InjectNeRF: a simple yet effective method for measuring and visualising epistemic uncertainty in few-shot novel view synthesis

Maja Drmač

Master of Science
School of Informatics
University of Edinburgh
2024

Abstract

This project introduces a novel method for displaying uncertainty of few-shot novel view synthesis models, with a focus on approaches that use neural radiance fields as a base. Leveraging inference-time dropout, our approach quantifies uncertainty without additional training costs, while improving rendering fidelity across various datasets. We support our design choices using theoretical reasoning, with tools from information theory and statistics, as well as empirically through a variety of experiments.

The results show that our method has strong positive correlation (formally quantified using rank correlation coefficients) with the output error, which is a key trait of a good uncertainty measure. We also discuss how our contribution can be utilised beyond this report for further improvements in output quality.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Maja Drmač*)

Acknowledgements

The past year has been the most challenging experience of my life so far – I had the opportunity to learn so much from so many talented individuals, both on a scientific and personal level. An important lesson I am taking from this degree is to approach every difficulty with hope and determination. No matter how impossible it may seem, great things come to those who persevere, and I am proud of my small contribution to the Informatics community with this thesis.

Huge thank you to my supervisor, Dr Luo Mai, for trusting me to be a part of this interesting and fulfilling project. I also express my deepest gratitude to Dr Chuanhao Sun for his guidance, patience and the deep and detailed discussions about all of our ideas. Finally, thank you to my family, friends and boyfriend who were there for me in all my highs and lows – I could not have done it without you.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Novel view synthesis (NVS) is the process of generating new, unseen views of a scene from a limited set of existing images or views. The ability to faithfully generate high-fidelity novel views is an important research problem in computer vision and beyond: it can be utilised for inferring scene information in robotics for navigation path planning or object manipulation, as well as for enhancing visualisation and 3D scene representation in augmented and virtual reality [3]. Its applications are expanding daily and are especially relevant in the era of generative AI. The current challenge in the field is few-shot novel view synthesis: although we have limited information about the scene, we still require rendering a full, faithful representation. While it is clear that with fewer views some areas of the scene might not be fully captured, our goal is to infer and approximate the function representing the whole scene. A simple illustration of a novel view sythesis task is shown in Figure 1.1.

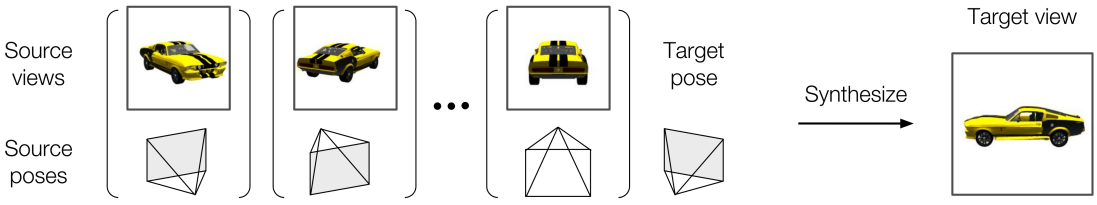


Figure 1.1: An example of a multi-view novel view synthesis task [4]. Given multiple source views, we aim to synthesise a target image with an arbitrary target pose only from given source images.

Novel view synthesis models work under the tacit assumption that the correct functions that are needed to generate new views can be uniquely determined from the

sparse set of given views. However, unseen views do not necessarily consist of the same frequency features. The known views can possibly be represented with different sets of features, so the set that a NVS model learns perfectly from the training process might not be the best one for the unseen views. Often the difference in performance on the training set versus the test set is drastic and highly noticeable. This suggests that the common hypotheses on the properties of the function that represents the 3D scene do not truly capture the underlying structure.

Furthermore, an interesting phenomenon observed in state of the art (SOTA) models for few-shot novel view synthesis is that different configurations of the model – for example, different error functions, adding adversarial training using networks with various structures, or even different weight initialisations – can yield SOTA performance for certain views. Marginal gains in performances often cannot be prescribed to the overall improvement of the solution, but rather the particular environment of the experiments: different random seeds, different GPUs or versions of libraries/software.

In summary:

- *the model that performs the best on average does not perform the best in every view*, and
- *a model that performs the best in one computational environment does not necessarily perform the best in a different one*.

Therefore, understanding the different outputs and corresponding performances a model can achieve, and exploring (un)certainty in its outputs becomes an interesting problem of high value, especially in fields where correctness and accuracy of the novel views are crucial. However, since novel view synthesis models are deterministic (evaluating each input multiple times in a constant environment will give the same output), they are unable to quantify this uncertainty associated with their learned functions. Quantifying the uncertainty contained in a NVS model is a relatively new area of study, and existing methods either use probabilistic techniques and/or elaborate changes to the conventional NeRF training pipeline that require costly computational power [5].

1.2 Aims and objectives

The goal of this project is to design a simple, yet effective uncertainty measurement technique of the SOTA performing model family (neural radiance fields). We are interested in analysing the variations we can produce, and what kind of uncertainty does

the network present in its key components, as this provides meaningful insight into the limitations of models for few-shot novel view synthesis. We expect that unseen views share certain features with known views, but different feature/function sets might share more than the others.

Our key objectives, on a high level, can be defined as follows.

- Integrate an uncertainty measurement method to our baseline model such that the features and functions perform perfectly on known views, on par with the baseline.
- Design and implement the optimisation algorithm of the method's hyperparameters, and support it with analytical reasoning.
- Evaluate the method's performance against the baseline, and interpret the results to devise potential future research directions.
- Verify the uncertainty measurement method by relating it to the errors of the model's outputs against the ground truth and using relevant statistical metrics.

Our contribution can be highlighted in several ways. As mentioned, there have not been many publications that have attempted to quantify uncertainty within the most popular novel view synthesis models, but as of recently it is a research area in rising [6, 5]. We also found that no previous work on uncertainty in novel view synthesis, particularly within the model family of the SOTA model, considered this approach. This makes a valuable addition to the relevancy and expected impact of this thesis. Furthermore, our method is model-agnostic, so it can be applied to other NVS models besides our baseline to produce similar results. Beyond quantifying uncertainty, we found that a large proportion of the results produced by our modifications outperform the baseline, showing promise that the method will be used to further enhance few-shot NVS models.

Finally, we aim for this work to be used as a comprehensive guide to model uncertainty and the current challenges regarding its quantification and measurement.

1.3 Thesis structure

First, we delve into the technical background relevant to the research, which is split into two chapters: introduction to NVS and introduction to uncertainty estimation. Chapter 2

provides an overview of the foundational concepts in 3D scene representation, focusing on NeRFs (Neural Radiance Fields). We also discuss the main challenges that NVS models face in few-shot scenarios, and how techniques such as Positional Encoding enhance the learning process. We also introduce our baseline model on which we will apply our modification. In Chapter 3, as a continuation of the background, we present the theoretical framework needed to understand quantifying uncertainty in few-shot NVS models. We explain the mathematical interpretation of uncertainty, including the two main types of model uncertainty and how this relates to our method. Related work in this domain is also reviewed, providing context to why a novel solution is needed.

Chapters 4 details the methodology employed in the thesis, beginning with an introduction to our novel uncertainty measurement method based on inference-time dropout. We elaborate on our implementation details and key design choices such as dropout rate optimisation and position, as well integrating the uncertainty into the model output. We also introduce our testing environment: the dataset and evaluation metrics.

Following our solution design, we discuss the experimental setup used to evaluate the proposed uncertainty quantification method in Chapter 5. The results are analysed using the predefined fidelity metrics, and we analyse the impact of different dropout configurations. We also present a visualisation of uncertainty and its correlation with model performance, demonstrating the effectiveness of the method in enhancing NVS models.

Finally, Chapter 6 summarises the key findings of the research and discusses the broader implications of the proposed method for few-shot NVS models. We outline potential directions for further research and improvements in uncertainty quantification. The appendices provide additional technical details, including implementation specifications, theoretical work, and results of the supplementary experiments. These materials are intended to facilitate the understanding of the results and to offer deeper insights into the experiments conducted during the thesis.

Chapter 2

Background

This chapter serves as both an introduction to NVS models, and as a deeper dive into the importance of our work and the relevant progress made in the field. First, we provide an overview of current solutions in NVS, focusing on the two main interpretations of 3D scene representation: Neural Radiance Fields (NeRF), and 3D Gaussian Splatting. We then discuss the current challenges with adapting these models for sparse inputs, and use this discussion to navigate our choice of baseline model. Finally, we summarise the potential applications of uncertainty for novel view synthesis models, the most recent publications that have tackled similar research questions to us and their flaws.

2.1 3D Scene representation in NVS

2.1.1 NeRFs

One of the major breakthroughs in view synthesis was made by [7] in 2020 with *neural radiance field scene representation* – NeRF. In NeRF, a scene is represented as a continuous volumetric field of particles which emit and block light. To learn the function of the field, it uses a multilayer perceptron (MLP) whose input consists of a location $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{R}^2$. NeRF renders each pixel of a camera as follows (see schema in Figure 2.1).

1. A ray $\mathbf{r}(t) = \mathbf{o} + t\,\overrightarrow{\mathbf{d}}$ is emitted from the camera's center of projection $\mathbf{o}$ along the direction vector determined from $\mathbf{d}$ such that it passes through the pixel.

2. A sampling strategy is used to determine a vector of sorted distances $\mathbf{t}$ between the camera's predefined near and far planes t_n and t_f. For each distance $t_k = \mathbf{t}(k)$, we compute its corresponding 3D position along the ray $\mathbf{x}_k = \mathbf{r}(t_k)$.

3. Each position is then transformed using a positional encoding $\mathsf{PE}(\cdot)$ (more on positional encodings is discussed in Section 2.2.1). The positional encoding of each ray position $\mathsf{PE}(\mathbf{r}(t_k))$ is provided as input to an MLP parameterised by weights $\mathbf{w}$, which we denote $F_{\mathbf{w}}$.

4. The output of the MLP is an emitted colour $\mathbf{c} = (r, g, b)$ and volume density α for each input pair $(\mathbf{x}, \mathbf{d})$. Finally, these function values are composited into an image using volume rendering techniques [8].

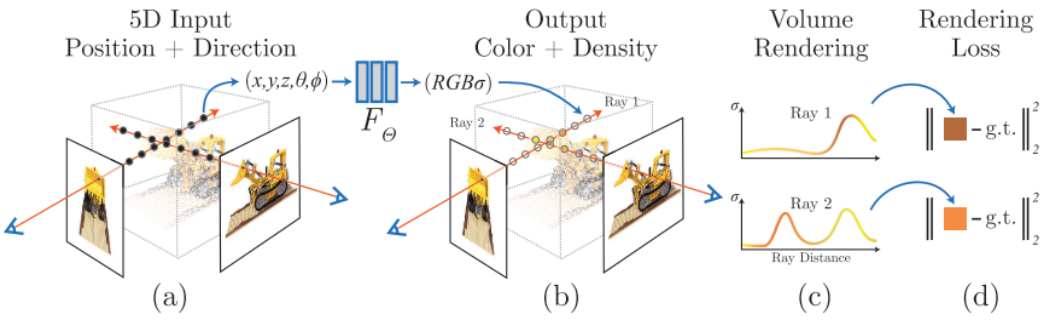


Figure 2.1: Pipeline of NeRF [7]. The 5D coordinates are sampled along camera rays and fed into a multilayer perceptron $F_{\mathbf{w}}$ (F_Θ on the schema) which outputs color (RGB) and volume density (σ). Using volume rendering techniques, this output is transformed into the output image.

The radiance field $\mathcal{F}$ describing the volumetric scene can be defined as

$$\mathcal{F} = \{(\mathbf{c}(\mathbf{x}, \mathbf{d}), \alpha(\mathbf{x})) : \mathbf{x} \in \mathbb{R}^3,\ \mathbf{d} \in \mathbb{R}^2\}, \tag{2.1}$$

where the volume density depends only on the spatial location. Since the publication of the original paper, many variations of NeRF have been presented – aiming to improve fidelity, runtime, memory consumption or generalisation to unseen objects. To this day, the fundamental approach of NeRF is a very popular research topic, and it has found applications beyond computer vision.

