Calibrated Uncertainties for Neural Radiance Fields

Niki Amini-Naieni Tomas Jakab Andrea Vedaldi Ronald Clark

University of Oxford

Abstract

Neural Radiance Fields have achieved remarkable results for novel view synthesis but still lack a crucial component: precise measurement of uncertainty in their predictions. Probabilistic NeRF methods have tried to address this, but their output probabilities are not typically accurately calibrated, and therefore do not capture the true confidence levels of the model. Calibration is a particularly challenging problem in the sparse-view setting, where additional held-out data is unavailable for fitting a calibrator that generalizes to the test distribution. In this paper, we introduce the first method for obtaining calibrated uncertainties from NeRF models. Our method is based on a robust and efficient metric to calculate per-pixel uncertainties from the predictive posterior distribution. We propose two techniques that eliminate the need for held-out data. The first, based on patch sampling, involves training two NeRF models for each scene. The second is a novel meta-calibrator that only requires the training of one NeRF model. Our proposed approach for obtaining calibrated uncertainties achieves state-of-the-art uncertainty in the sparse-view setting while maintaining image quality. We further demonstrate our method's effectiveness in applications such as view enhancement and next-best view selection.

1. Introduction

Recent advancements in scene representations have led to promising new approaches for novel view synthesis and scene reconstruction. Among these, Neural Radiance Fields (NeRFs) [15] have emerged as a particularly powerful tool, offering unprecedented levels of realism and detail in rendered images. The core idea behind NeRFs is to represent a scene as a continuous vector-valued function, parameterized by a neural network, that maps spatial coordinates and view directions to color and density values. This approach has facilitated the creation of highly detailed 3D representations from sets of 2D images, revolutionizing the field of 3D reconstruction.

However, despite their impressive capabilities, traditional

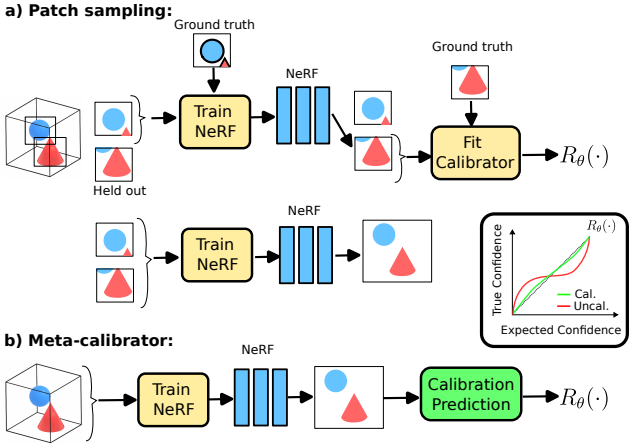

Figure 1. **NeRF calibration**: We address the problem of obtaining calibrated uncertainties from NeRF models that follow the true confidence levels of the rendered views. (a) We first introduce a patch-sampling approach that trains a NeRF on a subset of patches and fits the calibrator, $R_\theta(\cdot)$ to the held-out patches. (b) We then propose a meta-calibrator that learns to predict the calibration function from scene features, requiring no extra training to calibrate new scenes.

NeRF models often lack an essential component: an accurate measure of uncertainty in their predictions. This limitation is significant as it constrains the reliability and applicability of NeRF in critical areas such as autonomous systems, robotics, and augmented reality, where understanding the confidence level of predictions is crucial.

Recent efforts have been directed towards incorporating probabilistic modeling into NeRF. One such notable advancement is FlipNeRF [24], a variant that models color distributions along rays using a mixture of Laplacians, providing a probabilistic output that represents the uncertainty in color predictions. While FlipNeRF marks a significant step forward, it still suffers from a key limitation: the probabilities it outputs are not calibrated, meaning they do not necessarily correspond to true confidence levels. In this paper, we present a novel approach for obtaining accurate uncertainties for NeRF models. Our strategy integrates the Laplacian mixture distribution from FlipNeRF [24] with the calibration

techniques by Kuleshov et al. [10]. However, a naive application of this calibration method poses significant challenges in the context of NeRF. Firstly, the calibration framework in [10] does not explicitly define a method for deriving precise uncertainty values from the predicted distribution. The second challenge stems from the fact that the calibration method requires a held-out calibration set and the fact that NeRF models are fitted per-scene. In most cases there are only few-views available for training, and reserving data for calibration not only risks overfitting but also constrains the volume of data available for training the actual NeRF model. Furthermore, the calibration relies on ground-truth data and therefore complicates the calibration of uncertainties for outputs for which ground-truth values may not be available for most scenes.

To address the challenge of limited ground-truth data in the sparse-view setting, we propose two strategies illustrated in Figure 1. For the first strategy, we train two models: one on images with 20% of the rays removed in square patches, and another on all available training data. We use the former model for calibration and the latter for inference. Importantly, the model used for inference is still trained on all available data, ensuring that this method does not compromise the image quality of the rendered views.

The second strategy removes the need to train two models per target scene. Instead, we make use of a unique insight: while calibration curves exhibit significant variation across scenes, they also demonstrate a significant regularity in their structure. Using this observation, we propose to learn a low-dimensional representation of these calibration curves. The parameters of this representation can be predicted based on features of the scene, thus removing the need for reserving or synthesizing a calibration set using ground truth data for calibration on new scenes. While the first strategy results in more accurate uncertainties, it is less efficient than the second strategy when applying NeRFs to multiple target scenes. Specifically, our contributions are:

1. We propose a framework for obtaining calibrated uncertainties from NeRF models in the sparse-view setting. This consists of:
   (a) A robust and efficient way to compute uncertainty from the predictive posterior.
   (b) Two techniques to obtain the per-scene calibrator without requiring a separate held-out validation set, specifically a patch-sampling procedure and a meta-calibrator.
2. We show the effectiveness of our framework on real scenes in the Local Light Field Fusion (LLFF) dataset [14], as well as on two tasks: uncertainty guided NeRF view enhancement and next-best view selection.

2. Related Work

Neural Radiance Fields. Neural Radiance Fields (NeRFs) [15] have become a popular method for novel view synthesis due to their simplicity and impressive performance. From a set of 2D images, NeRFs learn a neural network representation of a single scene. A trained NeRF model outputs estimates of the volume density and emitted radiance at any 3D location and viewing direction. Novel views can be generated by applying volume rendering [8] to the density and radiance values predicted by the NeRF model for points along rays cast into the scene.

Over the last few years, several extensions of NeRFs have been explored [9, 17, 20, 28, 32]. These include speeding up training and inference [11, 22, 30], modeling dynamic scenes [3], and learning from a sparse set of training views [1, 6, 18, 24, 25]. Both MixNeRF [25] and FlipNeRF [24], designed for few-shot novel view synthesis, model the RGB color channels given a ray as independent random variables that follow a mixture model. FlipNeRF further uses the uncertainties of the pixel colors to regularize the training process. However, neither MixNeRF nor FlipNeRF considers the calibration of these uncertainties. Although our method also leverages a mixture model, unlike MixNeRF and FlipNeRF, we focus on NeRF uncertainty calibration and present a method to obtain calibrated uncertainties from NeRF models.

Uncertainty in NeRFs. Recently, methods for uncertainty estimation in NeRFs have been proposed [5, 13, 21, 26, 27, 29]. Stochastic Neural Radiance Fields (S-NeRF) [26] learns a probability distribution over all possible radiance fields by modeling the volume density and radiance as random variables that follow a joint distribution. S-NeRF employs variational inference to sample from an approximation to this distribution and uses the variances of the sampled pixel colors as the estimated uncertainties. Conditional-flow NeRF (CF-NeRF) [27] builds on S-NeRF by combining latent variable modeling and conditional normalizing flows to relax the strong constraints S-NeRF imposes over the radiance distribution. BayesRays [5] presents a method to obtain uncertainties from any pretrained NeRF model, further enhancing the flexibility of NeRF uncertainty estimation. While prior work has improved the practicality of NeRF uncertainty estimation, it has overlooked the problem of calibration, a well-known issue with Bayesian techniques [10]. We begin to explore this blindspot by presenting a method to obtain calibrated uncertainties from NeRF models.

