HyperFields: Towards Zero-Shot Generation of NeRFs from Text

Sudarshan Babu [*1] **Richard Liu** [*2] **Avery Zhou** [*1 2] **Michael Maire** [2] **Greg Shakhnarovich** [1] **Rana Hanocka** [2]

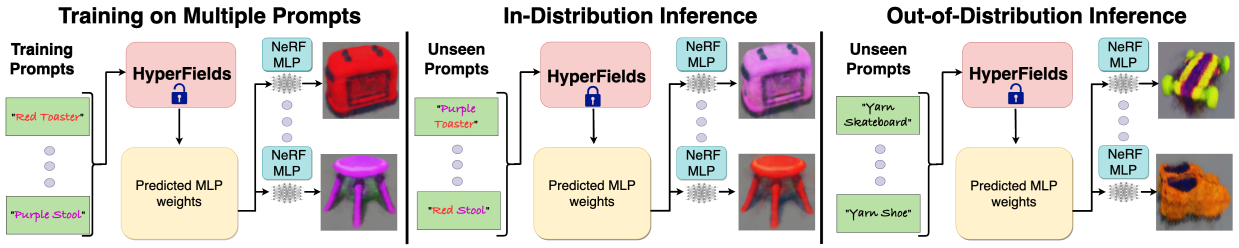

Figure 1. HyperFields is a hypernetwork that learns to map text to the space of weights of Neural Radiance Fields (first column). After training, HyperFields is capable of generating in-distribution scenes - *unseen during training* - in a feed forward manner (second column), and for out-of-distribution prompts HyperFields can be fine-tuned to yield scenes respecting prompt semantics with just a few gradient steps (third column).

.

Abstract

We introduce HyperFields, a method for generating text-conditioned Neural Radiance Fields (NeRFs) with a single forward pass and (optionally) some fine-tuning. Key to our approach are: (i) a dynamic hypernetwork, which learns a smooth mapping from text token embeddings to the space of NeRFs; (ii) NeRF distillation training, which distills scenes encoded in individual NeRFs into one dynamic hypernetwork. These techniques enable a single network to fit over a hundred unique scenes. We further demonstrate that HyperFields learns a more general map between text and NeRFs, and consequently is capable of predicting novel in-distribution and out-of-distribution scenes — either zero-shot or with a few finetuning steps. Finetuning HyperFields benefits from accelerated convergence thanks to the learned general map, and is capable of synthesizing novel scenes 5 to 10 times faster than existing neural optimization-based methods. Our ablation experiments show that both the dynamic architecture and NeRF distillation are critical to the expressivity of HyperFields.

[*]Equal contribution [1]Toyota Technological Institute at Chicago [2]University of Chicago. Correspondence to: Sudarshan Babu <sudarshan@ttic.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Recent advancements in text-to-image synthesis methods, highlighted by the works of Ramesh et al. (2021); Yu et al. (2022), have ignited similar interest in the field of text-to-3D synthesis. This interest has grown in tandem with the emergence of Neural Radiance Fields (NeRFs) (Mildenhall et al., 2020; Yu et al., 2021b; Jain et al., 2021), which is a popular 3D representation for this task, due to their ability to robustly depict complex 3D scenes.

To date, most text-conditioned 3D synthesis methods rely on either text-image latent similarity matching or diffusion denoising, both of which involve computationally intensive per-prompt NeRF optimization (Jain et al., 2022; Poole et al., 2022; Lin et al., 2022). Bypassing the need for per-prompt optimization remains a non-trivial challenge.

We propose to solve this problem through a hypernetwork-based neural pipeline, in which a single hypernetwork (Ha et al., 2016b) is trained to generate the weights of individual NeRF networks, each corresponding to a unique scene. Once trained, this hypernetwork is capable of efficiently producing the weights of NeRFs corresponding to novel prompts, either through a single forward pass or with minimal fine-tuning. Sharing the hypernetwork across multiple training scenes enables effective transfer of knowledge to new scenes, leading to better generalization and faster convergence. However, we find that a naive hypernetwork design is hard to train.

Our method, *HyperFields*, overcomes these challenges through several design choices. We propose predicting the weights of each layer of the NeRF network in a *progres-*

sive and *dynamic* manner. Specifically, we observe that the intermediate (network) activations from the hypernetwork-predicted NeRF can be leveraged to guide the prediction of subsequent NeRF weights effectively.

To enhance the training of our hypernetwork, we introduce a distillation-based framework rather than the Score Distillation Sampling (SDS) used in Poole et al. (2022); Wang et al. (2022). We introduce NeRF distillation, in which we first train individual text-conditioned NeRF scenes (using SDS loss) that are used as teacher NeRFs to provide fine-grained supervision to our hypernetwork (see Fig. 2). The teacher NeRFs provide exact colour and geometry labels, eliminating noisy training signals.

Our NeRF distillation framework allows for training HyperFields on a much larger set of scenes than with SDS, scaling up to 100 different scenes without any degradation in scene quality. Importantly, NeRF distillation is agnostic to the choice of text-to-3D model, so that HyperFields can learn high-quality and complex scenes from the latest generative model in a plug-and-play fashion. We show results of our method trained on high-detail scenes from Prolific Dreamer in Figures 7, 8 and 11.

Once trained, our model can synthesize novel in-distribution NeRF scenes in a single forward pass (Fig. 1, second column) and enables accelerated convergence for out-of-distribution scenes, requiring only a few fine-tuning steps (Fig. 1, third column). We clarify our use of the terms "in-distribution" and "out-of-distribution" in Sections 4.1 and 4.2 respectively. These results suggest that our method learns a semantically meaningful mapping. We justify our design choices through ablation experiments which show that both the dynamic hypernetwork architecture and NeRF distillation are critical to our model's expressivity.

Our successful application of dynamic hypernetworks to this difficult problem of generalized text-conditioned NeRF synthesis suggests a promising direction for future work on generalizing and parameterizing neural implicit functions through other neural networks.

2. Background and Related Work

Our work combines several prominent lines of work: neural radiance fields, score-based 3D synthesis, and learning function spaces using hypernetworks.

2.1. 3D Representation via Neural Radiance Fields

There are many competing methods of representing 3D data in 3D generative modeling, such as point-clouds (Nichol et al., 2022; Zhou et al., 2021), meshes (Michel et al., 2021; Hong et al., 2022; Metzer et al., 2022; Zeng et al., 2022), voxels (Sanghi et al., 2021; 2022), and signed-distance fields (Wang et al., 2021; Yariv et al., 2021; Esposito et al., 2022). This work explores the popular representation of 3D scenes by Neural Radiance Fields (NeRF) (Mildenhall et al., 2020; Xie et al., 2021; Gao et al., 2022). NeRFs were originally introduced to handle the task of multi-view reconstruction, but have since been applied in a plethora of 3D-based tasks, such as photo-editing, 3D surface extraction, and large/city-scale 3D representation (Gao et al., 2022).

There have been many improvements on the original NeRF paper, especially concerning training speed and fidelity (Chen et al., 2022a;b; Müller et al., 2022; Sun et al., 2021; Yu et al., 2021a). HyperFields uses the multi-resolution hash grid introduced in InstantNGP (Müller et al., 2022).

2.2. Score-Based 3D Generation

While many works attempt to directly learn the distribution of 3D models via 3D data, others opt to use guidance from 2D images due to the vast difference in data availability. Such approaches replace the photometric loss in NeRF's original objective with a guidance loss. The most common forms of guidance in the literature are from CLIP (Radford et al., 2021) or a frozen, text-conditioned 2D diffusion model. The former methods minimize the cosine distance between the image embeddings of the NeRF's renderings and the text embedding of the user-provided text prompt (Jain et al., 2022; Chen et al., 2022a; Jain et al., 2021).

Noteworthy 2D diffusion-guided models include DreamFusion (Poole et al., 2022) and Score Jacobian Chaining (SJC) (Wang et al., 2022), which feed noised versions of images rendered from a predicted NeRF into a frozen text-to-image diffusion model (Imagen (Saharia et al., 2022) and StableDiffusion Rombach et al. (2021), respectively) to obtain what can be understood as a scaled Stein Score (Liu et al., 2016). Our work falls into this camp, as we rely on score-based gradients derived from StableDiffusion to train the NeRF models which guide our hypernetwork training.

