# ProbNeRF: Uncertainty-Aware Inference of 3D Shapes from 2D Images

**Matthew D. Hoffman**[1] Google Research

**Tuan Anh Le**[1] Google Research

**Pavel Sountsov**[1] Google Research

**Christopher Suter** Google Research

**Ben Lee** Google Research

**Vikash K. Mansinghka** MIT, Google Research

**Rif A. Saurous** Google Research

## Abstract

The problem of inferring object shape from a single 2D image is underconstrained. Prior knowledge about what objects are plausible can help, but even given such prior knowledge there may still be uncertainty about the shapes of occluded parts of objects. Recently, conditional neural radiance field (NeRF) models have been developed that can learn to infer good point estimates of 3D models from single 2D images. The problem of inferring uncertainty estimates for these models has received less attention. In this work, we propose probabilistic NeRF (ProbNeRF), a model and inference strategy for learning probabilistic generative models of 3D objects' shapes and appearances, and for doing posterior inference to recover those properties from 2D images. ProbNeRF is trained as a variational autoencoder, but at test time we use Hamiltonian Monte Carlo (HMC) for inference. Given one or a few 2D images of an object (which may be partially occluded), ProbNeRF is able not only to accurately model the parts it sees, but also to propose realistic and diverse hypotheses about the parts it does not see. We show that key to the success of ProbNeRF are (i) a deterministic rendering scheme, (ii) an annealed-HMC strategy, (iii) a hypernetwork-based decoder architecture, and (iv) doing inference over a full set of NeRF weights, rather than just a low-dimensional code. Videos and code are available at `https://probnerf.github.io`.

[1]Equal contribution.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, TBD, USA. PMLR: Volume XXX. 

## 1 INTRODUCTION

Neural radiance fields (NeRFs; Mildenhall et al., 2020) are remarkably good at estimating the 3D geometry of an object from 2D images of that object. A neural network (often a simple multilayer perceptron) maps from 5D position-direction inputs to a 4D color-density output; this field is plugged into a volumetric rendering equation (Blinn, 1982) to obtain images of the field from various viewpoints, and trained to minimize the mean squared error in RGB space between the training images and their reconstructions.

This procedure works well when the training images are taken from enough viewpoints to fully constrain the geometry of the scene or object being modeled. But it fails when only one image is available; one cannot infer 3D geometry from a single 2D image without prior knowledge about what shapes are plausible. To address this, various extensions of NeRF have incorporated implicit and explicit priors, yielding impressive one- or few-shot novel-view reconstructions and/or unconditional samples (Yu et al., 2021; Kosiorek et al., 2021; Rebain et al., 2022a; Rematas et al., 2021; Dupont et al., 2022; Jang and Agapito, 2021; Wang et al., 2021; Trevithick and Yang, 2021; Chen et al., 2021).

But even with a good shape prior, there may still be uncertainty about the shape and appearance of unseen parts of the object. Although existing approaches can infer reasonable point estimates from a single image, they generally fail to account for this uncertainty.

We propose probabilistic NeRF (ProbNeRF), a system for learning priors on NeRF representations of 3D objects and for doing inference on those representations. ProbNeRF is trained using the variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014) framework, using amortized variational inference to speed up training. At test time, we use Hamiltonian Monte Carlo (Neal et al., 2011) to sample from the posterior over NeRFs that are consistent with a set of input views. Several technical contributions proved necessary to achieving high-fidelity reconstruction and robust shape uncertainty with this design:

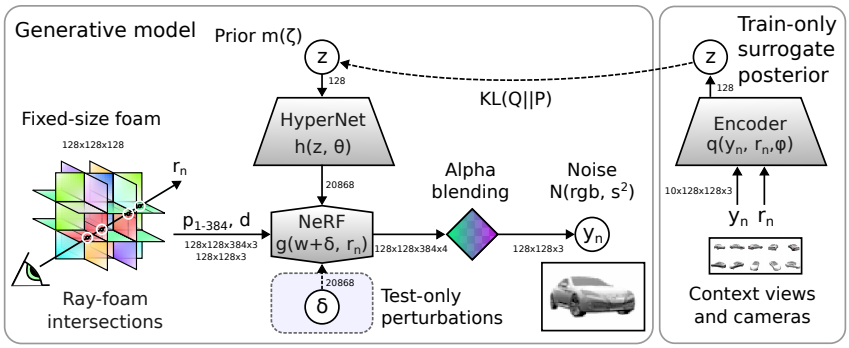


Figure 1: High-level diagram of the ProbNeRF generative process and training procedure. See Section 2 for details.

- Instead of the Monte Carlo rendering strategy employed by Mildenhall et al. (2020) and most subsequent work, we use an exact renderer based on (Chen et al., 2022). HMC works well with this renderer, but rejects nearly all proposals with the standard renderer.
- We employ a temperature-annealing strategy in our HMC sampler to make it more robust to isolated modes that arise from the non-log-concave likelihood.
- We employ a two-stage hypernetwork-based decoder, rather than a single-network strategy such as latent concatenation. This design lets us represent each object using a relatively small NeRF, which dramatically reduces per-pixel rendering costs (and therefore the cost of iterative test-time inference).
- In addition to a low-dimensional latent code, we treat the raw weights of each object's NeRF representation as random variables to be inferred. This eliminates the latent-code bottleneck from our model, allowing high-fidelity reconstruction of novel objects.

## 2 METHOD

In this section, we describe the ProbNeRF generative process, training procedure, and test-time inference procedure, as well as the neural architectures that implement them.

### 2.1 Generative Process

Let $f_w(x, v)$ be a function that, given some neural network weights $w$, a position $x \in \mathbb{R}^3$, and a viewing direction $v \in \mathbb{S}^2$, outputs a density $\sigma \in \mathbb{R}^+$ and an RGB color $c \in [0, 1]^3$. Let $g(w, r)$ be a rendering function that maps from a ray $r$ and the conditional field $f_w$ to a color $y \in [0, 1]^3$ by querying $f_w$ at various points along the ray $r$. (The renderer we use is defined in detail in Section 2.5.)

ProbNeRF assumes that, given a set of rays $r_{1:N}$ a set of pixels $y_{1:N}$ is generated by the following process: sample an abstract object code $z$ from a standard normal distribution pushed forward through an invertible RealNVP map $m$ (Dinh et al., 2017), run it through a hypernetwork to get a set of NeRF weights $w$, perturb those weights with low-variance Gaussian noise, render the resulting model, and add some pixelwise Gaussian noise. More formally,

$$\begin{aligned} \tilde{z} \sim \mathcal{N}(0, I); \quad z = m(\tilde{z}; \zeta); \quad w = h(z; \theta); \\ \delta \sim \mathcal{N}(0, I); \quad \tilde{w} = w + \sqrt{\alpha}\delta; \quad y_n \sim \mathcal{N}(g(\tilde{w}, r_n), s^2), \end{aligned} \tag{1}$$

where $m(\cdot; \zeta)$ is an invertible RealNVP (Dinh et al., 2017) function with parameters $\zeta$, $z \in \mathbb{R}^K$ is a latent code that summarizes the object's shape and appearance, $h(z; \theta)$ is a hypernetwork with parameters $\theta$ that maps from codes $z$ to NeRF weights $w$, and $\alpha$ and $s^2$ are scalar variance parameters. The generative process is summarized in Figure 1.

This generative process is similar to the one assumed by Kosiorek et al. (2021); Dupont et al. (2022): a latent code $z$ is sampled from a learned prior defined by a RealNVP, and used to index a learned family of NeRFs. There are two main differences. The first difference is architectural: we use a hypernetwork (Ha et al., 2017) to generate a full set of NeRF weights instead of concatenating the latent code $z$ to the input and activations[1]. This hypernetwork approach generalizes the latent-concatenation approach, and recent theoretical results (Galanti and Wolf, 2020) argue that hypernetworks should allow us to achieve a similar level of expressivity to the latent-concatenation strategy using a smaller architecture for $f$—intuitively, putting many parameters into a large, expressive hypernetwork makes it easier to learn a mapping to a compact function representation. This leads to large savings at both train and test time if we need to render many rays per object, since we can amortize the cost of an expensive mapping from $z$ to $w$ over hundreds or thousands of rays, each of which requires many function evaluations to render. For comparison, the NeRF architecture employed by Dupont et al. (2022) is an MLP with 15 layers of 512 hidden units, whereas in our experiments we get competitive results using a four-hidden-

[1]Dupont et al. (2022) frame their approach in terms of FiLM-style modulations (Perez et al., 2018) rather than concatenation; we show in the supplement that their latent-shift strategy is equivalent to concatenating a latent code to the activations at each layer.