Uncertainty Calibration in Deep Learning. Uncertainty calibration has been studied for Bayesian deep learning methods [4, 10]. However, calibration has not been adapted for or applied successfully to NeRF uncertainty estimates. This

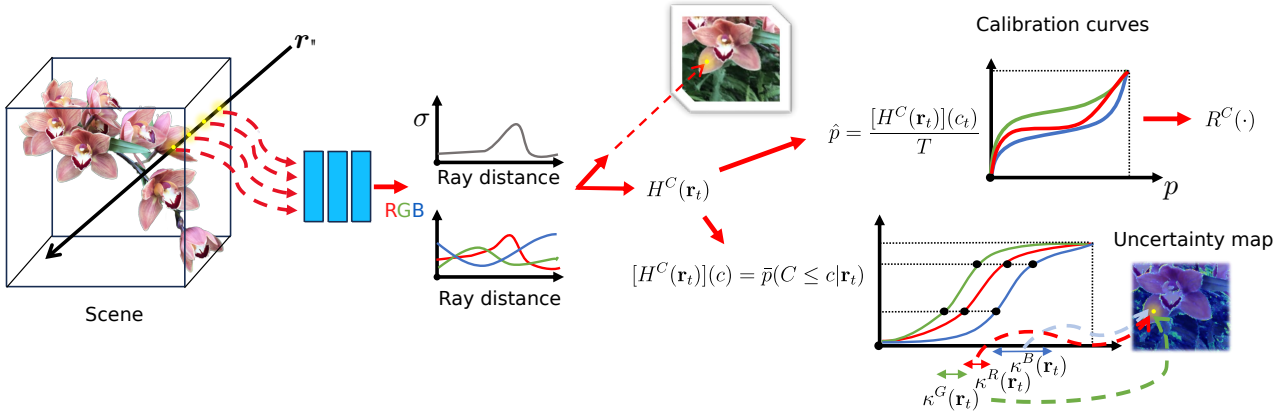

Figure 2. **NeRF calibration overview:** In this figure we show the base method we use for computing calibrated uncertainties from the NeRF model. The output of the NeRF model at each sample is a Laplace distribution containing the mean color, $\boldsymbol{\mu_j}$, and scale $\boldsymbol{\beta_i}$ parameters. The expected confidence levels of the true colour values are then computed using the ground truth input image, and the empirical distribution is formed. In Section 3.3 we present a patch sampling approach for computing this distribution, and in Section 3.4 we show how to predict it without using the ground truth image. The final uncertainty is then computed as the 0.25 - 0.75 inter-quartile range averaged over the RGB color channels.

may be because uncertainty estimation in NeRFs is a relatively new area of study that introduces additional complexity due to the unconventional nature of the NeRF framework. Rather than directly predicting a class or estimating a single scalar, NeRFs output a volume density and emitted radiance for every 3D location and viewing direction in space. Both of these quantities are uncertain and, hence, could be calibrated. Furthermore, these values are combined through volume rendering to estimate a 3D RGB color vector for every pixel. The uncertainties for these colors could also be calibrated. To control this complexity, in our method, we focus on calibrating the uncertainties of the rendered RGB colors and use a base probabilistic NeRF model [24, 25] that lends itself well to our calibration process.

3. Method

In this work, we propose a method for obtaining calibrated uncertainties from a NeRF model. Figure 2 shows an overview of our calibration procedure.

3.1. Preliminaries

Neural Radiance Fields. Neural Radiance Fields (NeRFs) [15] represent a scene as a continuous vector-valued function with inputs a Cartesian point $\mathbf{x} = (x, y, z)$ and unit viewing direction vector $\mathbf{d} = (u, v, w)$ and outputs an emitted radiance $\boldsymbol{c} = (r, g, b)$ and volume density σ. By optimizing the weights $\boldsymbol{\Theta}$ of a neural network approximation $\mathbf{F}_{\boldsymbol{\Theta}}$ to this representation, NeRFs can render the color of any pixel in a synthetic image of the scene. To achieve this, principles from classical volume rendering [8] are applied to the radiance and density values estimated by $\mathbf{F}_{\boldsymbol{\Theta}}$ for points along a ray cast from the origin $\mathbf{x_0}$ of the virtual camera, through the pixel, and into the scene. More specifically, the expected color $\mathbf{c}(\mathbf{r})$ of a camera ray $\mathbf{r}(t) = \mathbf{x_0} + t\mathbf{d}$ with near and far bounds t_n and t_f is:

$$\mathbf{c}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\boldsymbol{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{1}$$

where $T(t) = e^{-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds}$. Integral 1 is estimated with numerical quadrature to obtain the colors of the pixels in the synthetic image from the NeRF model outputs. Since the numerical quadrature is differentiable, NeRF optimizes $\boldsymbol{\Theta}$ according to equation 1 with gradient descent.

Probabilistic NeRF. We focus on FlipNeRF [24] which models the joint distribution of the color $\mathbf{C} = (R, G, B)$ given a ray $\mathbf{r}$ with a mixture of Laplacians:

$$p(\mathbf{C} = \mathbf{c}|\mathbf{r}) = \sum_{j=1}^{M} \pi_j \mathcal{F}(\mathbf{C} = \mathbf{c}; \boldsymbol{\mu_j}, \boldsymbol{\beta_j}), \tag{2}$$

where M is the number of sampled points along the ray $\mathbf{r}$. $\mathcal{F}(\mathbf{C} = \mathbf{c}; \boldsymbol{\mu_j}, \boldsymbol{\beta_j})$ is the 3D Laplacian probability density with location parameter $\boldsymbol{\mu_j} = (\mu_j^R, \mu_j^G, \mu_j^B)$ and scale parameter $\boldsymbol{\beta_i} = (\beta_j^R, \beta_j^G, \beta_j^B)$ evaluated at the color $\mathbf{c}$. The mixing coefficients $\{\pi_j\}_{j=1}^M$, location parameters $\{\boldsymbol{\mu_j}\}_{j=1}^M$, and scale parameters $\{\boldsymbol{\beta_j}\}_{j=1}^M$ are predicted by the FlipNeRF [24] model during inference. From this distribution, we can easily compute a closed form expression for the CDF for a given ray, which we denote as F_t. Finally, we can use the CDF to compute the confidence level by evaluating it at the given output value, $p_t = F_t(c_t)$. The model itself is learned using a regularised form of Maximum Likelihood

Estimation (MLE) over a training set $\mathcal{D} = \{(\mathbf{r_i}, \mathbf{c_i})\}_{i=1}^{N}$ containing the rays $\mathbf{r_i}$ and colors $\mathbf{c_i}$ from the pixels in the ground truth images of the scene. However, training FlipNeRF in this manner still has a major limitation in that the output probabilities are not calibrated. We build our method on top of FlipNeRF and show how we can preserve the view quality while obtaining calibrated uncertainties.

Calibrated regression. In [10], Kuleshov et al extend calibration methods for classification to regression. They define a forecaster $H : \mathcal{X} \to (\mathcal{Y} \to [0, 1])$ as a function that outputs for each $x_t \in \mathcal{X}$, a CDF F_t. Given a pretrained forecaster H, they suggest training an auxiliary model $R : [0, 1] \to [0, 1]$ by fitting R to a recalibration dataset $D = \left\{ \left([H(x_t)](y_t), \hat{P}([H(x_t)](y_t)) \right) \right\}_{t=1}^{T}$, where

$$\hat{P}(p) = |\{y_t : [H(x_t)](y_t) \leq p \text{ for } t = 1, \ldots, T\}|/T$$

This allows us to obtain predictive posterior values that closely match the true confidence levels using $\hat{F}_t \equiv R \circ F_t$. However, there are numerous challenges associated with applying this recalibration procedure: firstly, note that it requires ground-truth values (y_t) for the predictions we are recalibrating. This means that in order to prevent overfitting we need to reserve part of the training dataset specifically for fitting the calibrator (which leaves less data for training the model — a significant issue if we only have a few input views). This also makes it difficult to obtain calibrated uncertainties for quantities for which we might not have the ground-truth. Secondly, it does not actually prescribe how to compute a suitable uncertainty value from the predicted distribution.

3.2. Calculating uncertainty

While the predictive posterior in Equation 2 provides a distribution over likely color and depth values for a NeRF model, it does not inherently offer a straightforward metric for quantifying uncertainty at a specific point in the reconstruction. Intuitively, as the variance of this distribution increases, so does the uncertainty of the model's output at that point. Therefore, the variance or standard deviation is a popular choice for quantifying the uncertainty [24, 26, 27, 29]. However, in the case of the mixture distribution this can be slow to compute especially if it has to be done for each pixel.

To effectively quantify this uncertainty, we therefore propose to use the following metric based on the interquartile range:

$$\kappa^C(\mathbf{r_t}) = [\hat{F}_t^C]^{-1}\left(\frac{3}{4}\right) - [\hat{F}_t^C]^{-1}\left(\frac{1}{4}\right), \qquad (3)$$

where $\kappa^C(\mathbf{r_t})$ represents the uncertainty at a ray $\mathbf{r_t}$ in the color channel C.

In this formulation, the uncertainty is computed as the interquartile range of each calibrated distribution. This difference provides a measure of the statistical dispersion and thus serves as a robust measure of the spread of the output channel. By averaging the interquartile range over the color channels, we obtain a single scalar value that effectively quantifies the uncertainty of the NeRF model for the given ray. As shown in our experiments, this method is very computational efficient, and it provides an accurate estimate of uncertainty.