We use the following gradient motivated in DreamFusion:

$$\nabla_\theta \mathcal{L}(\phi, g(\theta)) \triangleq \mathbb{E}_{t,c}\left[w(t)(\hat{\epsilon}_\phi(z_t; y, t) - \epsilon)\frac{\partial x}{\partial \theta}\right] \quad (1)$$

which is similar to the gradient introduced in SJC, with the key difference being SJC directly predicts the noise score whereas DreamFusion predicts its residuals. We refer to optimization using this gradient as *Score Distillation Sampling* (SDS), following the DreamFusion authors. Followup work has aimed at improving 3D generation quality (Wang et al., 2023; Metzer et al., 2023; Chen et al., 2023), whereas we target an orthogonal problem of generalization and convergence of text-to-3D models.

Connections to ATT3D: We note that our work is concurrent and independent of ATT3D (Lorraine et al., 2023). We are similar in that we both train a hypernetwork to generate

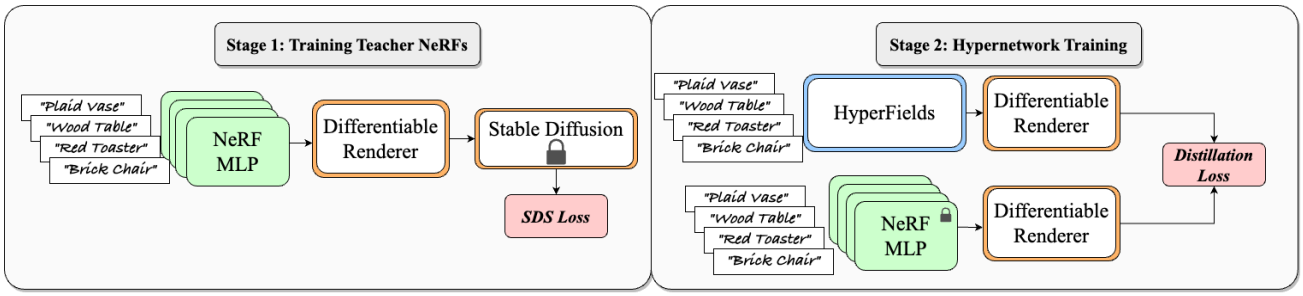

Figure 2. **Overview.** Our training pipeline proceeds in two stages. **Stage 1:** We train a set of single prompt text-conditioned teacher NeRFs using Score Distillation Sampling. **Stage 2:** We distill these single scene teacher NeRFs into the hypernetwork, through a photometric loss between the renders of the hypernetwork with the teacher network, which we dub our *distillation loss.*

NeRF weights for a set of scenes during training and generalize to novel in-distribution scenes without any test time optimization. On top of the in-distribution generalization experiments, we also demonstrate accelerated convergence to novel out-of-distribution scenes (defined in 4.2), which ATT3D does not.

On the technical side, we primarily differ in our novel dynamic hypernetwork architecture. Our hypernetwork generates the MLP weights of the NeRF, while ATT3D outputs the weights of the hash grid in their InstantNGP model. Importantly, our hypernetwork layers are conditioned on not just the input text prompt, but also the activations of the generated NeRF MLP (3). We show through our ablations that this dynamic hypernetwork conditioning is essential to the expressivity of our network, as it enables the network to change its weights for the same scene as a function of the view that is being rendered. In contrast, in ATT3D, the generated hash grid is the same regardless of the view being rendered, potentially resulting in the loss of scene detail.

Finally, ATT3D is built on Magic3D (Lin et al., 2022) which is a proprietary and more powerful text-to-3D model than the publicly available stable DreamFusion model (Tang, 2022) that we use in most of our experiments. We show that our model is capable of learning high quality and complex NeRF scenes produced by more powerful models such as ProlificDreamer without reduction in generation quality 4.3.

2.3. HyperNetworks

Hypernetworks are networks that are used to generate weights of other networks which perform the actual task (task performing network) (Ha et al., 2016a). Many works attempt to use hypernetworks as a means to improve upon conditioning techniques. Among these, some works have explored applying hypernetworks to implicit 2D representations (Sitzmann et al., 2020; Perez et al., 2017; Alaluf et al., 2021), and 3D representations (Sitzmann et al., 2019; 2021; Chiang et al., 2021). Very few works apply hypernetworks to radiance field generation. Two notable ones are HyperDiffusion and Shape-E, which both rely on denoising diffusion for generation (Erkoç et al., 2023; Jun & Nichol, 2023). HyperDiffusion trains an unconditional generative model which diffuses over sampled NeRF weights, and thus cannot do text-conditioned generation. Shap-E diffuses over latent codes which are then mapped to weights of a NeRF MLP, and requires teacher point clouds to train. Due to the memory burden of textured point clouds, scene detail is not well represented in Shap-E. Both of these methods have the same limitations of slow inference due to denoising sampling. In contrast, our method predicts NeRF weights dynamically conditioned on the 1) text prompt, 2) the sampled 3D coordinates, and 3) the previous NeRF activations.

An interesting class of hypernetworks involve models conditioned on the activations or inputs of the task-performing network (Chen et al., 2020). These models take the following form: let h, g be the hypernetwork and the task performing network respectively. Then $W = h(a)$, where W acts as the weights of g and a is the activation from the previous layer of g or the input to g. These are called dynamic hypernetworks, as the predicted weights change dynamically with respect to the layer-wise signals in g. Our work explores the application of dynamic hypernetworks to learning a general map between text and NeRFs.

3. Method

Our method consists of two key innovations, the dynamic hypernetwork architecture and NeRF distillation training. We discuss each of these two components in detail below.

3.1. Dynamic Hypernetwork

The dynamic hypernetwork consists of the Transformer $\mathcal{T}$ and MLP modules as given in figure 3. The sole input to the dynamic hypernetwork is the scene information represented as a text description. The text is then encoded by a frozen pretrained BERT model, and the text embedding z is processed by $\mathcal{T}$. Let conditioning token CT = $\mathcal{T}(z)$ be the intermediate representation used to provide the current scene information to the MLP modules. Note that the text

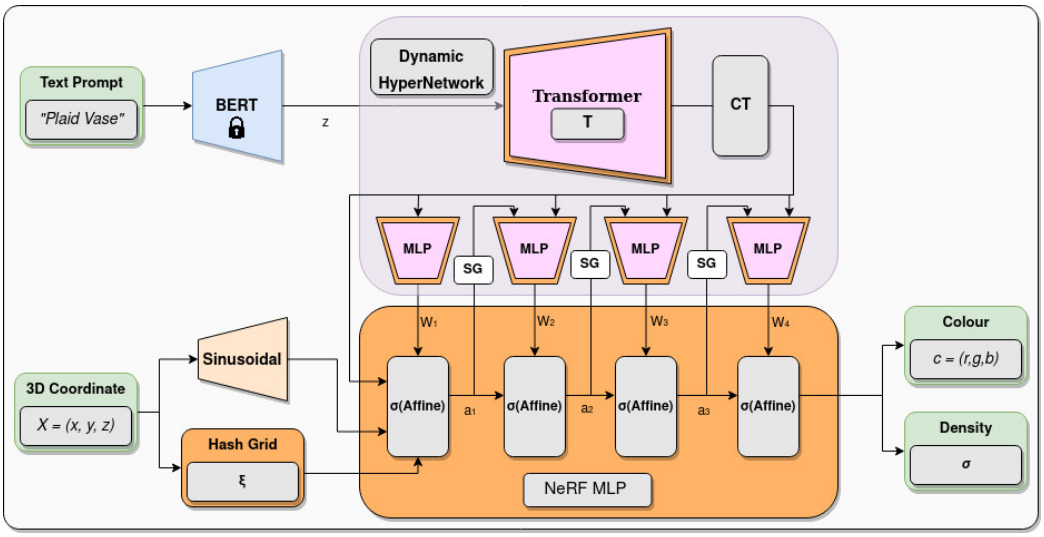

Figure 3. The input to the HyperFields system is a text prompt, which is encoded by a pre-trained text encoder (frozen BERT model). The text latents are passed to a Transformer module, which outputs a conditioning token (CT). This conditioning token (which supplies scene information) is used to condition each of the MLP modules in the hypernetwork. The first hypernetwork MLP (on the left) predicts the weights W_1 of the first layer of the NeRF MLP. The second hypernetwork MLP then takes as input both the CT and a_1, which are the *activations* from the first predicted NeRF MLP layer, and predicts the weights W_2 of the second layer of the NeRF MLP. The subsequent scene-conditioned hypernetwork MLPs follow the same pattern, taking the activations a_{i-1} from the previous predicted NeRF MLP layer as input to generate weights W_i for the ith layer of the NeRF MLP. We include stop gradients (SG) to stabilize training.

embeddings z can come from any text encoder, though in our experiments we use frozen BERT embeddings.

