Uncertainty in Implicit Neural Representations for Medical Imaging

Francisca Sousa Pereira Cruz de Vasconcelos
Keble College
University of Oxford

Submitted in partial completion of the
Master of Science in Statistical Science

September 13th, 2021

Acknowledgements

There are several people and institutions I would like to thank, without whom this research would not have been possible.

First and foremost, I would like to thank Professor Yee Whye Teh of the Oxford University Department of Statistics OxCSML group, for his supervision and guidance throughout this work. I appreciated meeting with him regularly to discuss results and gain valuable insights. Next, I would like to thank Oxford Statistics PhD candidate Bobby He, for his collaboration throughout the project. He provided an amazing starting point for the thesis, actively answered questions, and even occasionally helped to debug – all of which I am truly grateful for. I would further like to thank MIT PhD candidate Nalini Singh, for her guidance in the medical imaging domain, and look forward to further collaboration. More generally, I would like to thank the members of the OxCSML group for providing a welcoming, collaborative, and fun environment to do research, even during a global pandemic. I would like to give special thanks to PhD candidate Adam Goliński for maintaining and helping debug ziz; Professor Tom Rainforth for providing valuable insights during lunchtime discussions; and my office/lunch/football mates Jean-Francois Ton, Muhammad Faaiz Taufiq, Emilien Dupont, Valentin de Bortoli, and Andrew Campbell for the great company. Finally, I would like to thank those who made the masters degree program possible and the Statistics IT department, without which I would not have had access to the computational resources necessary for this work.

Beyond the Oxford Statistics Department, I would like to thank the many faculty, postdocs, and graduate students who previously taught and supervised me. I would not be at Oxford, nor able to work on such an interesting research problem without their prior mentorship. I particularly would like to thank Professor William Oliver of the MIT EQuS group, Professor Bill Freeman of the MIT CSAIL, and Professor Achuta Kadambi of the UCLA Visual Machines group for supporting me in my graduate and fellowship applications, beyond all that I learned in their labs and classes.

Institutionally, I would like to thank the Rhodes Trust for granting me the opportunity to pursue this masters degree, through the Rhodes Scholarship. Without their monetary and general support, I never would have been able to study in the UK. I also would like to thank Keble College, for providing great accommodation and general resources in pursuing my studies.

Personally, I would like to thank all my family and friends (both old and new) for their constant encouragement. In particular, thanks to Flat 27 for making even the worst of pandemic times enjoyable and fully immersing me in British culture; the Keble Rhodes scholars for many great times and adventures; the many friends I made through the degree program, especially the MCFC football lads; the OUAFC Women's Blues team for the great football, banter, and support through injuries; the many good friends from undergraduate who kept in touch via Zoom calls and social media; and the Pereira family for becoming my London family. I also would like to give special thanks to Christoher McDonald for being an amazing boyfriend who provided endless love, support, encouragement, and fun times throughout an amazing Oxford summer.

Finalmente, quero agradecer a toda a minha família em Portugal por todo o amor e suporte. I especially would like to thank my sister and parents, for their never-ending love, daily calls, constant support, and endless encouragement. Without them, I would not be where or who I am today. Obrigada a todos.

Abstract

In medical imaging, tissues and organs are visualized by measuring their interaction with electromagnetic waves or magnetic fields that penetrate the body from various angles. While highly successful, this practice usually involves exposing the patients to radiation, with the quality of the resulting images improving as the latter increases. This increases the risk of pathologies like cancer. In result, there is interest in reconstruction techniques that achieve high image quality from few measurements. Machine learning approaches based on deep learning have recently proven promising in this regard. However, they require large training datasets, which are difficult to collect in the medical setting. A significant recent advance was the introduction of implicit neural representations (INRs), which model images as continuous functions implemented by small neural networks.

This work proposes the first end-to-end INR architecture for computed tomography (CT) image reconstruction. The performance of this architecture is assessed in terms of image reconstruction quality and model calibration, i.e. the ability of the model to assign accurate confidence values to the reconstructed image pixels. Uncertainty quantification is particularly important for medical imaging since the reconstructed images inform doctor decisions and consequently patient outcomes. A good understanding of model confidence in reconstruction of each image region can affect those decisions, be leveraged for smart sensing acquisition of increased measurements in areas of larger uncertainty, or even be used for automated triage and assignment of images to healthcare providers with different degrees of specialization. Four established techniques from the uncertainty estimation literature are implemented and compared in this work: deep ensembles, Monte Carlo dropout, Bayes-by-backpropagation, and Hamiltonian Monte Carlo.

This study provides various interesting observations. It is shown that deep ensembles of Monte Carlo dropout base learners, with varied architectures, achieve the best image reconstruction performance and model calibration among the techniques tested; architecture parameters such as activation function and random Fourier feature embedding frequency can have large effects on model performance; some parameter choices (such as the use of the Sine activation function) can produce networks of high performance but require extensive tuning of the network architecture; previous intuitions about the role of certain network components (such as random Fourier feature embeddings enabling the learning of high-frequency image features) hold, but only when certain parameter values are precisely specified; and that Bayes-by-backpropagation is generally ill-suited for sampling from INR posterior distributions. Several metrics are also proposed and shown to be effective for monitoring the convergence of multi-chain Hamiltonian Monte Carlo methods in sampling from both the network parameter and output posterior distributions.

Contents

List of Figures **ix**

List of Tables **xi**

List of Abbreviations **xiii**

1 Introduction **1**

2 Background **5**

2.1 Medical Imaging 5

2.1.1 Radiation Exposure 6

2.1.2 Computed Tomography (CT) 7

2.1.3 X-Ray Attenuation 7

2.1.4 CT Projection Measurements 8

2.1.5 CT Image Reconstruction Problem 10

2.1.6 Implicit Neural Representations 10

2.2 Uncertainty Quantification for Neural Networks 12

2.2.1 Types of Uncertainty 12

2.2.2 Bayesian Neural Networks (BNNs) 12

2.2.3 Deep Ensembles 14

3 Methods **17**

3.1 Artificial Medical Data Generation 17

3.2 Classical CT Reconstruction 18

3.3 INRs with Uncertainty for Medical Imaging 18

3.3.1 Encoding CT Image Reconstruction in INRs 18

3.3.2 Uncertainty Quantification of INRs for CT Image Reconstruction 20

3.4 Tuneable MLP Model Parameters 21

3.5 Model Performance and Uncertainty Metrics 22

3.5.1 Calibration and Coverage 22

3.5.2 Assessing Model Quality 25

3.6 MLP Model Selection Procedures 26

3.6.1 Hardware, Software, and Contribution Notes . . . 26
3.6.2 BBB and MCD . . . 26
3.6.3 Deep Ensembles . . . 31
3.6.4 Hamiltonian Monte Carlo . . . 31