3.3. Calibration from patches

As previously discussed, to ensure an accurate fit of the calibration model, it is crucial to reserve a portion of the training data so that the calibrator can be trained on expected confidence levels representative of the test-time performance of the model. However, holding out entire images for calibration, especially in sparse-view scenarios where we have very limited views (*e.g.*, 3 views), is problematic. This approach could result in removing a large amount of the available data, creating a disparity between the calibration and inference models. Moreover, the calibration process might rely on uncertainties from a specific region, not adequately representing the whole scene. An initial, naive solution might be to hold out randomly-sampled pixels from the training images. However, this method is not sufficient as the calibration NeRF could still reconstruct these areas effectively by using context from adjacent pixels, which does not accurately reflect the model's performance on unseen data.

To overcome these challenges, we propose a more effective strategy: instead of holding out entire images or individual pixels, we sample and withhold square patches from the training images. This method ensures that the rays corresponding to these patches are excluded, allowing us to fit an effective calibration model. By selecting patches from various parts of the scene, we diversify the locations of the held-out rays. This approach not only maintains a balance in the available training data but also provides a more representative sample for calibration, ensuring that the NeRF's performance on these held-out regions is a better indicator of its performance on completely new test poses.

3.4. Meta-calibrator

In the previous section, we proposed a two-stage method for training the calibrator on sampled image patches. The calibration curve, however, varies across different scenes which, in the NeRF case, means we have to train a new calibration model for each scene. This is quite inefficient as it requires training a NeRF model in both stages and requires having ground-truth values available to compute the confidence levels needed for fitting the calibrator. To address this, we use the insight that the calibration curves demonstrate significant regularity (see Figure 3). We propose that a low-dimensional

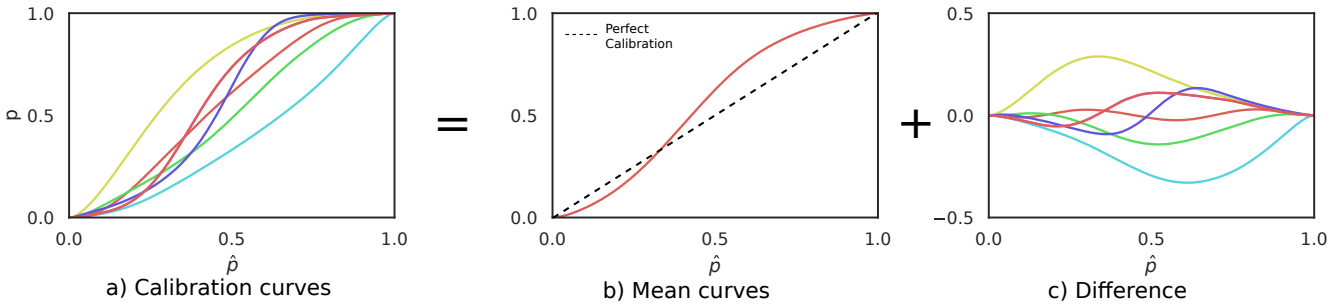

Figure 3. **Regularity in the calibration curves:** This figure shows the calibration curves obtained for seven of the real-world *DTU dataset* scenes [7]. While the calibration curves vary significantly across scenes there is a high degree of regularity in this variation. We use this insight to construct a low-dimensional parameterization of the curves that our meta-calibrator can predict from scene features.

model of the calibration function can be learned and predicted using scene features and the uncalibrated uncertainty map of the model, enabling us to predict the calibration function *without evaluating the empirical confidence levels or training an additional NeRF model for new scenes*.

A Parametric Model for Calibration Curves. To achieve this, we first fit a low-dimensional representation of the calibration curves via PCA. We form the training vectors, $\mathbf{v}_k$, from the calibration dataset as follows: $\mathbf{v}_k = \hat{P}([H(x_t)](y_t))$. We find that fitting the PCA using only a few scenes provides a good enough approximation to capture the variation in the test curves (see supplementary). Here, $\mathbf{V} = [\mathbf{v}_k]$ is a $K \times M$ matrix, with K representing the number of training curves and M the sample count along each curve. PCA is then used to determine the basis vectors $\mathbf{u}_i$ and coefficients α_i, so that each calibration curve can be represented as: $\mathbf{v}_k = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$. The parameters $\theta = (\alpha_1, \alpha_2, ..., \alpha_n)$ fully describe the calibration functions. To find the optimal number of components, we compute the explained variance, and find that in our case most of the variance is explained using only $n = 3$ components. To ensure the final calibrator is monotonically increasing, we derive the final calibration function $R_\theta(\cdot)$ from this, using isotonic regression applied to the curve approximated by θ.

Predicting Calibration Parameters. What remains is to estimate the calibration parameters, θ, for a new scene. As we do not want to train an additional NeRF model to obtain estimates for the test-time confidence levels, we propose predicting these parameters using scene-specific features that can be derived from the rendered images. This approach is motivated by the human ability to visually identify inaccuracies in the renderings. Specifically, we use a Multi-Layer Perceptron (MLP) regression model to estimate θ from scene-specific features. The MLP model utilizes features extracted by the DINOv2 model [19] from rendered images ($\mathbf{f}_I$) and uncalibrated uncertainty maps ($\mathbf{f}_\kappa$). The goal here is to have DINOv2 extract features that describe the rendering imperfections, correlating with the calibration curve. Once trained, the MLP model can predict θ for any new scene: $\theta = MLP([\mathbf{f}_I, \mathbf{f}_\kappa])$. We find that training the meta-calibrator on only a few scenes (30 in our case) allows it to generalize well to new test scenes, however, increasing the number of scenes can help further improve the accuracy of the predicted calibration curves (see supplementary).

In summary, the meta-calibrator involves rendering an image using the trained NeRF, extracting pertinent features with DINOv2, and feeding these to the MLP regression model. The resultant θ can be used to construct the scene-specific calibrator R_θ without relying on ground-truth output values. Compared to the method described in Section 3.3, this means we do not have to train any additional NeRF models specifically for obtaining the calibrator, which has significant performance advantages in applications like Next-Best View selection (see supplementary) where we have to train a model each time a new view is added.

3.5. Applications

Uncertainty-guided NeRF View Enhancement. When trained on limited views, such as in a 3-shot setting, NeRFs often produce images with unnatural artifacts. Therefore, we propose to mask regions based on our calibrated uncertainty and then use a denoising diffusion probabilistic model for inpainting [12] the most uncertain parts. Our method calculates the pixel uncertainty for each ray and masks pixels with uncertainties above a specific threshold selected using grid search on a validation scene. After masking, the images are fed to the inpainting model, resulting in enhanced images with fewer artifacts and smoother edges, as detailed in our experiments.

Next-best View Selection. In an active setting, an autonomous system may be tasked with selecting the next pose at which to capture more training data for the NeRF. Rather than selecting new poses randomly, such a system may use

uncertainty to guide a more efficient selection process. Similar to [29], we consider iteratively selecting the next-best view as the one that produces the highest uncertainty out of a set of candidate views, and then training on the new dataset composed of the original training views and the additional view. In our experiments, we show that next-best view selection guided by our calibrated uncertainties results in greater performance gains than the method proposed in [29] does.

4. Experiments

The primary objective of the experiments is to evaluate how well our proposed method captures the actual confidence levels of the NeRF model (i.e. the calibration error) and to demonstrate how this can benefit applications such as view enhancement and next-best view selection.

4.1. Datasets & Metrics

Metrics and calibration curves. We use a variant of the calibration error in [10] to evaluate the quality of the proposed calibration method with patches and the meta-calibrator. Specifically, given a test set $\mathcal{D} = \{(\mathbf{r_t}, \mathbf{c_t})\}_{t=1}^{T} = \{(\mathbf{r_t}, (r_t, g_t, b_t))\}_{t=1}^{T}$, we report:

$$ERR = \frac{1}{T}\sum_{t=1}^{T}(p_t - \hat{P}(p_t))^2, \tag{4}$$

where p_t is the expected confidence level for data point $(\mathbf{r_t}, c_t)$, and $\hat{P}(p_t)$ is the empirical frequency of data points within that confidence level. More specifically, for each $t \in \{1, \ldots, T\}$, we set $p_t = M_t^C(c_t)$ and,

$$\hat{P}(p) = |\{c_t : M_t^C(c_t) \leq p \text{ for } t = 1, \ldots, T\}|/T, \tag{5}$$

where $M \equiv F$ for uncalibrated errors and $M \equiv \hat{F}$ for calibrated errors, and $C \in \{R, G, B\}$. Note that this formulation of the calibration error is equivalent to the one in [10] with a confidence level for every unique p_t and weights that more significantly penalize errors from frequently predicted confidence levels.

Following [10], we plot $\{(p_t, \hat{P}(p_t))\}_{t=1}^{T}$ before and after calibration for each color channel to generate calibration curves. A perfectly calibrated forecaster would produce the straight line $p_t = \hat{P}(p_t)$ as each expected confidence level would equal the empirical one. Intuitively, our version of the calibration error is the mean squared vertical distance of points on the calibration curve from a perfectly straight line. If an expected confidence level occurs N times in the test data, its distance is counted N times in the mean. Following [27], we additionally report the area under the sparsification error (AUSE) with respect to the mean absolute error and the negative log-likelihood (NLL) of the test data averaged across all scenes in LLFF [14] for our method with patches and with the meta-calibrator. Following [24], we use the PSNR, SSIM [31], LPIPS [33], and geometric average [1] to evaluate image quality.

