enable downstream applications in single/few-image NeRF generation, text-to-NeRF, and reconstruction of real-world objects in cluttered scenes.

**Background.** We first provide a brief summary of hypernetworks and multi-resolution hash encodings that form the basis of HyP-NeRF. Hypernetworks are neural networks that were introduced as a meta-network to predict the weights for a second neural network. They have been widely used for diverse tasks, starting from representation learning for continuous signals \cite{65, 64, 57, 58}, compression \cite{36, 12}, few-shot learning \cite{50, 23}, continual learning \cite{63}. Our key insight is to use hypernetworks to generate both the network weights and instance-specific MRHEs.

**Neural Radiance Fields** (NeRF) \cite{32, 2} learn the parameters of a neural network \( f_n(x, \theta) = \{\sigma, c\} \), where \( x \) and \( \theta \) are the location and viewing direction of a 3D point, respectively, and \( \sigma \) and \( c \) denote the density and color predicted by \( f_n \) at that point. Once fully trained, \( f_n \) can be used to render novel views of the 3D scene. NeRF introduced *positionally encodings* of the input 3D coordinates, \( x \), to a higher dimensional space to capture high-frequency variations in color and geometry. InstantNGP \cite{55} further extended this idea to *instance-specific* multi-resolution hash encodings (MRHE) to encode \( x \) dynamically based on scene properties. These MRHEs, \( h_i \), are learned along with the MLP parameters, \( \phi \) for a given NeRF function, \( f \) and show improved quality and reduced training/inference time.

**Image Denoising** is the process of reducing the noise and improving the perceptual quality of images while preserving important structural details. Recent advancements in deep learning-based image restoration and denoising techniques \cite{26, 24, 10} have demonstrated remarkable success in removing noise and enhancing the perceptual quality of noisy images that may have suffered degradation. Such networks are trained on large datasets of paired noisy and clean images to learn a mapping between the degraded input and the corresponding high-quality output by minimizing the difference between the restored and the ground truth clean image. In our case, we use denoising to improve the quality of our NeRF renderings by reducing artifacts and improving the texture and structure at the image level.

### 3.1 Method

Given a set of NeRFs denoted by \( \{f_n(\phi_n, h_n)\}_{n=1}^N \), where \( N \) denotes the number of object instances in a given object category, we want to learn a prior \( \Phi = \{\Phi_S, \Phi_C\} \), where \( \Phi_S \) and \( \Phi_C \) are the shape and color priors, respectively. Each NeRF, \( f_n(\cdot) \), is parameterized by the neural network weights, \( \phi_n \), and learnable MRHEs, \( h_n \), as proposed in \cite{55}. \( f_n(\cdot) \) takes a 3D position, \( x \), and viewing direction, \( \theta \), as input and predicts the density conditioned on \( x \) denoted by \( \sigma_n^{(x)} \), and color conditioned on \( x \) and \( \theta \) denoted by \( c_n^{(x, \theta)} \). This is given as,

\[
\forall n \in N, \quad f_n(\phi_n, h_n)(x, \theta) = \{\sigma_n^{(x)}, c_n^{(x, \theta)}\}.
\]  

(1)

Our proposed method for learning NeRF priors involves two steps. First, we train a hypernetwork, \( M \), to learn a prior over a set of multiview consistent NeRFs of high-quality shape and texture. Second, we employ an image-based denoising network that takes as input an already multiview consistent set of images, rendered from the predicted NeRF, and improves the shape and texture of NeRF to higher quality by finetuning on a set of denoised images. Our architecture is outlined in Figure 3 and we explain each step in detail below.

**Step 1: Hypernetwork for Learning NeRF Prior.** We want to design our hypernetwork, \( M \), with trainable parameters, \( \Omega \) that can predict NeRF parameters \( \{\phi_n, h_n\} \) given a conditioning code \( z_n = \{S_n, C_n\} \), where \( S_n \) and \( C_n \) are the shape and color codes, respectively, for an object instance \( n \) belonging to a specific category. Here, \( S_n \) and \( C_n \) belong to codebooks, \( S \) and \( C \) that are trained along with \( \Omega \) in an auto-decoding fashion.