4 Experimental Results and Analysis 35
4.1 Classic CT Reconstruction . . . 35
4.2 MCD & BBB Model Analysis . . . 36
4.2.1 MCD Hyperparameter Analysis . . . 36
4.2.2 BBB Hyperparameter Analysis . . . 41
4.3 DE Performance Analysis . . . 43
4.4 HMC Run Analysis . . . 46
4.4.1 20-View Best Performing Single-Chain . . . 46
4.4.2 20-View Best Performing Multi-Chain . . . 47
4.4.3 5-View Best Performing Multi-Chain . . . 50
4.5 Uncertainty Quantification with INRs . . . 51

5 Conclusions 57

Appendices

A X-Ray Matter Interactions 61

B Classic CT Reconstruction Algorithms 63
B.1 Filtered Back-Projection (FBP) . . . 63
B.2 Algebraic and Iterative Reconstruction . . . 64
B.3 Simultaneous Iterative Reconstruction Technique (SIRT) . . . 65
B.4 Simultaneous Algebraic Reconstruction Technique (SART) . . . 65
B.5 Conjugate Gradient Least Squares (CGLS) . . . 66
B.6 Expectation Maximization (EM) . . . 66

C Neural Networks 69
C.1 Neural Network Mathematical Formulation . . . 69
C.2 Neural Network Training and Prediction . . . 71
C.2.1 Hyperparameter Sweeps . . . 72

D Tuneable MLP Model Parameters 73
D.1 Width and Depth . . . 73
D.2 Fourier Feature Mappings . . . 73
D.3 Activation Functions . . . 74
D.4 Optimizer . . . 75

E Model Performance and Uncertainty Metrics (Extended) 77

E.1 Peak Signal-to-Noise Ratio (PSNR) 77
E.2 Negative Log Likelihood (NLL) 77

F HMC Sampling Algorithm 79

References 81

List of Figures

2.1 CT Scanning Device 6
2.2 Tissue X-Ray Attenuation 7
2.3 CT Projective Measurements 8
2.4 Typical Set 14

3.1 Data-Generation Pipeline 18
3.2 CoiL Network vs Our End-to-End Approach 19
3.3 Activation Functions 21
3.4 Reliability Curve and δ Selection Example 23
3.5 Model Performance Metrics 24
3.6 Ground Truth Validation and Test Images 27
3.7 MCD Coarse Grid Hyperparameter Search 28
3.8 MCD Fine Bayesian Hyperparameter Search 29

4.1 Classical Image Reconstruction as a Function of View # 36
4.2 Classical Image Reconstruction Output 37
4.3 MCD PSNR by Activation Function Across Individual Images 37
4.4 MCD 20-View Best Image Reconstruction Across Images 38
4.5 MCD Hyperparameter Performance Boxplots 39
4.6 MCD Image Reconstruction Output by RFF Ω_0 40
4.7 BBB Hyperparameter Performance Boxplots 42
4.8 BBB Image Reconstruction Output by Prior Sigma and Width 43
4.9 DE Performance by Number of Base Learners 44
4.10 DE 5-View Image Reconstruction Output 45
4.11 DE 20-View Image Reconstruction Output 45
4.12 HMC 20-View Top Single-Chain Convergence Metrics 46
4.13 HMC 20-View Top Single-Chain Output 47
4.14 HMC 20-Views Top Multi-Chain Convergence Metrics 48
4.15 HMC 20-View Top Multi-Chain Outputs 49
4.16 HMC 20-View Best Multi-Chain Run $\hat{R}$ and t-SNE Plots 49
4.17 HMC 5-View Best Multi-Chain Output 50
4.18 HMC 5-View Best Multi-Chain t-SNE Visualization 51

4.19 5-View Validation Reconstruction and Calibration 54
4.20 20-View Validation Reconstruction and Calibration 55

B.1 FBP Image Reconstruction 64

C.1 Neural Network Building Blocks 70
C.2 Multi-Layer Perceptron 71
C.3 Sigmoid Function 71

List of Tables

3.1 MCD Sweep Relative Hyperparameter Importance 30

3.2 MCD Sweep Hyperparameter and Metric Correlation 30

4.1 Classical Reconstruction Performance 36

4.2 DE 5-View Top-10 Base Learners 43

4.3 DE 20-View Top-10 Base Learners 44

4.4 INR Uncertainty Quantification Results 53

List of Abbreviations

2D, 3D	Two- or three-dimensional.
FBP	Filtered back-projection.
ART	Algebraic reconstruction tomography.
CS	Compressed Sensing.
CNN	Convolutional neural network.
INR	Implicit neural representation.
CT	Computed tomography.
MRI	Magnetic resonance imaging.
PET	Positron emission tomography.
SPECT	Single-photon emission computed tomography.
FP	Forward-projection.
CGLS	Conjugate gradient least squares.
EM	Expectation-maximization.
SIRT	Simultaneous iterative reconstruction technique.
SART	Simultaneous algebraic reconstruction technique.
NN	Neural network.
GPU	Graphical processing unit.
MLP	Multilayer perceptron.
MSE	Mean-squared error.
NeRF	Neural radiance field.
SIREN	Sinusoidal representation network.
CoIL	Coordinate-based internal learning.
BNN	Bayesian neural network.
BBB	Bayes-by-Backprop.
MCD	Monte Carlo dropout.
HMC	Hamiltonian Monte Carlo.
MCMC	Markov chain Monte Carlo.

DE	Deep ensemble.
PSNR	Peak signal-to-noise ratio.
NLL	Negative log-likelihood.
NTK	Neural tangent kernel.
ReLU	Rectified linear unit.
SiLU	Sigmoid-weighted linear unit.
SGD	Stochastic gradient descent.
RFF	Random Fourier feature.
Adam	Adaptive moment estimation.
Adam-W	Adaptive moment estimation with weight decay.
CE	Calibration error.
ECE	Expected calibration error.
CMSE	Coverage mean-squared error.
PSRF	Potential scale reduction factor.
t-SNE	t-distributed stochastic neighbor embedding.