Datasets. For our experiments we use three datasets: the Realistic Synthetic 360° dataset [15], the subset of scenes in DTU [7] used in [24], and all the scenes in LLFF [14]. We specify the specific datasets used for each task in the descriptions below.

4.2. Implementation

Baselines. We compare our patches and meta-calibrator methods against the uncalibrated FlipNeRF [24] uncertainties and Density-aware NeRF Ensembles (DANE), the state-of-the-art NeRF uncertainty estimation method presented in [29] trained on the same data splits of LLFF [14]. Unlike [24], we assume the mixture of Laplacians distribution for the color and not a Gaussian when reporting the NLL of the ground truth test images for the former baseline. We additionally compare our method against the naive ensembles approach with five ensemble members presented in [29]. Following [29], we implement the ensembles-based methods using a public implementation of Instant-NGP [2, 16].

Our Approach. To evaluate our calibration method with patches, for each scene in LLFF [14], we train two models: one on the three training views with 20% of the rays removed in square patches one sixth the height of the images for calibration and one on all three training views for inference. To evaluate the meta-calibrator, we train it on the synthetic scenes in the Realistic Synthetic 360° dataset [15], the subset of scenes in DTU [7] used in [24], and all the scenes in LLFF except for the test scene.

For uncertainty-guided NeRF view enhancement, the uncertainty threshold is selected using a grid search over thresholds $[0, 0.25, 0.75, 1]$ in the *Room* scene of LLFF [14]. We use the denoising diffusion probabilistic model from [12] pretrained on the ILSVRC 2012 subset of ImageNet [23] at a resolution of 512×512 to inpaint the masked regions.

For next-best view selection, we start by training the NeRF model for 2000 iterations on a training set of three images. The next-best view is selected by evaluating the average pixel uncertainty for each of the candidate views, and the view with the highest uncertainty is added to the training set. The average PSNR of the test images is reported after each training iteration. Please refer to the supplementary materials for more details and results on the next-best view planning and inpainting tasks.

4.3. Results

Test Set Calibration Curves & Errors. We evaluate the calibration curves and errors on the test sets for all eight scenes in LLFF [14]. The calibration curves for the RGB channels are shown in Figure 4. The solid red, green, and

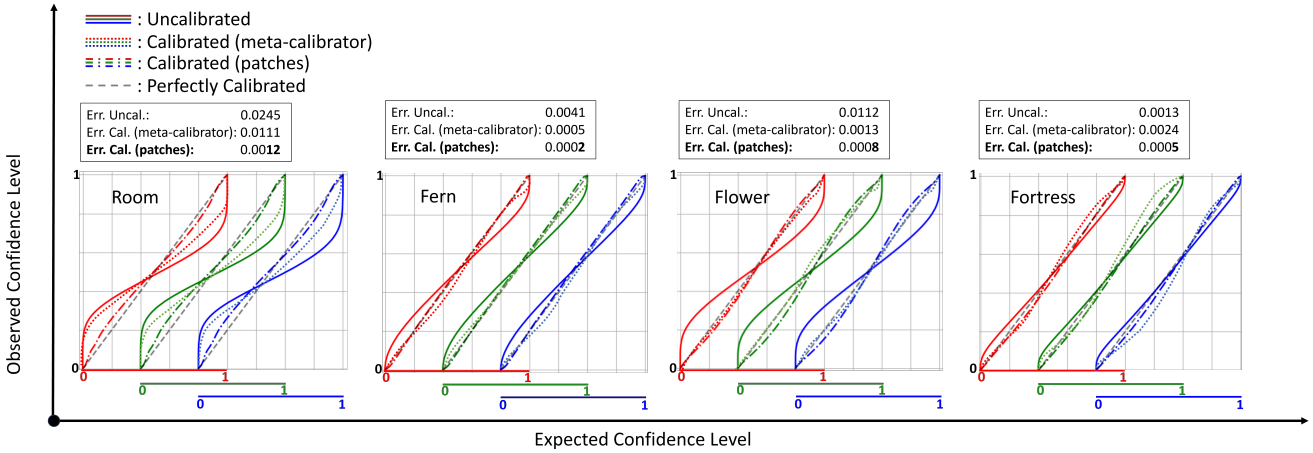

Figure 4. Calibration curves for test data from the first four scenes in LLFF [14]. The color of each curve indicates the color channel it corresponds to. The red, green, and blue solid curves are not closely aligned with the grey dashed lines, showing that the pretrained NeRF model is miscalibrated. The calibrated curves are much closer to the grey dashed lines for all scenes for the patches method and for all scenes but *Fortress* for the meta-calibrator, demonstrating that our proposed approach works very well. This is also verified by how significantly calibration reduces the error between the expected and empirical confidence levels.

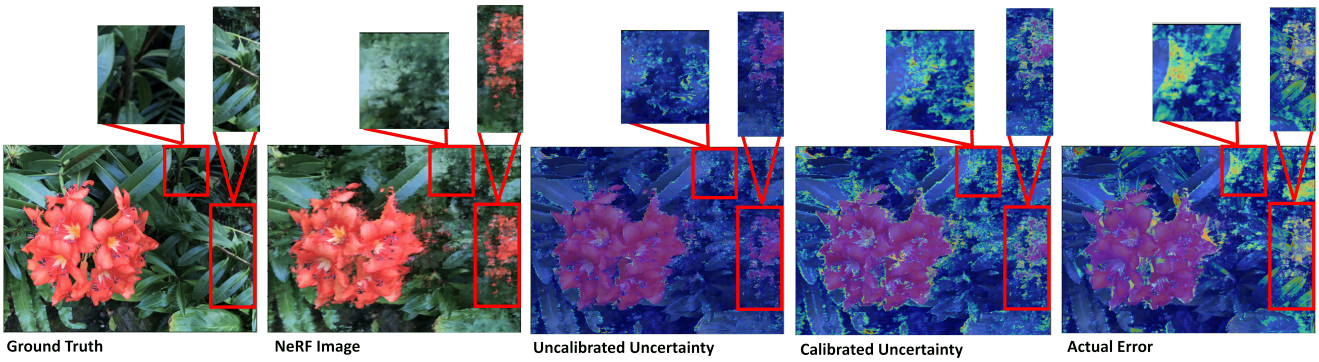

Figure 5. Uncalibrated and calibrated uncertainty maps from the *Flower* scene in LLFF [14]. The calibrated uncertainties clearly correlate better with the true errors (rightmost images) than the uncalibrated uncertainties do.

Table 1. Calibration errors for test data from the eight scenes in LLFF [14]. The calibration error is defined in 4 and 5. The calibration errors are averaged over the RGB color channels. The * indicates that the meta-calibrator does not require training an additional model for the target scene, while the patches method does.

Scene	Uncal.	Meta-cal.*	Patches
Room	0.0245	0.0111	**0.0012**
Fern	0.0041	0.0005	**0.0002**
Flower	0.0112	0.0013	**0.0008**
Fortress	0.0013	0.0024	**0.0005**
Horns	0.0085	**0.0016**	0.0031
Leaves	0.0024	0.0028	**0.0019**
Orchids	0.0070	**0.0006**	0.0023
Trex	0.0097	**0.0008**	0.0010

blue curves in Figure 4 illustrate that the initial confidence levels for the NeRF model are miscalibrated. The model is consistently overconfident for confidence levels closer to one and underconfident for confidence levels closer to zero. Clearly, the presented patches calibration method significantly improves the calibration quality as each calibrated curve is much closer to the perfectly calibrated line (dashed grey line in Figure 4) than each solid RGB curve is, and

Table 2. Our proposed method with patches and with the meta-calibrator result in significantly better uncertainties and image quality than the state-of-the-art NeRF uncertainty estimation method DANE [29] on the 3-view LLFF [14] sparse novel view synthesis dataset.