As shown in Figure 3, ideally we want \( M \) to learn a prior \( \{\Phi_S, \Phi_C\} \) over \( S \) and \( C \) such that given a random set of codes, \( \{Y_S \sim \Phi_S, Y_C \sim \Phi_C\} \), \( M \) should be able to generate a valid NeRF with consistent shape and texture for the given category of objects. To achieve this, we train \( M \) by assuming the same constraints as are needed to train a NeRF - a set of multiview consistent images \( I = \{I_{\theta \in \Theta}\}_{n=1}^N \) for a set of poses, \( \Theta \). In each training step, we start with a random object instance, \( n \), and use the corresponding codes \( S_n \) and \( C_n \) from the codebooks as an input for \( M \). Our key insight is that estimating both the MRHEs and MLP weights results in a higher quality than other alternatives. \( M \) then predicts the NeRF parameters \( \{\phi_n, h_n\} \), which is then used to minimize the
we train our hypernetwork end-to-end directly on images, so that it learns the implicit NeRF space

\[ \Phi = \{ \Phi_S, \Phi_C \} \]

to object instance \( n \). At this stage, the NeRF model acts as a set of differentiable layers to compute

the volumetric rendering loss as described in \([32]\). We use an instantiation of InstantNGP \([35]\) as our

following objective:

\[ \mathcal{L}(\Omega, S_n, C_n) = \sum_{r \in R} ||V'(r, \{ \sigma_n^{(x^r)}, c_n^{(x^r, \theta)} \}_i^{L_i=1}) - V_n(r)|| \]  

where \( V' \) denotes the volumetric rendering function as given in \([32] \) eqn. 3 and 5, \( r \) is a ray projected

along the camera pose \( \theta \), \( x^r \in x \) and \( L \) denote the number of points sampled along \( r \), and \( V_n \) denote

the ground truth value for projection of the \( n \)th object along \( r \).

Note that, in this step, the only trainable parameters are the meta-network weights, \( \Omega \), and the
codebooks \( S \) and \( C \). In this setting, the NeRF functions \( f^{(\cdot)} \) only act as differentiable layers that
allow backpropogation through to \( M \) enabling it to train with multiview consistency loss attained by

the volumetric rendering loss as described in \([32] \). We use an instantiation of InstantNGP \([35]\) as our

function \( f^{(\cdot)} \) consisting of MRHE and a small MLP.

A general limitation of hypernetworks arises from the fact that the intended output space (i.e. the
space of valid MLP weight matrices) is a subset of the actual output space, which is unrestricted and

can be any 2D matrix. Thus, a hypernetwork trained on loss functions in the weight space can result

in unstable training, and might require a lot of training examples to converge. To overcome this issue,
we train our hypernetwork end-to-end directly on images, so that it learns the implicit NeRF space

along with the category specific prior on it, which simplifies the setting for the hypernetwork and
allows for more stable training. As a causal effect of this, HyP-NeRF, when trained on less number

of examples, essentially acts as a compressing model.

Step 2: Denoise and Finetune. In the first step, \( M \) is trained to produce a consistent NeRF with
high-fidelity texture and shape. However, we observed that there is room to improve the generated
We provide evaluations of the prior learned by HyP-NeRF specifically focusing on the quality of the generated NeRFs. We consider three dimensions: (1) Generalization (Section 4.1): we validate whether HyP-NeRF can generate novel NeRFs not seen during training by conditioning on only a few training views. (2) Test Time Optimization: we augment (computationally less expensive) happens only once per scene. Only the NeRF predicted by the hypernetwork \( \Phi \) minimizes the following objective:

\[
\min \| \{ f_{\hat{I}_i} \}_{i=1}^m \| \text{subject to } \langle f_{\hat{I}_i} \rangle = \langle f_{\hat{I}_i} \rangle_{\text{known}}
\]

As shown in Figure 3 (bottom), we render novel views from the multiview consistent NeRF into different predefined poses given by \( \{ \theta_1, \theta_2, \ldots, \theta_m \} \) to produce a set of multiview consistent images \( \{ \hat{I}_i \}_{i=1}^m \). We then use a pre-trained image-level denoising autoencoder that takes \( \{ \hat{I}_i \}_{i=1}^m \) as input and produces images of improved quality given as \( \{ \hat{I}_i \}^{\text{im}} \). These improved images are then used to fine-tune \( f_{\hat{I}_i} \) to achieve \( f_{\hat{I}_i}^{\text{im}} \). Note that, we do not train the NeRFs from scratch on \( \{ \hat{I}_i \}^{\text{im}} \) and only finetune the NeRFs, which ensures fast optimization and simplifies the task of the denoising module that only needs to improve the quality and does not necessarily need to maintain the multiview consistency. While our denoising is image-level, we still obtain multiview consistent NeRFs since we finetune on the NeRF itself (as we also demonstrate through experiments in the supplementary).