1 Introduction

Image reconstruction, generating 2D or 3D images of an organ from indirect and noisy measurements, is central to medical imaging. Unlike standard photography, many medical imaging devices cannot simply photograph visible light reflected by the organ of interest. Instead, these devices emit higher frequency electromagnetic waves (X-rays), lower frequency electromagnetic waves (micro- and radio- waves), or strong magnetic fields from several angles or viewpoints to penetrate human tissue, interacting with matter inside the human body. Longer and stronger exposures gather more information about the organ of interest, ultimately improving reconstruction quality. However, the intrusive nature of these of scans can be harmful to the human body, with radiation exposure linked to increased risk of long-term cancer [1–3]. Thus, it is important to minimize scan duration and strength as much as possible, while still gathering enough information to generate a high-quality reconstruction, enabling proper diagnosis.

Three primary categories of image reconstruction methods have been developed by the medical imaging community to-date: (1) analytical and algebraic, (2) iterative, and (3) data-driven [4]. Analytical and algebraic techniques, such as filtered backprojection (FBP) [5], were proposed as early as the 1980s, leveraging signal processing techniques, including the Fourier transform and filtering. Although these procedures are fast, they suffer from poor resolution-noise trade-off. Iterative approaches, such as the early algebraic reconstruction tomography (ART) procedure [6], are more powerful but computationally intensive. They only became clinically available in 2009, as sufficient computational power made them feasible. These methods achieved improved image quality by incorporating knowledge of the imaging system physics, reducing reconstruction artifacts and noise. Able to generate high-quality reconstructions, the medical imaging community focused on reducing scan time and radiation dosages, leading to the development of reduced sampling techniques, such as compressed sensing (CS) [7, 8]. This enabled image reconstruction from sparse measurement data, at the cost of slower reconstruction times. Meanwhile, with the growing popularity of machine learning, the medical imaging community began using other data-driven approaches. These include unsupervised learning [9, 10] and supervised deep learning approaches, specifically convolutional neural networks (CNNs) [11, 12]. These machine learning approaches generally outperform competing classical methods. However, they require large datasets for training, which can be difficult to collect in many medical settings. Furthermore, many of these methods – deep learning in particular – lack model interpretability, which proves especially problematic in medicine [13–15].

Beyond medical imaging, machine learning has undergone major advances in image representation and reconstruction over recent years. A significant recent advance was the introduction of implicit neural representations (INRs), which represent complex coordinate-based signals as functions encoded by small neural networks. For example, an image can be represented as a function mapping (x, y) coordinates to (r, g, b) pixel intensities. INRs have taken the field of computer graphics by storm, achieving impressive results in novel view synthesis [16–19], shape representation [20–25], and texture synthesis [26, 27]. More recent work has also demonstrated the applicability of this technique to medical imaging [28, 29]. In all of these applications, INRs have been assessed primarily on their predictive accuracy and the plausibility of their reconstructed signal. However, for medical imaging, which affects doctor decisions and patient well-being, it is also important to understand the confidence of the model in the reconstructed image values. For example, a model can quantify its uncertainty about the reconstruction quality of each image pixel by outputting a variance per pixel location. If this variance is large in critical image regions, such as the location of a potential tumor, a doctor should order additional scans to ensure proper diagnosis. Uncertainty quantification can also be used for automated triage, e.g. by assigning images with different levels of uncertainty to healthcare providers of different degrees of expertise. This could decrease overall health care costs or be useful in certain settings, such as education or remote diagnosis. Finally, understanding of model uncertainty could inform more efficient measurement procedures, leveraging techniques such as active learning [30].

This work addresses these limitations by proposing the first end-to-end INR architecture for computed tomography (CT) image reconstruction and assessing model performance in terms of both image reconstruction quality and model calibration. In particular, we report the first study of model selection for uncertainty calibrated INR architectures. Four established neural network uncertainty estimation and calibration methods – deep ensembles [31], Monte Carlo dropout [32], Bayes-by-backpropagation [33], and Hamiltonian Monte Carlo [34–36] – are implemented and analyzed. This analysis produces several significant observations:

1. Deep ensembles of Monte Carlo dropout base learners with varied architectures achieve the best image reconstruction performance and model calibration among the techniques tested.

2. Architecture parameters such as activation function and random Fourier feature embedding frequency can have large affects on model performance. For example, the Sine activation function achieves the best results for Monte Carlo dropout but can be quite inconsistent, producing networks with large variation in reconstruction performance and calibration as network hyperparameters, such as width or depth, are changed. In practice, this implies that these networks require extensive parameter tuning to deliver their best performance. Other activations, such as SiLU, perform slightly worse but are more robust with respect to parameter configurations.

3. Previous intuitions that random Fourier feature embeddings can enable the INR network to learn high-frequency image features are confirmed. However, it was also found that the embedding frequency must match the amount of training data available. Poor specification of the frequency parameter can lead to reconstructions that are either too blurry or have too many high frequency artifacts.
4. Bayes-by-backpropagation is generally ill-suited for sampling from INR posterior distributions, achieving the worst performance of all uncertainty quantification approaches.
5. Several metrics can be used to monitor the convergence of multi-chain Hamiltonian Monte Carlo methods when sampling from both the network parameter and predictive posteriors.

2 Background

Contents

2.1 Medical Imaging 5
 2.1.1 Radiation Exposure 6
 2.1.2 Computed Tomography (CT) 7
 2.1.3 X-Ray Attenuation 7
 2.1.4 CT Projection Measurements 8
 2.1.5 CT Image Reconstruction Problem 10
 2.1.6 Implicit Neural Representations 10
2.2 Uncertainty Quantification for Neural Networks 12
 2.2.1 Types of Uncertainty 12
 2.2.2 Bayesian Neural Networks (BNNs) 12
 2.2.3 Deep Ensembles 14

2.1 Medical Imaging

Medical imaging plays an essential role in the diagnosis and monitoring of a range of medical conditions, such as broken bones, stroke, cancer, blood clots, gastrointestinal issues, and, more recently, COVID-19 [37]. The interior of the human body is generally imaged for clinical analysis, medical intervention, and visual representation of organ or tissue physiology. There are a wide range of imaging technologies, including X-ray radiography, magnetic resonance imaging (MRI), computed tomography (CT), ultrasound, positron emission tomography (PET), and single-photon emission computed tomography (SPECT). Such imaging devices emit high-frequency X-rays, microwaves, radio-waves, or strong magnetic fields that penetrate the human body from several viewpoints or angles, interacting with internal tissue. Reconstruction algorithms use the measurements of these attenuated signals to reconstruct a 2D or 3D image of the tissue. In general, image reconstruction accuracy improves with longer and stronger exposures, from an increased number of views or angles.