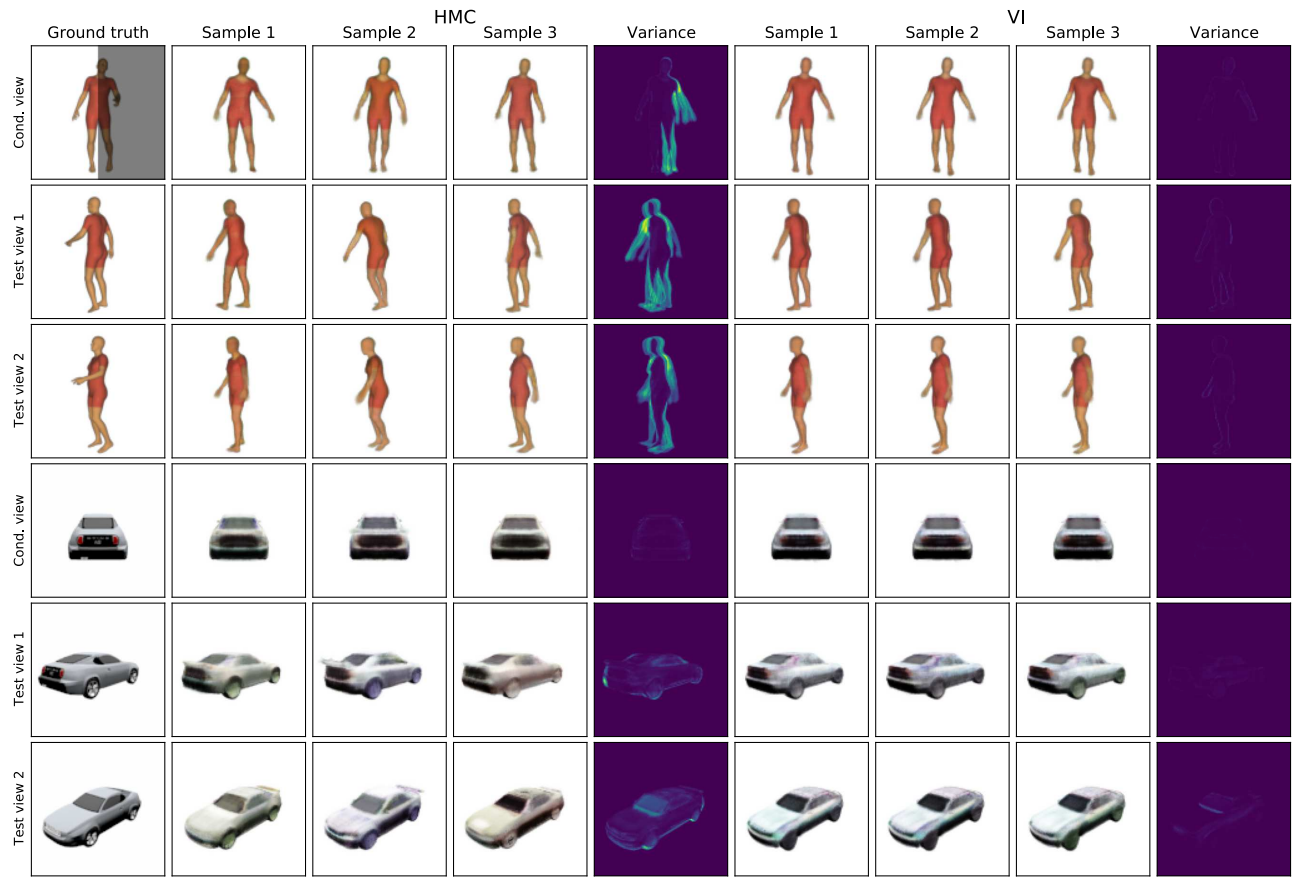


Figure 2: Conditioned on either the left half a view of a GHUM body (top left) or a back view of an SRN car (1st column, 4th row), HMC produces samples (columns 2–4) that are realistic, consistent with the conditioned-on view, and diverse as shown by the per-pixel variance (column 5). VI produces realistic and consistent samples (columns 6–8), but they have almost no diversity (last column). Renderings from novel views (rows 2–3/5–6) highlight the diversity of the HMC samples, in e.g. the poses of the left arm and the left leg or the variation of the car's shape, spoiler and color.

layer architecture with 64 hidden units—a cost savings of more than two orders of magnitude per function evaluation. Without this reduction in rendering costs, iterative MCMC methods for test-time inference would be impractical.

The second main difference between the ProbNeRF generative process and previous generative models of NeRFs is that we allow for small perturbations of the weights $w$. This is essentially a measure to address *misspecification* (cf. e.g., Kleijn and van der Vaart, 2012); it ensures that our prior on NeRFs has positive support on the full range of functions $\{f_{\tilde{w}} \mid \tilde{w} \in \mathbb{R}^D\}$, rather than the much smaller manifold of functions $\{f_w \mid w = h(z;\theta)$ for some $z \in \mathbb{R}^K\}$. We choose the variance $\alpha = 0.025^2$ on the weights to be small enough not to introduce noticeable artifacts, but large enough that the likelihood signal from a high-resolution image can overwhelm the prior preference to stay near the manifold defined by the mapping from $z$ to $w$. That way, even if the range of the hypernetwork does not include a parameter vector $w$ that accurately represents an object (for example, due to limited capacity or overfitting), the posterior $p(\tilde{w} \mid r, y)$ will still concentrate around a good set of parameters $\tilde{w}$ with more data.

### 2.2 Training Procedure

We train ProbNeRF models using a variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014) strategy, with a simplified generative process that omits the perturbation from $w$ to $\tilde{w}$:

$$\begin{aligned} \tilde{z} \sim \mathcal{N}(0, I); \quad z = m(\tilde{z}; \zeta); \quad w = h(z; \theta); \\ y_n \sim \mathcal{N}(g(w, r_n), s^2). \end{aligned} \tag{2}$$

We omit these perturbations at training time to force the model to learn hypernet parameters $\theta$ and RealNVP parameters $\zeta$ that can explain the training data well without relying on perturbations. The perturbations $\delta$ are intended to allow the model as an inference-time "last resort" to explain factors of variation that were not in the training set; at training time we do not want $\delta$ to explain away variations

that could be explained using $z$, since the model lacks a mechanism to learn a meaningful prior on $\delta$.

To compute a variational approximation $q(z \mid y, r)$ to the posterior $p(z \mid y, r)$, we use a convolutional neural network (CNN; LeCun and Bengio, 1998) to map from each RGB image and camera matrix to a diagonal-covariance $K$-dimensional Gaussian potential, parameterized as locations $\mu_j$ and precisions $\tau_j$ for the $j$th image; these potentials are meant to approximate the influence of the likelihood function on the posterior (Johnson et al., 2016; Sønderby et al., 2016). We combine these $J$ potentials with a learned "prior" potential parameterized by location $\mu_0$ and precisions $\tau_0$ via the Gaussian update formulas

$$\hat{\tau} = \textstyle\sum_{j=0}^{J} \tau_j; \quad \hat{\mu} = \hat{\tau}^{-1} \sum_{j=0}^{J} \tau_j \mu_j \tag{3}$$

and set $q(z_k \mid y, r) = \mathcal{N}(z_k; \hat{\mu}_k, \hat{\tau}_k^{-1})$.

We train the encoder parameters $\phi$, the hypernet parameters $\theta$, and the RealNVP parameters $\zeta$ by maximizing the evidence lower bound (ELBO) using Adam (Kingma and Ba, 2015):

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_q[\log p(y \mid z, r) - \log \tfrac{q(z|y,r)}{p(z)}] \\ &= \mathbb{E}_q\left[\log p(y \mid z, r) - \log \frac{q(z|y,r)}{\mathcal{N}(m^{-1}(z); 0, I)\left|\frac{dm^{-1}}{dz}\right|}\right].\end{aligned} \tag{4}$$

We train on minibatches of 8 objects and 10 randomly selected images per object to give the encoder enough information to infer a good latent code $z$. The encoder sees all 10 images, but to reduce rendering costs we compute an unbiased estimate of the log-likelihood $\log p(y \mid z, r)$ from a random subsample of 1024 rays per object.

### 2.3 Architectures

For each object's NeRF, we use two MLPs, each with two hidden layers of width 64. The first MLP maps from position to density; the second maps from position, view direction, and density to color. All positions and view directions are first transformed using a 10th-order sinusoidal encoding (Mildenhall et al., 2020). The number of parameters per object is 20,868, relatively few for a NeRF.

The RealNVP network that implements the mapping from $\tilde{z}$ to $z$ comprises two pairs of coupling layers. Each coupling layer is implemented as an MLP with one 512-unit hidden layer that shifts and rescales half of the variables conditioned on the other half; each pair of coupling layers updates a complementary set of variables. The variables are randomly permuted after each pair of coupling layers.

The hypernetwork that maps from the 128-dimensional code $z$ to the 20,868 NeRF weights is a two-layer 512-hidden-unit MLP. This mapping uses a similar number of FLOPs to rendering a few pixels (see Section 2.5).

The encoder network applies a 5-layer CNN to each image and a two-layer MLP to its camera-world matrix, then linearly maps the concatenated the image and camera activations to locations and log-scales for each image's Gaussian potential. (Full architectural details in supplement.)

All networks use ReLU nonlinearities.

### 2.4 Test-Time Inference

The training procedure outlined above is able to learn a good RealNVP prior on codes and to reconstruct training-set examples accurately. However, we found that the trained encoder generalizes poorly to held-out examples—it is useful scaffolding for training the model, but fails to accurately reconstruct objects it was not trained on. Furthermore, variational inference is well known to underestimate posterior uncertainty (e.g., Yao et al., 2018), and one of our primary goals is to capture uncertainty about object shape and appearance.

As an alternative, at inference time we turn to Hamiltonian Monte Carlo (HMC; Neal et al., 2011), a gradient-based Markov chain Monte Carlo (MCMC) method that uses momentum to mitigate poor conditioning of the target log-density function. Rather than sample in $z, \tilde{w}$ space, we use the noncentered parameterization and sample from $p(\tilde{z}, \delta \mid y, r)$ (Betancourt and Girolami, 2015), since the joint prior for $\tilde{z}$ and $\delta$ is a well-behaved spherical normal. (Note that only the prior's contribution to the posterior is simple; the likelihood still makes things difficult.)

HMC is a powerful MCMC algorithm, but it can still get trapped in isolated modes of the posterior. Running multiple chains in parallel can provide samples from multiple modes, but it may be that some chains find (but cannot escape from) modes that have negligible mass under the posterior. A conditioning problem also arises in inverse problems where some degrees of freedom are poorly constrained by the likelihood: as the level of observation noise decreases it becomes necessary to use a smaller step size, but the distance in the latent space between independent samples may stay almost constant (Langmore et al., 2021).