For our denoising autoencoder, we use VQVAE2 [42] as the backbone. To train this network, we simply use images projected from the NeRF, predicted by the hypernetwork (lower quality relative to the ground truth) as the input to the VQVAE2 model. We then train VQVAE2 to decode the ground truth by minimizing the L2 loss objective between VQVAE2’s output and the ground truth.

### 3.2 HyP-NeRF Inference and Applications

Training over many NeRF instances, \( M \) learns a prior \( \Phi \) that can be used to generate novel consistent NeRFs. However, \( \Phi \) is not a known distribution like Gaussian distributions that can be naively queried by sampling a random point from the underlying distribution. We tackle this in two ways:

**Test Time Optimization.** In this method, given a single-view or multi-view posed image(s), we aim to estimate shape and color codes \( \{ S_n, C_n \} \) of the NeRF that renders the view(s). To achieve this, we freeze \( M \)’s parameters and optimize the \( \{ S_n, C_n \} \) using the objective given in Equation (2).

**Query Network.** We create a query network, \( \Delta \), that maps a point from a known distribution to \( \Phi \). As CLIP’s [41] pretrained semantic space, say, \( C \), is both text and image aware, we chose \( C \), as our known distribution and learn a mapping function \( \Delta(z \sim C) \rightarrow \Phi \). Here, \( \Delta \) is an MLP that takes \( z \) as input and produces \( \mathcal{V}_z \in \Phi \) as output. To train \( \Delta \), we randomly sample one pose from the ground truth multiview images \( I_n^I \in \{ I_{\theta} \}_{\theta \in \Theta} \) and compute the semantic embedding \( z^I_n = \text{CLIP}(I_n^I) \) and map it to \( \{ S_n, C_n \} \in \Phi \) given as \( \{ S_n, C_n \} = \Delta(z^I_n) \). We then train our query network by minimizing the following objective:

\[
\mathcal{L}_\Delta = \sum_{\theta} ||\{ S_n, C_n \}, \{ S_n, C_n \}||.
\]

At the time of inference, given a text or image modality such as a text prompt, single-view unposed (in-the-wild) image, or segmented image, we compute the semantic embedding using CLIP encoder and map it to \( \Phi \) using \( \Delta \), from which we obtain the shape and color codes as input for the HyP-NeRF.

Note that for \( N \) query points in a scene, the forward pass through the hypernetwork (computationally expensive) happens only once per scene. Only the NeRF predicted by the hypernetwork (computationally less expensive) is run for each query point.

### 4 Experiments

We provide evaluations of the prior learned by HyP-NeRF specifically focusing on the quality of the generated NeRFs. We consider three dimensions: (1) Generalization (Section 4.1), we validate whether HyP-NeRF can generate novel NeRFs not seen during training by conditioning on only a few training views.
Figure 4: Qualitative Comparison of Generalization on ABO. The NeRFs are rendered at a resolution of $512 \times 512$. HyP-NeRF is able to preserve fine details such as the legs, creases, and texture even for novel instances. PixelNeRF fails to preserve details and to model the structure.

Table 2: Compression. We randomly sample 250 datapoints from our training dataset and compare the NeRFs learned using InstantNGP [35] on the individual instances against HyP-NeRF that learns the entire dataset. Note, we do not employ the denoising module (see Section 3.1) for this evaluation.

Datasets and Comparisons. We primarily compare against two baselines, PixelNeRF [75] and InstantNGP [35] on the Amazon-Berkeley Objects (ABO) [11] dataset. ABO contains diverse and detailed objects rendered at a resolution of $512 \times 512$ which is perfect to showcase the quality of the NeRF generated by HyP-NeRF. Rather than use a computationally expensive model like VisionNeRF (on the SRN [56] dataset) on a resolution of $128 \times 128$, we show our results on $512 \times 512$ and compare with PixelNeRF. Additionally, we compare with the other baselines on SRN at $128 \times 128$ resolution qualitatively in the main paper (Figure 5) and quantitatively in the supplementary. For compression, we directly compare with InstantNGP [35], that proposed MRHE, trained to fit on individual objects instance-by-instance.