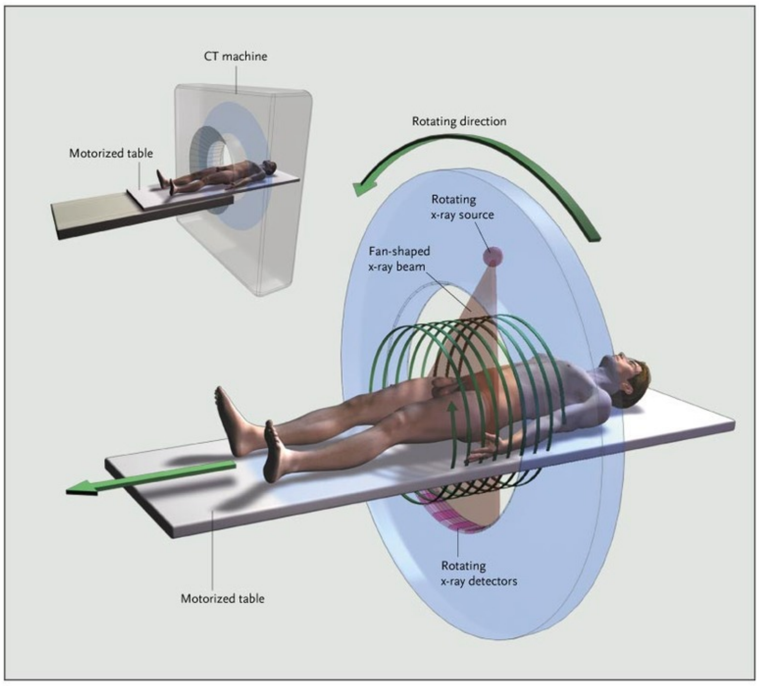

Figure 2.1: Illustration of a CT scanning device. Reproduced with permission from [38], Copyright Massachusetts Medical Society.

2.1.1 Radiation Exposure

As of 2010, 5 billion medical imaging studies were performed worldwide, two-thirds of which employed ionizing radiation [2]. Since then, the use of radiology has only grown, with a 16% increase in the United Kingdom from 2013 to 2018 and over 42 million examinations carried out by the UK National Health Service in 2016-17 [39]. Whilst the increased use of radiology has undoubtedly improved medical care, diagnostic X-rays are the largest man-made source of radiation exposure to the general population [3]. CT comprises the majority of this exposure [38] and will be the focus of this work.[1]

Such radiation exposure has been known to increase the risk of cancer, with an estimated 29,000 current or future cancer cases linked to CT scans performed in the United States of America in 2007 alone [40]. While there are alternative imaging platforms available, CT scanners provide a compromise between the highly detailed images generated by MRI and the quicker scan time and lower cost of plain X-ray imaging. Further, different imaging modalities enable better visualization of different tissue types, with not all patients able to use all imaging modalities. For example, patients with metal implants, such as pacemakers, cannot use MRIs. This makes it impossible to simply use other imaging modalities and creates interest in reducing radiation exposure of CT exams. Minimizing radiation strength and time is an active, ongoing area of research [41–43]. In this work, we focus on improving algorithmic reconstruction performance of CT images, especially in the limited measurement regime, through uncertainty quantification. In addition to enabling better medical diagnosis, good model calibration could eventually enable reduced measurements and radiation exposure, via machine learning techniques such as active learning.

[1]It should be noted that while we focus on CT imaging, the techniques developed in this work can be applied more broadly to other medical imaging techniques.

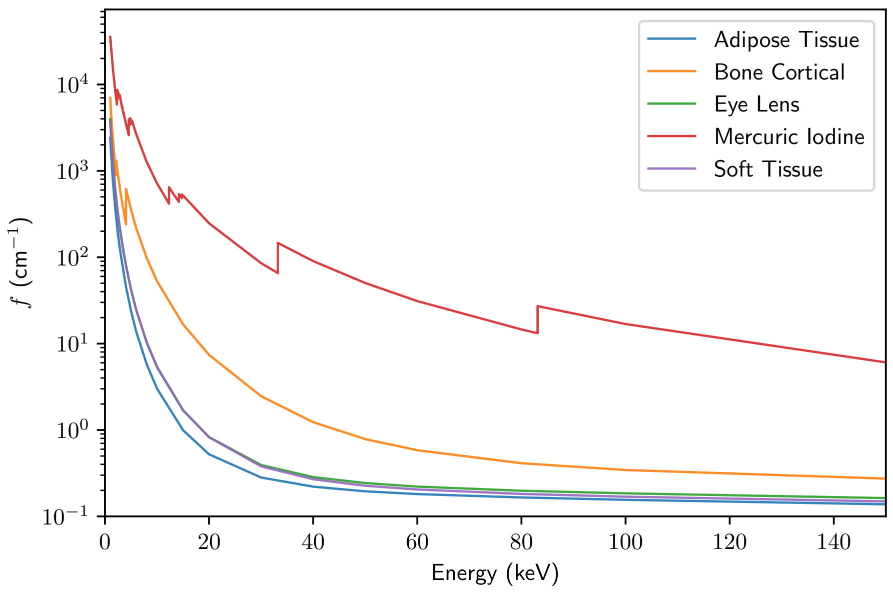

Figure 2.2: The attenuation coefficient (f) of different tissues found in the body, as a function of energy (keV). The plot was generated using X-ray mass attenuation coefficients (f/ρ) and densities (ρ) from the NIST Standard Reference Database [45, 46].

2.1.2 Computed Tomography (CT)

Computed tomography (CT), also known as computed axial/assisted tomography (CAT), is a noninvasive medical imaging technique frequently used in radiology to generate detailed images of the body. Since its original development in the 1970s, CT has become widespread in medical imaging – with over 70 million CT scans taken and reported annually in the United States, since 2007 [44]. There are multiple types of CT scanners, such as spiral CT, electron beam CT, and CT perfusion imaging. For now, we focus on spiral, also know as spinning tube or helical, CT.