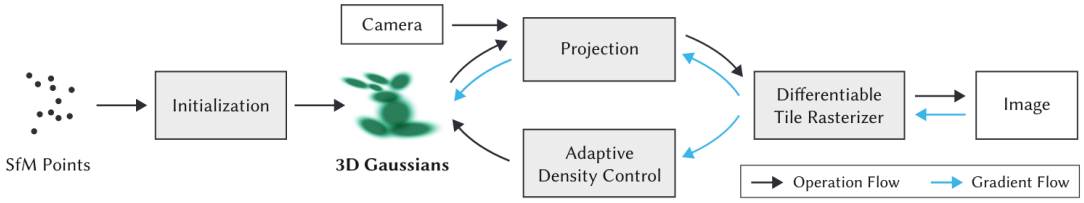
2.1.2 3D Gaussian Splatting

Another approach for view synthesis is 3D Gaussian Splatting (3DGS), a novel rasterisation technique that uses 3D Gaussians to represent scenes instead of continuous volumetric radiance fields. This means a scene is defined a set of millions of points, where each point is a 3D Gaussian with its own unique parameters that are fitted per

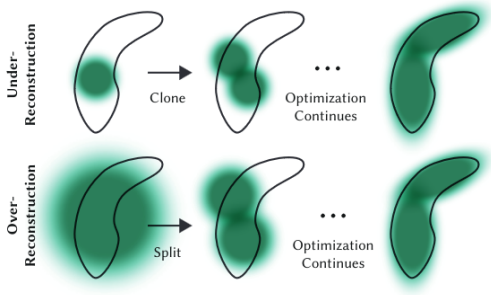
scene [9]. A 3D Gaussian is defined as

$$G(\mathbf{x}) = \exp\left(-\frac{\mathbf{x}^\intercal \Sigma^{-1} \mathbf{x}}{2}\right), \tag{2.2}$$

where Σ is the covariance matrix. Representing the Gaussians as discrete ellipsoids with opacity and view-dependent color information contributes to better efficiency, while the representation's differentiability makes it suitable for optimisation.



(a)



(b)

(c)

Figure 2.2: Pipeline of 3D Gaussian Splatting (Figure 2.2a). First, Structure from Motion is used to estimate a point cloud from a set of images, followed by converting each point to a Gaussian. The model is trained using SGD, differentiable Gaussian rasterisation and automated densification and pruning (pictured in Figure 2.2b). Example of a rendered view with each Gaussian rasterised fully opaque is shown in Figure 2.2c [9].

While 3DGS has significant training acceleration compared to NeRF and real-time rendering, it requires much higher memory capacity than NeRF-based models and it is not compatible with current, most popular rendering techniques (the official implementation suggests using the System for Image-Based Rendering viewer). Furthermore, it also struggles with low-detail, splotchy artifacts that are usually different from NeRF – more coarse and anisotropic, and especially noticeable in the case of sparse inputs like in Figure 2.3.

Figure 2.3: Example of a 3DGS rendering with 8 training views. The artifacts can be clearly seen in the center and bottom left.

2.2 Challenges with few-shot NVS

State of the art NVS solutions can achieve great fidelity and accuracy when given a large amount of input views – the original NeRF implementation assumes 100-150 inputs available per object [10]. However, these solutions struggle when it comes to few-shot NVS (e.g. 3 to 10 input views, as presented in Figure 2.4). Multiple modifications have been designed with the idea to require less inputs but retain output quality: leveraging extra information using external models, training on large, curated datasets, or geometry regularisation. These methods noticeably increase the computational complexity of the baseline NeRF model.

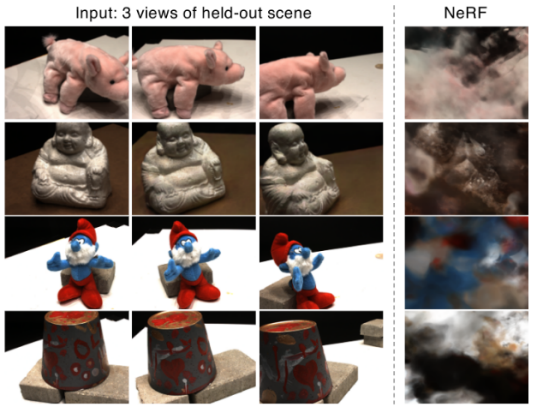


Figure 2.4: Example of NeRF's outputs when trained on three views per object [11].

A notable flaw of deep networks is the appearance of *spectral bias* – bias towards learning low frequency functions. Over-parameterised networks prioritise learning simple patterns that generalise across data samples[1]. This is especially evident in NeRF

[1]For a more detailed mathematical explanation on the spectral bias using Fourier analysis, see [12].

for few-shot scenarios, as the model is even more sensitive to susceptible noise since there are fewer images to learn coherent geometry.

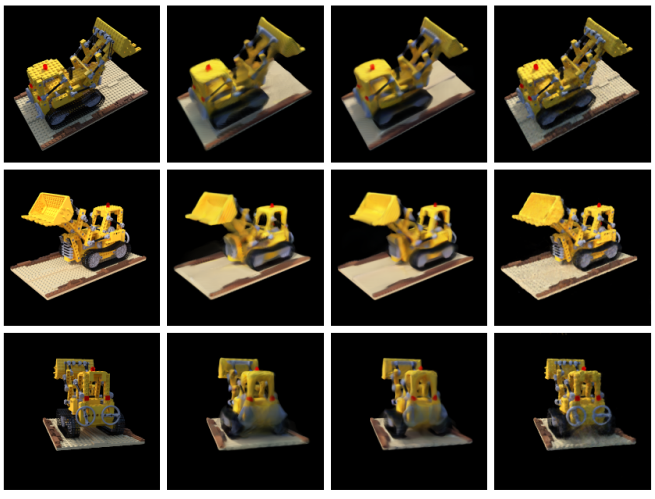
2.2.1 Positional encoding

One way NeRF heuristically allows its MLP to better fit data that contains high frequency variation is using positional encoding: instead of $F_{\mathbf{w}}$ operating directly on the input, the coordinates are first mapped to a higher dimensional space using a high frequency function, which in this case was a simple sinusoidal mapping $\mathsf{PE} : \mathbb{R}^d \to \mathbb{R}^{2L \cdot d}$ with

$$\mathsf{PE}(\mathbf{p}) = \left[\sin\left(2^0\pi \cdot \mathbf{p}\right), \cos\left(2^0\pi \cdot \mathbf{p}\right), \ldots, \sin\left(2^{L-1}\pi \cdot \mathbf{p}\right), \cos\left(2^{L-1}\pi \cdot \mathbf{p}\right)\right], \ \mathbf{p} \in \mathbb{R}^d.$$

The inputs were normalised to the $[-1,1]$ interval first, and the value of L was empirically set to 4 for the viewing direction values and to 10 for the location vector. We can express this addition as reformulating $F_{\mathbf{w}}$ as a composition of two functions, $F_{\mathbf{w}} = f_{\mathbf{w}} \circ \mathsf{PE}$, where $f_{\mathbf{w}}$ is the learnable function of the MLP.

This encoding is a special case of Fourier features [13]. While a Fourier feature mapping can be used to overcome the spectral bias of coordinate-based MLPs towards low frequencies by allowing them to learn much higher frequencies, bringing noticeable improvement as in Figure 2.5, the over-fast convergence on high-frequency components that this mapping defines impedes NeRF from exploring low-frequency information, making the model significantly biased towards undesired high-frequency artifacts.



(a) Ground Truth (b) No mapping (c) Basic (d) Positional enc.

Figure 2.5: Influence of the positional encoding on NeRF outputs – more high-frequency details are present [13].

Furthermore, stationary positional encodings are difficult to configure without

adequate prior knowledge, which is why they still often remain ineffective in retaining high-frequency components. This makes having the positional encoding's number of components and corresponding frequencies dependant on the inputs extremely desirable. To address this issue, [14] proposes the Adaptive Positional Encoding whose frequency parameters are trainable. More formally,

$$\gamma(\mathbf{p}) = [\gamma_0(\mathbf{p}), \gamma_1(\mathbf{p}), \ldots, \gamma_{L-1}(\mathbf{p})]^\mathsf{T}, \tag{2.3}$$

where $\gamma_k(\mathbf{p}) = [\sin(\omega_k \mathbf{p}), \cos(\omega_k \mathbf{p})]$, and $[\omega_1, \omega_2, \ldots, \omega_L]^\mathsf{T}$ are the trainable frequency parameters. The parameters are adjusted during training and converge to the proper frequency bands of the scene representations, ensuring a better frequency search space for the current scene [14]. The downside of this approach is that the learning task for training these parameters is non-trivial and requires learning high dimensional functions in an unbounded domain.

2.3 Baseline model

2.3.1 FreeNeRF

The state of the art base model for few-shot novel view synthesis is **FreeNeRF**, developed as an enhancement to NeRF's performance under data-scarce conditions while performing comparable to the original model, but with with far fewer inputs [15]. As of now, there have not been any adaptations of 3DGS for sparse inputs, with the basic model performing significantly worse than FreeNeRF.

FreeNeRF introduces two important additions to the model that are simple to implement and do not significantly increase the model's training or inference time. The first is a frequency regularisation mechanism: initially, the model is restricted to learning low-frequency details, which ensures stable training and prevents overfitting to noise. As training advances, higher frequencies are progressively introduced through a frequency mask, enabling the model to capture finer details such as edges, textures, and subtle lighting variations. The other regularisation mechanism FreeNeRF introduces is for occlusion. When parts of the scene are obscured from certain viewpoints, NeRFs usually produce artifacts or incorrect reconstructions since the data in those areas is incomplete. The occlusion regularisation term is applied during training, particularly near the camera's position, where dense fields of information may be lacking. By discouraging the model from overfitting to sparse or noisy data in these regions, FreeNeRF reduces

the likelihood of artifacts and improves the overall quality of novel view synthesis, even in challenging occluded scenes [15].

The model performs 200 thousand training steps, with 8 input (training) views and 25 output (test) views. We set the frequency regularisation mask to gradually decrease to 0% over the course of first 50% (100,000) of the steps.

2.3.2 Experimenting with positional encoding

We replace the standard PE with the novel **Sinusoidal Positional Encoding (SPE)** [16] which can be simply represented as

$$\mathrm{SPE}(\mathbf{x}) = \sin(\omega \mathrm{PE}(\mathbf{x})),$$

where the ω is a trainable vector that represents the learned features. This is equivalent to applying the function $\sin(\cdot)$ by a sinusoidal activation on the first layer of the MLP (after the positional encoding):

$$\begin{aligned}\sin(W_{SPE}\mathrm{PE}(\mathbf{x})) = \big[&\sin\left(\omega_1^\intercal \sin(\pi\mathbf{x})\right), \sin\left(\omega_2^\intercal \cos(\pi\mathbf{x})\right), \\ &\ldots, \sin\left(\omega_{2L-1}^\intercal \sin(2^{L-1}\pi\mathbf{x})\right), \sin\left(\omega_{2L}^\intercal \cos(2^{L-1}\pi\mathbf{x})\right)\big]^\intercal, \quad (2.4)\end{aligned}$$

where $W_{SPE} = [\omega_1, \ldots, \omega_{2L}]^\intercal$ is the trainable matrix of the weights of the first fully connected layer in the MLP. The benefit of this approach, besides its simple implementation and plug-and-go nature, is that it is designed to adaptively learn frequency features that are closely aligned with the true underlying function. It essentially has the standard PE as a subcase (so if the PE manages to capture the appropriate features, SPE can easily approach PE), but it can also learn the number and frequencies of Fourier series of the encoding from the inputs, making it more tailored to the data.

As mentioned in the introduction, although this baseline performs the best in the environment of [16], small changes in its components like overall loss function, discriminator network type, or even different weight initialisations, can affect the model performance. We run experiments both with SPE and without (i.e. with the standard PE) to support our evaluation.

2.4 Why uncertainty matters?

While we have touched upon the importance of quantifying model uncertainty for enhancing models or making accurate comparisons between models, it is important

Figure 2.6: Example of ground truth vs baseline that despite all the improvements continues to struggle with learning high-frequency colour and geometry.

to showcase that knowing a model's uncertainty is also a critical aspect of practical applications of NVS models.