To make our sampling procedure more robust to minor modes and poor conditioning, we use a temperature-annealing strategy (e.g., Kirkpatrick et al., 1983; Neal, 2001). Over the course of $T$ HMC iterations, we reduce the observation-noise scale $s$ logarithmically from a high initial value $s_0$ to a low final value $s_T$, with $s_t = s_0^{(T-t)/T} s_T^{t/T}$ (for a Gaussian likelihood, this is equivalent to annealing the "temperature" of the likelihood). That is, we start out targeting a distribution that is close to the prior, and gradually increase the influence of the likelihood until we are targeting the posterior. We also anneal the step size so that it is proportional to $s_t$. This procedure lets the sampler explore the latent space thoroughly at higher temper-

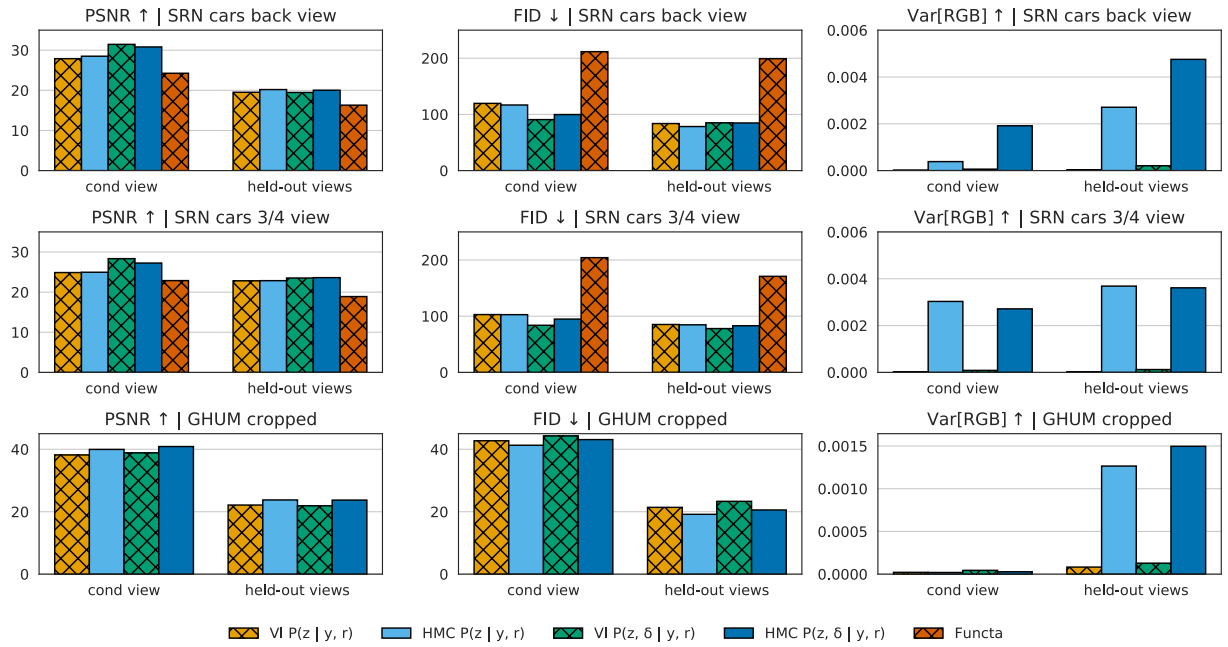


Figure 3: Posterior-predictive metrics for VI and HMC on the ProbNeRF model and MAP inference on Functa (where appropriate) on held-out data for SRN Cars and GHUM. For SRN cars, the top two rows condition on a uninformative back view and an informative 3/4 view respectively, and evaluate on 3 held-out views. For GHUM the bottom row is conditioned on the left-half crop of a front view, evaluated on 4 held-out views. We compare VI and HMC on full ProbNeRF ($p(\tilde{z}, \delta \mid y, r)$) and an ablation where the $\delta$ is omitted ($p(\tilde{z} \mid y, r, \delta = 0)$). The first column shows reconstruction quality in PSNR. The middle column is FID score, measuring whether reconstructions look like test examples from the corresponding view(s). The last column shows the variance of RGB values in the reconstructed images, measuring posterior diversity.

atures before settling into a state that achieves low reconstruction error. In Section 3.3.2, we show that this annealing procedure yields more-consistent results than running HMC at a low fixed temperature.

### 2.5 Exact Rendering

NeRFs (Mildenhall et al., 2020) generally employ a stochastic quadrature approximation of the rendering integral. Although this procedure is deterministic at test time, we have found empirically that its gradients are not reliable enough to use in HMC (see Section 3.3.1).

While stochastic-gradient methods are robust to the noise from this procedure, standard HMC methods are not (Betancourt, 2015). Stochastic-gradient HMC methods do exist (Ma et al., 2015), but require omitting the Metropolis correction, which perturbs the stationary distribution unless one uses a small step size and/or can accurately estimate the (high-dimensional) covariance of the gradient noise.

Instead, we employ a simplified version of the renderer used in Chen et al. (2022). We assume all density is concentrated in a "foam" consisting of the surfaces of a 128x128x128 lattice of cubes. Since there is no density inside the cubes, we can render a ray by enumerating all ray-cube intersection points, computing opacities and colors at each intersection, and alpha-compositing the result. This simplification avoids the need to map the latent code to grid vertices as in Chen et al. (2022). Rendering a ray requires at most $128 \times 3 = 384$ function evaluations (*not* $128^3$). In Section 3.3.1 we show that this renderer works well with HMC, while HMC with the standard quadrature scheme cannot achieve high acceptance rates.

## 3 EXPERIMENTS

In this section, we qualitatively and quantitatively evaluate ProbNeRF's ability to generate realistic-looking conditional and unconditional samples, as measured by FID score (Heusel et al., 2017); accurately reconstruct the views we condition on; and, given a single image, generate diverse and plausible hypotheses about what an object looks like from other views. We also apply a set of ablations to demonstrate the value of using an exact renderer when doing MCMC, using temperature annealing in our HMC procedure, and doing posterior inference over the raw NeRF weights as well as the latent code.

We evaluate ProbNeRF on two datasets: SRN Cars (Sitzmann et al., 2019), and a set of renderings from the GHUM

generative model (Xu et al., 2020). We use the standard SRN Cars dataset with 2458 train cars and 704 test cars. GHUM is a generative model fit on a dataset of 60,000 photo-realistic body scans that comprises of normally distributed latent codes corresponding to the facial expression, body pose, and body shape which are decoded into a 3D mesh which can be rendered given a camera pose. To isolate the representation of uncertainty over body poses, we construct a dataset by sampling a code for body pose while setting the facial expression and body shape codes to zero. We generate 2500 training examples, each containing 50 views from cameras pointing to the middle of the body with poses sampled uniformly from a fixed-radius circle around the body, parallel to the ground plane and elevated to the middle of the body. We generate 100 test examples, each containing 50 views in a similar way except the camera poses are equally spaced points on the circle. All images have resolution $128 \times 128$. All models were trained for one million iterations with an observation noise scale $s = 0.1$.

### 3.1 Posterior Inference

We want ProbNeRF to accurately reconstruct the view it is conditioned on and produce realistic and diverse reconstructions of held out views. To quantitatively assess this, we measure reconstruction quality using the PSNR metric (Wang et al., 2004), realism using the FID score (Heusel et al., 2017), and diversity using average per-pixel variance.

We compare ProbNeRF with baseline methods along the inference and the modeling axes. For inference, we compare ProbNeRF's HMC inference procedure with an iterative mean-field variational inference (VI) procedure fit using Adam and sticking-the-landing gradients (Roeder et al., 2017). For modeling, we compare ProbNeRF with "Functa" (Dupont et al., 2022) which proposes separately extracting a compressed representation—a functum—for each training example and doing probabilistic modeling and inference on top of the functa.

To compare between HMC and VI, we use the same trained ProbNeRF model and for each test example pick a conditioning view $(y, r)$ and a set of $H = 4$ held out views $\{(y'_h, r'_h)\}_{h=1}^H$. For both HMC and VI, we produce a set of $L$ samples $\{(\tilde{z}_\ell, \delta_\ell)\}_{\ell=1}^L$ targeting the single view posterior $p(\tilde{z}, \delta | y, r)$ which are used to produce renderings from the conditioning view $y_\ell = g(\tilde{w}_\ell, r)$ and from the held-out view $y'^{\ell}_h = g(\tilde{w}_\ell, r'_h)$. We chose a perturbation variance $\alpha$ of $0.025^2$ by rendering perturbed weights with different amounts of Gaussian noise and choosing the largest variance that did not produce significant artifacts.

In HMC, we obtain samples by running 8 independent chains with the annealing procedure described in Section 2.4 and taking the last 16 samples in each chain. In all experiments, we run annealed HMC for $T = 100$ steps with 100 leapfrog steps per HMC step, for a total of 10,000 gradient evaluations. The noise scale $s$ is annealed from $s_0 = 5$ to $s_T = 0.1$ In VI, we approximate the posterior using an independent normal distribution for each element of the latent variable $(\tilde{z}, \delta)$ parameterized by the location $\mu$ and log scale $\log \sigma$. We maximize the evidence lower bound with respect to $(\mu, \log \sigma)$ using 1,500 gradient steps and generate 16 samples using the final parameters. All experiments were run on 8 TPU cores; each HMC sampling run took about 87 minutes, and each VI run took about 12 minutes. Reducing these times was out of scope for this paper, but there are many ways we might speed things up that could apply to both HMC and VI, such as subsampling pixels, annealing from a coarse rendering grid to a finer one, learning a preconditioner, or using some of the myriad strategies that have been proposed in the NeRF literature.

We find that while both HMC and VI produce high-quality (high PSNR) and realistic (low FID) reconstructions, HMC produces significantly more-diverse held-out reconstructions (higher mean per-pixel variance) for both SRN Cars and GHUM (Fig. 3). This is qualitatively illustrated in Fig. 2 where we show three samples for each HMC and VI rendered from the conditioning view and two held out views alongside the per-pixel variance shown as a heatmap (more HMC samples can be found in the supplement). HMC samples show diversity in the pose of the left arm and the left leg in the GHUM example and in the variation of the spoiler, bumper and taillights in the SRN Cars example. VI samples show almost no diversity.

To compare between ProbNeRF and Functa, we train a Functa model on SRN Cars using the open-sourced code to obtain a functaset of training-set modulations. We fit the RealNVP prior used for our model instead of the neural spline flow (Durkan et al., 2019) used by Dupont et al. (2022), since the prior-fitting code is not released. (Dupont et al. (2022) also fit a denoising diffusion probabilistic model (DDPM; Ho et al., 2020), but they only use it for sample generation, not for MAP inference, since it is very expensive to evaluate the density of a DDPM.) We note that this may be a source of difference in our reproduction. For each test example, we obtain a modulation code by performing a 1000-step gradient-based MAP search given the observed view $(y, r)$ as per equation 2 in (Dupont et al., 2022). These modulation codes are then rendered for the conditioned view and the held-out views.

We find that Functa produces less accurate, less realistic, and less diverse reconstructions on both conditioned on and held-out views than ProbNeRF (Fig. 3).

To compare against other novel-object-inference baselines from the literature, we adopted the standard SRN Cars one-shot evaluation setup: we condition on an informative 3/4 view, estimate the posterior-predictive mean rendering of each of 250 other views from the dataset using the HMC samples, and compute PSNR and SSIM metrics (higher

| | ProbNeRF | PixelNeRF | CodeNeRF |
|---|---|---|---|
| PSNR ↑ | 23.00 | 23.17 | 23.80 |
| SSIM ↑ | 0.89 | 0.90 | 0.91 |

Table 1: Comparison to one-shot SRN Cars results from the literature.