In **spiral CT**, illustrated in Figure 2.1, the patient lies along the central axis of a cylindrical measurement tube. As the scan is performed, an X-ray generator rotates around the patient while moving along the axis of measurement. X-rays are emitted, which pass through the patient and are attenuated at various rates by the different types of tissues in the body, as described in Section 2.1.3. After exiting the body, the attenuated X-rays are measured by X-ray detectors positioned and moving opposite to the X-ray source. These measurements are used to create a sinogram, as described in Section 2.1.4, which is not understandable by doctors. This sinogram is then input to a reconstruction algorithm, which solves an under-determined inverse problem, described in Section 2.1.5, to generate a human-understandable 2D or 3D image of the organ of interest. This image can then be used by the doctor for medical diagnosis.

2.1.3 X-Ray Attenuation

X-rays produced by CT scanners can interact with matter via the photoelectric effect, the Compton effect, and coherent scattering, as described in Appendix A. Through these interactions, some

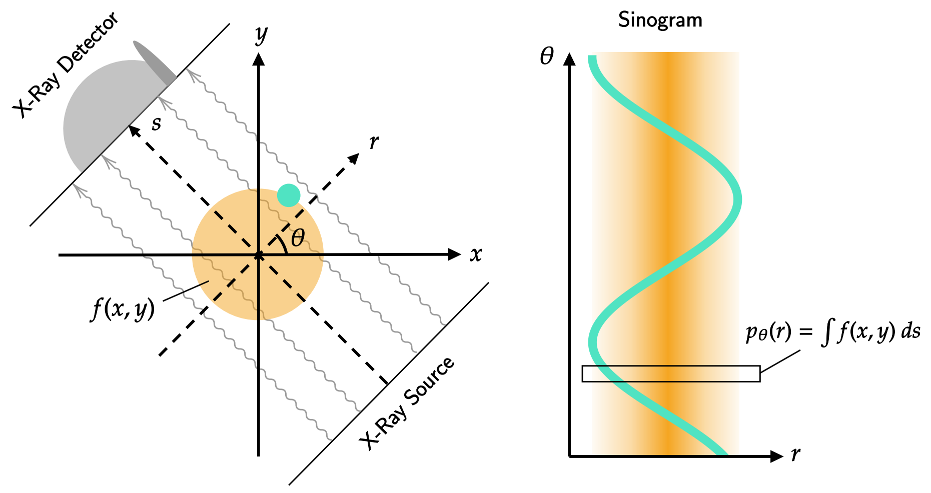

Figure 2.3: An abstraction of the CT measurement process and resultant sinogram data. In a CT scan, an X-ray source and detector spin around a patient, in this case a large orange circle and small blue point. The measurement axes, defined by r and s, rotate around the origin of the fixed frame axes, defined by x and y, according to measurement angles θ. At each angle, X-rays are sent in parallel along r through the patient and are attenuated according to the different tissue attenuations, $f(x, y)$, in the patient before being measured by the detectors. The measurement values for each each angle, $p_\theta(r)$, are stacked to produce a sinogram. Notice that the blue off center point appears as a sinusoid in the sinogram.

of the emitted X-ray photons are absorbed or scattered when passing through different tissues in the body. The attenuation is described by the **Beer-Lambert Law**,

$$J = J_0 e^{-fL}, \tag{2.1.3.1}$$

where J and J_0 are the incident and transmitted X-ray intensities; L is the material thickness; and f is the linear attenuation coefficient of the material,

$$f = \tau_1 + \tau_2 + \tau_3, \tag{2.1.3.2}$$

with photoelectric (τ_1), Compton (τ_2), and coherent scattering (τ_3) attenuation coefficients. Attenuation coefficients for common materials in the body – iodine, bone, water, and soft-tissue – are plotted, in Figure 2.2, over the range of incident X-ray energies used in CT imaging. It is clear that the attenuation of X-ray photons can be used to distinguish and, thus, image various tissues in the body [47].

2.1.4 CT Projection Measurements

As described in Section 2.1.2, the CT scan relies on an X-ray generator which rotates around the patient, emitting X-ray photons. In this work, we only consider the restricted case of the spiral CT setup, in which there is no motion along the patient axis. Instead, we focus on reconstructing singular 2D image cross-sections and assume a parallel-beam geometry, in which photons are emitted and detected with the linear geometry of Figure 2.3.

We begin by considering the measurements of a single detector, measuring at angle θ. Assume that the X-ray generator outputs monoenergetic X-rays of intensity J_0. If the patient were simply a homogeneous block of tissue, with length $\Delta\ell$ and attenuation coefficient f, we could directly apply the Beer-Lambert law (Equation 2.1.3.1),

$$J = J_0 e^{-f\Delta\ell}, \tag{2.1.4.1}$$

to solve for the output attenuated X-ray intensity J. In reality, several blocks of tissue will be present in the patient, each with its own attenuation coefficient. However, since the exit X-ray flux from one block of tissue is the entrance X-ray flux to its neighboring block, we can simply apply the Beer-Lambert law in a cascading fashion over intervals of length $\Delta\ell$ and attenuation coefficients $(f_1, f_2, ..., f_n)$,

$$J = J_0 e^{-f_1\Delta\ell} e^{-f_2\Delta\ell} ... e^{-f_n\Delta\ell} = J_0 e^{-\sum_{i=1}^{n} f_i \Delta\ell}. \tag{2.1.4.2}$$

As $\Delta\ell \to 0$, the summation term becomes an integration over the length, L, of the patient,

$$J = J_0 e^{-\int_L f(\ell)d\ell}. \tag{2.1.4.3}$$

Finally, dividing both sides of the expression by J_0 and taking a negative logarithm, we define the **projection measurement** term,

$$p_\theta = -\ln\left(\frac{J}{J_0}\right) = \int_L f(\ell)d\ell. \tag{2.1.4.4}$$