In addition to conditioning token CT, each MLP module takes in the activations from the previous layer a_{i-1} as input. Given these two inputs, the MLP module is tasked with generating parameters W_i for the ith layer of the NeRF MLP. For simplicity let us assume that we sample only one 3D coordinate and viewing direction per minibatch, and let h be the hidden dimension of the NeRF MLP. Then $a_{i-1} \in \mathbb{R}^{1\times h}$. Now the weights $W_i \in \mathbb{R}^{h\times h}$ of the ith layer are given as follows:

$$W_i = \text{MLP}_i(CT, a_{i-1}) \tag{2}$$

The forward pass of the ith layer is:

$$a_i = W_i * a_{i-1} \tag{3}$$

where $a_i \in \mathbb{R}^{1\times h}$ and $*$ is matrix multiplication. This enables the hypernetwork MLPs to generate a different set of weights for the NeRF MLP that are best suited for each given input 3D point and viewing direction pair. This results in effectively a unique NeRF MLP for each 3D point and viewing direction pair.

In practice training with minibatch size 1 is impractical, so during training we sample a non-trivial minibatch size and generate weights that are best suited for the given minibatch, as opposed to weights unique to each 3D coordinate and viewing direction pair as illustrated above.

In order to generate a unique set of weights for a given minibatch we do the following:

$$\overline{a}_{i-1} = \mu(a_{i-1}) \tag{4}$$

$$W_i = MLP_i(CT, \overline{a}_{i-1}) \tag{5}$$

Where $\mu(.)$ averages over the minibatch index. So if the minibatch size is n, then $a_{i-1} \in R^{n\times h}$, and $\overline{a}_{i-1} \in \mathbb{R}^{1\times h}$ and the forward pass is still computed as given in equation 3. This adaptive nature of the predicted NeRF MLP weights leads to the increased flexibility of the model. As shown in our ablation experiments in Figure 9, it is an essential piece to our model's large scene capacity.

3.2. NeRF Distillation

As shown in figure 2, we first train individual DreamFusion NeRFs on a set of text prompts, following which we train the HyperFields architecture with supervision from these single-scene DreamFusion NeRFs.

The training routine is outlined in Algorithm F, in which at each iteration, we sample n prompts and a camera viewpoint for each of these text prompts (lines 2 to 4). Subsequently, for the ith prompt and camera viewpoint pair we render image $\mathcal{I}_i$ using the ith pre-trained teacher NeRF (line 5). We then condition the HyperFields network ϕ_{hf} with the ith prompt, and render the image $I_i^{'}$ from the ith camera view point (line 6). We use the image rendered by the pre-trained teacher NeRF as the ground truth supervision to

HyperFields (line 7). For the same sampled n prompts and camera viewpoint pairs, let $\mathcal{I}_1^{'}$ to $\mathcal{I}_n^{'}$ be the images rendered by HyperFields and $\mathcal{I}_1$ to $\mathcal{I}_n$ be the images rendered by the respective pre-trained teacher NeRFs. The distillation loss is given as follows:

$$\mathcal{L}_d = \frac{\sum_{i=1}^{n}(I_i - I_i^{'})^2}{n} \tag{6}$$

We observe through our ablations in Figure 10 that this simple distillation scheme greatly helps HyperFields in learning to fit multiple text prompts simultaneously, as well as learn a more general mapping of text to NeRFs.

4. Results

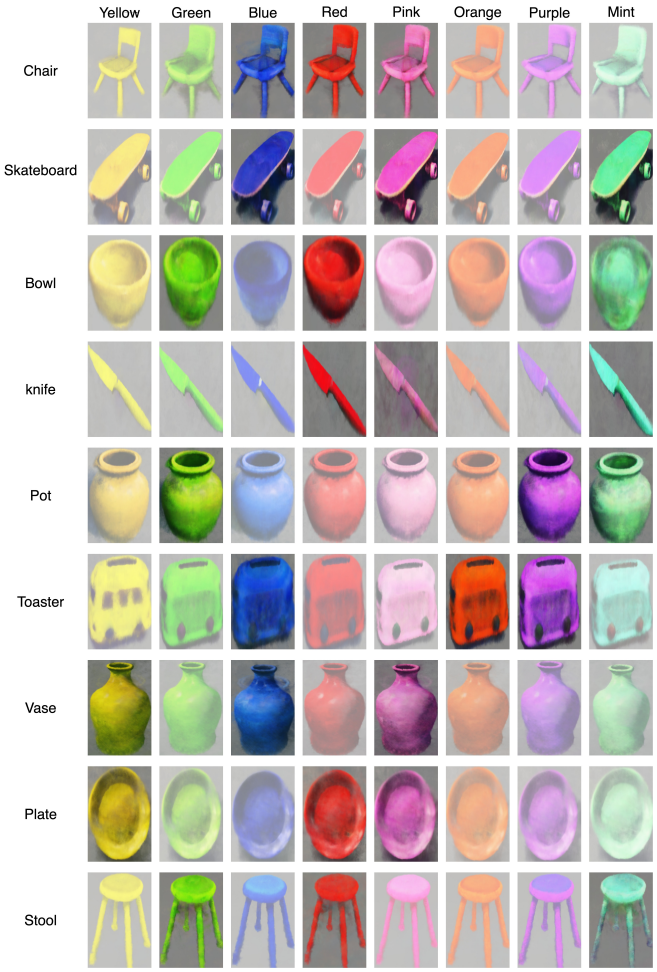

Figure 4. **Zero-Shot In-Distribution Generalization.** We train HyperFields on a 9x8 grid of object/color combination scenes, and hold out a subset of combinations. The faded scenes are in the training set and the bright scenes are the trained model's **zero-shot predictions** of the holdout set.

We evaluate HyperFields by demonstrating its generalization capabilities, out-of-distribution convergence, amortization benefits, and ablation experiments. In Sec. 4.1 and Sec. 4.2 we evaluate the model's ability to synthesize novel scenes, both in and out-of-distribution. We quantify the amortization benefits of having this general model compared to optimizing individual NeRFs in Sec. 4.4. Finally, our ablations in Sec. 4.5 justify our design choices of dynamic conditioning and NeRF distillation training.

	Top-1	Top-3	Top-5	Top-6	Top-10
Unseen	57.1	85.7	85.7	90.4	95.2
Seen	69.5	88.1	94.9	94.9	96.6

Table 1. CLIP Retrieval Scores: We report the average retrieval scores for the scenes shown in Fig. 4. We achieve similar scores between the seen and unseen prompts, indicating that our zero-shot generations are of similar quality to the training scenes.

4.1. In-Distribution Generalization

Fig. 4 shows the results of training on a subset of combinations of 9 shapes and 8 colours, while holding out 3 colours for each shape. Our model generates NeRFs in a zero-shot manner for the held-out prompts (opaque scenes in Fig. 4) with quality nearly identical to the trained scenes.

We call this *in-distribution generalization* as both the shape and the color are seen during training but the inference scenes (opaque scenes in Fig.4) are novel because the combination of color and shape is unseen during training. For example: "Orange toaster" is a prompt the model has not seen during training, though it has seen the color "orange" and the shape "toaster" in its training set.

We quantitatively evaluate the quality of our zero-shot predictions with CLIP retrieval scores. The support set for the retrieval consists of all 72 scenes (27 unseen and 45 seen) shown in Fig. 4. In Table 1 we compute the top-k retrieval scores by CLIP similarity. The table reports the average scores for top-k retrieval, separated by unseen (zero-shot) and seen prompts. The similarity in scores between unseen and seen prompts indicates that our model's zero-shot predictions are of similar quality to the training scenes.

4.2. Accelerated Out-of-Distribution Convergence

We further test HyperFields's ability to generate shapes and attributes that it has *not seen* during training. We call this *out-of-distribution inference* because the specified geometry and/or attribute are not within the model's training set.