	Uncertainty			Image Quality	
	Cal. Err. (↓)	AUSE (↓)	NLL (↓)	PSNR (↑)	LPIPS (↓)
Naïve Ens.	0.0505	0.0559	13.16	15.19	0.646
DANE	0.0441	0.0577	11.25	15.19	0.646
Ours (Uncal.)	0.0086	0.0221	-1.54	**19.34**	**0.235**
Ours (Meta-cal.*)	0.0026	0.0223	-2.04	**19.34**	**0.235**
Ours (Patches)	**0.0014**	**0.0207**	**-2.53**	**19.34**	**0.235**

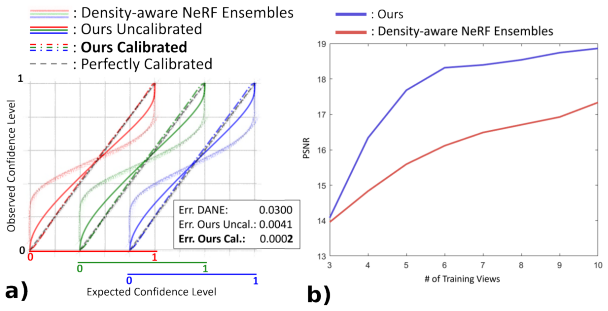

Figure 6. **Comparison to DANE:** Here we compare our proposed approach to DANE [29]. In (a) we show the calibration errors for our method with patches and DANE on the *Fern* scene, and in b) we show the gain in PSNR as we add more views in the *Horns* scene of LLFF [14]. Our method results in lower calibration errors and greater performance gains in next-best view planning.

the meta-calibrator performs very well, sometimes outperforming the patches approach. This means that the expected confidence levels predicted by the calibrated CDFs are much closer to the empirical ones than the expected confidence levels from the original CDFs are. Furthermore, our calibration approach significantly reduces the calibration error for all channels and all scenes in the LLFF test set. These results are shown in Table 8. Finally, qualitative examples from visualizing the calibrated and uncalibrated uncertainty maps are included in Figure 5.

Comparison to DANE [29]. In the sparse-view setting, where uncertainty estimation is crucial, our approach results in better uncertainties and image quality than the state-of-the-art uncertainty estimation method, Density-aware NeRF Ensembles (DANE) [29], as shown in Figure 6 and Table 2.

Uncertainty-guided NeRF View Enhancement. The proposed method for enhancing views rendered by the pre-trained NeRF model improves the quality of the predicted images for the LLFF [14] *Flower* scene both quantitatively (see Table 3) and qualitatively (see Figure 7).

Table 3. The proposed Uncertainty-guided NeRF View Enhancement method improves all metrics over the original NeRF renderings (top row) and using the uncalibrated uncertainties (2nd row).

	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Geom. Avg.(↓)
FlipNeRF	20.25	0.629	0.216	0.119
Ours (Uncal.)	20.26	0.630	0.214	0.119
Ours (Cal.)	**20.29**	**0.634**	**0.209**	**0.117**

Next-best View Selection. In Figure 6 (b), we show that the uncertainties calibrated by the proposed meta-calibrator can be leveraged to select more informative views for training, resulting in more significant and efficiently realized performance gains than DANE [29]. Fitting the calibration

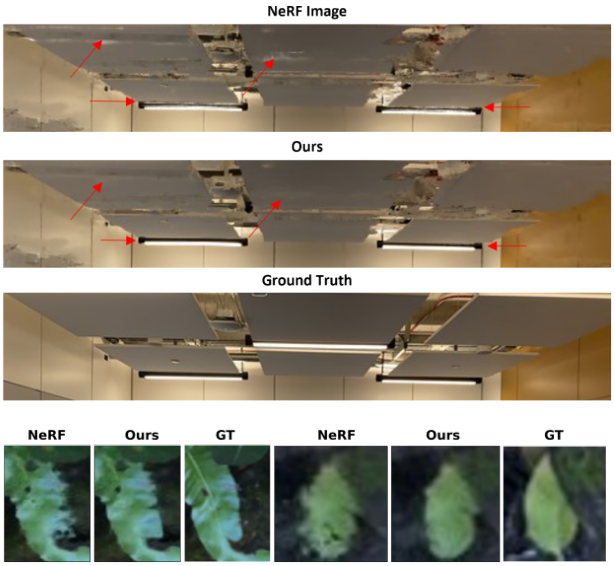

Figure 7. Uncertainty-guided NeRF View Enhancement of predicted images from the *Room* (top) and *Flower* (bottom) scenes in LLFF [14]. Our proposed method replaces unnatural textures and fills in holes with more realistic and semantically meaningful pixels.

model in each step of next-best view planning takes 2m40s with patch sampling using 4x RTX-4090's, whereas estimating the calibration model with the meta-calibrator takes 0.1s for each step. Taking into account the time for training the scene model, this equates to a 50% speedup.

5. Conclusion

In this paper we addressed the problem of obtaining calibrated uncertainties from NeRF models. Our approach achieves state-of-the-art uncertainty while maintaining image quality in the sparse-view setting. We provide two novel techniques that trade off accuracy and efficiency to address the lack of views available. The first method synthesizes a held-out recalibration set using square patches sampled from the training views, while the second method trains a meta-calibrator to infer the calibration curves from scene features with no access to ground-truth data.

We demonstrate the effectiveness of our approach on the 3-shot novel view synthesis LLFF [14] dataset as well as on uncertainty-guided NeRF view enhancement and next-best view selection. Our method represents a significant step forward in the practical application of NeRF to real-world scenarios. By enabling efficient and accurate calibration of NeRF models without relying on extensive GT data, we open up new avenues for the use of NeRF in situations where data is limited and uncertainty is critical.

Acknowledgements. The authors would like to thank Seunghyeon Seo for his support on FlipNeRF, and Oishi Deb for her insightful feedback on uncertainty estimation. Niki Amini-Naieni is funded by an AWS Studentship, the Reuben Foundation, and the AIMS CDT program at the University of Oxford. Tomas Jakab is sponsored by EPSRC VisualAI EP/T028572/1 and Andrea Vedaldi by ERC-CoG UNION 101001212.

Ethics. We used the Realistic Synthetic 360° dataset [15], the subset of scenes in DTU [7] used in [24], and all the scenes in LLFF [14] following their terms and conditions. There is no personal data. For further details on ethics, data protection, and copyright please see https://www.robots.ox.ac.uk/~vedaldi/research/union/ethics.html.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5835–5844, 2021. 2, 6

[2] Yash Bhalgat. Hashnerf-pytorch. https://github.com/yashbhalgat/HashNeRF-pytorch/, 2022. 6

[3] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *CVPR*, pages 8649–8658, 2021. 2

[4] Biraja Ghoshal and Allan Tucker. On calibrated model uncertainty in deep learning. In *ECML*, 2022. 2

[5] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' Rays: Uncertainty quantification in neural radiance fields. *ArXiv*, abs/2309.03185, 2023. 2

[6] Ajay Jain, Matthew Tancik, and P. Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, pages 5865–5874, 2021. 2

[7] Rasmus Ramsbøl Jensen, A. Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 5, 6, 9

[8] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *SIGGRAPH*, page 165–174, 1984. 2, 3

[9] Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sovna Mokr'a, and Danilo Jimenez Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *ICML*, 2021. 2

[10] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *ICML*, pages 2796–2804, 2018. 2, 4, 6

[11] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NIPS*, 2020. 2

[12] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022. 5, 6, 14

[13] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2

[14] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. In *TOG*, 2019. 2, 6, 7, 8, 9, 12, 13, 14, 15, 16

[15] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3, 6, 9

[16] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *ACM Trans. Graph.*, 2022. 6

[17] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, and M. Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *CGF*, pages 45–59, 2021. 2

[18] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 2

[19] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 5, 11, 15

[20] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, 2021. 2

[21] Yunlong Ran, Jing Zeng, Shibo He, Lincheng Li, Yingfeng Chen, Gim Hee Lee, Jiming Chen, and Qianru Ye. Neurar: Neural uncertainty for autonomous 3d reconstruction. In *RAL*, 2023. 2

[22] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, pages 14148–14156, 2020. 2

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. pages 211–252, 2015. 6

[24] Seunghyeon Seo, Yeonjin Chang, and Nojun Kwak. Flipnerf: Flipped reflection rays for few-shot novel view synthesis. In *ICCV*, 2023. 1, 2, 3, 4, 6, 9, 13, 14

[25] Seunghyeon Seo, Donghoon Han, Yeonjin Chang, and Nojun Kwak. Mixnerf: Modeling a ray with mixture density for novel view synthesis from sparse inputs. In *CVPR*, pages 20659–20668, 2023. 2, 3

[26] Jianxiong Shen, Adria Ruiz, Antonio Agudo, and Francesc Moreno-Noguer. Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. In *3DV*, pages 972–981, 2021. 2, 4, 13

[27] Jianxiong Shen, Antonio Agudo, Francesc Moreno-Noguer, and Adria Ruiz. Conditional-flow nerf: Accurate 3d modelling with reliable uncertainty quantification. In *ECCV*, 2022. 2, 4, 6, 13, 14

[28] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NIPS*, 2020. 2

[29] Niko Sünderhauf, Jad Abou-Chakra, and Dimity Miller. Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields. In *ICRA*, 2023. 2, 4, 6, 7, 8, 13

[30] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *CVPR*, 2023. 2

[31] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. In *TIP*, 2004. 6, 14

[32] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *ArXiv*, abs/2010.07492, 2020. 2

[33] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

A. Meta-calibrator Design

The goal of the meta-calibrator is to predict the calibration curve needed to correct the model's confidence using only data from the model. In this section we give more details of the meta-calibrator's design.

A.1. Model Details

In Figure 8 we detail the two main stages of the meta-calibrator. We use a pretrained DINOv2 [19] model and an MLP with 3 layers of output size: $[128, 128, 3]$ with leaky ReLU activations throughout except the last layer.