Architectural Details. We use InstantNGP as $f_{\ell_n}$, with 16 levels, hashtable size of $2^{11}$, feature dimension of 2, and linear interpolation for computing the MRHE; the MLP has a total of 5, 64-dimensional, layers. We observed that a hashtable size $2^{11}$ produces NeRF of high-quality at par with...
the a size of $2^{14}$. Hence, we use $2^{13}$ to speed up our training. Our hypernetwork, $M$, consists of 6 MLPs, 1 for predicting the MRHE, and the rest predicts the parameters $\phi$ for each of the MLP layers of $f$. Each of the MLPs are made of 3, 512-dimensional, layers. We perform all of our experiments on NVIDIA RTX 2080Tis.

**Metrics.** To evaluate NeRF quality, we render them at 91 distinct views and compute metrics on the rendered images. Following PixelNeRF, we use PSNR($\uparrow$), SSIM($\uparrow$), and LPIPS($\downarrow$) [78]. Additionally, we compute Fréchet Inception Distance (FID)($\downarrow$) [17] to further test the visual quality. Although these metrics measure the quality of novel-view synthesis, they do not necessarily evaluate the geometry captured by the NeRFs. Therefore, we compute Chamfer’s Distance (CD) whenever necessary by extracting a mesh from NeRF densities [30]. Please see the supplementary for additional details.

### 4.1 Generalization

One way to evaluate if HyP-NeRF can render novel NeRF instances of high quality is through unconditional sampling. However, our learned prior $\Phi$ is a non-standard prior (like a Gaussian distribution) and thus random sampling needs carefully designed mapping between such a known prior and $\Phi$. Therefore, we instead rely on a conditional task of single-view novel NeRF generation: given a single arbitrarily-chosen view of a novel object, we generate the corresponding NeRF, $f_{(\cdot)}$, through test-time optimization (see Section 3.2). We compare quantitatively with PixelNeRF on ABO at a high resolution of $512 \times 512$ and qualitatively with the rest of the baselines on SRN at $128 \times 128$.

As shown in Table 1, we significantly outperform PixelNeRF on all of the metrics. Further, the qualitative results in Figure 5 clearly shows the difference between the rendering quality of HyP-NeRF against PixelNeRF. Specifically, PixelNeRF fails to learn details, especially for the Sofa category. On the other hand, HyP-NeRF preserves intricate details like the texture, legs, and folds in the objects even at a high resolution. Further, we show our results on the widely used SRN dataset at the resolution of $128 \times 128$ in Figure 5. Here, our quality is comparable with the baselines.

### 4.2 Compression

Unlike InstantNGP, which is trained on a single 3D instance, HyP-NeRF is trained on many NeRF instances which effectively results in the compression of these NeRFs into the latent space (or the codebook). We evaluate this compression capability by computing NeRF quality degradation compared to single-instance-only method, InstantNGP.

We randomly sample 250 instances from the training set and train InstantNGP separately on each of them. These samples are a subset of the training data used in HyP-NeRF’s codebook. We show degradation metrics in Table 2. Note that we do not perform denoising on the generated NeRFs as we want to only evaluate the compression component of HyP-NeRF in this section. As can be seen in Table 2 there is a significant degradation in terms of PSNR (an average of 11%), but the overall geometry is preserved almost as well as InstantNGP. However, InstantNGP is trained on a single instance, whereas we train on 1000s of NeRF instances (1038, 783, and 517 instances for ABO Chairs, Sofa, and Tables, respectively). This results in a $60 \times$ compression gain: for ABO Chairs,
with 1038 training instances, HyP-NeRF needs 163MB to store the model, whereas a single instance of InstantNGP needs on average 8.9MB. Note that we use the same network architecture for HyP-NeRF and InstantNGP making this a fair comparison. Moreover, the storage complexity for InstantNGP-based NeRFs is linear with respect to the number of instances, whereas our degradation in visual quality is sublinear.