Several works have shown that the 3D representations learned by NeRF can be used for different downstream tasks that have relevant applications in fields such as robotics or augmented reality: previously explored examples include camera-pose recovery, 3D semantic segmentation, and depth estimation [17]. In these tasks, providing information about the confidence associated with the model outputs is crucial and would expand the possible real-life applications. By incorporating uncertainty-aware mechanisms, systems can manage visual artifacts more effectively and allow users to understand the limitations of the synthesised views, helping them adjust their expectations and interpret the views more accurately [5]. Systems that incorporate uncertainty can also highlight regions of high uncertainty, which is particularly valuable in applications that require real-time feedback and adjustments from users based on the model's confidence levels. Another example is the possible high-stakes applications of NeRF or other NVS model in general, such as medical diagnostics [18] or autonomous driving [19]. In such fields, quantifying uncertainty is vital for risk management and decision-making. Accurate uncertainty estimates enable more informed and safe decisions, as they provide insights into the potential risks associated with the synthesised views. Managing these risks is crucial for ensuring the reliability and safety of systems that rely on novel view synthesis [1].

It is clear that understanding and addressing uncertainty in novel view synthesis models is essential for improving model reliability, enhancing user experience, enabling adaptive systems, supporting model evaluation, and managing risks. Research in the field of model uncertainty enables progress and evolution of possible applications of NVS models.

2.5 Previous work

One of the first and simplest methods developed for accurate measurement of predictive uncertainty in neural networks was the Deep Ensembles method [20]. This method involves training multiple instances of the same model architecture, each initialised with different random weights – it is expected that these models converge over a very large number of steps to the same model – and the variance between the obtained predictions of the individual models is interpreted as the level of uncertainty. The method is simple to implement and requires almost no tuning for the training process. However, the primary drawbacks of Deep Ensembles are their computational and resource demands: training and maintaining several models simultaneously require significantly more processing power and memory than single-model approaches [5]. Still, the ensemble learning approach to uncertainty measurement remains a foundation of many newer solutions.

There have not been many publications specifically targeting uncertainty measurement in NeRF, but we highlight the two main directions from the same group of authors. S-NeRF (*Stochastic NeRF*, [3]) extends NeRF by modeling the radiance field as a distribution rather than a deterministic value. It uses a Bayesian learning approach to estimate the posterior distribution over all possible radiance fields for a given scene by modeling radiance-density pairs as stochastic variables following a distribution whose parameters are optimised during learning. In this manner, uncertainty estimations can be obtained during inference by computing the variance over multiple predictions obtained from different radiance fields sampled from this distribution.

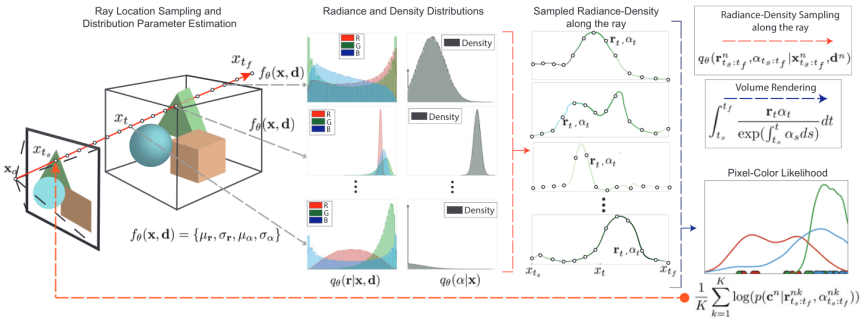


Figure 2.7: Pipeline of Stochastic NeRF, presented in [3].

The current state-of-the-art CF-NeRF (*Conditional-Flow NeRF*, [1]) uses a similar strategy to S-NeRF – it also learns a parametric distribution approximating the posterior, but it avoids the limitations of S-NeRF by not imposing that the distribution can be fully factorised. Instead, it incorporates latent variable modeling to jointly learn

radiance and density distributions for all points in the scene. The radiance-density joint distribution is efficiently modeled using Conditional Normalising Flows: a class of machine learning models used to transform samples from a simple known distribution to variables following an arbitrary, more complex probability density function by applying a series of invertible (bijective) transformations. The similarities and differences in the approaches of these two methods can be more clearly seen in Figure 2.7 and Figure 2.8.

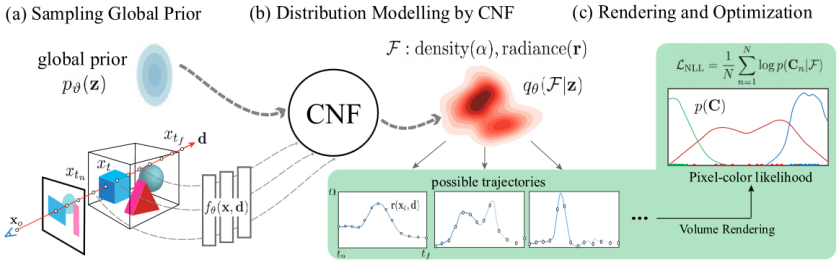


Figure 2.8: Pipeline of Conditional-Flow NeRF, presented in [1].

Both of the aforementioned methods have fundamental flaws in their approaches. S-NeRF's distribution approximation assumes a simple and fully-factorised distribution for $q_{\mathbf{w}}$, as well as (conditional) independence of the radiance and density for each location-view pair in the scene, which is usually not true for adjacent spatial locations and leads to noisy or low-quality images and depth maps [3]. On the other hand, CF-NeRF can be viewed as a more complex and computationally heavy ensemble method, with the full training taking up to 50 hours. Finally, both CF-NeRF and S-NeRF rely on extra information in the form of depth maps, so their performance is not reproducible when it comes to a dataset like `nerf_synthetic` and the results obtained have no real interpretation or meaning, such as in Figure 2.9.

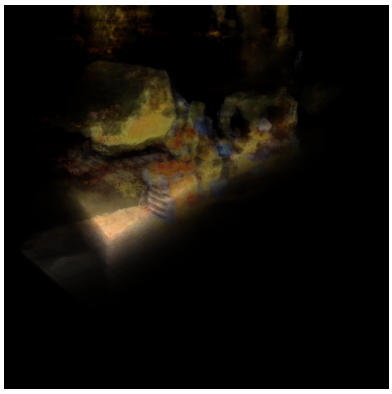


Figure 2.9: Example of an extremely poor quality result using CF-NeRF.

Chapter 3

Quantifying uncertainty

Model uncertainty in machine learning can be classified into two types: aleatoric and epistemic. Commonly, those uncertainties are assumed to be additive, and their sum describes the model's predictive uncertainty. Aleatoric uncertainty describes the variability of an experiment outcome due to inherently random effects that cannot be reduced by any additional source of information (for example, data generation has a stochastic component). Epistemic uncertainty refers to uncertainty caused by a lack of knowledge about the best model, which can in theory be reduced using additional information. Simply put, epistemic uncertainty refers to the reducible part, while aleatoric uncertainty refers to the irreducible part of the predictive uncertainty [21, 22]. The goal of this chapter is to deepen the reader's understanding of what is uncertainty and how its interpretations using probability, statistics and information theory will drive the logic behind our solution.

3.1 Why epistemic uncertainty?

Our method, along with the vast majority of the research in the field of novel view synthesis, focuses on epistemic uncertainty, as we know that with sufficient inputs, like in standard NeRF or 3DGS, we can achieve (close to) perfect accuracy with standard datasets (such as `nerf_synthetic`). Therefore, few-shot NVS models can improve their knowledge using additional input, and we expect that most types of input data do not bring any aleatoric uncertainty as they have little to no stochastic components.

We still empirically verify this assumption by observing output of the deterministic model when random Gaussian noise is added to the input. The noise was added to the input vectors before applying positional encoding. The two methods of noise addition

Figure 3.1: Examples of outputs produced using noise concatenation of different dimensions and variance to the input.

that we implemented were *noise concatenation* and *adding a noise 'dimension'* to the input tensor. We also considered different variances of the noise, as well as different noise lengths/number of layers added. Adding input to the noise either deteriorated the fidelity of all outputs significantly (such as in Figure 3.1) or produced close to no variation, which means inputs themselves have no significant noise or uncertainty for the model.

3.2 Mathematical interpretation of uncertainty

Predictive uncertainty is traditionally modeled using a probabilistic approach, by inferring the predictive distribution $p(\mathbf{y}|\mathbf{x})$ of outcomes $\mathbf{y}$ for inputs $\mathbf{x}$. One way to learn a predictive distribution is to learn a distribution over functions, i.e. model parameters [23]. If our machine learning model has parameters $\mathbf{w}$, then the predictive distribution under the model is denoted as $p(\mathbf{y}|\mathbf{x},\mathbf{w})$. In the case of NeRF, $\mathbf{w}$ are simply the weights of the neural network. From now on, we assume that the true model can be represented by our chosen model class: for example, a multilayer perceptron in NeRF. For the true model parameters $\mathbf{w}^*$ we have that $p(\mathbf{y}|\mathbf{x},\mathbf{w}^*) = p(\mathbf{y}|\mathbf{x})$.

Diving deeper into the probabilistic formulation, we can further describe predictive uncertainty and its components with the help of a distribution that depends on the training dataset $\mathcal{D} = \{\mathbf{X},\mathbf{Y}\}$: that is, the posterior probability $p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$ of certain model parameters $\mathbf{w}$ being the true model parameters $\mathbf{w}^*$ given the training datset. This posterior captures the set of plausible model parameters given the data [22].

To relate the posterior and the predictive distribution, we can use Bayesian model averaging (BMA). BMA is a statistical method designed to account for model uncertainty. Rather than relying on a single model to make predictions, we consider a set of

candidate (plausible) models based on the observed data, each representing different hypotheses about the underlying data-generating process, and average their results to obtain a more robust prediction [24, 25]. BMA is particularly useful in applications with several plausible models, which is appropriate for our setup.

The Bayesian model average predictive distribution is defined as

$$p(\mathbf{y}|\mathbf{x},\mathcal{D}) = \int_{\mathbf{w}} p(\mathbf{y}|\mathbf{x},\tilde{\mathbf{w}})p(\tilde{\mathbf{w}}|\mathcal{D})d\tilde{\mathbf{w}}, \tag{3.1}$$

i.e. the likelihood $p(\mathbf{y}|\mathbf{x},\mathbf{w})$ is weighted by its corresponding posterior probability $p(\mathbf{w}|\mathcal{D})$ (the "plausibility" of the model) and summed over all possible values of $\mathbf{w}$ in the parameter space [26]. Equivalently, this marginal probability over $\mathbf{w}$ can be rewritten using the properties of a continuous variable's expectation:

$$p(\mathbf{y}|\mathbf{x},\mathcal{D}) = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[p(\mathbf{y}|\mathbf{x},\mathbf{w})]. \tag{3.2}$$

Now we can introduce the predictive uncertainty of our model using tools from information theory [27, 28]. One of the most common measures of predictive uncertainty is the Shannon entropy H (also known as the average level of information, surprise or uncertainty inherent to the variable's possible outcomes) of the BMA predictive distribution:

$$H(p(\mathbf{y}|\mathbf{x},\mathcal{D})) = H\Big(\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[p(\mathbf{y}|\mathbf{x},\mathbf{w})]\Big),$$

which can be further decomposed into aleatoric and epistemic components [23]:

$$H(p(\mathbf{y}|\mathbf{x},\mathcal{D})) = \underbrace{\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}\Big[H\Big(p(\mathbf{y}|\mathbf{x},\mathbf{w})\Big)\Big]}_{aleatoric} \tag{3.3}$$

$$+ \underbrace{\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}\Big[\mathsf{KL}\big(p(\mathbf{y}|\mathbf{x},\mathbf{w})||p(\mathbf{y}|\mathbf{x},\mathcal{D})\big)\Big]}_{epistemic}, \tag{3.4}$$

where $\mathsf{KL}(p||q)$ denotes the Kullback-Leibler (KL) divergence from the probability distribution p to the probability distribution q, defined as

$$\mathsf{KL}(p||q) = \sum_{x\in\mathcal{X}} p(x)\cdot\log\frac{p(x)}{q(x)}. \tag{3.5}$$

The KL-divergence is also called the relative entropy, and it measures the inefficiency of assuming that the distribution is q when the true distribution is p [29]. In our application, we can interpret the KL-divergence as the epistemic uncertainty created from sampling according to some model's predictive distribution instead of the true model's predictive distribution. The aleatoric uncertainty is the uncertainty in the outcome $\mathbf{y}$ given the true

model, which can be represented by the entropy of the predictive distribution under the true model $\mathbf{w}^*$. However, since we do not know the true model parameters, we compute the expected aleatoric uncertainty by averaging over the posterior distribution of the model parameters.