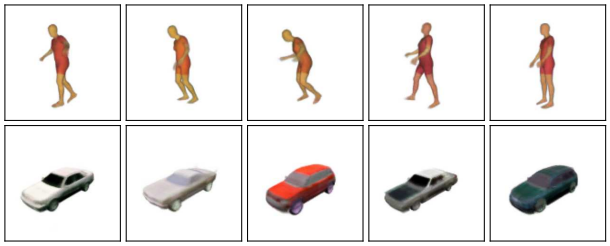

Figure 4: Unconditional GHUM and SRN Cars samples.

| | ProbNeRF | Functa | Functa (DDPM) | $\pi$-GAN |
|---|---|---|---|---|
| FID ↓ | 84.6 | 158.1 | 80.3 | 36.7 |

Table 2: FID for SRN Cars prior samples.

is better for both). Table 1 shows the results compared with PixelNeRF (Yu et al., 2021) and CodeNeRF (Jang and Agapito, 2021). ProbNeRF can produce point estimates that are comparable to recent methods. But we emphasize that these baselines can only produce point estimates, whereas ProbNeRF produces a diverse set of hypotheses.

### 3.2 Generative Modeling

Our unconditional samples look realistic for both GHUM and SRN Cars (Fig. 4; more in the supplementary material). SRN Cars samples are on par with Functa in FID when it is trained using a denoising diffusion probabilistic model (DDPM) prior (Ho et al., 2020) (Table 2). Both ProbNeRF and Functa have a worse FID than $\pi$-GAN (Chan et al., 2021) which focuses only on generation and cannot be trivially extended to do uncertainty-aware inference. We hypothesize that the gap between our retrained Functa model and the results reported by Dupont et al. (2022) are due to the lower expressivity of the RealNVP prior compared to a DDPM prior (although DDPMs are much more expensive to sample from and do posterior inference with).

### 3.3 Ablations

#### 3.3.1 Exact Renderer

To demonstrate the difficulty of doing HMC with the approximate quadrature-based volumetric renderer (Mildenhall et al., 2020) versus the exact foam renderer (Section 2.5), we first trained models using the quadrature- and foam-based renderers on SRN cars. We evaluate HMC on each of these models using the appropriate renderer as follows. We sample a fixed $\tilde{z}, \delta$ pair from the prior for each model and render a single view $y$ of the resulting NeRF; conditioned on this view, $\tilde{z}, \delta$ is a sample from the posterior $p(\tilde{z}, \delta \mid y, r)$. For each of a variety of step sizes, we then run 8 HMC chains with 10 leapfrog steps initialized from this sample and targeting $p(\tilde{z}, \delta \mid y, r)$ for 20 iterations, and report the average Metropolis acceptance rate across the chains on the last iteration. HMC sampling using the quadrature renderer suffers from low acceptance rates even with tiny step sizes, while the foam renderer yields high acceptance rates for small-enough step sizes (Fig. 5).

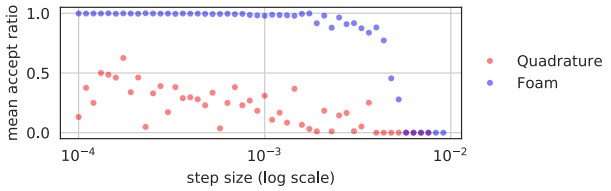


Figure 5: HMC acceptance rates for quadrature and foam renderers.

#### 3.3.2 Temperature Annealing

In this section we qualitatively demonstrate the value of our annealed-HMC strategy. Conditioning on the rear view of an ambulance, we ran HMC with annealing as described in Section 2.4, and compare with HMC initialized in the same way but without annealing and with a fixed step size of 0.0005 (we found it necessary to use a lower step size for fixed-temperature HMC to ensure reasonable acceptance rates across all chains). We ran both methods with 8 parallel chains. Fig. 6 shows the last samples from each method.

The annealed-HMC procedure's samples are both more consistent and more faithful to the ground truth. This result is consistent with the hypothesis that the annealing procedure allows HMC to avoid low-mass modes of the posterior and focus on more-plausible explanations of the data.

#### 3.3.3 Inference Over Raw NeRF Weights

In addition to running HMC on ProbNeRF targeting both the latent code and the raw NeRF weights, $p(\tilde{z}, \delta | y, r)$, we run HMC on ProbNeRF targeting only the latent code, $p(\tilde{z}|y, r)$. Like in Section 3.1, we obtain $L = 8$ samples by running $L$ independent chains. We pick the 16 samples from each chain and render reconstructions from the conditioning view and $H = 4$ held-out views.

We show that performing inference over the raw NeRF weights significantly increases the quality (higher PSNR) and realism (lower FID) of the conditioned-on view reconstruction while not having negative effects on held-out view reconstruction performance (Fig. 3). Further, when condi-

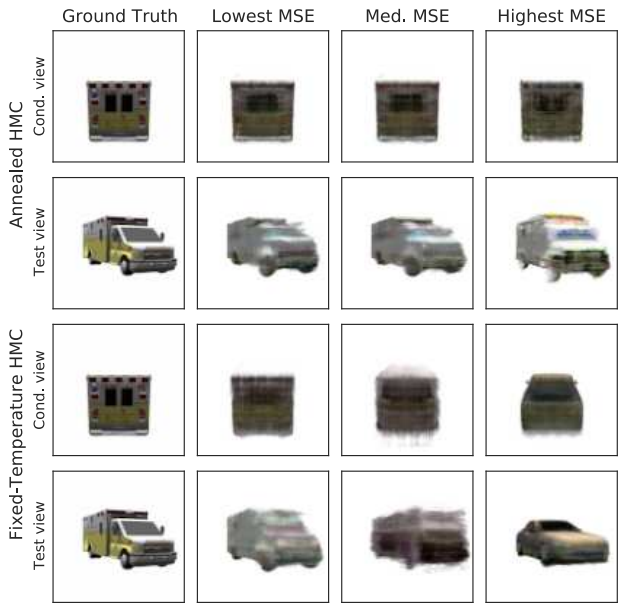


Figure 6: Renderings of samples generated by annealed and fixed-temperature HMC conditioned on an image of the rear of an ambulance (columns 2–4) and corresponding ground-truth images (column 1). Rows 1 and 3 show the conditioned-on view, rows 2 and 4 show a held-out view. Samples are selected to have the lowest (column 2), 4th-lowest (column 3), or highest (column 4) squared-error reconstruction of the conditioned-on view. Samples from all 8 chains are in the supplement.

tioning on high-information views, the quality and realism of held-out views improves relative to when we only infer the latent code. This supports our hypothesis that adding raw NeRF weights as latent variables increases the support with a positive prior over the radiance fields which lets our system adapt to novel views given sufficiently informative observations. Importantly, despite doing test-time inference in a model that is different from the model used during training does not harm performance on reconstruction of far away held-out views or prior sampling. For GHUM, the discrepancy between ProbNeRF and latent-only ProbNeRF conditioned-on view reconstruction quality is not as large, which we attribute to (i) a smaller discrepancy between training and test distributions and (ii) both models being able to fit the training distribution well.

## 4 RELATED WORK

**Neural fields for novel-scene inference.** While classic NeRFs (Mildenhall et al., 2020) are only fit on single scenes, there have been many recent extensions allowing novel scene or novel object inference. CodeNeRF (Jang and Agapito, 2021) and LOLNeRF (Rebain et al., 2022a) condition neural fields by concatenating inputs with per-scene latent codes, and learn priors that are able to generate coherent 3D geometry. PixelNeRF (Yu et al., 2021), IBRNet (Wang et al., 2021), GRF (Trevithick and Yang, 2021), and MVSNeRF (Chen et al., 2021) exploit the geometry of the conditioning view, known as "local conditioning", to inform novel-view reconstructions. As noted by Sajjadi et al. (2022), local conditioning generalizes less well to views far from observations, and is more computationally expensive. Sajjadi et al. (2022) explore using attention-based mechanisms alongside a set-based latent representation which Rebain et al. (2022b) observes to be generally superior to concatenation or hypernetwork-based methods. However, hypernetworks are shown to perform nearly as well as attention mechanisms, and attention is expensive, especially for iterative posterior inference (Kosiorek et al., 2021). Scene representation networks (Sitzmann et al., 2019) also use hypernetwork conditioning, although on a different neural representation. Similar to ShaRF (Rematas et al., 2021), we also find that test-time inference of NeRF weights alongside latent codes improves reconstructions, especially when input images are highly informative.

**Probabilistic neural scene representations.** Like ProbNeRF, these methods go beyond single point estimates and can represent multiple plausible scenes consistent with the potentially low-information views. Generative Query Networks (Eslami et al., 2018; Rosenbaum et al., 2018) also train a VAE system but use a convolutional decoder which doesn't enforce multi-view consistency. NeRF-VAE (Kosiorek et al., 2021) also combines VAEs and NeRFs, but relies on latent-concatenation conditioning and amortized VI, resulting in low-diversity novel-view reconstructions. Concurrently with our work, Anonymous (2023) extends NeRF-VAE to produce larger posterior diversity by using normalizing flows, attention, and set-based latent representations; we were unable to evaluate it at the time of submission. GAUDI (Bautista et al., 2022) fits a diffusion-based conditional generative model that can sample diverse and plausible large-scale real world scenes given few observed views, although the conditioning mechanism must be trained from paired data, whereas ProbNeRF can condition on arbitrary sets of pixels and camera positions. 3DiM (Watson et al., 2022) is a diffusion-based image-to-image model that can synthesize diverse novel views of scenes, but it does not guarantee multi-view consistency. Shen et al. (2022) propose conditional-flow NeRF (CF-NeRF), a method that uses variational inference to quantify uncertainty in a NeRF representation. The most important difference between ProbNeRF and CF-NeRF is that ProbNeRF learns a prior over objects, while CF-NeRF uses a default uniform prior. Inferring the shapes of objects from single and/or occluded views is underconstrained without an informative prior, so CF-NeRF is fit to 4–5 informative views. Giving the model so much information about the scene leaves relatively little epistemic uncertainty (e.g., where precisely an edge is), whereas ProbNeRF can cope

with more extreme uncertainty about shape, pose, and appearance.