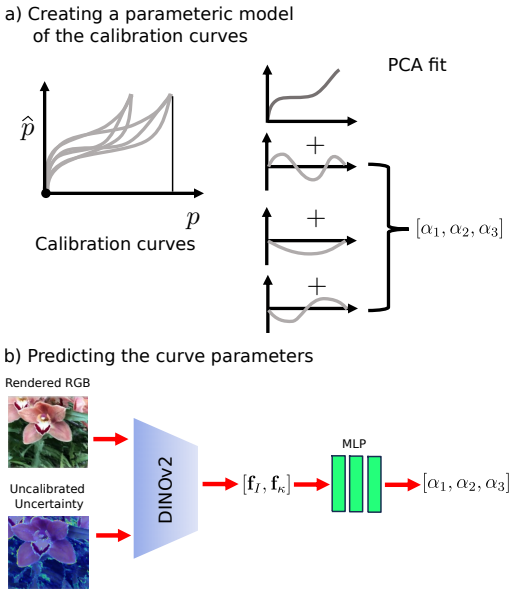

Figure 8. **Further illustration of the meta-calibrator design:** In stage (a) we fit a low-dimensional parameteric model of the calibration curves. The meta-calibrator then predicts these curve parameters from rendered images of the scene and their associated uncalibrated uncertainty maps (b).

A.2. Optimal Number of Coefficients

A key aspect of the meta-calibrator design is determining the optimal number of PCA coefficients to effectively capture the calibration curve's essence without overfitting. To this end, we use an *explained variance* plot, which graphs the cumulative explained variance against the number of PCA components. This plot provides a visual representation of the variance in the calibration data that each additional PCA component accounts for (see Figure 9). As the number of components increases, the rate of increase in explained variance diminishes, indicating diminishing returns. In our analysis, we observed a clear elbow at the 3rd component, where the explained variance reached a plateau, suggesting that additional components contribute minimally to explaining the data. This inflection point guided our decision to select three PCA coefficients for the meta-calibrator.

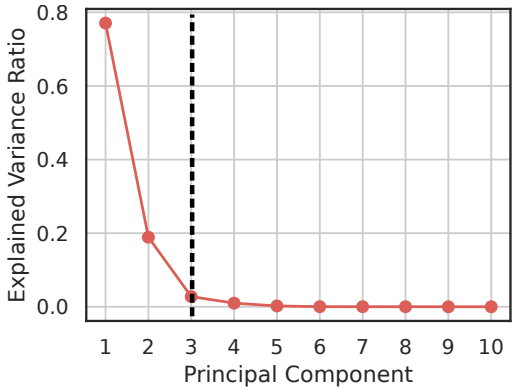

Figure 9. **Optimal number of coefficients for calibration curve model:** This figure shows that most of the variance of the calibration curves is explained using only three PCA components. We therefore use three components in our experiments.

A.3. Number of Scenes for PCA Fit

Figure 10 shows the error in the PCA approximation as a function of the number of training scenes used to construct the PCA representation of the calibration curves. As can be seen in the figure, we observe a diminishing return in the reduction of approximation error with an increasing number of scenes beyond 21. This indicates that a relatively small dataset can be effectively utilized to fit the PCA.

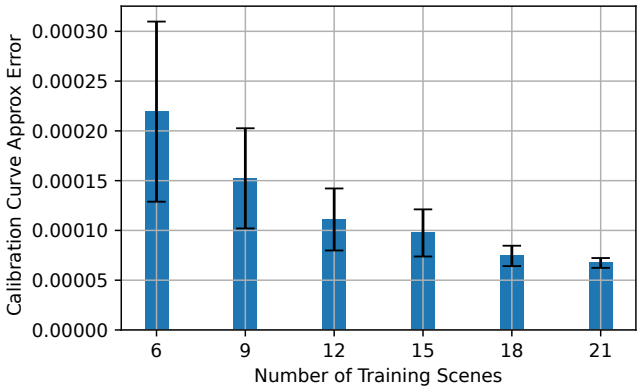

Figure 10. **Impact of number of training scenes on calibration curve approximation:** Here we show the error of the 3-coefficient curve approximation as a function of the number of training scenes. By using only 21 scenes, the error of the PCA approximation of the test curves is sufficiently low to indicate a good approximation (i.e. it's an order of magnitude lower than we can expect from the final calibration).

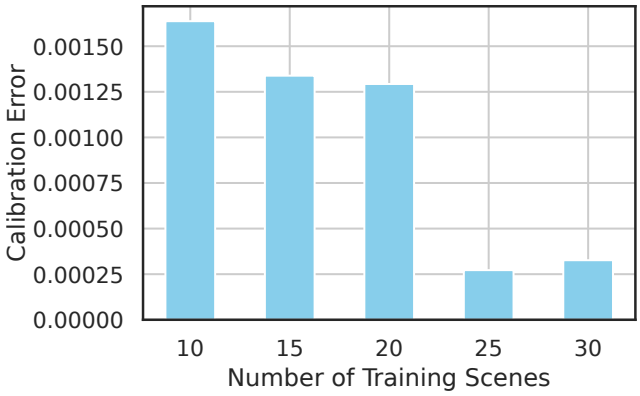

Figure 11. **Effect of the number of training Scenes on meta-Calibrator error:** The graph demonstrates that increasing the number of training scenes leads to a consistent decrease in the meta-calibrator error, highlighting the benefits of using larger datasets for training. This test was conducted on the *trex* scene from the LLFF dataset.

A.4. Number of Scenes for Training Meta-calibrator

Here we investigate the influence of the number of training scenes on the performance of our proposed meta-calibrator. To quantify this relationship, we conducted a series of experiments with varying numbers of training scenes, ranging from 30 to larger datasets. The results are shown in Figure 11, which clearly illustrates the inverse relationship between the number of training scenes and the meta-calibrator error for the *trex* scene from the LLFF dataset [14]. Our results indicate that a relatively small dataset comprising 30 scenes is sufficient for the meta-calibrator to generalize effectively to new test scenes. This finding suggests that the calibration information can be successfully extracted from the rendered RGB images and uncalibrated uncertainty maps using the DINO features. However, we did observe that the test performance is dependent on the type of scenes used in training. This means if the test domain consists of indoor scenes, it's beneficial to use these types of scenes during training.

A.5. Why is the PCA Representation Necessary?

One might wonder why the PCA parameterization of the curves is necessary - why not simply predict a discretized representation of the curve directly? In essence, the PCA parameterization of the calibration curves allows us to simplify the complex, high-dimensional data into a low-dimensional, manageable form. This approach is favored over direct prediction of the calibration curve primarily because it is difficult to predict a high-dimensional output without a large amount of training data. The low-dimensional representation therefore improves the model's generalization capabilities for new scenes. To demonstrate this, we show an example where we compare predicting the PCA coefficients to directly predicting a discretized 384-dim representation of the curve (with an MLP of size [128,128,384]). From 12 we see that the direct prediction leads to a noisier curve with higher error.

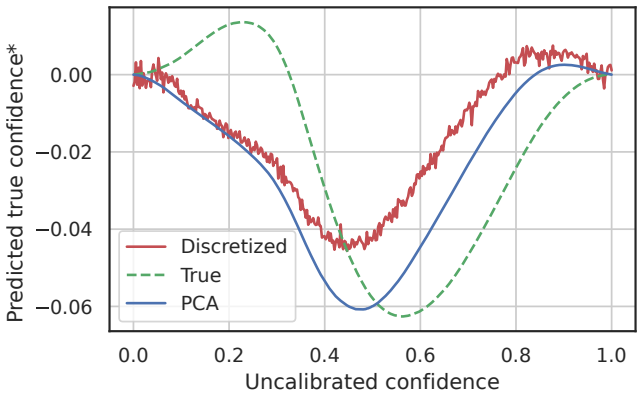

Figure 12. **PCA representation vs direct prediction of the calibration curve.** Here we show an example *mean-normalized* (*) calibration prediction for the *fern* scene from the LLFF dataset. The low-dimensional PCA parameterization allows the model to generalize better and preserve the characterisitics of the true calibration curves.

B. Does Calibration Preserve the Order of the Uncertainties?

One might think that the order of the uncertainties is preserved by calibration as we're fitting a monotonic curve to the expected confidences. However, this is not the case. *Rather than preserving the order of the uncertainties with respect to the pixels, the calibration preserves the monotonicity of the individual CDFs at each pixel.* To elucidate this concept, consider an example where calibration can reverse the order of uncertainties for two pixel CDFs as shown in Figure 13.

Initially, the CDF for ray 1, corresponding to pixel 1, might indicate higher uncertainty compared to ray 2 (pixel 2). However, after applying the calibration process, the order of uncertainties can be reversed. This reversal is attributed to the differing shapes and slopes of the CDFs, which are altered non-linearly during calibration. The implications of this observation are significant. It underscores the non-trivial nature of the calibration process in uncertainty modeling and suggests that calibration does not merely scale or shift uncertainties but can fundamentally alter the relation between the uncertainty values. In summary, this highlights the complexity and nuanced impact of calibration on the predicted uncertainties.