Unfortunately, except in trivial cases, $p(\mathbf{w}|\mathcal{D})$ is intractable. Therefore, we seek to find an approximation $p(\mathbf{w}|\mathcal{D}) \approx q(\mathbf{w};\phi)$ from a family of simpler distributions with parameters ϕ whose probability density functions have a known closed form.

For clarity, we return to the related work (Section 2.5). The parametric distribution approximation used by S-NeRF can be written as

$$p_{\mathbf{w}}(\mathcal{F}\,|\mathcal{D}) \approx q_{\mathbf{w}}(\mathcal{F}) = \prod_{\mathbf{x}\in\mathbb{R}^3} \prod_{\mathbf{d}\in\mathbb{R}^2} q_{\mathbf{w}}(\mathbf{c}|\mathbf{x},\mathbf{d}) q_{\mathbf{w}}(\alpha|\mathbf{x}), \tag{3.6}$$

where the chosen distributions are logistic normal over the radiance values, and rectified normal for the volume density. Variational inference approximates a conditional density of latent variables given observed variables using optimisation over a family of densities over the latent variables, choosing the one that is closest in KL-divergence to the conditional of interest [30].

Similarly, CF-NeRF models the radiance-density joint distribution as

$$q_{\mathbf{w}}(\mathcal{F}) = \int q_{\vartheta}(\mathbf{z}) \prod_{\mathbf{x}\in\mathbb{R}^3} \prod_{\mathbf{d}\in\mathbb{R}^2} q_{\mathbf{w}}(\mathbf{c}|\mathbf{x},\mathbf{d},\mathbf{z}) q_{\mathbf{w}}(\alpha|\mathbf{x},\mathbf{z})\, \mathrm{d}\mathbf{z}, \tag{3.7}$$

where $q_{\vartheta}(\mathbf{z})$ denotes the learnable Gaussian prior from which the latent variable $\mathbf{z}$ is sampled. The likelihood of different samples according to the transformed distribution can be computed using the *change-of-variables* formula from calculus [1].

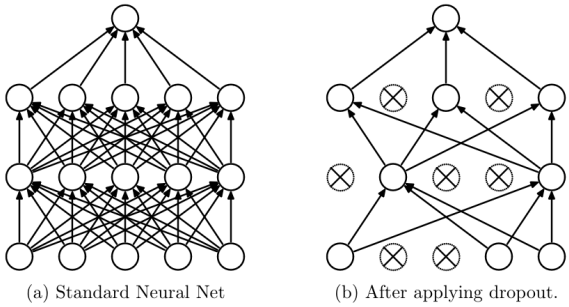
Chapter 4

Methodology

Now that we are equipped with the required background and understanding of the fundamentals of epistemic uncertainty, we can lay out the design of our solution and provide well-supported reasoning for each component. We also discuss our experimental environment and evaluation methods.

4.1 Inference time dropout

As mentioned in Section 3.2, the BMA predictive distribution is usually intractable and requires approximation. One of the most common and simple approximation techniques is Monte Carlo sampling of model parameters, implemented using the so-called **dropout technique** that was first proposed in [31] (see schema in Figure 4.1). Originally, dropout was presented for training only: that is, a unit at training time that is present with probability p, while at inference time it is always present (with its weights being scaled by p).



(a) Standard Neural Net (b) After applying dropout.

Figure 4.1: Dropout technique random neurons are set to zero during training [31].

However, activating dropout at inference time makes the output of the network no longer deterministic. Each dropout configuration corresponds to a different sample from

the approximate parametric posterior distribution, which means multiple simulations will approximate the parametric posterior. Formally, inference time dropout results in the approximation

$$q(\mathbf{w};\Phi) = Bern(\mathbf{w};\Phi) \approx p(\mathbf{w}|\mathcal{D}) \tag{4.1}$$

where $Bern(\cdot;\cdot)$ denotes the Bernoulli distribution and Φ is the dropout rate (probability) on the weights.

This approach of introducing inference time dropout alongside training dropout is also known as the Monte Carlo (MC) dropout technique [32]. However, instead of traditional MC dropout, we chose to implement *inference time only* dropout, also called the *dropout injection technique*. The uncertainty measured by this approach is known as post-hoc model uncertainty, as it measures uncertainty after the model has already been trained and tested [33].

We chose this method over traditional MC dropout for several key reasons. Dropout during training is typically employed to prevent overfitting. However, when used for uncertainty quantification it is no longer conceived as a regulariser, which means that activating dropout layers during training is redundant. This removal also makes our method model-agnostic, and we only need to train our model once which significantly reduces tuning and training costs as the computional overhead only impacts inference time. Furthermore, our goal with NeRF is to encourage overfitting on the training set, so using dropout during training for regularisation and overfit prevention would directly combat our baseline model and negatively impact the performance. We will refer to our solution in the rest of the report as **InjectNeRF** (dropout injection with NeRF).

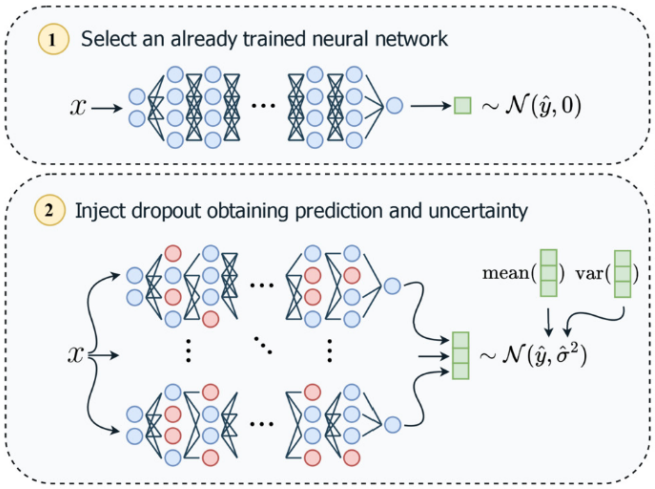


Figure 4.2: Schema of inference-time dropout approach presented in [33].

It is important to emphasise that with this approach, the model whose uncertainty

we are approximating is no longer the same model that was trained. The inference time dropout is equivalent to removing neurons from the model after training, making the architecture of the InjectNeRF model slightly different than the trained baseline. We can still use the same logic due to the sparse nature of our inputs – since the scene has more parameters than the model can infer from few views, the network develops redundant representations. Therefore, as long as dropout is optimised so that InjectNeRF captures/learns approximately the same information as the baseline, we can consider the models sufficiently close and treat them as almost equivalent.

4.2 Measuring uncertainty

To compute our uncertainty, we complete T forward passes during inference per each novel view, with injected dropout. Based on similar published experiments concerning approximations of statistical measures [34], we sample 30 times – that is, we run the inference pipeline 30 times with a fixed dropout rate. The output of the model is the mean of the samples

$$\overline{\mathbf{y}}_i = \frac{1}{T} \sum_{t=1}^{T} \mathbf{y}_{i,t}, \tag{4.2}$$

and we save the standard deviation of each channel of each pixel as a separate image representing the model uncertainty [35]. We use standard deviation because it is directly comparable to the data and therefore better for visualisation since it can directly be interpreted as RGB values. It also fits naturally into many statistical frameworks such as confidence intervals, and it provides a more intuitive sense of the variability present in a dataset. As our images are represented as three-dimensional tensors (image height $\times$ image width $\times$ three colour channels), the mean and standard deviation are calculated separately for each of the elements in the tensor.

4.3 Dropout position

The position of the dropout is another key parameter in our setup. Traditionally, dropout is positioned before every layer of the neural network. However, dropout position in our application provides crucial additional context: the uncertainty of which *model component* are we measuring.

We briefly overview the standard NeRF model architecture, as depicted in Figure 4.3. The positional encoding of the input location is passed through 8 fully-connected ReLU

layers, each with 256 channels. In the fifth layer, the encoded input is concatenated to its activation. An additional layer outputs the volume density and a 256-dimensional feature vector, which is concatenated with the positional encoding of the input viewing direction and is processed by an additional fully-connected ReLU layer with 128 channels. A final layer (with a sigmoid activation) outputs the emitted RGB radiance [7]. As mentioned, the sinusoidal positional encoding simply replaces the ReLU activation of the first layer with a sine activation function.

We can now determine two potential positions for the dropout that are of high interest: the first layer (after the Sinusoidal/Regular PE is applied), and the last layer (corresponding to the neurons that predict the RGB values). Our hypothesis is that dropout in the layer of the PE will produce higher uncertainty in high-frequency details, while the dropout before RGB prediction will be more visible on blurry or inconsistent colour patches or larger surfaces that have limited information due to occlusion. We perform experiments with dropout in only one of the layers.

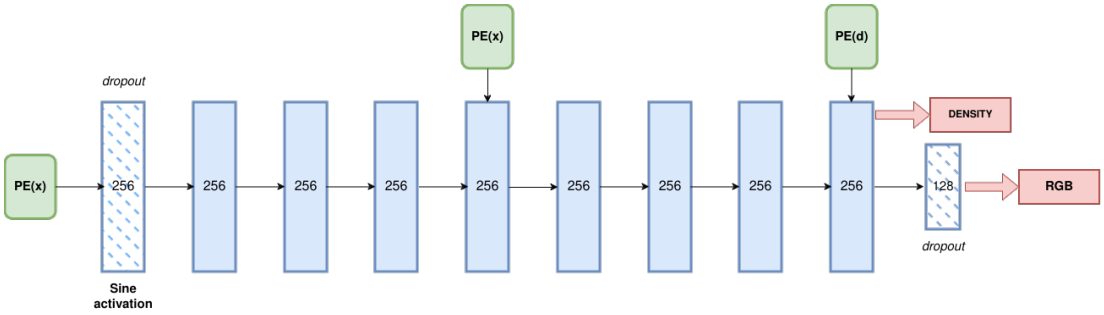


Figure 4.3: NeRF architecture with our proposed dropout additions, used during inference only (based on schema provided in [7]). The architecture of FreeNeRF with and without SPE is the same except for the change of activation function in the first layer.

4.4 Selecting the appropriate dropout rate

We now face the challenge of finding the optimal dropout rate Φ for approximating the epistemic uncertainty. In MC dropout, the optimal dropout rate is taken to be the minimiser of the distance between the real and the hypothesis weight distribution [33]. That is,

$$\Phi = \arg\min_{\phi} \mathrm{KL}(p(\mathbf{w}|\mathcal{D}) \,||\, q(\mathbf{w};\phi)). \tag{4.3}$$

This expression is intractable, so we must derive an approximation, but also adapt it to be independent from the training procedure. One such method from variational

inference is minimising an approximation of the NLL over a set of N input samples – a validation set, for example. Instead of a separate validation set, we use the training set to maintain our goal of matching the baseline's training set accuracy.

While the equation defined in [33] is meant for one dimensional output:

$$\Phi = \arg\min_{\phi} \sum_{i=1}^{N} \left(\frac{\log \sigma_i^2(\phi)}{2} + \frac{(y_i^{gt} - \bar{y}_i)^2}{2\sigma_i^2(\phi)} \right), \tag{4.4}$$

we can adjust it for our purposes. Introducing $\overline{\sigma}_i^2$ as the mean of all the values of the variance tensor, we can restate the equation as

$$\Phi = \arg\min_{\phi} \sum_{i=1}^{N} \left(\frac{\log \overline{\sigma}_i^2(\phi)}{2} + \frac{MSE(\mathbf{y}_i^{gt}, \overline{\mathbf{y}}_i)}{2\overline{\sigma}_i^2(\phi)} \right). \tag{4.5}$$

The one-dimensional approximation of Equation (4.4) can be derived under the normality assumption on the predictive distribution and the model's approximation (for a short proof, see Appendix B.1). However, our modification can independently be interpreted as a suitable goal regardless of the distribution of the output.

- The network is trained without dropout, so significant increase in the dropout rate after training poses risk of removing neurons beyond the model's redundancy, which will result in noticeable decrease of prediction quality. We also want to keep our InjectNeRF model close in accuracy to the baseline, as discussed in Section 4.1.

- For very small values of the dropout rate, the variability induced on the network predictions is almost unnoticeable (close to zero, especially considering the layer dimensions) and therefore it does not accurately represent the model's uncertainty.