**Other NeRF generative models.** Schwarz et al. (2020); Niemeyer and Geiger (2021); Chan et al. (2021) train NeRF generative models using a discriminator loss from generative adversarial nets (GANs) (Goodfellow et al., 2014) which produce high-quality, diverse samples. However, GAN-style training often results in models that struggle at reconstruction (Wu et al., 2017). DreamFusion (Poole et al., 2022) generates impressive NeRF scenes given a text prompt, but also does not focus on inference from images.

**Exact rendering.** DIVeR (Wu et al., 2021) introduces a deterministic and exact renderer based on integrating trilinearly interpolated features exactly on a voxel grid. This requires four times as many function evaluations or table lookups per intersected voxel as the MobileNeRF strategy we adapt (Chen et al., 2022).

**Probabilistic programming for computer vision.** Although several probabilistic programming approaches to computer vision use test-time Monte Carlo inference (Mansinghka et al., 2013; Le et al., 2017; Kulkarni et al., 2015; Gothoskar et al., 2021), they mainly focus on finding one probable scene interpretation per 2D image (though Mansinghka et al. (2013) demonstrate limited domain-specific uncertainty reporting on a restricted class of road scenes). In contrast, ProbNeRF characterizes shape and appearance uncertainty for open-ended classes of 3D objects.

## 5 DISCUSSION

Given a single low-information view of a novel object, ProbNeRF can not only produce reasonable point estimates of that object's shape and appearance, but can also guess what range of shapes and appearances are consistent with the available data. Making these sorts of diverse (but coherent) guesses about unseen features of objects is a fundamental problem in in vision. ProbNeRF shows that it is possible to simultaneously achieve high-fidelity reconstruction and robust characterization of uncertainty within the NeRF framework. One next step for future research could be to quantify tradeoffs between model fidelity, inference efficiency, and uncertainty characterization, to support variations on ProbNeRF suitable for real-time applications in robotics. More broadly, we hope ProbNeRF encourages more research at the interface of Bayesian inference, 3D graphics, and computer vision, enabling computer vision systems that entertain diverse hypotheses about the world.

### Acknowledgments

Thanks to Katie Colins, Varun Jampani, Adam Kosiorek, Despoina Paschalidou, and Sharad Vikram for their helpful comments, and to Alex Alemi, Kevin Murphy, and Andrea Tagliasacchi for their timely and helpful feedback on the manuscript.

### References

Anonymous (2023). Laser: Latent set representations for 3d generative modeling. In *Submitted to The Eleventh International Conference on Learning Representations*. under review.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*.

Bautista, M. A., Guo, P., Abnar, S., Talbott, W., Toshev, A., Chen, Z., Dinh, L., Zhai, S., Goh, H., Ulbricht, D., et al. (2022). Gaudi: A neural architect for immersive 3d scene generation. *arXiv preprint arXiv:2207.13751*.

Betancourt, M. (2015). The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 533–540.

Betancourt, M. and Girolami, M. (2015). Hamiltonian monte carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79(30):2–4.

Blinn, J. F. (1982). Light reflection functions for simulation of clouds and dusty surfaces. *SIGGRAPH Comput. Graph.*, 16(3):21–29.

Chan, E. R., Monteiro, M., Kellnhofer, P., Wu, J., and Wetzstein, G. (2021). pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809.

Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., and Su, H. (2021). Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133.

Chen, Z., Funkhouser, T., Hedman, P., and Tagliasacchi, A. (2022). Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. In *5th International Conference on Learning Representations*.

Dupont, E., Kim, H., Eslami, S. A., Rezende, D. J., and Rosenbaum, D. (2022). From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, pages 5694–5725. PMLR.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. *Advances in neural information processing systems*, 32.

Eslami, S. A., Jimenez Rezende, D., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A.,

Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science*, 360(6394):1204–1210.

Galanti, T. and Wolf, L. (2020). On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33:10409–10419.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *NeurIPS*.

Gothoskar, N., Cusumano-Towner, M., Zinberg, B., Ghavamizadeh, M., Pollok, F., Garrett, A., Tenenbaum, J., Gutfreund, D., and Mansinghka, V. (2021). 3dp3: 3d scene perception via probabilistic programming. *Advances in Neural Information Processing Systems*, 34:9600–9612.

Ha, D., Dai, A. M., and Le, Q. V. (2017). Hypernetworks. In *International Conference on Learning Representations*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Jang, W. and Agapito, L. (2021). Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958.

Johnson, M. J., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. (2016). Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems*, 29.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations*.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kleijn, B. J. and van der Vaart, A. W. (2012). The bernstein-von-mises theorem under misspecification. *Electronic Journal of Statistics*, 6:354–381.

Kosiorek, A. R., Strathmann, H., Zoran, D., Moreno, P., Schneider, R., Mokrá, S., and Rezende, D. J. (2021). Nerf-vae: A geometry aware 3d scene generative model. In *International Conference on Machine Learning*, pages 5742–5752. PMLR.

Kulkarni, T. D., Kohli, P., Tenenbaum, J. B., and Mansinghka, V. (2015). Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399.

Langmore, I., Dikovsky, M., Geraedts, S., Norgaard, P., and von Behren, R. (2021). Hamiltonian monte carlo in inverse problems; ill-conditioning and multi-modality. *arXiv preprint arXiv:2103.07515*.

Le, T. A., Baydin, A. G., and Wood, F. (2017). Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*, pages 1338–1348. PMLR.

LeCun, Y. and Bengio, Y. (1998). Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks*, pages 255–258.

Ma, Y.-A., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28.

Mansinghka, V. K., Kulkarni, T. D., Perov, Y. N., and Tenenbaum, J. (2013). Approximate bayesian image interpretation using generative probabilistic graphics programs. *Advances in Neural Information Processing Systems*, 26.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing*, 11(2):125–139.

Neal, R. M. et al. (2011). MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Niemeyer, M. and Geiger, A. (2021). Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. (2018). Film: Visual reasoning with a general conditioning layer. In *AAAI*.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. (2022). Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

Rebain, D., Matthews, M., Yi, K. M., Lagun, D., and Tagliasacchi, A. (2022a). Lolnerf: Learn from one look. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1558–1567.

Rebain, D., Matthews, M. J., Yi, K. M., Sharma, G., Lagun, D., and Tagliasacchi, A. (2022b). Attention beats concatenation for conditioning neural fields. *arXiv preprint arXiv:2209.10684*.

Rematas, K., Martin-Brualla, R., and Ferrari, V. (2021). Sharf: Shape-conditioned radiance fields from a single view. In *International Conference on Machine Learning*, pages 8948–8958. PMLR.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286.

Roeder, G., Wu, Y., and Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *Advances in Neural Information Processing Systems*, 30.

Rosenbaum, D., Besse, F., Viola, F., Rezende, D. J., and Eslami, S. M. A. (2018). Learning models for visual 3d localization with implicit mapping.

Sajjadi, M. S., Meyer, H., Pot, E., Bergmann, U., Greff, K., Radwan, N., Vora, S., Lučić, M., Duckworth, D., Dosovitskiy, A., et al. (2022). Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238.

Schwarz, K., Liao, Y., Niemeyer, M., and Geiger, A. (2020). Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166.

Shen, J., Agudo, A., Moreno-Noguer, F., and Ruiz, A. (2022). Conditional-flow nerf: Accurate 3d modelling with reliable uncertainty quantification. In *Proceedings of the European Conference on Computer Vision*, page 540–557, Berlin, Heidelberg. Springer-Verlag.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. (2019). Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. *Advances in neural information processing systems*, 29.

Trevithick, A. and Yang, B. (2021). Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192.

Wang, Q., Wang, Z., Genova, K., Srinivasan, P. P., Zhou, H., Barron, J. T., Martin-Brualla, R., Snavely, N., and Funkhouser, T. (2021). Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Watson, D., Chan, W., Martin-Brualla, R., Ho, J., Tagliasacchi, A., and Norouzi, M. (2022). Novel view synthesis with diffusion models.

Wu, L., Lee, J. Y., Bhattad, A., Wang, Y., and Forsyth, D. (2021). DIVeR: Real-time and accurate neural radiance fields with deterministic integration for volume rendering.

Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. (2017). On the quantitative analysis of decoder-based generative models. In *International Conference on Learning Representations*.

Xu, H., Bazavan, E. G., Zanfir, A., Freeman, W. T., Sukthankar, R., and Sminchisescu, C. (2020). Ghum & ghuml: Generative 3d human shape and articulated pose models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6184–6193.

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018). Yes, but did it work?: Evaluating variational inference. In *International Conference on Machine Learning*, pages 5581–5590. PMLR.

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. (2021). pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587.

## Appendix A Additional figures

Figures 7, 8, 9, and 10 show eight uncurated samples from HMC and VI for GHUM and SRN Cars (cf. Fig. 2). Figures 11, 12, 13, and 14 show eight uncurated HMC and VI samples generated by conditioning on uncropped single-view images of two SRN cars. Both HMC and VI produce models that are realistic and consistent with the observed data, but samples from the variational distribution obtained by VI are all essentially the same, while HMC produces samples with noticeable diversity in pose (for GHUM) and shape and color (for SRN Cars).

Figures 15 and 16 show uncurated unconditional samples from the trained models.

Fig. 17 shows uncurated samples conditioned on the rear view of an ambulance (cf. Fig. 6) generated using HMC with temperature annealing and with a fixed temperature. Annealed HMC consistently finds solutions that are consistent with the conditioned-on view; fixed-temperature HMC does not.

Fig. 18 is a version of Fig. 2 with high-resolution (256x256 pixels) renderings. Some aliasing is noticeable for GHUM; using a finer grid for rendering does not eliminate it. This might be alleviated by either using ideas from MipNeRF (Barron et al., 2022) or jittering camera distances at training time.