As illustrated in Figure 2.3, a CT scanner with a parallel beam geometry contains several detectors side-by-side, collectively measuring a plane of attenuated X-ray photons. In this 2D imaging space, the projection measurement becomes a function of detector position, r. Thus, Equation 2.1.4.4 is re-expressed as the line integral,

$$p_\theta(r) = -\ln\left(\frac{J}{J_0}\right) = \int f(\theta, r)ds, \tag{2.1.4.5}$$

known as a **forward-projection (FP)**. It follows from the coordinate system of Figure 2.3 that, for measurements at angle θ, point (x, y) within the patient cross-section is projected onto detector position

$$r' = x\cos\theta + y\sin\theta. \tag{2.1.4.6}$$

Combining this with (2.1.4.5), we derive the **Radon transform** [48] of the patient cross-section,

$$p_\theta(r) = -\ln\left(\frac{J}{J_0}\right) = \int_{\mathcal{Y}}\int_{\mathcal{X}} f(x, y)\ \delta(x\cos\theta + y\sin\theta - r)\ dxdy, \tag{2.1.4.7}$$

where δ is the Dirac delta function and $\mathcal{X} \times \mathcal{Y}$ is the set of image pixels (x, y). Sweeping over all the angles, these projective measurements are stacked to form a Radon transform image. As depicted in Figure 2.3, the projection measurements of the blue point across several angles produces a sinusoidal curve. The representation of all the CT scan measurements is thus known as a **sinogram**.

2.1.5 CT Image Reconstruction Problem

Sinograms are not human-interpretable. They depict the integrated attenuation coefficients, or projection measurements ($p_\theta(r)$), of the patient cross-section from several angles (θ) over all detector positions (r). Instead, the desired outcome of a CT scan is a reconstructed image of the cross-section itself. This corresponds to the attenuation coefficient function, $f(x, y)$, which is the inverse of the Radon transform of (2.1.4.7), or

$$f(x, y) = \frac{1}{2\pi} \int_0^\pi u_\theta(x \cos\theta + y \sin\theta)\, d\theta, \tag{2.1.5.1}$$

where u_θ is the derivative of the Hilbert transform of $p_\theta(r)$ [49]. The **projection-slice theorem** [50] ensures that f can be fully reconstructed with infinite measurement angles, θ. In practice, however, it is not possible to acquire infinite measurements. Typically, reconstruction quality improves with number of measurements, but this increases radiation exposure. In practice, hundreds of measurements are performed in a CT scan, but there is interest in reducing this number. In this work, we study algorithm performance in the very low measurement data regime, where uncertainty quantification over the value of $f(x, y)$ becomes especially important. The combination of limited and noisy real-world data renders the reconstruction of the desired image much more complex than simply evaluating the integral of (2.1.5.1). Leveraging assumptions or data-driven insights about the measurements and physics at play, several statistical models have been developed and used to derive various image reconstruction algorithms. For a discussion of some of the main, clinically approved, classical image reconstruction algorithms considered in this work – **filtered back-projection (FBP)**, **conjugate gradient least squares (CGLS)**, **expectation-maximization (EM)**, **simultaneous iterative reconstruction technique (SIRT)**, and **simultaneous algebraic reconstruction technique (SART)** – we refer the reader to Appendix B.

Note: For the remainder of this work, we will denote the linear attenuation coefficient function as $f \in \mathcal{F}$. Further, $f(x, y)$ denotes the linear attenuation coefficient at each pixel (x, y) in the ground truth image I. Our statistical model will use sinogram measurement data, $p_\theta(r)$, to produce a reconstructed image, K, corresponding to a predicted attenuation coefficient function $\hat{f}(x, y)$.

2.1.6 Implicit Neural Representations

In this work, we investigate an alternative to the CT image reconstruction problem based on artificial neural networks (ANNs), frequently referred to as **neural networks (NNs)**. These are formally defined and described in the broader context of deep learning in Appendix C. **Implicit neural representations (INRs)**, also known as coordinate-based representations or neural function representations, are a novel means of parameterizing signals, using small NNs. Typically, signals used in computation are discrete. For example, 2D images are generally represented as grids of pixels, while 3D objects are often represented as point-clouds, grids of voxels, or meshes. INRs, however, parameterize all these signals as *continuous functions*, mapping between low-dimensional spaces, from coordinate to signal value. This function is learned and approximated by a small NN.

For example, a 2D image is parameterized by the function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^3$, implemented with a NN which takes as input the pixel location (x, y) and outputs RBG pixel values. The image can be fully reconstructed by sampling the NN at each pixel location. Similarly, a 3D object is parameterized by the function $h : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \Theta \times \Phi \rightarrow \mathbb{R}^3 \times \Gamma$, implemented with a NN which takes as input the pixel coordinate (x, y, z) and view direction (θ, ϕ), outputting RGB pixel values and opacity γ. In this case, the object is reconstructed via volume rendering, by strategically sampling pixels in the object point-cloud and integrating pixel colors and opacity over the view direction.

In 2019, INRs were demonstrated to outperform traditional grid, mesh, and point-cloud approaches for shape modeling and reconstruction. A 3D object shape is instead represented as a signed distance function (SDF)[2] learned from a set of training shapes [20, 25, 54]. Since then, INRs have been used as continuous, memory-efficient representations of shape parts [22, 23], objects [55–57], and scenes [24, 58, 59]. In addition to representing shapes via SDFs, INRs were also extended to encode object appearance [17, 18, 27], as in the previously described 2D image and 3D object examples. Among the most impactful of these works are **neural radiance fields (NeRF)** [16], which achieve state-of-the-art results in novel view synthesis by using random Fourier features as positional encodings, facilitating the representation of high frequency functions by the MLP [60].

In the year since the publication of the original NeRF paper, there has been an explosion of literature applying and improving the technique [61]. Particularly impressive among NeRF extensions is NeRF in the Wild (NeRF-W), which renders high-resolution 3D scenes from unstructured collections of 2D images ‘in the wild’ and encodes transient scene features as tuneable latents [62]. Among attempts to improve NeRF performance, **sinusoidal representation networks (SIRENs)**, which use sinusoidal activation functions, were argued to outperform ReLU-based INRs [63]. Meta-learning has also been shown to improve INR performance [19, 64–66], by improving model initialization to enable faster convergence and improving performance in the case of limited data. As far as we are aware, no work has been done to quantify uncertainty of INRs, barring the use of active learning to further improve performance in limited-data regimes. More recently, INRs were applied directly to the CT image reconstruction challenge. The **coordinate-based internal learning (CoIL)** network [28] uses an INR with a Fourier feature mapping to learn the sinogram measurement field from sparse measurements. Meanwhile, the AutoInt network [67] automates integration by instantiating an integral NN but training over the derivative NN, offering an end-to-end means of calculating the Radon transform with only two network evaluations. This work found that, in the context of CT image reconstruction, although SIREN fit the measurements best, the swish activation function with positional encoding generalized the best, especially to unseen views. Finally, INRs have also been used as static priors for parametric motion fields in the reconstruction of dynamic (4D) CT [29]. As was generally the case with INRs, none of these works mention or quantify uncertainty in the final model and reconstruction.