We train our model on a rich source of prompts, across multiple semantic dimensions (material, appearance, shape). The list of prompts used is provided in the appendix material section D using NeRF distillation loss. Post training, we test our model on the prompts in Fig. 5. The prompts are

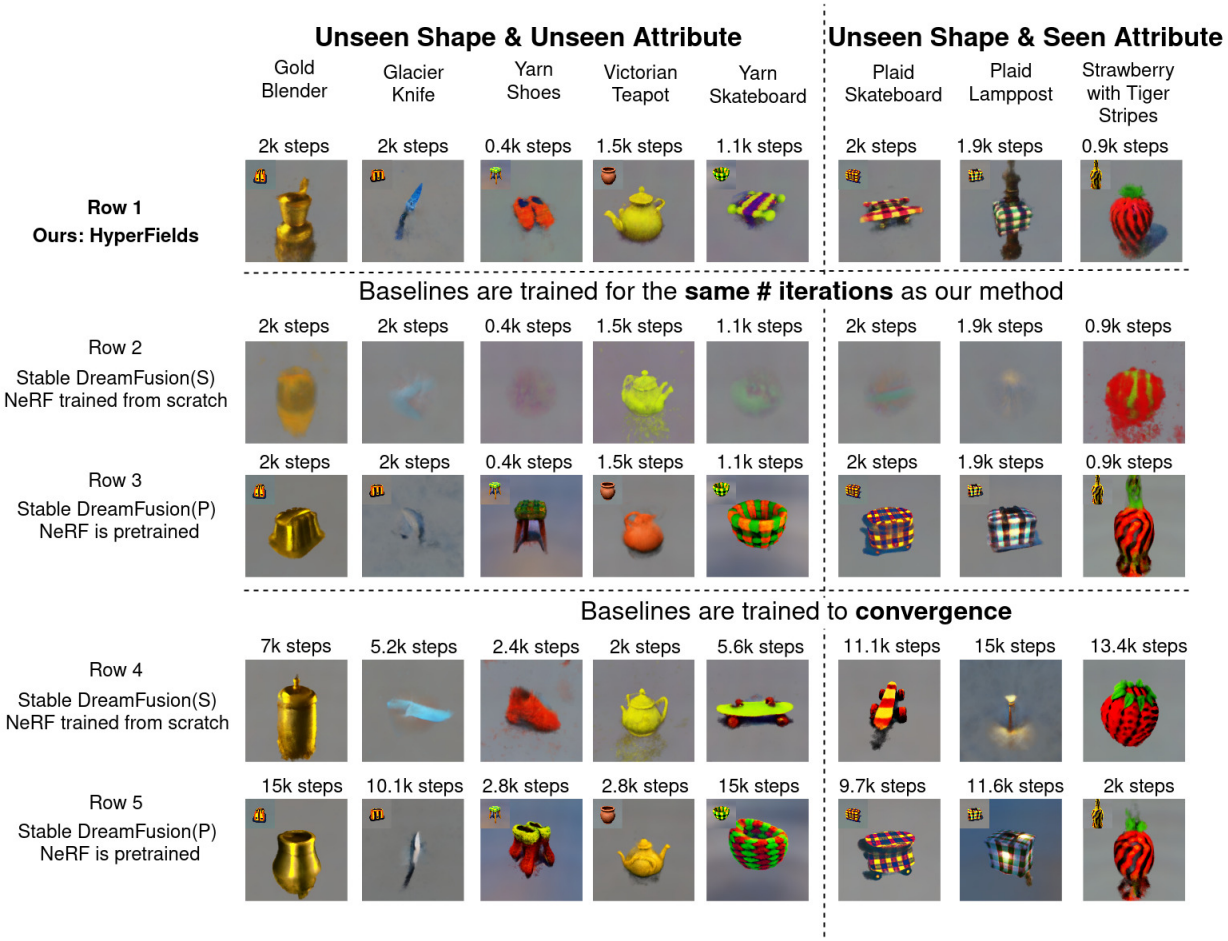

Figure 5. **Finetuning to out-of-distribution prompts: unseen shape and or unseen attribute.** Our method generates out-of-distribution scenes in at most 2k finetuning steps (row 1), whereas the baseline models are far from the desired scene at the same number of iterations (rows 2 and 3). When allowed to fine-tune for significantly longer (rows 4 and 5) the baseline generations are at best comparable to our model's generation quality, demonstrating that our model is able to adapt better to out-of-distribution scenes.

Model	Golden Blender	Yarn Shoes	Yarn Skateboard	Plaid Skateboard	Plaid Lamppost	Strawberry with tiger stripes
Our Method (↓)	1.3 ± 0.14	1.0 ± 0.09	1.3 ± 0.07	1.3 ± 0.11	1.4 ± 0.03	1.1 ± 0.14
Best DreamFusion Baseline (↓)	2.5 ± 0.11	2.4 ± 0.13	2.3 ± 0.11	1.7 ± 0.09	2.0 ± 0.09	2.2 ± 0.08
P-Score (↓)	1.0×10^{-25}	2.01×10^{-34}	2.8×10^{-30}	1.1×10^{-8}	1.3×10^{-25}	2.0×10^{-26}

Table 2. **Average User-Reported Ranks (N=450):** We report the average rank submitted by all users for our method, and compute the average rank for all 33 of the baselines. We report the average rank of the best performing baseline for each prompt (with $\pm$95% confidence intervals). Our method is consistently preferred over the best baseline, despite the best baseline consuming 33x more computational resources than our method to find. We report the p-value for the difference in rank between our method and the next best DreamFusion baseline, and find it is significant at the 1% level across all our prompts.

grouped based on whether both shape and attribute are unseen (column 1, Fig. 5) or just the shape is unseen (column 2, Fig. 5). For example, in "gold blender" both material "gold" and shape "blender" are unseen during training.

Since these prompts contain geometry/attributes that are unseen during training, we do not expect high quality generation without additional optimization. Instead, we demonstrate that fine-tuning the trained HyperFields model on SDS loss for the given the out-of-distribution prompt can lead to accelerated convergence especially when compared to the DreamFusion baselines.

Figure 6. **Prolific Dreamer scenes distilled into HyperFields:** We distill 30 high quality and complex scenes generated by Prolific Dreamer into a single HyperFields model, which underscores the modeling capacity of our novel architecture.

We consider two baselines, 1) **Stable Dreamfusion (S):** Publicly available implementation of Dreamfusion trained from **S**cratch, 2) **Stable Dreamfusion (P):** Stable Dreamfusion model **P**re-trained on a semantically close scene and finetuned to the target scene. The motivation in using Stable Dreamfusion (P) is to have a pre-trained model as a point of comparison against HyperFields model.

4.2.1. Qualitative Evaluation

We show out-of-distribution generation results for 8 different scenes in Fig. 5. The inset images in the upper left of row 1 of Fig. 5 are the scenes generated zero-shot by our method, *with no optimization*, when provided with the out-of-distribution prompt. The model chooses the *semantic nearest neighbour* from its training data as the initial guess for out-of-distribution prompts. For example, when asked for a "golden blender" and "glacier knife", our model generates a scene with "tiger striped toaster", which is the only related kitchenware appliance in the model sees during training. We pretrain the Stable Dreamfusion(P) baselines to the same scenes predicted by our model zero-shot. The pretrained scenes for Stable Dreamfusion(P) are given as insets in the upper left of row 3 and 5 in Fig. 5.

By finetuning on a small number of epochs for each out-of-distribution target scene using score distillation sampling, our method can converge much faster to the target scene than the baseline DreamFusion models. In row 2 and 3 of Fig. 5, we see that both Dreamfusion(S) and (P), barely learn the target shape for the same amount of training budget as our method. In rows 4 and 5 of Fig. 5 we let the baselines train to convergence, and even then the quality of the converged baseline scenes are worse or at best comparable to our model's generation quality. On average we see a 5x speedup in convergence.

Importantly, DreamFusion(P) which is pre-trained to **the same zero-shot predictions of our model** is unable to be fine-tuned to the target scene as efficiently and at times get stuck in suboptimal local minima close to the initialization (see "yarn skateboard" row 3 and 5 in Fig. 5). This demonstrates that HyperFields learns a semantically meaningful mapping from text to NeRFs that cannot be arbitrarily achieved through neural optimization. We further explore the smoothness of this mapping through interpolation experiments in Sec. I of the appendix.

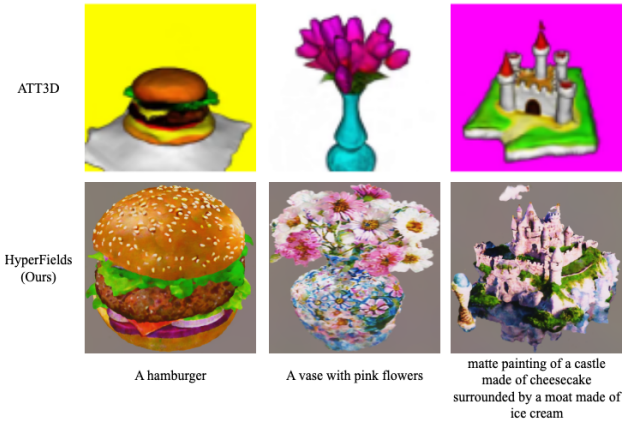

Figure 7. **Visual Comparison to ATT3D.** We visually compare scenes packed into HyperFields against the same scenes shown in ATT3D. NeRF distillation allows HyperFields to inherit the high generation quality of Prolific Dreamer, so the scenes we generate are of higher visual quality and complexity.