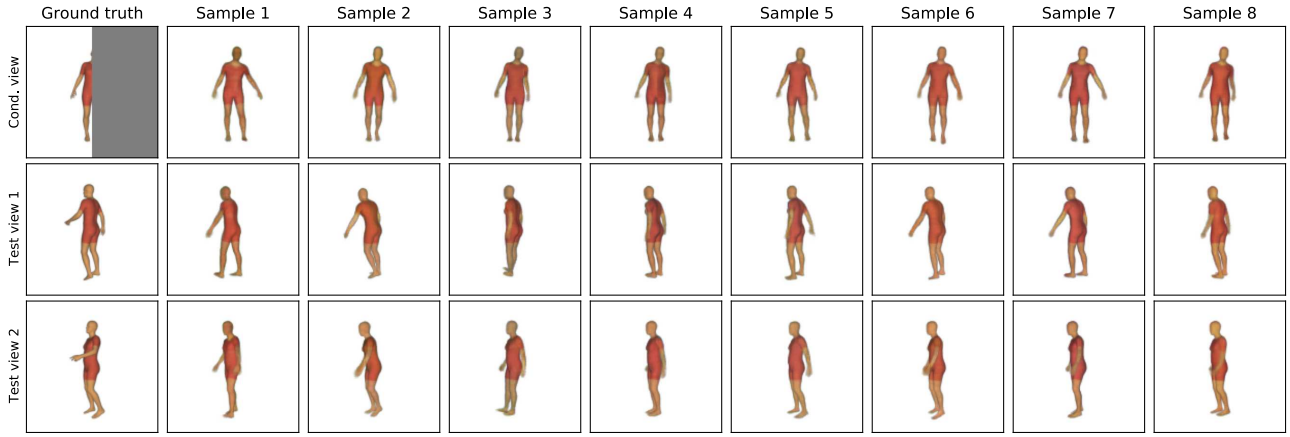


Figure 7: Uncurated GHUM HMC samples.

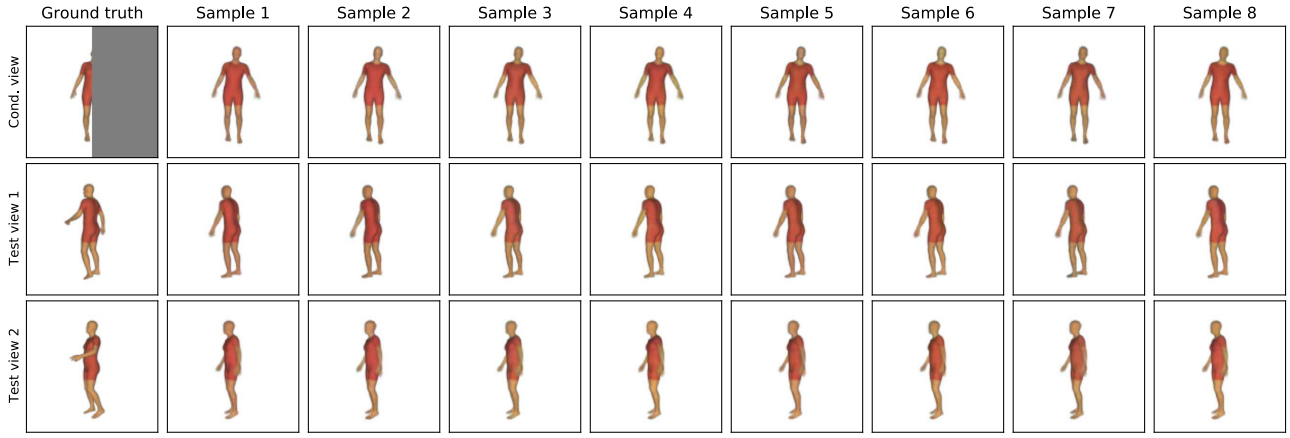


Figure 8: Uncurated GHUM VI samples.

## Appendix B Architecture details

In this section we briefly describe the architectural details of ProbNerf neural nets. Fig. 19 contains the architectures used in this work.

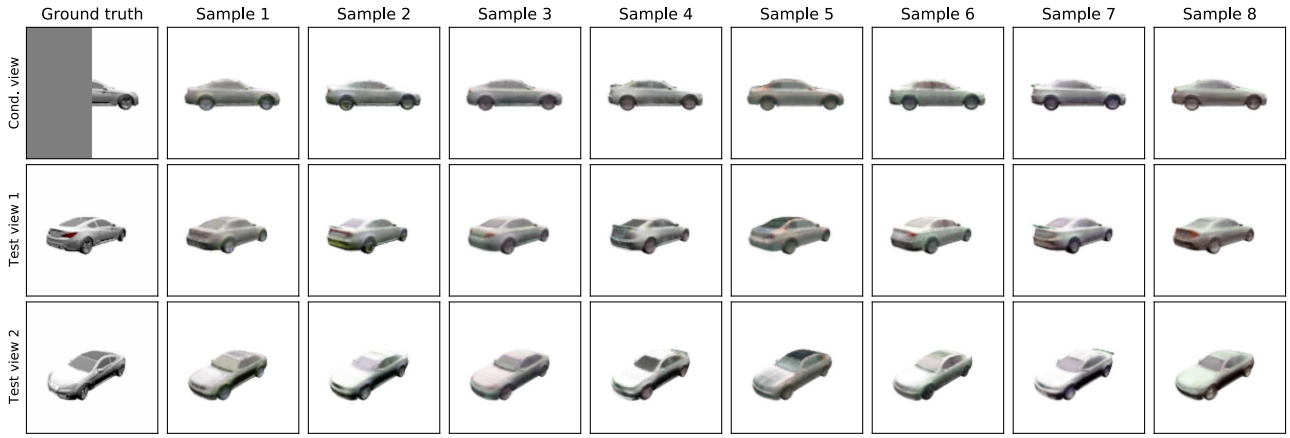


Figure 9: Uncurated SRN Cars HMC Samples

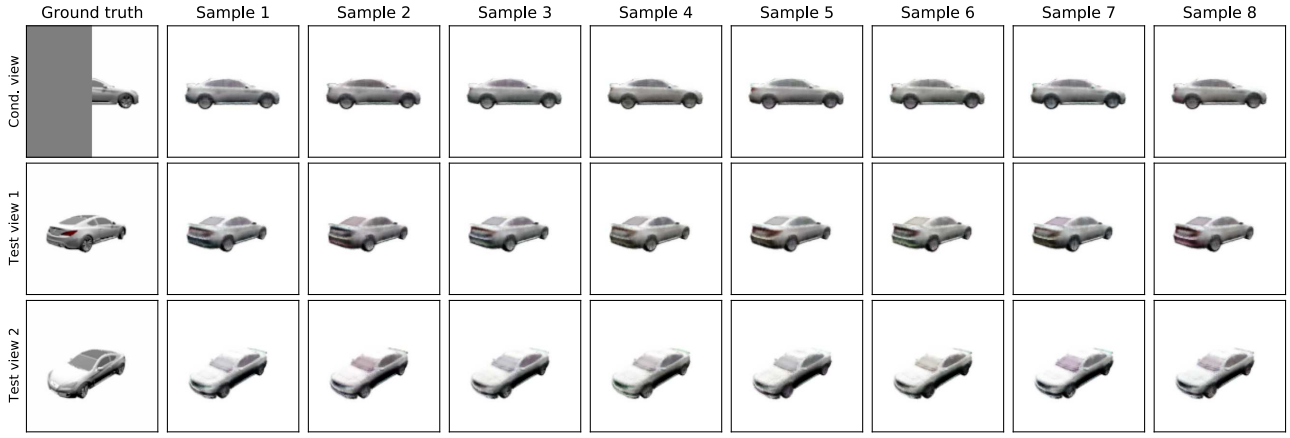


Figure 10: Uncurated SRN Cars VI samples.


Figure 11: Uncurated full-view grey SRN Cars HMC samples.

**HyperNet** $h(z;\theta)$ is a simple MLP with two shared hidden layers, followed by a learnable linear projection and reshape operations to produce the parameters of the two NeRF networks.

**RealNVP** $m(\tilde{z};\zeta)$ consists of 4 RealNVP blocks which act on a latent vector split into 2 parts (designated as $z_0$ and $z_1$ in

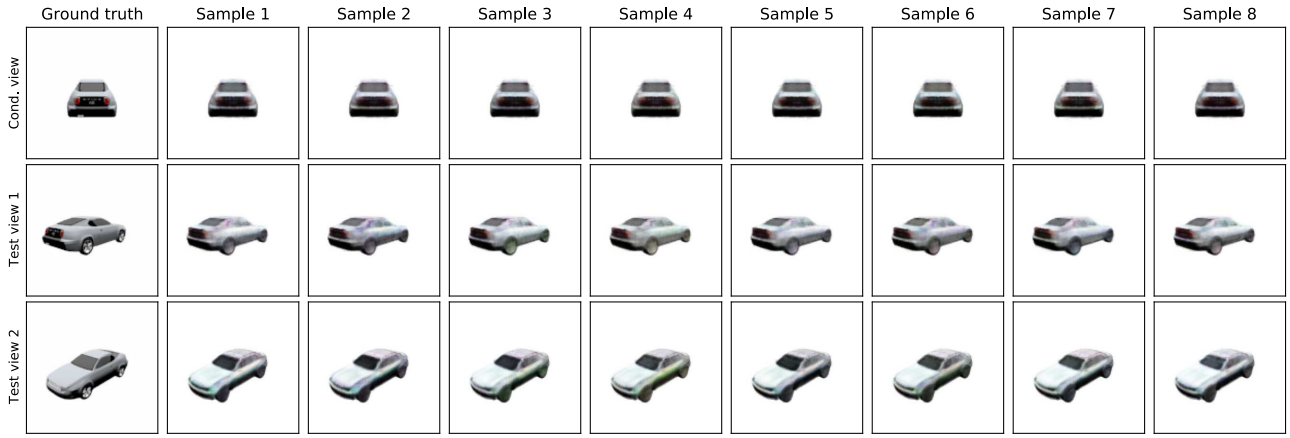


Figure 12: Uncurated full-view grey SRN Cars VI samples.