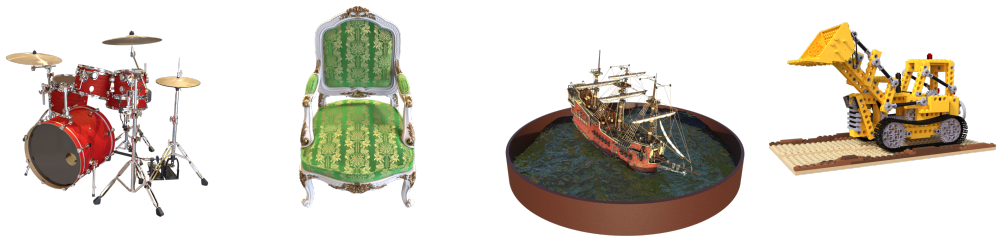
Having the variance term as both a denominator and numerator in the expression, as well as including the effect of the output error, encapsulates our aims in an elegant manner. Once the optimal value of Φ is found, the approximate distribution can be used for computing a prediction and associated predictive uncertainty as described earlier.

4.5 Dataset

The main dataset we use for our evaluations is the `nerf_synthetic`[1] which consists of eight objects rendered in Blender. Our experiments use four of these objects, chosen to

[1] Provided by authors of [7] at `www.matthewtancik.com/nerf`.

cover a wide range of complexity (high frequency inputs, intricate patterns or details). For more details on the full dataset and its authors, see Figure A.1 in the Appendix.



(a) `drums` (b) `chair` (c) `ship` (d) `lego`

Figure 4.4: The four Blender objects used in experiments.

The dataset includes camera intrinsics and extrinsics, necessary for accurate 3D scene reconstruction. Since the objects are synthetically rendered, they have clean geometric structures, as well as fine details, textures, and lighting variations, making the dataset the most frequent choice for benchmarking NeRFs.

4.6 Fidelity metrics

The standard metric used in novel view synthesis tasks is the Peak Signal-to-Noise Ratio (PSNR). The PSNR (in decibel) is the ratio between the maximum possible power of a signal, and the power of corrupting noise that affects the fidelity of its representation. In our application, we take the noisy image to be the output of our model. For each ground truth/model output pair, the PSNR is calculated as the average of the metric's channelwise value. We give the equations for one channel below.

Setting G to be the ground truth and M to be the output of the model, both of dimensions $m \times n$, the PSNR is defined as

$$\mathsf{PSNR}(G,M) = 10 \cdot \log_{10}\left(\frac{256^2}{\mathsf{MSE}}\right) = 20 \cdot \log_{10}(256) - 10 \cdot \log_{10}(\mathsf{MSE}), \tag{4.6}$$

where

$$\mathsf{MSE}(G,M) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\big(G(i,j) - M(i,j)\big)^2 \tag{4.7}$$

is the Mean Squared Error. The higher the PSNR, the higher is the quality of the model output [36].

Another popular metric is the SSIM (Structulary Similarity Index Measure), which also considers important perceptual phenomena like luminance masking and contrast

masking terms. SSIM is calculated on various windows of an image. The measure between two windows X and Y of common size $N \times N$ is

$$\text{SSIM}(X,Y) = \frac{(2\mu_X\mu_Y + c_1)(2\sigma_{XY} + c_2)}{(\mu_X^2 + \mu_Y^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)}.$$

where $\mu_X, \sigma_X^2, \mu_Y, \sigma_Y^2$ denote the corresponding pixel sample means and variances of x and y and $\sigma_{x,y}$ is the covariance. The two variables c_1 and c_2 are defined to stabilize the division with weak denominator: $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ where L is the *dynamic range* of the pixel-values (typically taken as $2^{\text{bits per pixel}} - 1$) and $k_1 = 0.01$ and $k_2 = 0.03$ by default. The Mean SSIM (MSSIM) is calculated by the average SSIM over all the windows [37]. The standard size of the windows is 7×7. The value of the metric ranges between -1 and 1, where 1 denotes a complete match with the original image.

The metrics are implemented as a part of the `skimage.metrics` library. The evaluations of the two baseline models are given below in Table 4.1.

Model: FreeNeRF				
Object	Train PSNR	Test PSNR	Train SSIM	Test SSIM
`chair`	38.4798	27.005	0.9901	0.9253
`ship`	33.0575	23.3482	0.9213	0.7761
`drums`	32.4471	20.271	0.9755	0.8525
`lego`	36.2977	25.199	0.9802	0.8866

Model: FreeNeRF + SPE				
Object	Train PSNR	Test PSNR	Train SSIM	Test SSIM
`chair`	39.3806	26.9811	0.9919	0.9204
`ship`	33.384	23.4326	0.9154	0.77
`drums`	33.7015	20.3036	0.978	0.8489
`lego`	32.9735	23.1805	0.9571	0.8482

Table 4.1: Evaluation of the baseline models on the chosen objects. We see that with our computational environment, the two models perform quite similarly on the testset with marginal differences, except in the case of `lego`. In most cases, SPE does help the model fit better to the training views.

Chapter 5

Evaluation

In this chapter, we evaluate the proposed method through a series of experiments. Our main focus are the following research questions.

- *Is our proposed method effective in quantifying uncertainty of our baseline?*
- *Does the dropout position in the network impact the evaluated uncertainty?*
- *How does our method impact the fidelity of generated views?*

Additionally, we discuss the visualisation of our outputs, which is a key component in interpreting uncertainty. The results demonstrate that the method can effectively capture uncertainty, particularly in areas of high variability or artifacts, and often improves upon the baseline model.

5.1 Obtaining the optimal dropout rate

Using Equation (4.5), we perform a sweep over seven possible values of the dropout rate in the interval $[0.02, 0.3]$ – separately for each dropout position. We decided on these boundaries and the chosen values based on the analysis of the impact of dropout increase that was discussed in Section 4.4: considering the SPE layer has 256 neurons and the RGB layer has 128, less than 2% dropout is insignificant, and we still want to keep the majority of the neurons active.

The results presented in Figure 5.1 and Table 5.1 show that the optimal dropout rate for the (S)PE layer is in most cases smaller than the RGB optimal rate. This is not surprising, as the positional encoding's goal is to perfectly learn the frequency functions present in the training set, making each component key for the model's performance on

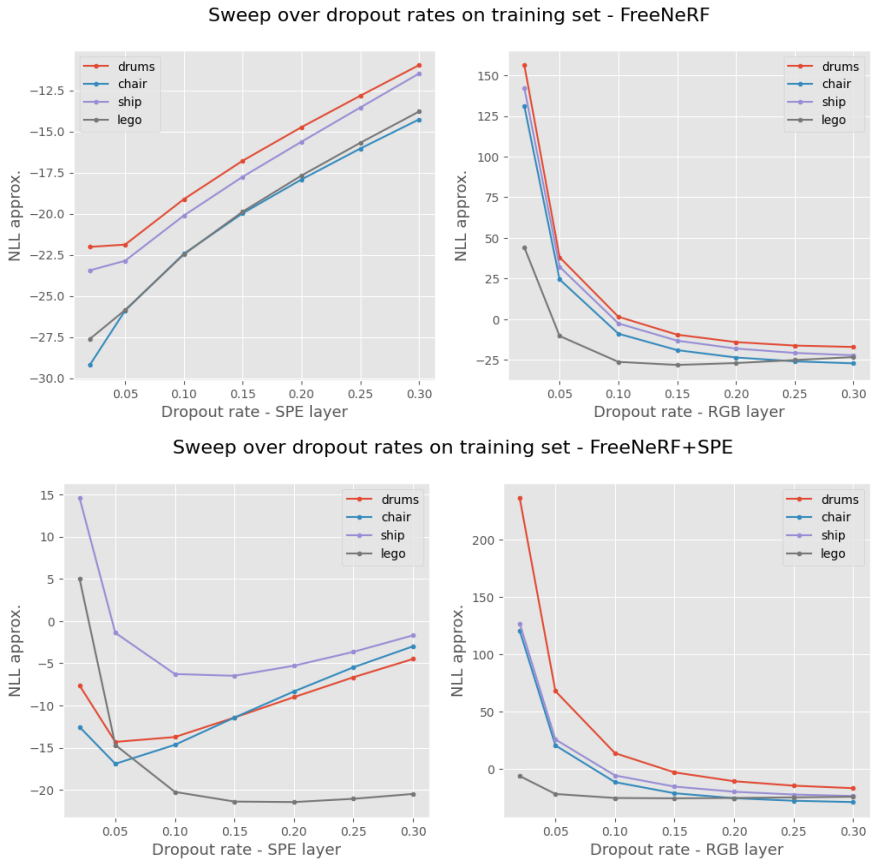


Figure 5.1: Detailed results of the dropout rate sweeps.

Model: FreeNeRF		
Object	Φ_{SPE}	Φ_{RGB}
`chair`	0.02	0.30
`ship`	0.02	0.30
`drums`	0.02	0.30
`lego`	0.02	0.15

Model: FreeNeRF + SPE		
Object	Φ_{SPE}	Φ_{RGB}
`chair`	0.05	0.30
`ship`	0.15	0.30
`drums`	0.05	0.30
`lego`	0.20	0.15

Table 5.1: Optimal dropout rates based on training set sweep, per object, for both baselines.

the same set of views. We also verify the performances of the InjectNeRF models on the training set are indeed sufficiently close to the baselines', as displayed in Figure 5.2.

5.2 Uncertainty visualisation

Naturally, we want to be able to show the uncertainty of our model per each generated view. To aid the uncertainty visualisation and metric evaluation, we created a set of

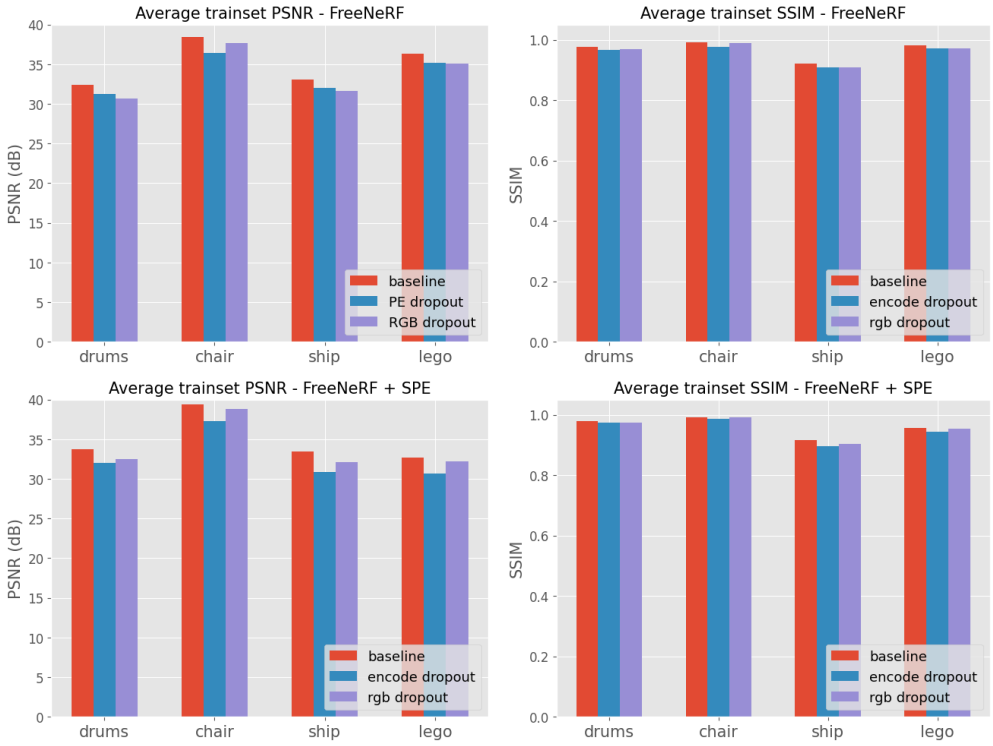


Figure 5.2: Inference time dropout's performance on the trainset, compared to the baselines.

visualisation functions (see examples on the `lego` object in Figure 5.3). The first option is a separate plot of the standard deviation, as well as the standard deviation relative to the ground truth pixel value. The second option overlays the standard deviation on top of the (mean) model output and separately highlights the values in the top 10% of all of the pixelwise standard deviations in the testset. For a cleaner display, the standard deviation is taken to be averaged per pixel, i.e. the mean of the channels is taken.