[2]A SDF is a continuous function that encodes an object as a surface. For a given point in 3D, the SDF outputs the point’s distance to the closest surface point. The sign of the distance determines whether the point lies inside (negative) or outside (positive) of the surface. If the distance is zero, the point lies exactly on the object surface [51–53].

2.2 Uncertainty Quantification for Neural Networks

For a given measurement data set, many NN parameters settings, W, can achieve decent image reconstruction. In this work, we leverage this non-uniqueness in network parameterization to quantify the uncertainty of the network predictions $\hat{f}$.

2.2.1 Types of Uncertainty

Bayesian modeling can address two distinct types of uncertainty: aleatoric and epistemic [68, 69]. **Aleatoric uncertainty** is due to *measurement noise*, such as X-ray detector noise. This type of uncertainty cannot be reduced, even if more measurements are taken, since it is inherent to the measurement. To see why, think of rolling an unbiased die. Irrespective of how many times you roll the die, you will always be uncertain of the outcome of the next roll, since each outcome has a $\frac{1}{6}^{\text{th}}$ probability. On the other hand, **epistemic** or **model uncertainty** accounts for *uncertainty in the model parameters*. This type of uncertainty can be reduced with more measurement data. To see why, imagine a model that aims to predict the outcome of a biased die roll, with no prior information about the bias. As more data is taken, the variance in the model parameters decreases, and the model output distribution better approximates the true biased die distribution. This work primarily utilizes noiseless data, meaning we are most interested in quantifying epistemic/model uncertainty. Namely, we consider how well the model reconstructs the ground truth attenuation coefficient image from the sinogram data.

2.2.2 Bayesian Neural Networks (BNNs)

In deep learning, epistemic uncertainty can be quantified with resort to **Bayesian neural networks (BNNs)** [70]. Recall that a NN outputs $\hat{f}$, an approximation of the desired ground truth, f. Assume we are given: a function over ground truth values $h(f_1, ..., f_n)$[3], a prior over the network weights $p(W)$, and an observation likelihood $p(h|W)$. Bayes' rule can be applied to calculate the weight posterior,

$$p(W|h) \propto p(h|W)\, p(W), \tag{2.2.2.1}$$

which is then used to quantify the uncertainty over the the network output, $\hat{f}$, through the posterior predictive function

$$p(\hat{f}|h) = \int p(\hat{f}|W)\, p(W|h)\, dW. \tag{2.2.2.2}$$

Due to the high-dimensional parameter space, this integral must be approximated to enable computationally efficient BNN inference.

One such popular method is **Bayes-by-Backprop (BBB)** [33], a variational approximation to exact Bayesian updates. Variational learning aims to find the optimal parameters ψ of an approximate distribution on the NN weights, $q(W|\psi)$, also known as the variational posterior. This is achieved by maximizing the variational free energy / evidence lower bound (ELBO),

$$\mathcal{F}(h, \psi) = \mathbb{E}_{q(W|\psi)}[\log p(h|W)] - \mathbb{KL}\left[\, q(W|\psi) \,||\, p(W) \,\right], \tag{2.2.2.3}$$

[3]In the context of medical imaging with INRs, $h(f_1, ..., f_{|\mathcal{X}\times\mathcal{Y}|})$ is the Radon transform of the network output at all pixel locations (Equation 2.1.4.7), since our data is a sinogram consisting of projective measurements $p_\theta(r)$.

with respect to the weight distribution parameters ψ, i.e. computing $\arg\max_{\psi}\mathcal{F}(h,\psi)$. The ELBO can be approximated by Monte Carlo,

$$\mathcal{F}(h,\psi) \approx \sum_{i=1}^{n} \log p(h|W_{(i)}) + \log p(W_{(i)}) - \log q(W_{(i)}|\psi), \tag{2.2.2.4}$$

where $W_{(i)}$ are samples drawn from variational posterior $q(W_{(i)}|\psi)$. In this work, a Gaussian distribution is used as the variational posterior, parameterized by $\psi = (\mu_\psi, \sigma_\psi)$, with mean μ_ψ and standard deviation σ_ψ. The elements of σ_ψ comprise a diagonal covariance matrix, meaning weights are assumed to be uncorrelated. A Gaussian prior, $p(W) = \mathcal{N}(W|\sigma^2)$, with tunable σ is used to initialize the network. Training the network requires computing a forward-pass and backward-pass. Although the network is parameterized by a distribution of weights, in each forward pass a single sample is drawn from the variational posterior and propagated through the network to perform updates. A re-parameterization trick, in which the sample ϵ is transformed by the function $\mu_\psi + \sigma_\psi \odot \epsilon$, is used to ensure a gradient can be calculated for backpropagation. Finally, to aid learning, it is common to modify the ELBO as

$$\tilde{\mathcal{F}}(h,\psi) = \mathbb{E}_{q(W|\psi)}[\log p(h|W)] - \xi \cdot \mathbb{KL}\left[\, q(W|\psi) \,||\, p(W) \,\right], \tag{2.2.2.5}$$

where $\xi << 1$ is an added hyperparameter, known as the KL factor. This is beneficial for training because it puts greater emphasis in the loss on the training data, through the $\mathbb{E}_{q(W|\psi)}[\log p(h|W)]$ term[4].

Another popular approach to BNN approximate inference is **Monte Carlo dropout (MCD)** [32]. A neural network, of arbitrary non-linearities and depth, is mathematically equivalent to an approximate deep Gaussian process if dropout is applied to every weight layer. Thus, the network's dropout objective minimizes the KL-divergence between an approximate distribution and the deep Gaussian process posterior. The first two moments of this distribution can be empirically approximated, using a Monte Carlo estimate of B Bernoulli sampled weight matrices, $W_{(b)}$. The predictive mean is thus calculated as,

$$\mathbb{E}_{q(\hat{f}|h)}(\hat{f}) \approx \frac{1}{B}\sum_{b=1}^{B} \hat{f}(W_{(b)}), \tag{2.2.2.6}$$

which is equivalent to averaging the results of B stochastic forward passes through the network.