4.2.2. Quantitative Evaluation

Model/Metric	CLIP Top-3		KID ↓	SSIM ↑
	Precision	Recall		
Stable DreamFusion	0.50	0.37	0.17	0.55
Our Model	0.77	0.63	0.13	0.62

Table 3. The HyperFields-generated renders for the out-of-distribution prompts (see Fig. 5) demonstrate superior performance compared to Stable DreamFusion's renders across multiple metrics.

Additionally, in order to get a quantitative evaluation of our generation quality for the out-of-distribution prompts (in Fig. 5) we conduct a human study where we ask participants to rank the render that best adheres to the given prompt in descending order (best render is ranked 1). We compare our method's generation with 33 different DreamFusion models. 1 is trained from scratch and the other 32 are finetuned from checkpoints corresponding to the prompts in section D. Of these 33 models we pick the best model for each of the out-of-distribution prompts, so the computational budget

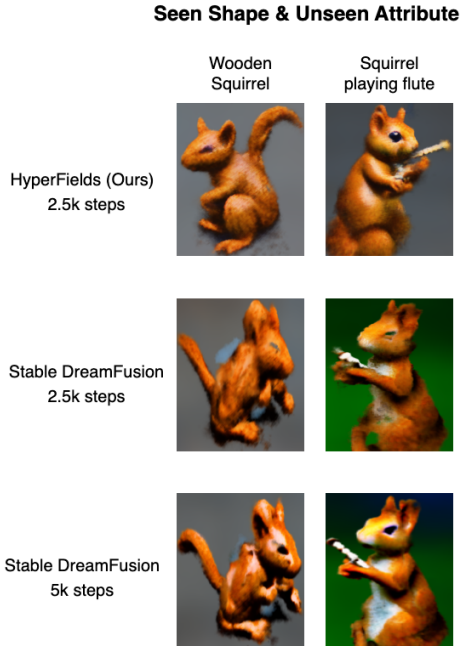

Figure 8. **Prolific Dreamer OOD Comparison.** We finetune a HyperFields model trained on the scenes in Fig. 11 on novel attributes, and compare against the Stable DreamFusion baseline trained for the same number of steps and **double** the number of steps.

to find the best baseline for a given prompt is almost 33x that our of method. Note each of these models, including ours, are trained for the same number of steps. We report average user-reported rank for our method and the average best baseline chosen *for each prompt* in Tab. 2. We outrank the best DreamFusion baseline consistently across all our out-of-distribution prompts.

Furthermore, in Tab. 3 we evaluate our method's out-of-distribution generation quality using precision and recall metrics for top-3 CLIP retrieval tasks. For each given out-of-distribution prompt (in Fig. 5), we check if the associated render is among the top three retrievals according to CLIP's ranking. Additionally, we assess the quality of our renders using KID and SSIM scores. Across all these quantitative metrics, HyperFields outperforms the stable DreamFusion baseline.

4.3. HyperFields with Prolific Dreamer Teachers

NeRF distillation training means that our pipeline is agnostic to the choice of text-to-3D model, and thus can inherit high-quality generation properties from the latest open-source models. We demonstrate this in Fig. 6, where we generate teacher NeRFs using Prolific Dreamer (Wang et al., 2023) and distill them into a single HyperFields model. Our model generates the distilled scenes with virtually no quality degradation. We provide a visual comparison of our generations

against the same scenes from ATT3D in Fig. 7.

We also show accelerated out-of-distribution convergence of our high-quality HyperFields model in Fig 8. Note that even with double the amount of training, the Stable DreamFusion baseline is unable to match our model's generation quality.

4.4. Amortization Benefits

The cost of pre-training HyperFields and individual teacher NeRFs is easily amortized in both in-distribution and out-of-distribution prompts. Training the teacher NeRFs is not an additional overhead; it's the cost of training a DreamFusion model on each of those prompts. The only overhead is the NeRF distillation training in stage 2 (Fig. 2), which takes roughly two hours. This overhead is offset by our ability to generate unseen combinations in a feedforward manner.

For comparison, the DreamFusion baseline takes approximately 30 minutes to generate each test scene in Fig. 4, totaling $\sim$14 hours for all 27 test scenes. Our model can generate all 27 test scenes in less than a minute, making it an order of magnitude faster than DreamFusion, even with the 2 hour distillation overhead.

Our method's ability to converge faster to new out-of-distribution prompts leads to linear time-saving for each new prompt. This implies a practical use case of our model for rapid out-of-distribution scene generation in a real world setting. As shown in Fig. 5, the baseline's quality only begins to match ours after 3-5x the amount of training time.

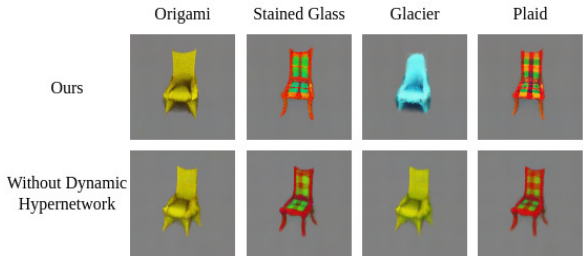

Figure 9. **Dynamic Hypernet Packing.** Without dynamic conditioning, the network collapses the origami/glacier attributes and stained glass/plaid attributes.

4.5. Ablations

We ablate on the activation conditioning in our dynamic hypernetwork ("without dynamic hypernetwork") in Fig. 9. Row 2 shows that even in the simple case of 4 scenes the static hypernetwork collapses the "glacier" and "origami" styles, and the "plaid" and "stained glass" styles.

If we attempt to pack the dynamic hypernetwork using just Score Distillation Sampling (SDS) from DreamFusion, we experience a type of mode collapse in which the SDS optimization guides similar shapes towards the same common geometry. See Fig. 10 for an example of this mode collapse.

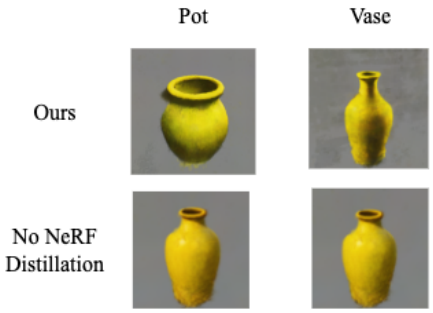

Figure 10. **NeRF Distillation.** We compare packing results when training with score distillation ("No NeRF Distillation") versus our NeRF distillation method ("Ours"). The iterative optimization of score distillation causes mode collapse in geometry.

5. Conclusion

We present HyperFields, a novel framework for generalized text-to-NeRF synthesis, which can produce individual NeRF networks in a single feedforward pass. Our results highlight a promising step in learning a general representation of semantic scenes. Our novel dynamic hypernetwork architecture coupled with NeRF distillation learns an efficient mapping of text token inputs into a smooth and semantically meaningful NeRF latent space. Our experiments show that with this architecture we are able to fit over 100 different scenes in one model, and predict high quality unseen NeRFs either zero-shot or with a few finetuning steps. Comparing to existing work, our ability to train on multiple scenes greatly accelerates convergence of novel scenes. In future work we would like to explore the possibility of generalizing the training and architecture to achieving zero-shot open vocabulary synthesis of NeRFs and other implicit 3D representations.

6. Limitations

Our model is trained through distillation from teacher models, thus, the quality of our generated scenes is bound by the quality of the current state-of-the-art open source models. Similarly, our model inherits the limitations of these SOTA models. For instance, it is well known that Stable Diffusion struggles with long prompts with complex compositionality and janusing, which are also limitations of our model.

Impact Statement

As mentioned above our model inherits limitations from the teacher models. Similarly, our model inherits potential harmful biases of the teacher models. Any stereotypes or biases from the teacher models will be reproduced by our

model.

Acknowledgements

This work was partially supported by NSF grants CNS-1956180 and 2304481, and BSF grant 2022363. Additional support was provided by gifts from Snap Research, Adobe Research, and Google Research. We also thank Adam Bohlander for his assistance in setting up the computing infrastructure.