4.3 Retrieval

A generalizable learned prior has the ability to generate NeRFs based on different input modalities like text, images, segmented and occluded images, random noise, and multiview images. We now demonstrate additional querying and retrieval capabilities as described in Section 3.2.

This experiment’s goal is to retrieve specific NeRF instances that HyP-NeRF has encountered during training from a single-view unposed image of that instance. Section 4.3 presents the number of times we could correctly retrieve from an arbitrary view of seen NeRF instances. We achieve almost 100% accuracy for Chair and Sofa datasets. However, we take this a step further and try to retrieve the closest training instance code corresponding to unseen views of seen instances taken from in-the-wild internet images. Figure 6 (top) shows examples from this experiment in which we are able to retrieve a NeRF closely matching the input query. This demonstrates the ease of designing a simple mapping network that can effectively interact with HyP-NeRF’s prior.

Along with retrieving a seen instance, we use the query network to generate novel NeRFs of unseen instances as shown in Figure 6 (middle and bottom). In the middle row, we take an image of a cluttered scene, segment it with SAM [22], and pass this as input to the query network, from which we obtain a set of latent codes given as input to HyP-NeRF (see Figure 2). Finally, in the bottom row, we show text-to-NeRF capabilities enabled by HyP-NeRF.

4.4 Ablation

Two key designs of HyP-NeRF include incorporating the MRHE and the denoising network. We present the affect of removing these two components in Table 4 and Figure 1 for MRHE and Table 1, Figure 2, and Figure 4 for denoising. In the first ablation, we change the design of our neural network by using a hypernetwork to predict the parameters of a standard nerf with positional encodings [32]. Since we remove the MRHE, we also increase the number of layers in the MLP to match the layers mentioned in [32]. Since there is a significant increase in the view rendering time, we randomly sample 70 training examples for evaluation. As seen in Table 4, the quality of the rendered views lags significantly in all the metrics including the CD (measured against NeRFs rendered on InstantNGP individually). This is showcased visually in Figure 1 and the supplementary. Similarly, we find significant differences between the quality of the NeRFs before and after denoising (Table 1, Figure 2, and Figure 4), particularly in the Chair category with more diverse shapes.
4.5 Color and Shape Disentanglement

We start with two object instances from the train set - A and B and denote their corresponding shape and geometry codes as $A_s$, $B_s$ and $A_g$, $B_g$. We switch the geometry and shape code and generate two novel NeRFs given by $A_s$, $B_g$ and $B_s$, $A_g$. In Figure 7, we can clearly see the disentanglement: geometry is perfectly preserved, and the color is transferred faithfully across the NeRFs. In Figure 8, we show clusters of color and shape codes using TSNE plots and visualize instances from the clusters.

5 Conclusion, Limitation, and Future Work

We propose HyP-NeRF, a learned prior for Neural Radiance Fields (NeRFs). HyP-NeRF uses a hypernetwork to predict instance-specific multi-resolution hash encodings (MRHEs) that significantly improve the visual quality of the predicted NeRFs. To further improve the visual quality, we propose a denoising and finetuning technique that result in an improved NeRF that preserves its original multiview and geometric consistency. Experimental results demonstrate HyP-NeRF’s capability to generalize to unseen samples and its effectiveness in compression. With its ability to overcome limitations of existing approaches, such as rendering at high resolution and multiview consistency, HyP-NeRF holds promise for various applications as we demonstrate for single- and multi-view NeRF reconstruction and text-to-NeRF.

Limitation and Future Work. One limitation of our work is the need for the pose to be known during test-time optimization (Section 3.2). Although we propose the query network to predict novel NeRFs conditioned on an unposed single view, the result may not exactly match the given view because of the loss of detail in the CLIP embedding. Future work should design a mapping network that can preserve fine details. An iterative pose refinement approach that predicts the pose along with the shape and color codes could also be adopted. A second limitation of our work is the non-standard prior $\Phi$ that was learned by HyP-NeRF which makes unconditional generation challenging. GAN-based generative approaches solve this problem by randomly sampling from a standard distribution (like Gaussian distribution) and adversarially training the network. However, those methods often focus more on image quality than 3D structure. Future work could address this by incorporating latent diffusion models that can map a standard prior to HyP-NeRF’s prior.
References


Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.