5.3 RGB and (S)PE dropout comparison

We empirically confirm our hypothesis that there is a clear difference in the uncertainties created by the two different dropout positions. Dropout in the (S)PE layer results in higher uncertainty of the fine, high frequency details of the object: its outline, edges, detailed patterns. We can strengthen our argument further by analysing Figure 5.4, which shows a comparison between the SPE dropout uncertainty and the Daubechies 4 wavelet decomposition on one of the test views of the drums with the SPE baseline.

The Daubechies 4 (Db4) wavelet decomposition is a multiresolution analysis technique, derived from the Daubechies family of orthogonal wavelets that define a discrete wavelet transform. It systematically decomposes a signal into its frequency components

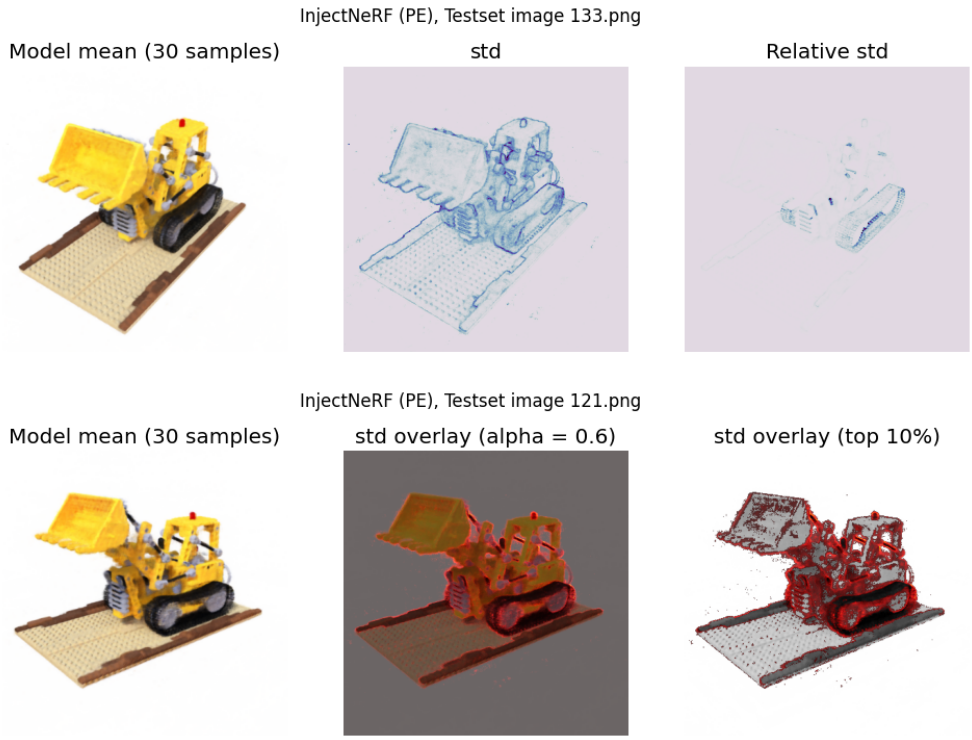


Figure 5.3: The two different visualisation styles of the uncertainty outputted by InjectNeRF.

by applying successive low-pass and high-pass filters, with the number 4 in the name signifying it has four wavelet and scaling function coefficients. At each level of decomposition, the signal is divided into an approximation (low-frequency component) and a detail (high-frequency component), with the approximation being further decomposed in subsequent levels. This process creates a hierarchical structure where each level isolates distinct frequency bands, allowing for the analysis of the signal's frequency content across different scales [38]. The wavelet coefficients in the high-frequency sub-bands quantify the strength of high-frequency details at different scales and orientations. Larger coefficients correspond to stronger high-frequency details, such as more prominent edges or sharper textures.

Comparing the three levels of the Db4 decomposition with our model's output, we see that the details of darker colour (meaning higher uncertainty) appear in the higher levels of the wavelet decomposition, meaning dropout in the SPE layer indeed quantifies the uncertainty of the model about the set of frequency functions learnt during training. On the other hand, dropout in the RGB layer shows higher uncertainty in the *inside areas* of the object, particularly in areas of low quality in the model output, such as back of the chair in Figure 5.5 which in the model output is a mix of beige, yellow and green pixels (as opposed to a monochrome backing in the ground truth). Furthermore,

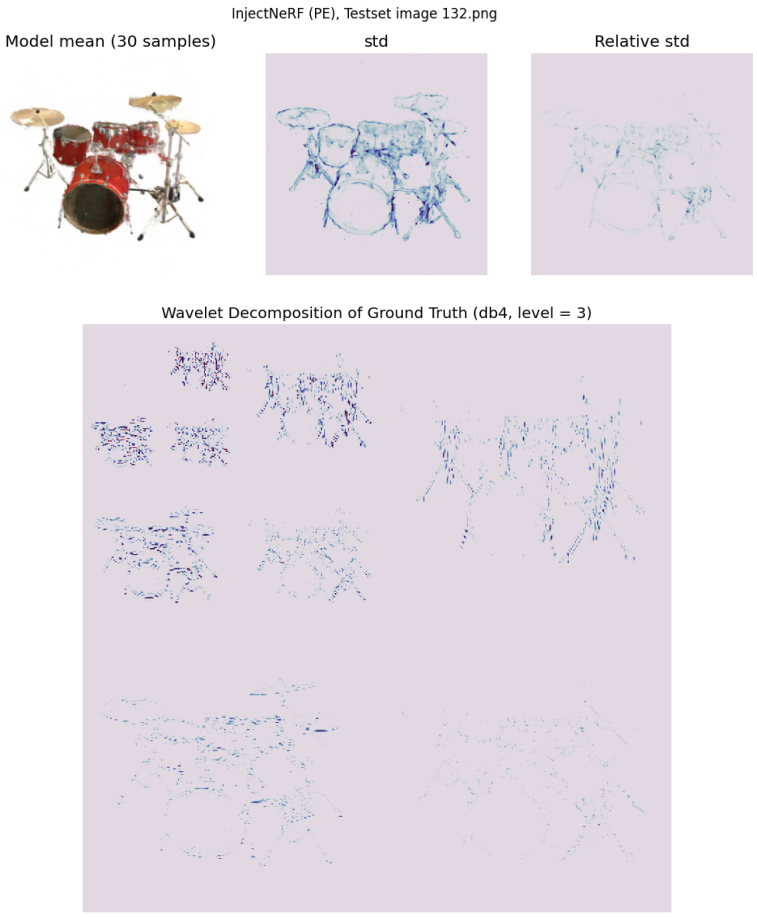


Figure 5.4: Comparison of the epistemic uncertainty from InjectNeRF with SPE dropout and the Db4 discrete wavelet decomposition.

we observe high uncertainty in areas with unwanted artifacts, such as random floating pixels and missing parts of the ship model's base in Figure 5.6.

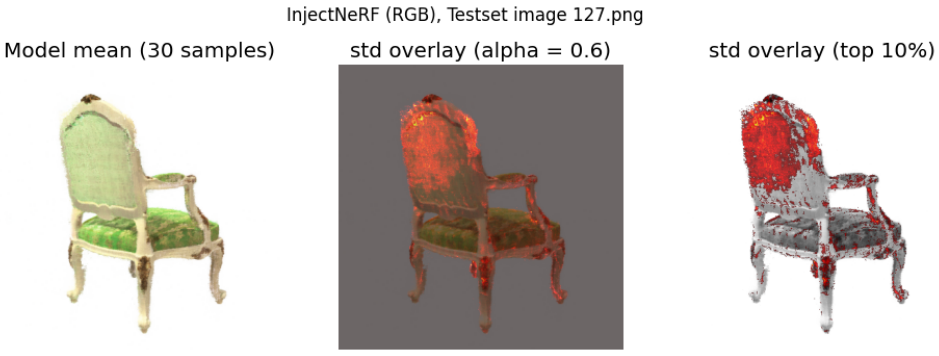


Figure 5.5: InjectNeRF - RGB dropout correctly displays high uncertainty.

Finally, the visualisation reveals a meaningful difference between the model's uncertainty between FreeNeRF with regular PE and Sinusoidal SPE, particularly when the dropout is applied at the relevant PE layer (an example is shown in Figure 5.7).

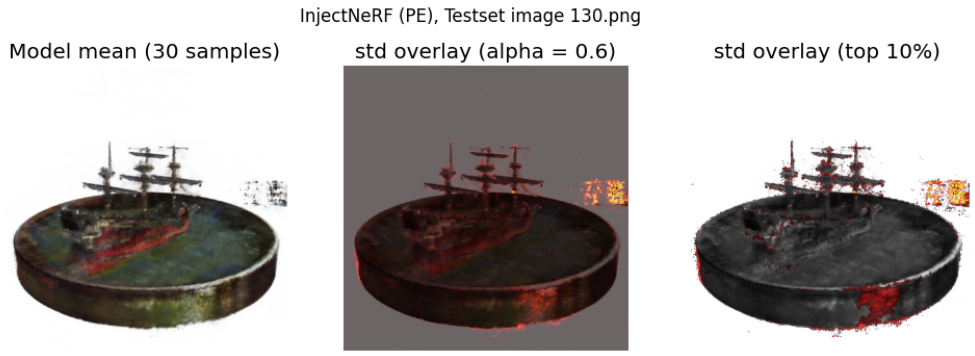
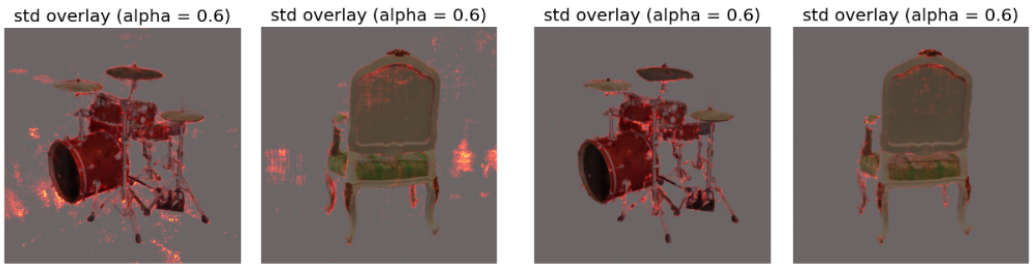


Figure 5.6: InjectNeRF correctly captures high uncertainty around undesirable artifacts and low quality patches.

While the overall performance might not be better, the addition of Sinusoidal Positional Encoding reduced the model's uncertainty in the captured frequency components that in regular FreeNeRF manifest as random off-white pixels around the object.



(a) PE dropout – regular PE (b) PE dropout – SPE

Figure 5.7: Addition of Sinusoidal Positional Encoding benefits the model's confidence.

5.4 Fidelity comparison

Finally, we evaluate the output generated by InjectNeRF (i.e. the sample mean) against the baseline. From Figure 5.8 in more detail, we see that InjectNeRF can outperform our FreeNeRF + SPE baseline in PSNR in almost all test views with SPE dropout – with three of them having at least 24 out of 25 views showing better performance. With RGB dropout, the score is significantly lower, but with 96% outperformance on the most difficult object `drums`. We see less success with the SSIM metric, with InjectNeRF - SPE dropout still obtaining better results than the baseline in the majority of views.

In the baseline with the regular PE, whose results are outlined in Figure 5.9, we still observe a large number of PSNR improvements with either RGB or PE dropout, most noticeable in the `drums` and `ship` objects. However, the performance in SSIM is

Figure 5.8: The difference in performance between InjectNeRF's mean outputs and the baseline: **FreeNeRF with SPE**.