References

Alaluf, Y., Tov, O., Mokady, R., Gal, R., and Bermano, A. H. Hyperstyle: Stylegan inversion with hypernetworks for real image editing, 2021. URL https://arxiv.org/abs/2111.15666.

Chen, A., Xu, Z., Geiger, A., Yu, J., and Su, H. Tensorf: Tensorial radiance fields, 2022a. URL https://arxiv.org/abs/2203.09517.

Chen, A., Xu, Z., Geiger, A., Yu, J., and Su, H. Tensorf: Tensorial radiance fields, 2022b. URL https://arxiv.org/abs/2203.09517.

Chen, R., Chen, Y., Jiao, N., and Jia, K. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation, 2023.

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., and Liu, Z. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11030–11039, 2020.

Chiang, P.-Z., Tsai, M.-S., Tseng, H.-Y., Lai, W.-s., and Chiu, W.-C. Stylizing 3d scene via implicit representation and hypernetwork, 2021. URL https://arxiv.org/abs/2105.13016.

Erkoç, Z., Ma, F., Shan, Q., Nießner, M., and Dai, A. HyperDiffusion: Generating Implicit Neural Fields with Weight-Space Diffusion, March 2023. URL http://arxiv.org/abs/2303.17015. arXiv:2303.17015 [cs].

Esposito, S., Baieri, D., Zellmann, S., Hinkenjann, A., and Rodolà, E. Kiloneus: A versatile neural implicit surface representation for real-time rendering, 2022. URL https://arxiv.org/abs/2206.10885.

Gao, K., Gao, Y., He, H., Lu, D., Xu, L., and Li, J. Nerf: Neural radiance field in 3d vision, a comprehensive review, 2022. URL https://arxiv.org/abs/2210.00379.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016a.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks, 2016b. URL https://arxiv.org/abs/1609.09106.

Hong, F., Zhang, M., Pan, L., Cai, Z., Yang, L., and Liu, Z. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *ACM Transactions on Graphics (TOG)*, 41(4):1–19, 2022.

Jain, A., Tancik, M., and Abbeel, P. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5885–5894, October 2021.

Jain, A., Mildenhall, B., Barron, J. T., Abbeel, P., and Poole, B. Zero-shot text-guided object generation with dream fields. 2022.

Jun, H. and Nichol, A. Shap-E: Generating Conditional 3D Implicit Functions, May 2023. URL http://arxiv.org/abs/2305.02463. arXiv:2305.02463 [cs].

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., and Lin, T.-Y. Magic3d: High-resolution text-to-3d content creation. *arXiv preprint arXiv:2211.10440*, 2022.

Liu, Q., Lee, J. D., and Jordan, M. I. A kernelized stein discrepancy for goodness-of-fit tests and model evaluation, 2016. URL https://arxiv.org/abs/1602.03253.

Lorraine, J., Xie, K., Zeng, X., Lin, C.-H., Takikawa, T., Sharp, N., Lin, T.-Y., Liu, M.-Y., Fidler, S., and Lucas, J. Att3d: Amortized text-to-3d object synthesis. *arXiv*, 2023.

Metzer, G., Richardson, E., Patashnik, O., Giryes, R., and Cohen-Or, D. Latent-nerf for shape-guided generation of 3d shapes and textures, 2022. URL https://arxiv.org/abs/2211.07600.

Metzer, G., Richardson, E., Patashnik, O., Giryes, R., and Cohen-Or, D. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12663–12673, 2023.

Michel, O., Bar-On, R., Liu, R., Benaim, S., and Hanocka, R. Text2mesh: Text-driven neural stylization for meshes. *arXiv preprint arXiv:2112.03221*, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127.

Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M. Point-e: A system for generating 3d point clouds from complex prompts, 2022. URL https://arxiv.org/abs/2212.08751.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. Dreamfusion: Text-to-3d using 2d diffusion, 2022. URL https://arxiv.org/abs/2209.14988.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the spectral bias of neural networks. 2018. doi: 10.48550/ARXIV.1806.08734. URL https://arxiv.org/abs/1806.08734.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation, 2021. URL https://arxiv.org/abs/2102.12092.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487.

Sanghi, A., Chu, H., Lambourne, J. G., Wang, Y., Cheng, C.-Y., and Fumero, M. Clip-forge: Towards zero-shot text-to-shape generation. *arXiv preprint arXiv:2110.02624*, 2021.

Sanghi, A., Fu, R., Liu, V., Willis, K., Shayani, H., Khasahmadi, A. H., Sridhar, S., and Ritchie, D. Textcraft: Zero-shot generation of high-fidelity and diverse shapes from text, 2022. URL https://arxiv.org/abs/2211.01427.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2019. URL https://arxiv.org/abs/1906.01618.

Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

Sitzmann, V., Rezchikov, S., Freeman, W. T., Tenenbaum, J. B., and Durand, F. Light field networks: Neural scene representations with single-evaluation rendering, 2021. URL https://arxiv.org/abs/2106.02634.

Sun, C., Sun, M., and Chen, H.-T. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction, 2021. URL https://arxiv.org/abs/2111.11215.

Tang, J. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion.

Wang, H., Du, X., Li, J., Yeh, R. A., and Shakhnarovich, G. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation, 2022. URL https://arxiv.org/abs/2212.00774.

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction, 2021. URL https://arxiv.org/abs/2106.10689.

Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., and Zhu, J. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. Neural fields in visual computing and beyond, 2021. URL https://arxiv.org/abs/2111.11426.

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. Volume rendering of neural implicit surfaces, 2021. URL https://arxiv.org/abs/2106.12052.

Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. Plenoctrees for real-time rendering of neural radiance fields, 2021a. URL https://arxiv.org/abs/2103.14024.

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021b.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson,

B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., and Wu, Y. Scaling autoregressive models for content-rich text-to-image generation, 2022. URL https://arxiv.org/abs/2206.10789.

Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., and Kreis, K. Lion: Latent point diffusion models for 3d shape generation, 2022. URL https://arxiv.org/abs/2210.06978.

Zhou, L., Du, Y., and Wu, J. 3d shape generation and completion through point-voxel diffusion, 2021. URL https://arxiv.org/abs/2104.03670.

A. Model Details

Baselines: Our baseline is 6 layer MLP with skip connections every two layers. The hidden dimension is 64. We use an open-source re-implementation (Tang, 2022) of DreamFusion as both our baseline model and architecture predicted by HyperFields, because the original DreamFusion works relies on Google's Imagen model which is not open-source. Unlike the original DreamFusion, the re-implementation uses Stable Diffusion (instead of Imagen). We use Adam with a learning rate of 1e-4, with an epoch defined by 100 gradient descent steps.

HyperFields: The architecture is as described in Figure 2 in the main paper. The dynamic hypernetwork generates weights for a 6 layer MLP of hidden dimension 64. The transformer portion of the hypernetwork has 6 self-attention blocks, each with 12 heads with a head dimension of 16. We condition our model with BERT tokens, though we experiment with T5 and CLIP embeddings as well with similar but marginally worse success. Similar to the baseline we use Stable Diffusion for guidance, and optimize our model using Adam with the a learning rate of 1e-4. We will release open-source code of our project in a future revision of the paper.

We use the multiresolution hash grid developed in InstantNGP Müller et al. (2022) for its fast inference with low memory overhead, and sinusoidal encodings γ to combat the known spectral bias of neural networks (Rahaman et al., 2018). The NeRF MLP has 6 layers (with weights predicted by the dynamic hypernetwork), with skip connections every two layers. The dynamic hypernetwork MLP modules are two-layer MLPs with ReLU non-linearities and the Transformer module has 6 self-attention layers. Furthermore, we perform adaptive instance normalization before passing the activations into the MLP modules of the dynamic hypernetwork and also put a stop gradient operator on the activations being passed into the MLP modules (as in figure 3).

B. Packing

Figure 11. **Prompt Packing.** Our dynamic hypernetwork is able to pack 9 different objects across 12 different prompts for a total of 108 scenes. Dynamic hypetnetwork coupled with NeRF distillation enables packing these scenes into one network.