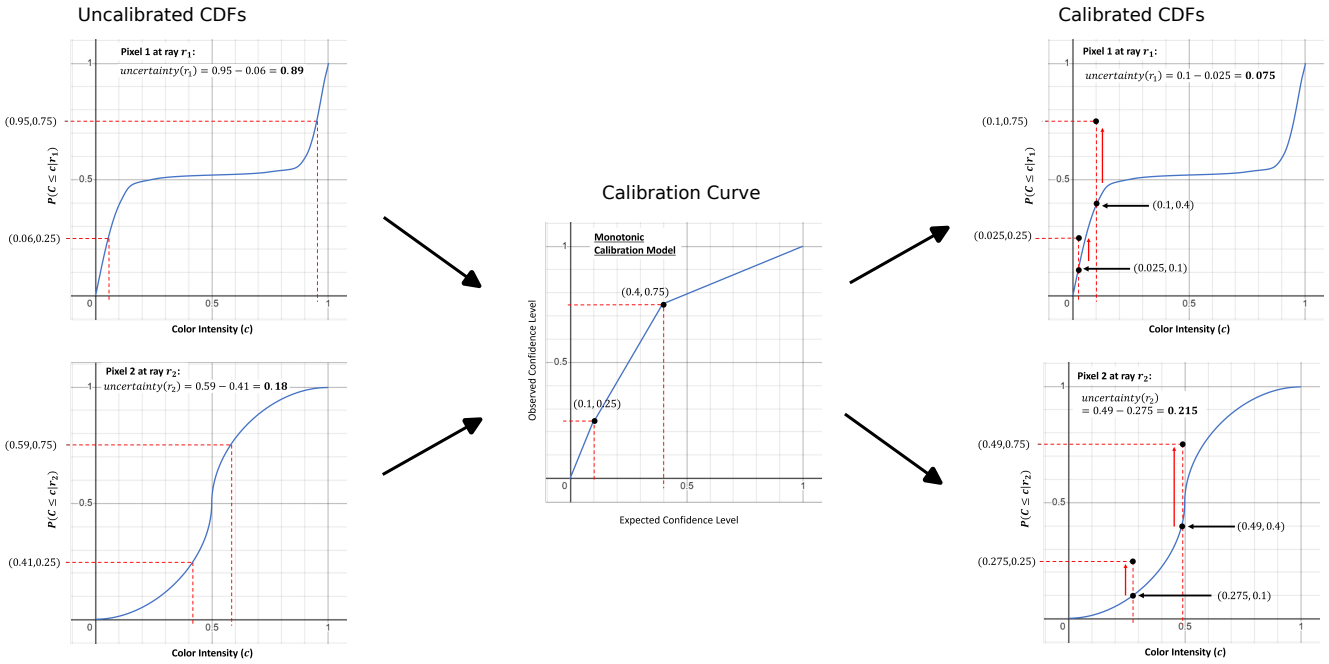

Figure 13. **Is the order of uncertainties necessarily preserved during calibration?** This illustration shows that the order of uncertainties for two pixels (corresponding to rays 1 and 2) are not necessarily preserved during calibration. We start with $uncertainty(r_1) > uncertainty(r_2)$ for the left two uncalibrated CDFs and end up with $uncertainty(r_1) < uncertainty(r_2)$ after calibration on the right.

C. Standard NeRF Uncertainty Estimation Benchmark

In our main paper, following prior work [24] on novel view synthesis from a sparse set of training views, we report results on 3-view LLFF [14]. Uncertainty estimation from three views is more difficult than it is in the standard NeRF uncertainty estimation setting used by [26, 27, 29], where 4-12 views are available. However, one may wonder how our uncertainties compare to the uncertainties from [26, 27, 29] in this standard setup. To answer this question, in this section, we follow the evaluation procedure of [29] and report the negative log-likelihood obtained from our method on 80 % of the views available while training on the remaining 20 %. As shown in Table 4, our approach significantly beats the state-of-the-art uncertainty estimation method Density-aware NeRF Ensembles (DANE) [29] for all scenes in LLFF.

D. Efficiency of Uncertainty Calculation

In this section, we provide further details on why we use the interquartile range, rather than the variance or standard deviation, to quantify the uncertainty at each pixel. While the variance of a Laplacian mixture model can be obtained in closed form from the parameters of the component distributions, in our approach, the parameters of the *calibrated* CDF (e.g., the location and scale parameters of the component CDFs) are not known. Hence, to obtain the variance of the distribution for each pixel, we would either need to sample from it or differentiate the predicted CDF to obtain the corresponding PDF and then integrate to estimate the variance. As shown in Table 5, both of the aforementioned methods are much slower than estimating the interquartile range. This is because the interquartile range can be calculated from the calibrated CDF directly.

E. Information Gain in Next-best View Selection

In the main paper, we consider the problem of next-best view planning, where an active agent picks novel views to train on. To show that calibration, specifically, helps an agent select views that have the most potential for improving the NeRF's performance, we compare the information gain from rays selected according to the highest calibrated uncertainties to the information gain from rays selected according to the highest uncalibrated uncertainties. Specifically, for evenly spaced fractions $\gamma_i \in [0, 0.5]$, we plot the average PSNR over the test set assuming the top $100\% \times \gamma_i$ most uncertain pixel colors are predicted perfectly by the NeRF model. We use the updated PSNR to quantify information gain. Intuitively, better uncertainties should result in selecting pixels with higher information gain. For all fractions and all scenes in LLFF [14], the calibrated uncertainties produce higher average PSNRs on the test set. The resulting plots for

Table 4. **Comparison to SOTA on standard NeRF uncertainty estimation benchmark.** Our proposed approach using the meta-calibrator achieves significantly better negative log-likelihoods on the test sets for all scenes in LLFF [14] than existing state-of-the-art NeRF uncertainty estimation methods do. M indicates the number of ensemble members, and MC-DO refers to Monte Carlo Dropout sampling with M sample configurations. Note, the NLL calculation for these results differs from the one used in Table 2 of the main paper. Here, following [27], the NLL is averaged over the color channels and all the rays in the test set. In Table 2 of the main paper, the joint NLL of the color channels averaged over all the rays in the test set is reported.

	Negative Log-likelihood (↓)								
Scene	# of Train. Views	MC-DO $M = 5$	Naïve Ens. $M = 5$	NeRF-W	S-NeRF	CF-NeRF	DANE $M = 5$	DANE M = 10	**Ours Meta-cal**
Flower	7	4.63	1.63	1.71	1.27	—	1.00	0.85	**-2.05**
Fortress	8	5.19	2.29	1.04	-0.03	—	-1.30	-1.30	**-1.99**
Leaves	5	2.72	2.66	0.79	0.68	—	0.97	0.73	**-1.19**
Horns	12	4.18	2.17	0.78	0.60	—	-0.55	-0.66	**-2.18**
T-Rex	11	4.10	2.28	1.91	1.37	—	-0.31	-0.69	**-1.49**
Fern	4	4.90	2.47	2.16	2.01	—	-0.98	-1.00	**-1.41**
Orchids	5	5.74	2.23	2.24	1.95	—	-0.28	-0.31	**-0.84**
Room	8	5.06	2.13	4.93	2.35	—	-1.35	-1.35	**-2.17**
Avg.		4.57	2.23	1.95	1.27	0.57	-0.35	-0.47	**-1.67**

Table 5. **Timing of obtaining different uncertainty measures for the distribution of 1 pixel.** Calculating the interquartile range is much faster than calculating the variance.

Uncertainty Metric	Method	Time (s)
Variance	Integration	9.807
Variance	Sampling	1.759
Interquartile Range (Ours)	**Interpolation**	**0.008**

two scenes in LLFF are included in Figure 14.

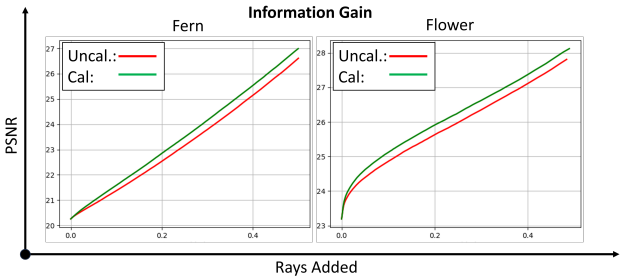

Figure 14. **Information gain from uncalibrated and calibrated uncertainty-guided ray selection.** Picking rays according to the calibrated uncertainties (green) consistently results in higher PSNRs than picking rays according to the uncalibrated uncertainties (red).

F. Further Results on Uncertainty-Guided NeRF View Enhancement

In this section, we present additional results to demonstrate the effectiveness of our uncertainty-guided NeRF approach for view enhancement. Our method significantly enhances the quality of rendered images, as evidenced by the Structural Similarity Index Measure (SSIM) [31] comparisons with the original FlipNeRF [24] renderings.