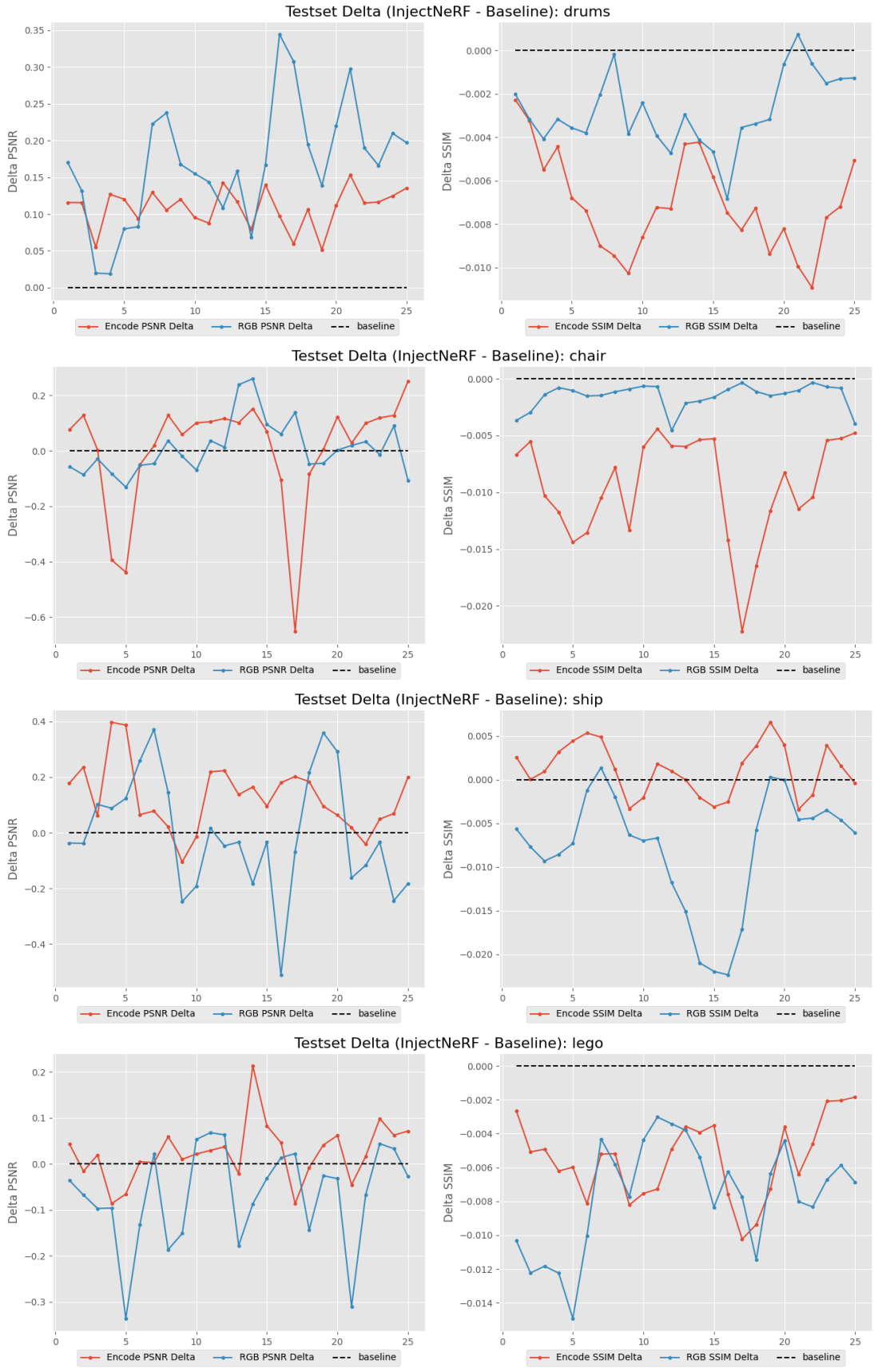


Figure 5.9: The difference in performance between InjectNeRF's mean outputs and the baseline: **FreeNeRF**.

noticeably worse, with only notable gains being with the `ship` object. An interesting observation is that for both baselines, we see the most improvements with the two more complex objects.

The output of our model is a mean of 30 samples, which means some samples performed better than the mean on some (or perhaps all) views, and some performed worse – therefore, we could even further improve our gains over the baseline by choosing the best sample rather than the mean. These results empirically confirm that there exist multiple continuous functions that fit the training views well, but have different performances on the test views. This phenomenon has been observed in several studies such as [2, 20] (not specifically for NVS), where it is further discussed why ensemble methods may work better than a single model. The two main arguments in our experimental setup are *statistical* and *representational* (depicted in Figure 5.10). First, considering our training dataset is small, the learning algorithm can find many different hypotheses from the same family that all have the same accuracy on the training data, so the construction of an ensemble out of all of these accurate classifiers reduces the risk of choosing the worst performing one in inference. Furthermore, as mentioned in the introduction, the true function of a 3D scene may not be a continuous function, but rather a combination of multiple components of different types (discrete, continuous, ...). Forming weighted sums of hypotheses might expand the space of representable functions which can bring us closer to the true function.

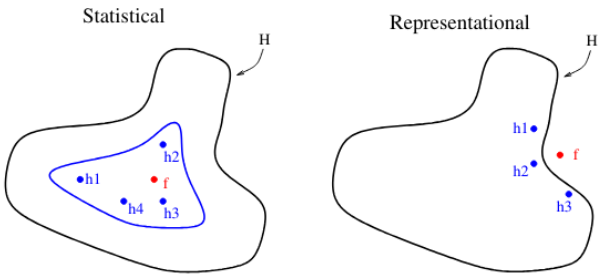


Figure 5.10: Possible benefits of using ensembles rather than a singular model [2].

5.5 Evaluating the method as an uncertainty measure

It remains to evaluate how well our solution represents epistemic uncertainty. This can be done by quantifying the relationship between the model's pixelwise mean squared error and the returned pixelwise standard deviation. Generally, we expect that pixels with higher error will have higher uncertainty, therefore, we consider methods and

metrics for measuring correlation between two random vectors - the flattened tensors of the error and uncertainty. Our evaluation consists of two rank-correlation metrics, since they do not assume any specific distribution of the data or a concrete type of relationship (linear, polynomial, ...) between the data. While S-NeRF [3] and CF-NeRF [1] use the Pearson correlation coefficient, we believe this is not a suitable measure as the data we are trying to correlate is not normally distributed – this can be easily checked via the Kolmogorov-Smirnov test for goodness of fit – which is one of the necessary conditions for the Pearson coefficient to be a meaningful measure of a linear relationship [39, 40].

Spearman's ρ (also known as Spearman's rank correlation coefficient) is a non-parametric measure of the strength and direction of association between two ranked (ordinal) variables. It assesses how well the relationship between two variables can be described using a monotonic function. The coefficient is calculated by first converting the raw data into ranks. Given a set of n pairs (X_i, Y_i), where X and Y are the two variables being compared, the ranks of X_i and Y_i are denoted as $\text{rank}(X_i)$ and $\text{rank}(Y_i)$, respectively. Finally, using the Pearson correlation formula applied to the ranks, we arrive at

$$\rho = \frac{\text{cov}(\text{rank}(X), \text{rank}(Y))}{\sigma_{\text{rank}(X)} \sigma_{\text{rank}(Y)}},$$

where $\text{cov}(\text{rank}(X), \text{rank}(Y))$ is the covariance of the ranks and $\sigma_{\text{rank}(X)}$ and $\sigma_{\text{rank}(Y)}$ are the standard deviations of the ranks.

Spearman's ρ does not take into account possible ties, i.e. pairs of observations with the same value, so the score might be significantly higher than the realistic result. Therefore, we utilise Kendall's τ (tau) coefficient, another rank correlation coeffficient that measures the ordinal association between two measured quantities, to support our results. Specifically, we use the τ_b coefficient which is an extension of the standard coefficient designed to handle ties more effectively.

Given a set of n pairs (X_i, Y_i), where X and Y are the two variables being compared, τ_b is calculated based on the number of concordant and discordant pairs. A pair (X_i, Y_i) and (X_j, Y_j) is *concordant* if both $X_i > X_j$ and $Y_i > Y_j$, or if both $X_i < X_j$ and $Y_i < Y_j$. Similarly, a pair (X_i, Y_i) and (X_j, Y_j) is *discordant* if $X_i > X_j$ and $Y_i < Y_j$, or if $X_i < X_j$ and $Y_i > Y_j$.

Let P be the number of concordant pairs, Q the number of discordant pairs, T_X the number of ties only in variable X and T_Y the number of ties only in variable Y. Then, Kendall's τ_b coefficient is given by:

$$\tau_b = \frac{P - Q}{\sqrt{(P + Q + T_X) \cdot (P + Q + T_Y)}}.$$

Both coefficients attain values in the $[-1, 1]$ interval: value greater than 0 indicates a positive association, less than 0 indicates a negative association, and equal to 0 suggests no association between the variables [41]. Results in Table 5.2 and Table 5.3 show that across all objects, both coefficients show strong (> 0.68 for ρ and > 0.49 for τ_b) or even very strong positive association, which is exactly the result we expected from a good uncertainty measure.

Correlation coefficients, baseline = FreeNeRF				
	InjectNeRF - SPE dropout		**InjectNeRF - RGB dropout**	
Object	Spearman's ρ	Kendall's τ_b	Spearman's ρ	Kendall's τ_b
`chair`	0.8524	0.7374	0.9466	0.8547
`ship`	0.7848	0.6315	0.8398	0.6696
`drums`	0.921	0.8091	0.8326	0.7036
`lego`	0.7445	0.6165	0.8373	0.6935

Table 5.2: Results showing success of our uncertainty measure on FreeNeRF.

Correlation coefficients, baseline = FreeNeRF + SPE				
	InjectNeRF - SPE dropout		**InjectNeRF - RGB dropout**	
Object	Spearman's ρ	Kendall's τ_b	Spearman's ρ	Kendall's τ_b
`chair`	0.8648	0.7854	0.9334	0.8425
`ship`	0.7964	0.6577	0.8287	0.6615
`drums`	0.8314	0.7353	0.9059	0.7962
`lego`	0.8538	0.7324	0.833	0.6914

Table 5.3: Results showing success of our uncertainty measure on FreeNeRF with SPE.

Chapter 6

Conclusion

6.1 Possible improvements

While our method is simple and effective, the dropout rate optimisation process and final uncertainty evaluation introduce noticeable additional computational overhead. Although we implemented a coarse search over dropout values and reduced the number of forward passes in the optimisation process from 30 to 25, on average the runtime of InjectNeRF is around 3.5 hours – which is still a large saving compared to the costly Deep Ensembles method (200,000 iterations of FreeNeRF training take close to 7 hours) or CF-NeRF. Still, for real-time applications, this overhead might be a bottleneck, particularly in scenarios requiring immediate feedback. We include an implementation of InjectNeRF with an option to skip the optimisation process by passing the desired dropout rate as an input flag to reduce the runtime to 90 minutes. Further improvements can be made by parallelising the optimisation and evaluation process, so each InjectNeRF instance runs on a separate GPU, and gathering and comparing the results at the end to select the optimal dropout rate. With the right high performance computing resources (i.e. enough GPUs), the parallelisation can reduce the runtime to real-time rendering of the baseline. These improvements would also allow for a finer search grid for the optimal dropout rate, as well as more samples for variance estimation, which could potentially bring more improvements to our results.

Additionally, our testing dataset is rendered by software, so it would be beneficial for the model to be tested on more challenging scenarios to strengthen our arguments. Real-world application may present further challenges due to increased complexity in lighting, texture, and occlusions. This can be done by running InjectNeRF on more

complex, real-world datasets such as Local Light Field Fusion (LLFF, Figure 6.1)[1]. Incorporating additional modalities, such as depth maps or a complex background, could give us further valuable insight into the epistemic uncertainty of the model. Future iterations of this research could focus on improving model generalisation in more challenging scenarios, such as dynamic scenes or environments with significant occlusions.

Figure 6.1: Objects `horns` and `flower` from the LLFF dataset.

6.2 Future work

Since our results show that InjectNeRF can improve the performance of its baseline, this opens several further research directions for applications of our method. One of the major advantages of our method is its model-agnostic nature, so applying this uncertainty measurement framework to a variety of NVS models (3D Gaussian Splatting or more complex, newer architectures like Diffusion Models), would be the next biggest milestone – we expect similar results as with NeRFs. With multiple different baseline structures, we can expand our *ensemble* and further improve the accuracy of outputs by combining pixels outputted by different models based on their confidence levels. The simplicity of the method implementation would allow us to integrate the uncertainty measure quickly, and after parallelisation is implemented, we would have a computationally efficient output enhancement method using pre-trained models.

Another benefit of our approach is that we can further break down the epistemic uncertainty using the position of the dropout, which gives us deeper insight into our model's behaviour. We can use this to further explore NeRF's components, or the components of any other NVS model, which could lead to more curated improvement suggestions for the baselines themselves.