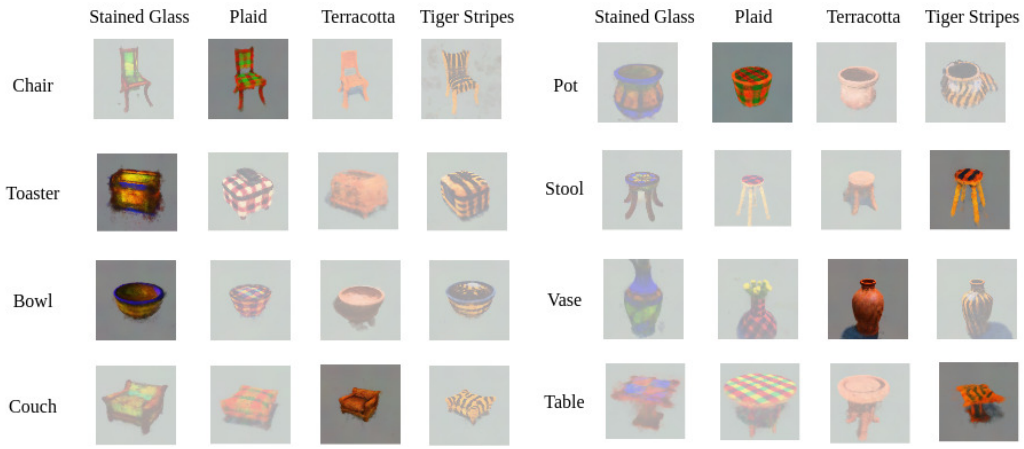

Figure 12. **Fine-Tuning In-Distribution: seen shape, seen attribute, unseen combination.** During training, the model observes every shape and color, but some combinations of shape and attribute remain unseen. During inference, the model generalizes by generating scenes that match prompts with previously unseen combinations of shape and attribute, with small amount of finetuning (atmost 1k steps).

C. In-Distribution Generalization with Complex Prompts

For additional attributes ("plaid", "Terracotta" etc.), our model produces reasonable zero-shot predictions, and after fewer than 1000 steps of finetuning with SDS is able to produce unseen scenes of high quality. We show these results in Fig. 12 with 8 objects and 4 styles, where 8 shape-style combinational scenes are masked out during training (opaque scenes in Fig. 12).

D. Out-of-Distribution Convergence

In Fig 5 we show the inference time prompts and the corresponding results. Here we provide the list of prompts used to train the model: "Stained glass chair", "Terracotta chair", "Tiger stripes chair", "Plaid toaster", "Terracotta toaster", "Tiger stripes toaster", "Plaid bowl", "Terracotta bowl", "Tiger stripes bowl", "Stained glass couch", "Plaid couch", "Tiger stripes couch", "Stained glass pot", "Terracotta pot", "Tiger stripes pot", "Stained glass vase", "Plaid vase", "Tiger stripes vase", "Stained glass table", "Plaid table", "Terracotta table".

Since the training prompts dont contain shapes such as "Blender", "Knife", "Skateboard", "Shoes", "Strawberry", "Lamp-post", "Teapot" and attributes such as "Gold", "Glacier", "Yarn", "Victorian", we term the prompts used in Fig 5 as out-of-distribution prompts–as the model does not see these shapes and attributes during training.

E. User Study Renders

We link to the images (including baselines) used in the user study described in Section 4.2.2 here. All renders are taken from the same camera angle and the baseline scenes are finetuned with the same number of iterations as our model.

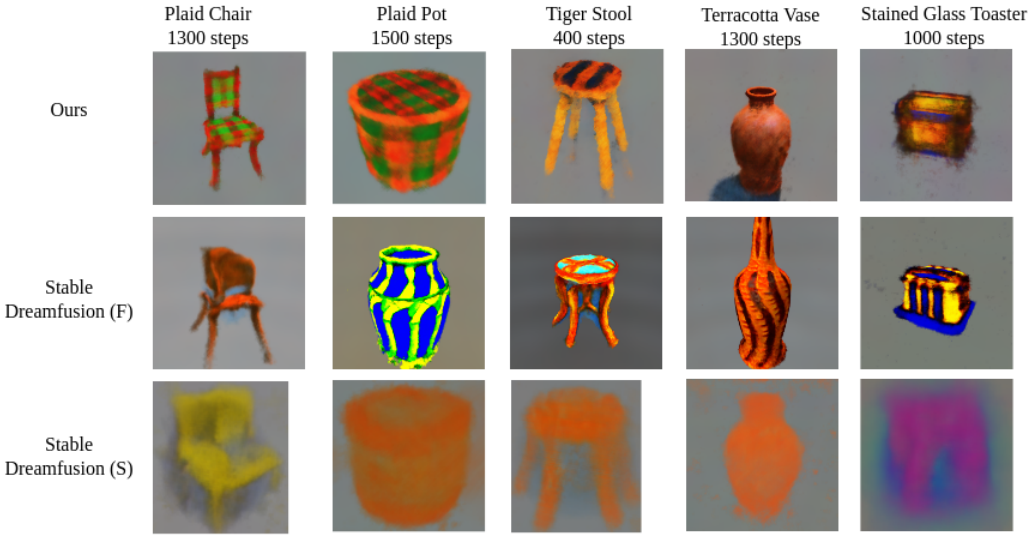

Figure 13. **Generalization Comparison.** We train a single HyperFields model and compare Stable DreamFusion. "Stable DreamFusion (F)" indicates finetuning from an initialized DreamFusion model. "Stable DreamFusion (S)" indicates the DreamFusion model trained from scratch. Zero-shot results and initializations are shown in the upper left of "Ours" and "Stable DreamFusion (F)", respectively. Above each column indicates the number of training epochs for each method add figures in the upper left.

F. Algorithm for training HyperFields

Require: $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots \mathcal{T}_N\}$ ▷ Set of text prompts
Require: $\mathcal{C}$ ▷ Set of Camera view points
Require: $\theta_1, \theta_2, \cdots \theta_N$ ▷ pre-trained NeRFs
Require: ϕ_{HF} ▷ Randomly initialized HyperFields
Require: $\mathcal{R}$ ▷ Differentiable renderer function

for each step **do**
  $\mathcal{T}_l, \mathcal{T}_m, \mathcal{T}_n \sim \mathcal{T}$ ▷ Sample text prompts from $\mathcal{T}$
  for $\mathcal{T}_i \in \{\mathcal{T}_l, \mathcal{T}_m, \mathcal{T}_n\}$ **do**
    $\mathcal{C}_i \sim \mathcal{C}$
    $\mathcal{I}_i = \mathcal{R}(\theta_i(\mathcal{C}_i))$ ▷ i^{th} nerf renders image for given camera $\mathcal{C}_i$
    $\mathcal{I}_i{}' = \mathcal{R}(\phi_{HF}(\mathcal{T}_i, \mathcal{C}_i))$ ▷ Condition ϕ_{HF} on i^{th} prompt
    $\mathcal{L}_i = (\mathcal{I}_i - \mathcal{I}_i{}')^2$
  end for
  $\mathcal{L}_d = \sum_{i \in \{l,m,n\}} \mathcal{L}_i$
end for

G. HyperFields Trained on Additional Prolific Dreamer Teachers

In addition to the scenes shown in Fig. 6, train HyperFields on a different set of ProlificDreamer teachers and the scenes generated by our single HyperFields model is shown in Fig. 14. This demonstrates the ability of HyperFields to learn another varied set of scenes with complex geometries.

H. Multi-View Consistency of Generated Scenes

In Fig. 15, we show multiple scenes generated by a single HyperFields model from various camera poses. Across multiple views we see that geometry is well formed and consistent with the geometry in other views.

I. BERT Token Interpolation

Another option for interpolation is to interpolate the input BERT embeddings fed in the our Dynamic HyperNet. We show results in Figure 16 where we interpolate across two chair colors in a Dynamic HyperNet trained on only chair colors. The interpolation is highly non-smooth, with a single intermediate color shown at $\delta = 0.4$ and discontinuities on either end at