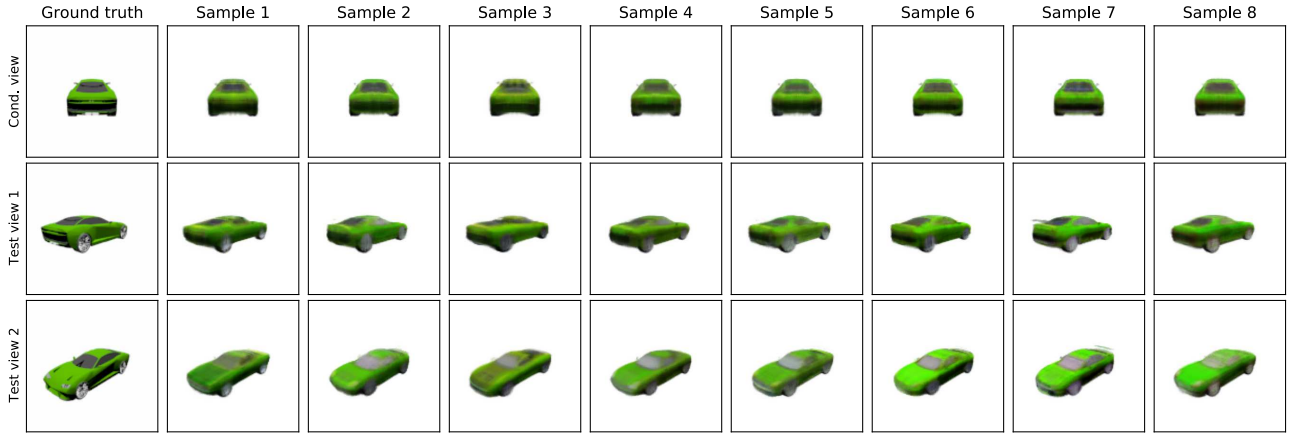


Figure 13: Uncurated full-view green SRN Cars HMC samples.

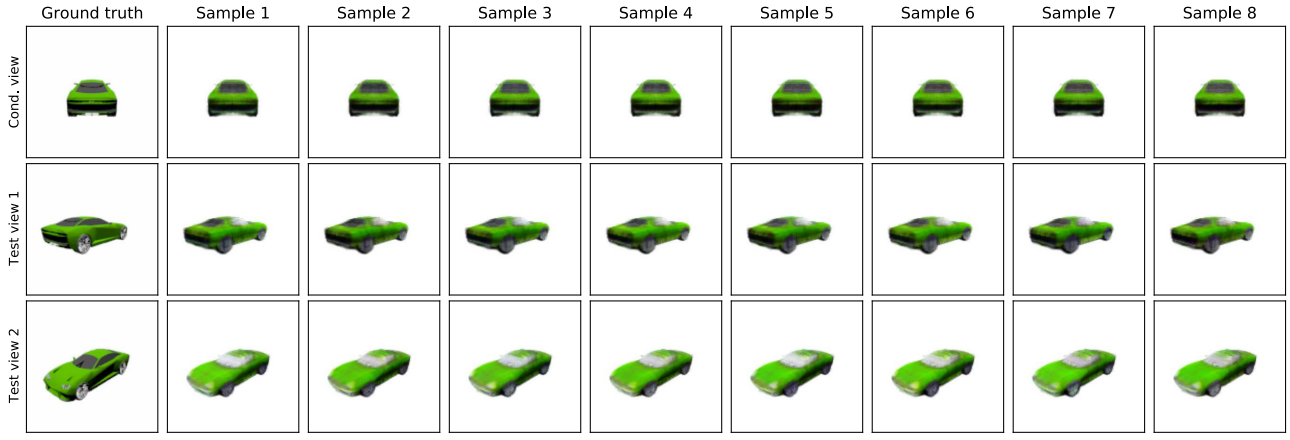


Figure 14: Uncurated full-view green SRN Cars VI samples.

the diagram). The split sense is reversed between the RealNVP blocks.

**NeRF** The NeRF is split into two sub-networks, one for density and one for color. The input position $p$ and ray direction $d$ are encoded using a 10th order sinusoidal positional encoding. For a scalar component of the input vector $x_i$ we produce a

Figure 15: GHUM unconditional samples.

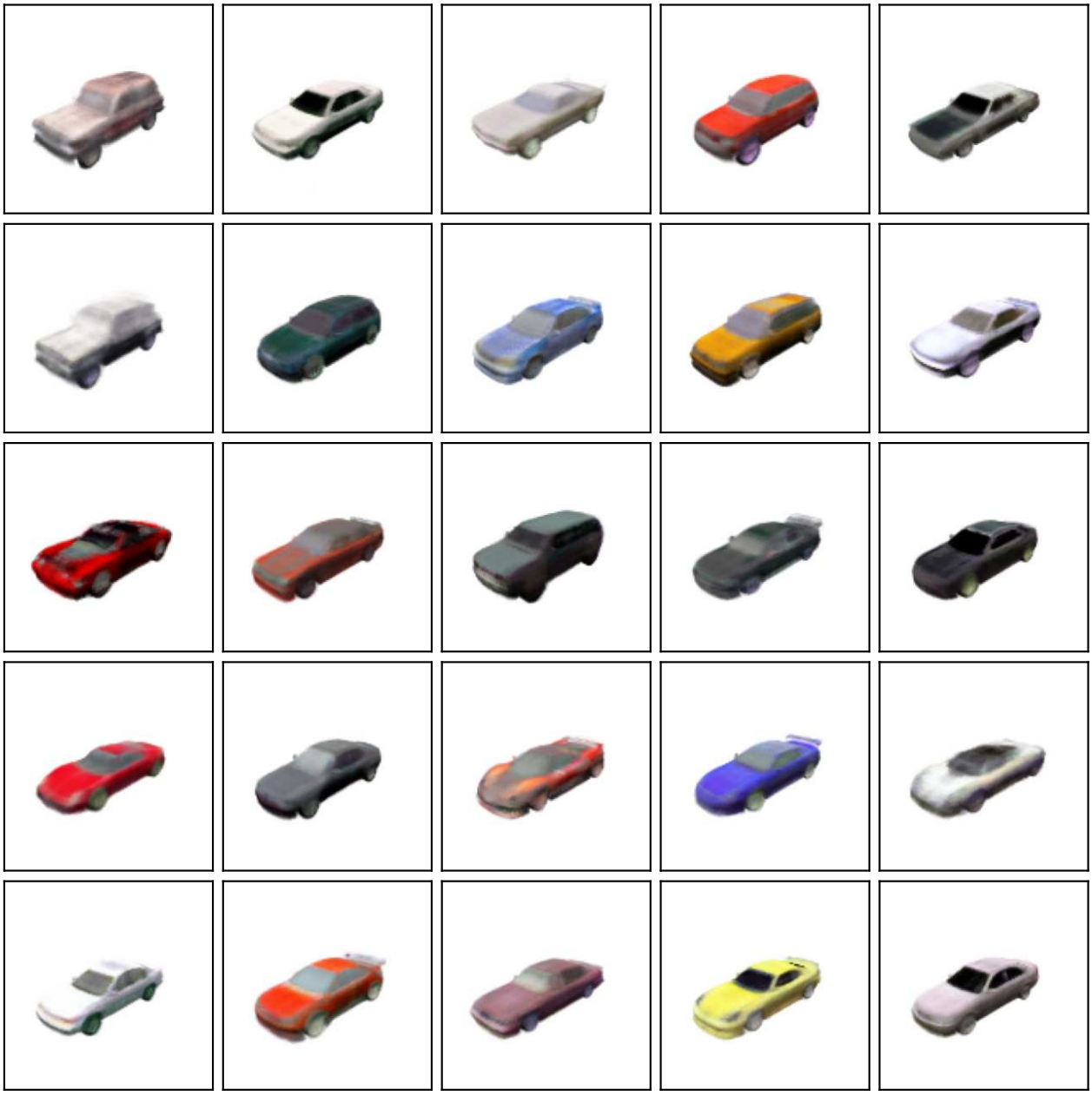

Figure 16: SRN Cars unconditional samples.

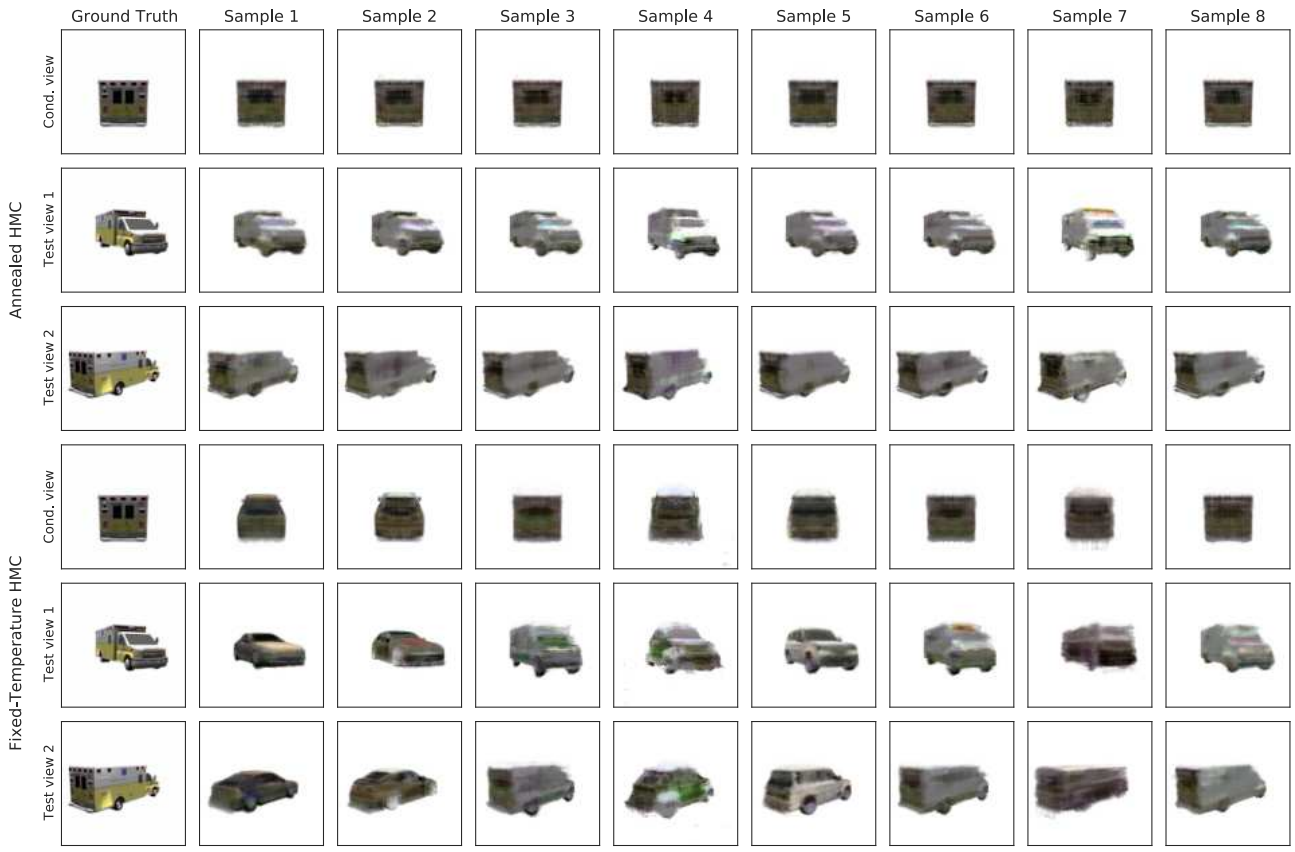


Figure 17: Uncurated annealed and fixed-temperature HMC samples.