Figure 15 shows an overview of the process of our view enhancement technique. Initially, FlipNeRF generates a base rendering, which is then processed through our method to obtain calibrated uncertainties. These are then used with a DDPM [12] to "correct" the uncertain pixels. This process effectively refines the visual details and corrects any inaccuracies in the initial rendering.

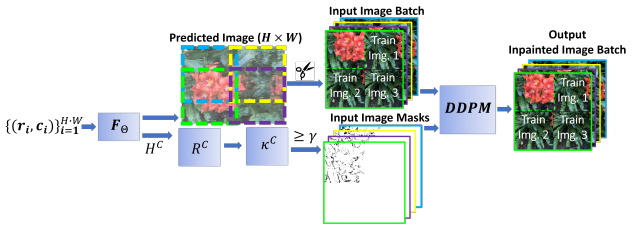

Figure 15. **Overview of proposed Uncertainty-Guided NeRF View Enhancement.** The NeRF model F_Θ outputs a predicted image and an uncalibrated forecaster H^C for each color channel C. Our approach produces calibrated uncertainties κ^C by fitting a calibrator R^C to confidence levels output by H^C. Pixels corresponding to calibrated uncertainties above a threshold γ (determined using a validation scene) are then masked and inpainted using a DDPM [12] conditioned on the training images.

Table 6. **Uncertainty-guided NeRF view enhancement.** With our calibrated uncertainties we can correct uncertain regions of the scene using a DDPM. This improves the SSIM of the original FlipNeRF renderings on seven out of the eight scenes in LLFF [14]. We use *room* as the validation scene in these results.

	SSIM (↑)	
Scene	FlipNeRF	Ours
Room	0.793	**0.797**
Fern	0.648	**0.651**
Flower	0.629	**0.634**
Fortress	0.671	**0.672**
Horns	0.580	**0.585**
Leaves	**0.528**	0.514
Orchids	0.464	**0.471**
Trex	0.730	**0.731**

Table 6 presents a quantitative comparison of SSIM values between the original FlipNeRF renderings and our enhanced views for eight different scenes. Our method shows

noticeable improvements in SSIM scores for most scenes, indicating a higher level of structural similarity to the ground truth. The only exception is the "Leaves" scene, where our method slightly underperforms, which could be due to the difficulty the DDPM has in completing scenes with highly complex textures and structures.

Overall, these results validate the efficacy of our uncertainty-guided approach in enhancing the visual quality of NeRF-rendered views. By incorporating calibrated uncertainties, our method not only preserves the structural integrity of the scenes but also enhances the fine details, leading to more realistic and visually pleasing renderings.

G. Comparing Calibration Fitting Procedures

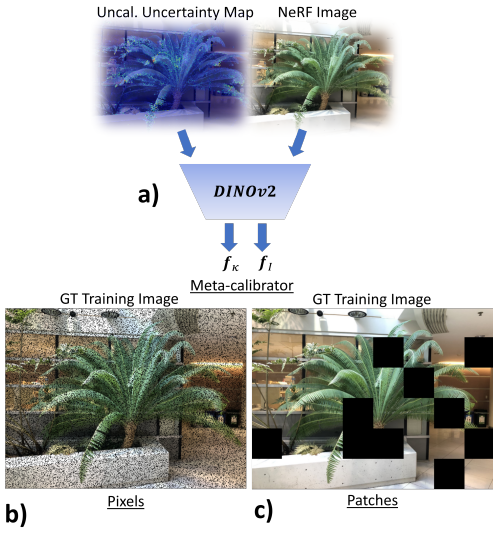

Figure 16. **Different approaches to addressing a lack of held-out data.** In our paper, as shown in (a), we propose training a meta-calibrator on a set of training scenes to predict the calibration curves of a test scene from the DinoV2 [19] features of the uncalibrated uncertainty map and the predicted NeRF image. We also propose holding out patches from the training images and fitting a calibrator to the held-out rays as illustrated in (c). In (b), we additionally depict holding out *pixels* instead of *patches*, and in Table 7 show that this is inferior to our proposed approaches in (a) and (c).

In this section we provide a more detailed comparison of the three paradigms for fitting the calibration model using either: 1) **the training set**, 2) **held-out data** or 3) **no held-out data**. The **no held-out data** approach comprises of the meta-calibrator and patch sampling proposed in this paper - illustrated again in Figure 16 for easy comparison. The main results of this comparison are shown in Table 7.

G.1. Using the Training Set

One might consider fitting the calibrator on the training set. To show why this will not work, in this section, we present the calibration curves for the red, green, and blue color channels of the training rays for four scenes in LLFF [14] in Figure 17. These curves reveal that not only are the confidence levels of the pretrained NeRF model miscalibrated for the training set, but the pattern they follow is different from the one observed in the test set. The model is consistently overconfident for confidence levels closer to zero and underconfident for confidence levels closer to one. Thus, calibration using the training set would result in very poor generalization to held-out data. Interestingly, the pretrained NeRF model was trained with a loss that incorporates the negative log-likelihood of the training data. Despite this, the confidence levels are still miscalibrated.

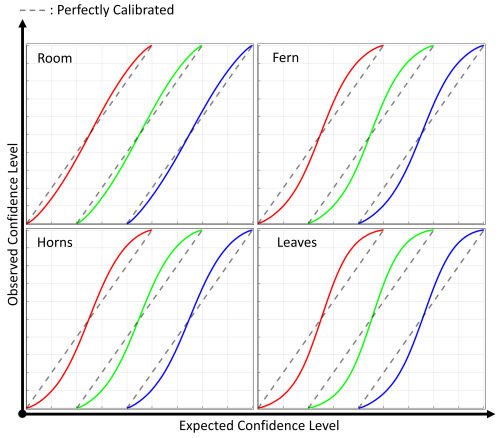

Figure 17. Calibration curves for training data from four scenes in LLFF [14]. The color of each curve indicates the color channel it corresponds to. The solid red, green, and blue curves are not closely aligned with the grey dashed lines in general, showing that the pretrained NeRF model is miscalibrated, even for the training set. Interestingly, the expected confidence levels for the training set follow a different pattern from the one followed by the uncalibrated confidence levels of the test set (see Figure 4 of main paper).

G.2. Using No Held-out Data

As shown in Section G.1, using the training set to fit the calibrator will not allow it to generalize to held-out data. We also consider sampling pixels to train the calibrator on (see Figure 16 (b)), rather than patches (see Figure 16 (c)), but show in Table 7, this leads to poor performance. Sampling patches results in significantly lower calibration errors because the NeRF is provided with less context to accurately fill in the masked regions than it is provided with pixels. These methods do not use held out data as we train the final NeRF model on the whole image. This is unlike in Section G.3 where we do not train the NeRF on the held-out data.

Finally, as can be seen in Table 7, the meta-calibrator also leads to significantly lower calibration error, with the added advantage that it does not require training an additional NeRF

model for the scene, or GT data for the calibration.

Table 7. **Average calibration error for different methods of dealing with no held-out data.** Note, * indicates that the meta-calibrator does not require training an additional model for the target scene, while the method with patches does. Holding out pixels results in worse calibration accuracy, while both patch sampling and the meta-calibrator improve the calibration accuracy. Patch sampling achieves the lowest calibration error, but is less efficient at calibrating new scenes than the meta-calibrator is.

	Cal. Err.
Uncal.	0.0086
Pixels	0.0213
Meta-calibrator*	0.0026
Patches	**0.0014**

G.3. Using Held-out Data

In this section, we consider the case when held-out data is available for fitting a calibrator. For instance, in safety-critical scenarios such as medical applications and autonomous driving, the uncertainty of estimates may be more important than the accuracy of the model. In these cases, it might be desirable to hold out data from the training set to improve the quality of the uncertainty estimates. As shown in Table 8, additional held-out data results in lower calibration errors overall. However, our proposed patch sampling approach and meta-calibrator achieve lower calibration errors for four out of the eight scenes in LLFF [14], further demonstrating that our methods for addressing a lack of held-out data work very well.

Table 8. **Calibration errors for test data from the eight scenes in LLFF** [14]. The calibration errors are averaged over the RGB color channels. The * indicates that the meta-calibrator does not require training an additional model for the target scene, while the patches method does.

Scene	Uncal.	Meta-cal.*	Patches	Held-out Data
Room	0.0245	0.0111	**0.0012**	0.0047
Fern	0.0041	0.0005	**0.0002**	0.0009
Flower	0.0112	0.0013	0.0008	**0.0001**
Fortress	0.0013	0.0024	**0.0005**	0.0007
Horns	0.0085	0.0016	0.0031	**0.0005**
Leaves	0.0024	0.0028	0.0019	**0.0004**
Orchids	0.0070	0.0006	0.0023	**0.0002**
Trex	0.0097	**0.0008**	0.0010	0.0022

H. The Code

We plan to release the full code for our method for reproducibility.