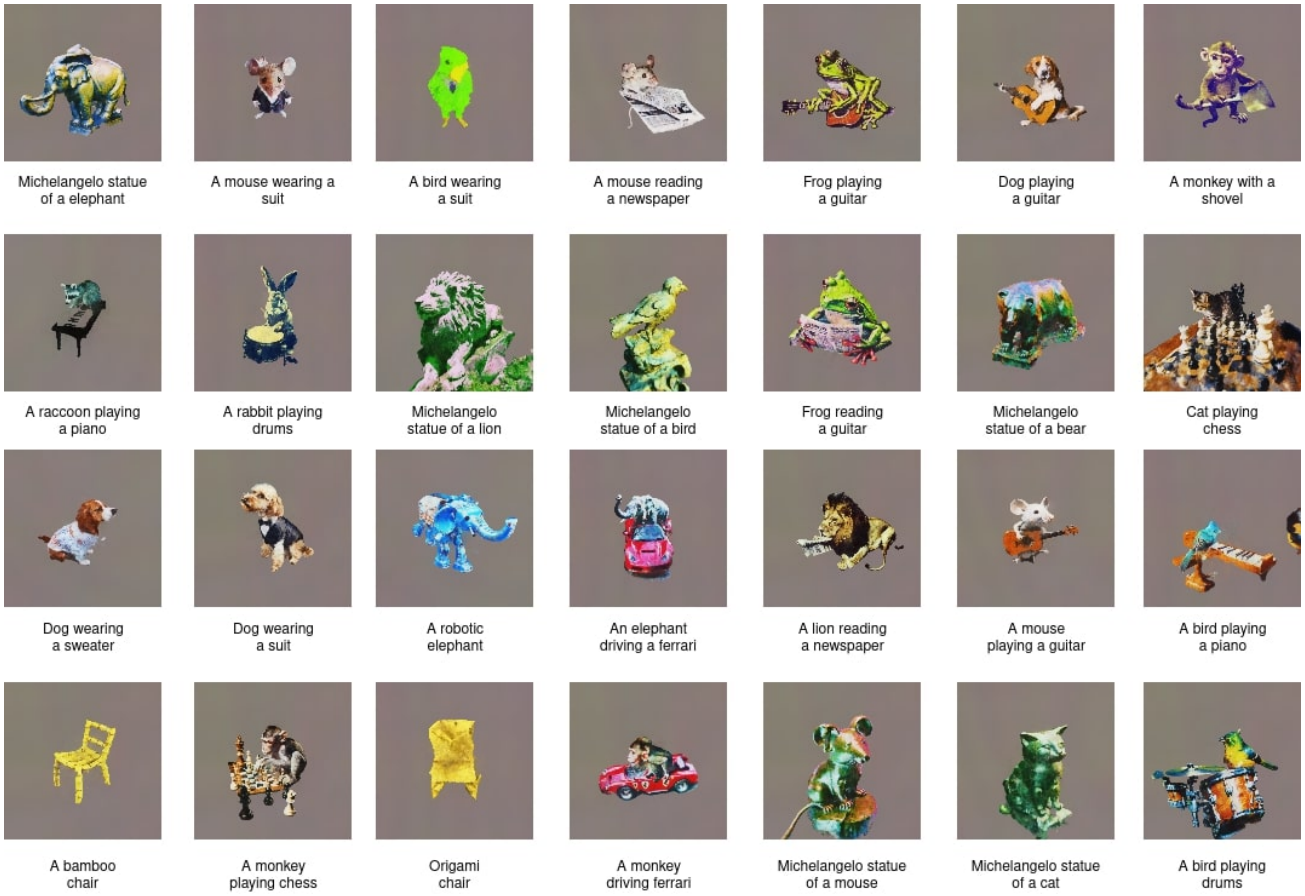

Figure 14. Additional set of Prolific Dreamer scenes distilled into HyperFields model, showcasing the ability of Hyperfields to learn a significantly diverse set of geometries.

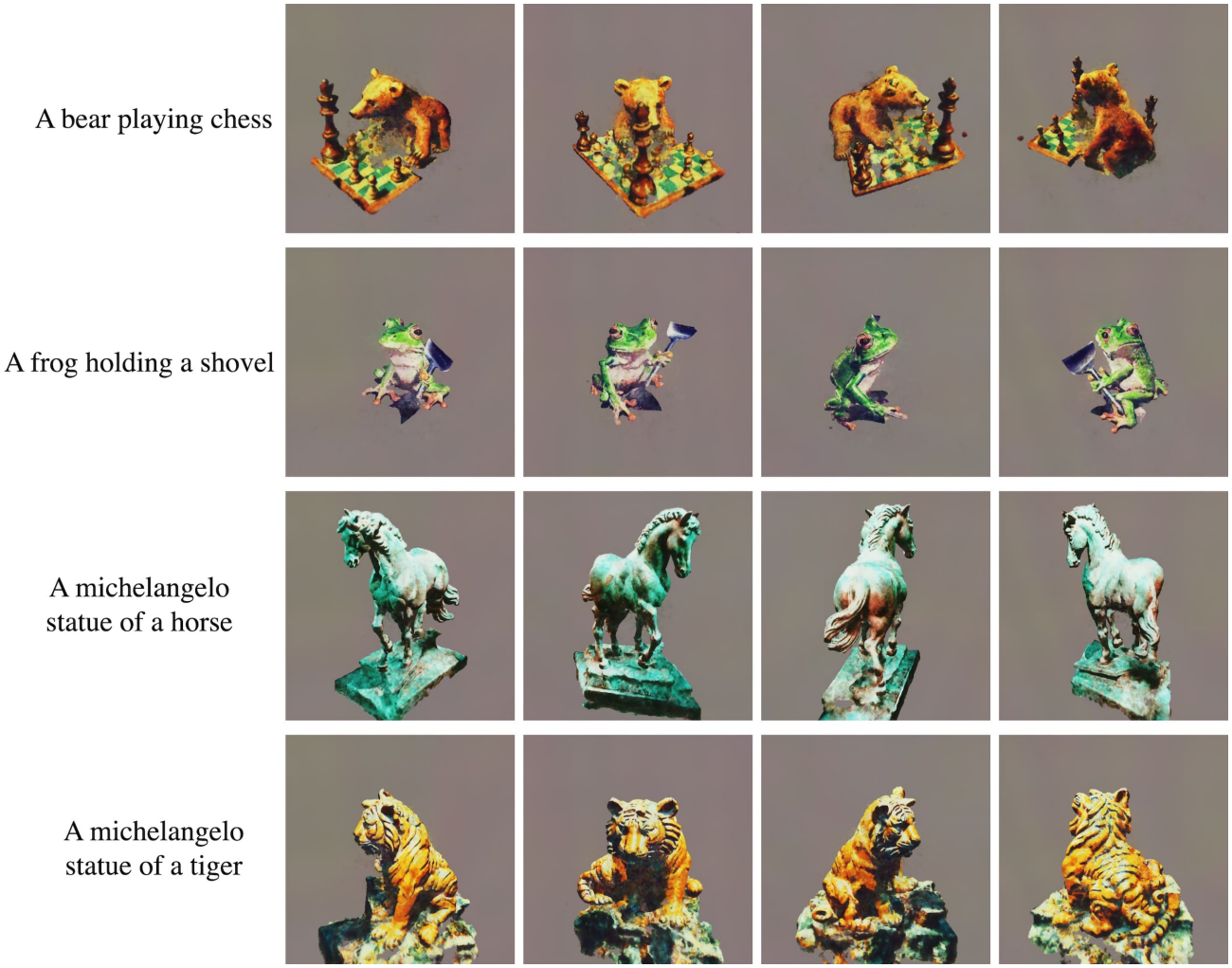

Figure 15. Renders of various scenes generated by HyperFields from various camera poses. The geometry is well formed and consistent across multiple views.

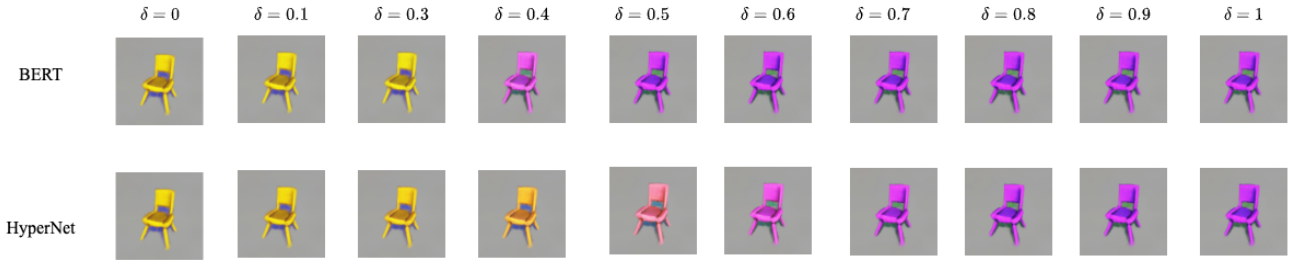

Figure 16. **BERT Token Interpolation.** We show results of interpolating the BERT tokens corresponding to the prompts "yellow chair" and "purple chair". In contrast, interpolation on the level of the hypernetwork ("HyperNet") is smoother than interpolating the BERT tokens.

$\delta - 0.3$ and $\delta = 0.5$. On the other hand, our HyperNet token interpolation shown in Figure 10 demonstrates a smooth and gradual transition of colors across the interpolation range. This demonstrates that our HyperNet learns a smoother latent space of NeRFs than the original BERT tokens correspond to.