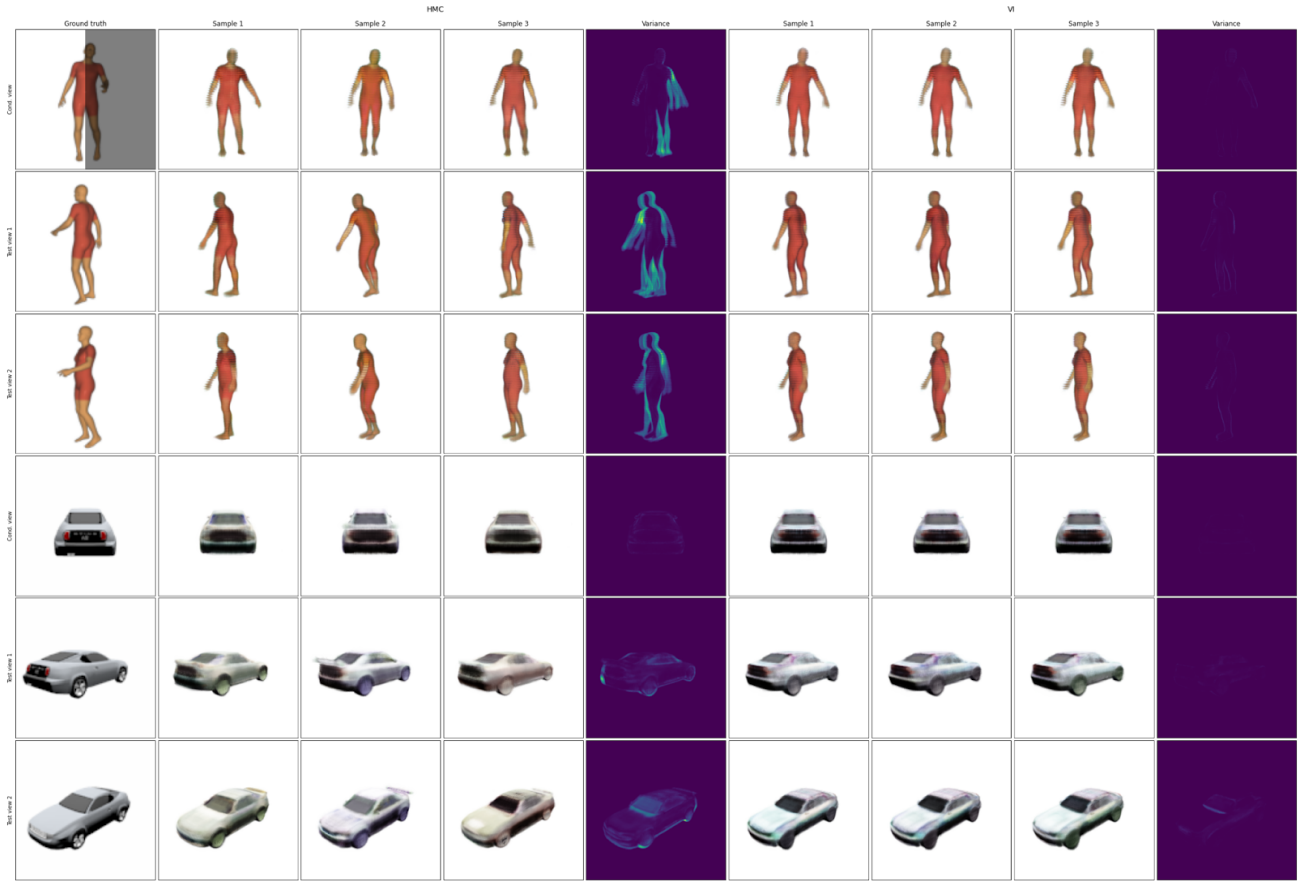


Figure 18: Fig. 2 with 256x256 renderings.

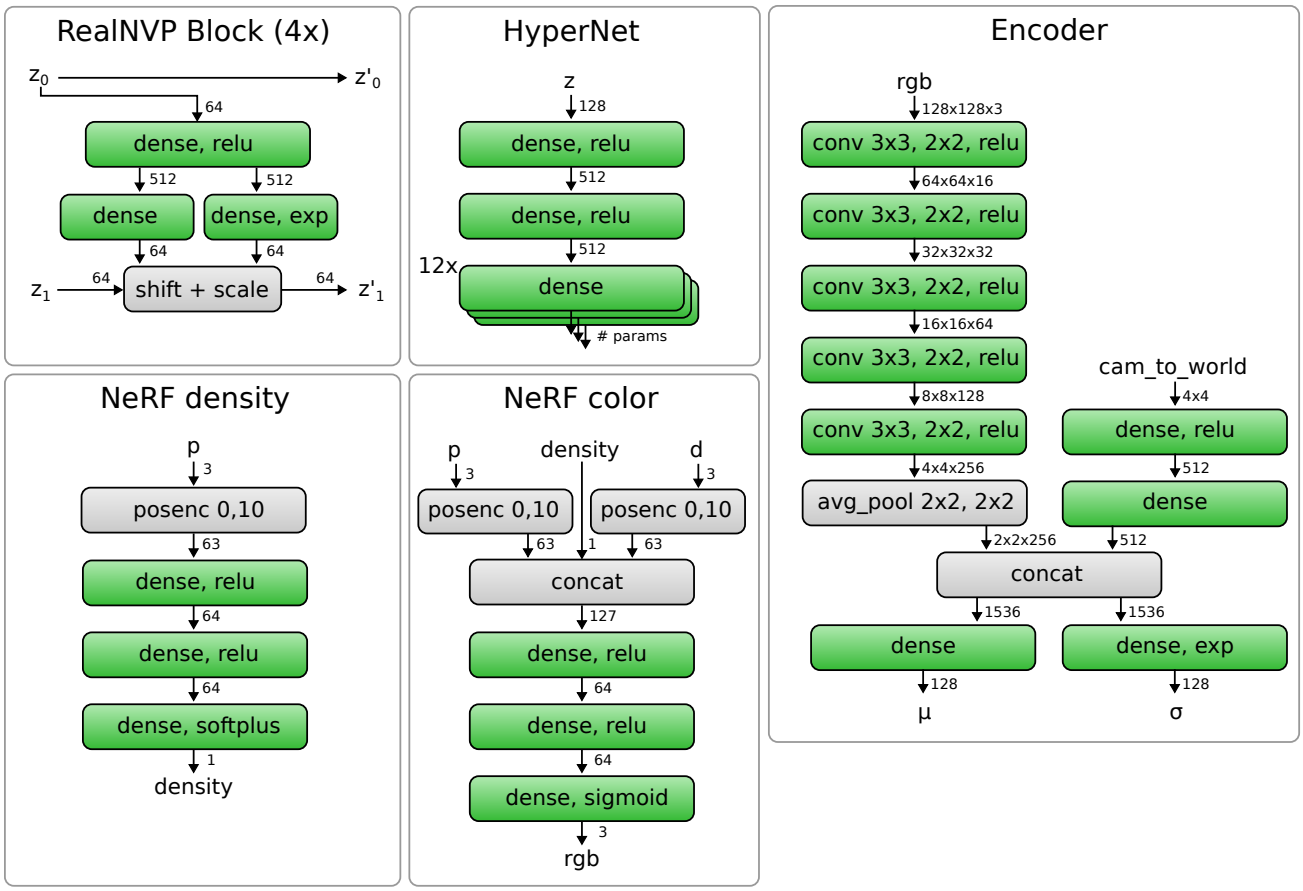


Figure 19: Neural Nets used in ProbNerf

feature:

$$f_i = \{\sin(2^j \pi x_i + 0.5k) | j \in [0, 10), k \in [0, 1]\}. \tag{5}$$

We flatten and concatenate this array with the original input value to produce a 21 element feature vector for each $x_i$. To convert output density $\sigma \in \mathbb{R}^+$ to $\alpha \in [0, 1]$ we squash it as $\alpha = 1 - \exp(-\sigma/128)$, where 128 is the grid size.

**Encoder** Each potential of the variational posterior is modeled as a diagonal covariance Gaussian with mean $\mu$ and scale $\sigma$ computed via a CNN.

## Appendix C Equivalence of linear latent-shift modulations and latent concatenation

The linear shift-only modulations used by Dupont et al. (2022) work as follows for an MLP: given a latent vector $z$, for each layer's output pre-nonlinearity activations $a^{(i)}$ (treating the input as an activation vector $a^{(0)}$), add a shift vector $s^{(i)} = V^{(i)} z$ that is a linear function of $z$ to get $a'^{(i)} = a^{(i)} + s^{(i)}$, and propagate $a'$ forward through the network instead of $a$.

The same effect can be achieved by concatenating $z$ to the activations $a'^{(i)}$ at each level $i$ of the network. The resulting computation is

$$\begin{aligned} a'^{(i)} &= W^{(i)}(\sigma(a'^{(i-1)})^\top, z^\top)^\top + b^{(i)} \\ &\triangleq \tilde{W}^{(i)} \sigma(a'^{(i-1)}) + b^{(i)} + V^{(i)} z \\ &\equiv a^{(i)} + s^{(i)}, \end{aligned} \tag{6}$$

where $W^{(i)}$ denotes the weight matrix at layer $i$, $b^{(i)}$ denotes the biases at layer $i$, $\sigma$ denotes a nonlinear activation function,

and we define $\tilde{W}^{(i)}$ and $V^{(i)}$ to be the submatrices of $W^{(i)}$ that are multiplied by the previous layer's activations $\sigma(a'(^{i-1}))$ and the concatenated latents $z$ respectively.