[1] `https://www.kaggle.com/datasets/arenagrenade/llff-dataset-full`

6.3 Summary of key points

This thesis has introduced a novel, model-agnostic approach for measuring epistemic uncertainty in few-shot novel view synthesis, with neural radiance fields as the main focus. Through the implementation of inference-time dropout, we successfully quantified the uncertainty of state-of-the-art (SOTA) NVS models. We empirically proved that InjectNeRF provides meaningful uncertainty estimates, while bringing no additional training costs and therefore outperforming various previously published uncertainty measurement techniques. Our experiments also show that our method can efficiently bring fidelity improvements to the model outputs, which opens possibility for further research using this work as foundation. We utilised our method to further research the structure of NeRF models and its behavior based on the dropout position or change in the positional encoding approach.

In conclusion, our method has shown significant promise in improving the robustness and reliability of NVS models, particularly in sparse input scenarios. We believe that this work can serve as a foundation for further advancements in uncertainty quantification within computer vision and related fields, ultimately enhancing the accuracy and reliability of novel view synthesis models in both academic research and practical applications.

Bibliography

[1] J. Shen, A. Agudo, F. Moreno-Noguer, and A. Ruiz, "Conditional-Flow NeRF: Accurate 3D Modelling with Reliable Uncertainty Quantification," 2022.

[2] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, (Berlin, Heidelberg), pp. 1–15, Springer Berlin Heidelberg, 2000.

[3] J. Shen, A. Ruiz, A. Agudo, and F. Moreno-Noguer, "Stochastic Neural Radiance Fields: Quantifying Uncertainty in Implicit 3D Representations," 2021.

[4] S.-H. Sun, M. Huh, Y.-H. Liao, N. Zhang, and J. J. Lim, "Multi-view to Novel View: Synthesizing Novel Views with Self-Learned Confidence," in *Computer Vision – ECCV 2018*, (Cham), pp. 162–178, Springer International Publishing, 2018.

[5] L. Goli, C. Reading, S. Sellán, A. Jacobson, and A. Tagliasacchi, "Bayes' Rays: Uncertainty Quantification for Neural Radiance Fields," 2023.

[6] L. Savant, D. Valsesia, and E. Magli, "Modeling uncertainty for Gaussian Splatting," 2024.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, p. 99–106, dec 2021.

[8] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields," 2021.

[9] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," 2023.

[10] N. R. Projects, "Tips for training NeRF models with Instant Neural Graphics Primitives." `https://github.com/NVlabs/instant-ngp/blob/master/docs/nerf_dataset_tips.md`. Accessed: April 12, 2024.

[11] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural Radiance Fields from One or Few Images," 12 2020.

[12] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, "On the Spectral Bias of Neural Networks," 2019.

[13] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," 2020.

[14] Z. Gao, W. Dai, and Y. Zhang, "Adaptive positional encoding for bundle-adjusting neural radiance fields," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3261–3271, 2023.

[15] J. Yang, M. Pavone, and Y. Wang, "FreeNeRF: Improving Few-shot Neural Rendering with Free Frequency Regularization," 3 2023.

[16] C. Sun, Z. Yuan, K. Xu, L. Mai, S. N, S. Chen, and M. K. Marina, "Learning High-Frequency Functions Made Easy with Sinusoidal Positional Encoding," in *Proceedings of the 41st International Conference on Machine Learning* (R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, eds.), vol. 235 of *Proceedings of Machine Learning Research*, pp. 47218–47233, PMLR, 21–27 Jul 2024.

[17] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "INeRF: Inverting Neural Radiance Fields for Pose Estimation," 2021.

[18] A. Shamsi, H. Asgharnezhad, S. Jokandan, A. Khosravi, P. Kebria, D. Nahavandi, S. Nahavandi, and D. Srinivasan, "An Uncertainty-Aware Transfer Learning-Based Framework for COVID-19 Diagnosis," *IEEE transactions on neural networks and learning systems*, vol. PP, 02 2021.

[19] K. Chitta, A. Prakash, and A. Geiger, "NEAT: Neural Attention Fields for End-to-End Autonomous Driving," 2021.

[20] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," 2017.

[21] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," 2021.

[22] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?," 2017.

[23] K. Schweighofer, L. Aichberger, M. Ielanskyi, and S. Hochreiter, "Introducing an Improved Information-Theoretic Measure of Predictive Uncertainty," 2023.

[24] M. Hinne, Q. F. Gronau, D. van den Bergh, and E.-J. Wagenmakers, "A Conceptual Introduction to Bayesian Model Averaging," *Advances in Methods and Practices in Psychological Science*, vol. 3, no. 2, pp. 200–215, 2020.

[25] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: a tutorial (with comments by M. Clyde, David Draper and E. I. George, and a rejoinder by the authors," *Statistical Science*, vol. 14, no. 4, pp. 382 – 417, 1999.

[26] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt, "Advances in Variational Inference," 2018.

[27] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning," 2018.

[28] R. Ash, *Information Theory*. Dover books on advanced mathematics, Dover Publications, 1990.

[29] T. Cover and J. Thomas, *Entropy, Relative Entropy, and Mutual Information*, ch. 2, pp. 13–55. John Wiley & Sons, Ltd, 2005.

[30] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, p. 859–877, Apr. 2017.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[32] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," 2016.

[33] E. Ledda, G. Fumera, and F. Roli, "Dropout injection at test time for post hoc uncertainty quantification in neural networks," *Information Sciences*, vol. 645, p. 119356, 2023.

[34] S. G. Kwak and J. H. Kim, "Central limit theorem: the cornerstone of modern statistics," *Korean Journal of Anesthesiology*, vol. 70, pp. 144–156, April 2017. Epub 2017 Feb 21.

[35] M.-H. Laves, S. Ihler, J. F. Fast, L. A. Kahrs, and T. Ortmaier, "Well-Calibrated Regression Uncertainty in Medical Imaging with Deep Learning," in *Proceedings of the Third Conference on Medical Imaging with Deep Learning* (T. Arbel, I. Ben Ayed, M. de Bruijne, M. Descoteaux, H. Lombaert, and C. Pal, eds.), vol. 121 of *Proceedings of Machine Learning Research*, pp. 393–412, PMLR, 06–08 Jul 2020.

[36] D. Sethi, S. Bharti, and C. Prakash, "A comprehensive survey on gait analysis: History, parameters, approaches, pose estimation, and future work," *Artificial Intelligence in Medicine*, vol. 129, p. 102314, 2022.

[37] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600 – 612, 05 2004.

[38] B. Li and X. Chen, "Wavelet-based numerical analysis: A review and classification," *Finite Elements in Analysis and Design*, vol. 81, pp. 14–31, 2014.

[39] S. Prion and K. A. Haerling, "Making Sense of Methods and Measurement: Pearson Product-Moment Correlation Coefficient, journal = Clinical Simulation in Nursing," vol. 10, no. 11, pp. 587–588, 2014.

[40] H. Liu, "Chapter 7 - Description methods of spatial wind along railways," in *Wind Forecasting in Railway Engineering* (H. Liu, ed.), pp. 251–282, Elsevier, 2021.

[41] K. M. Abadir, R. D. H. Heijmans, and J. R. Magnus, *Sample statistics and their distributions*, p. 327–370. Econometric Exercises, Cambridge University Press, 2018.

[42] A. Loquercio, M. Segu, and D. Scaramuzza, "A General Framework for Uncertainty Estimation in Deep Learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153–3160, 2020.

Appendix A

Computational environment

Here we give details of the environment used to obtain the results presented in the dissertation. This also serves as a guide for result reproduction.

A.1 Technical specifications

All the training and testing was done on the Edinburgh ML-Systems Group's Gala server cluster (`gala1`). The technical specifications of Gala are as follows.

- 4U server - PCIe 4.0
- CPU: 2 x 28 CPU cores (AMD EPYC Zen 3, 7453)
- GPU: 8 x NVIDIA A5000 (NVLinked, 4 pairs)
- Memory: 1TB DDR4 3200MHz (16 x 64GB)
- NVMe SSD: 2 x 3.84TB (Intel P5510, PCIe 4.0)
- SATA SSD: 1.92TB (Intel S4510)
- 1 Gbps NIC

A.2 Full details of NeRF Synthetic dataset

The renders are from modified Blender models located on `blendswap.com`.

(a) drums by bryanajones (CC-BY): `https://www.blendswap.com/blend/13383`

(b) chair by 1DInc (CC-0): `https://www.blendswap.com/blend/8261`

(c) ship by gregzaal (CC-BY-SA): `https://www.blendswap.com/blend/8167`

(d) mic by up3d.de (CC-0): `https://www.blendswap.com/blend/23295`

(e) hotdog by erickfree (CC-0): `https://www.blendswap.com/blend/23962`

(f) materials by elbrujodelatribu (CC-0): `https://www.blendswap.com/blend/10120`

(g) lego by Heinzelnisse (CC-BY-NC): `https://www.blendswap.com/blend/11490`

(h) ficus by Herberhold (CC-0): `https://www.blendswap.com/blend/23125`

All the respective renders are depicted in Figure A.1.

(a) `drums` (b) `chair` (c) `ship` (d) `mic`

(e) `hotdog` (f) `materials` (g) `lego` (h) `ficus`

Figure A.1: `nerf_synthetic` dataset – object breakdown

Appendix B

Theoretical work

B.1 Proof of NLL approximation

We aim to prove that minimising

$$\mathsf{KL}(p(\mathbf{w}|\mathcal{D}) \,||\, q(\mathbf{w};\phi))$$

is equivalent to minimising

$$\sum_{i=1}^{N} \left(\frac{\log \sigma_i^2(\phi)}{2} + \frac{(y_i^{gt} - \bar{y}_i)^2}{2\sigma_i^2(\phi)} \right).$$

We expand on the proof provided in [42]. Assume we have a successfully trained network $p_{net}(y|x,\mathbf{w})$ that can predict the ground truth very well. That is, using BMA properties,

$$p_{gt}(y|x) \approx p_{pred}(y|x,\mathcal{D}) = \int_{\mathbf{w}} p_{net}(y|x,\tilde{\mathbf{w}}) p(\tilde{\mathbf{w}}|\mathcal{D}) d\tilde{\mathbf{w}}$$

where $p_{pred}(y|x,\mathcal{D})$ is the prediction of our successful model. However, we do not have the true $p(\mathbf{w}|\mathcal{D})$, and instead we approximate $p(\mathbf{w}|\mathcal{D})$ with $q(\mathbf{w};\phi)$. The real predicted distribution is actually:

$$\hat{p}_{pred}(y|x;\phi) = \int_{\mathbf{w}} p_{net}(y|x,\tilde{\mathbf{w}}) q(\tilde{\mathbf{w}};\phi) d\tilde{\mathbf{w}}.$$

From these two equations, considering the integrals have $p_{net}(y|x,\tilde{\mathbf{w}})$ in common, we can infer that

$$\arg\min_{\phi} \mathsf{KL}(p(\mathbf{w}|\mathcal{D}) \,||\, q(\mathbf{w};\phi)) \Leftrightarrow \arg\min_{\phi} \mathsf{KL}(p_{gt}(y|x) \,||\, \hat{p}_{pred}(y|x;\phi)).$$

If we take both p_{gt} and $\hat{p}_{pred}$ to be normally distributed, that is $p_{gt} \sim \mathcal{N}(\mu_{gt}, \sigma_{gt}^2)$ and $\hat{p}_{pred} \sim \mathcal{N}(\mu_{pred}, \sigma_{pred}^2)$, the KL divergence between the two distributions is given by:

$$\mathsf{KL}(p_{gt}(y|x) \,||\, \hat{p}_{pred}(y|x;\phi)) = \frac{1}{2} \left(\log \left(\frac{\sigma_{pred}^2}{\sigma_{gt}^2} \right) + \frac{\sigma_{gt}^2 + (\mu_{gt} - \mu_{pred})^2}{\sigma_{pred}^2} - 1 \right).$$

Finally, as ground truth is (quasi-)deterministic, $\sigma_{gt}^2 \to 0$ i.e. the ground truth distribution effectively has no variance, and it can be treated as a constant. Therefore, since it does not affect the point in which the minimum is attained, we can remove it from consideration for minimising our expression to obtain

$$\frac{1}{2}\left(\log(\sigma_{pred}^2) + \frac{(\mu_{gt} - \mu_{pred})^2}{\sigma_{pred}^2} - 1\right).$$

Removing the constant $\frac{1}{2}$ and summing over all model predictions and ground truths from the dataset gives us exactly Equation (4.4).