A final means of approximate BNN inference, which we explore in this work, is **Hamiltonian Monte Carlo (HMC)** [34–36]. Originally proposed for calculations in lattice quantum chromodynamics [34], HMC is an instance of the Metropolis-Hastings algorithm [71] for Markov Chain Monte Carlo (MCMC) [72]. In the context of quantifying NN uncertainty, HMC can be used to obtain a sequence of random samples that converge to samples from the BNN posterior distribution, $p(W|h)$ [73]. For the high-dimensional distributions associated with BNNs, the density $p(W|h)$ is concentrated at the distribution mode, while the volume dW is concentrated at the distribution tails. As illustrated in Figure 2.4, the resulting distribution expectation – a product $p(W|h)dW$ of distribution volume and density – concentrates in a nearly-singular neighborhood,

[4]In the context of medical imaging with INRs, this emphasizes how closely the Radon transform of the network outputs are to the sinogram measurement data.

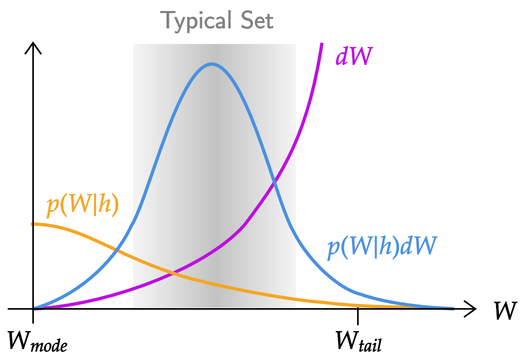

Figure 2.4: An illustration of concentration of measure, in which the expectation, $p(W|h)dW$, of the BNN's high-dimensional weight posterior is concentrated in the typical set. This can be attributed to the fact that, while most of the distribution density $p(W|h)$ is concentrated about the distribution modes, the volume dW is concentrated at the tails. Unlike other MCMC methods, HMC efficiently explores and samples from the entire distribution typical set. (Figure inspired by [36].)

known as the *typical set*. Traditional Metropolis-Hastings MCMC, using a Gaussian random walk proposal distribution, typically fails to explore the full typical set of these distributions. HMC, however, leverages the physics of Hamiltonian dynamics via a time-reversible and volume-preserving integrator, to simulate and propose points scattered around the typical set. This significantly improves exploration of the full distribution and decreases the correlation between consecutive samples, reducing the total number of required MCMC samples.

2.2.3 Deep Ensembles

Alternatively, predictive uncertainty can be quantified by aggregating the outputs of several NNs trained for the same task, a method known as **deep ensembles (DEs)** [31]. This is inspired by ensembling statistical models, i.e. trees, where randomization is achieved by bagging, or training several models on different data bootstrap samples. If the base learner lacks intrinsic randomness, meaning it can be trained efficiently via convex optimization, bagging induces diversity and reduces variance. However, since NN training can have many local optima and should be trained with as much data as possible, such a bootstrap procedure can be detrimental to performance [74]. DEs of NNs instead induce randomization by randomly initializing each network's parameters and randomly shuffling the data. For models that already quantify uncertainty, DEs can be used to improve uncertainty calibration. Combining base learners by ensembling helps to capture model uncertainty, by averaging predictions over multiple models consistent with the training data. This makes the ensemble robust to model misspecification and out-of-distribution examples. It was shown in the original DE work [31], that even as few as five base-learners can significantly improve uncertainty calibration. In principle, more base learners results in better performance, but training large ensembles is computationally expensive, given the need to train each individual base learner (if not more, in order to prune for the best performing models).

In this work, we consider DEs in which each base learner has a different architecture and a unique set of weights. Recent work [75] has shown that ensembling over architectures can outperform the more common single-architecture DEs for uncertainty estimation. The ensemble of NN base learners is defined as a uniformly weighted mixture model, combining predictions as

$$p(\hat{f}) = \frac{1}{M} \sum_{m=1}^{M} p(\hat{f}_{(m)} | W_{(m)}), \tag{2.2.3.1}$$

where M is the total number of NNs in the ensemble and each $\hat{f}_{(m)}$ is a unique architecture with learned weights $W_{(m)}$. For regression tasks, the predictive uncertainty of the DE is approximated by a Gaussian distribution of mean and variance

$$\mu(\hat{f}) = \frac{1}{M} \sum_{m=1}^{M} \mu(\hat{f}_{(m)} | W_{(m)}) \tag{2.2.3.2}$$

$$\sigma^2(\hat{f}) = \frac{1}{M} \sum_{m=1}^{M} (\sigma^2(\hat{f}_{(m)} | W_{(m)}) + \mu^2(\hat{f}_{(m)} | W_{(m)})) - \mu^2(\hat{f}). \tag{2.2.3.3}$$

3 Methods

Contents

3.1 Artificial Medical Data Generation 17
3.2 Classical CT Reconstruction 18
3.3 INRs with Uncertainty for Medical Imaging 18
3.3.1 Encoding CT Image Reconstruction in INRs 18
3.3.2 Uncertainty Quantification of INRs for CT Image Reconstruction . . 20
3.4 Tuneable MLP Model Parameters 21
3.5 Model Performance and Uncertainty Metrics 22
3.5.1 Calibration and Coverage 22
3.5.2 Assessing Model Quality 25
3.6 MLP Model Selection Procedures 26
3.6.1 Hardware, Software, and Contribution Notes 26
3.6.2 BBB and MCD 26
3.6.3 Deep Ensembles 31
3.6.4 Hamiltonian Monte Carlo 31

3.1 Artificial Medical Data Generation

In this work, we use the **Shepp-Logan phantom** data generation method [76], implemented via the `phantominator` Python package, to create artificial images of 2D brain cross-sections, known as **phantoms**. These artificial brain images are produced by overlaying randomly oriented and sized ellipses with different attenuation coefficients on a background of 256×256 pixels. The Radon transform, discussed in Section 2.1.4, is used to generate sinograms of these ground truth images, producing artificial measurement data. The number of **views** is defined as the number of measurement angles, θ, used in generating the sinogram. Gaussian noise of zero mean and tunable standard deviation is added to all pixels, to create sinograms that more realistically approximate actual measurement data. The data generation pipeline is depicted in Figure 3.1. In this work, we use noiseless data for most experiments. Assessing model performance in the noisy data regime is an important next step, which is left for future work.

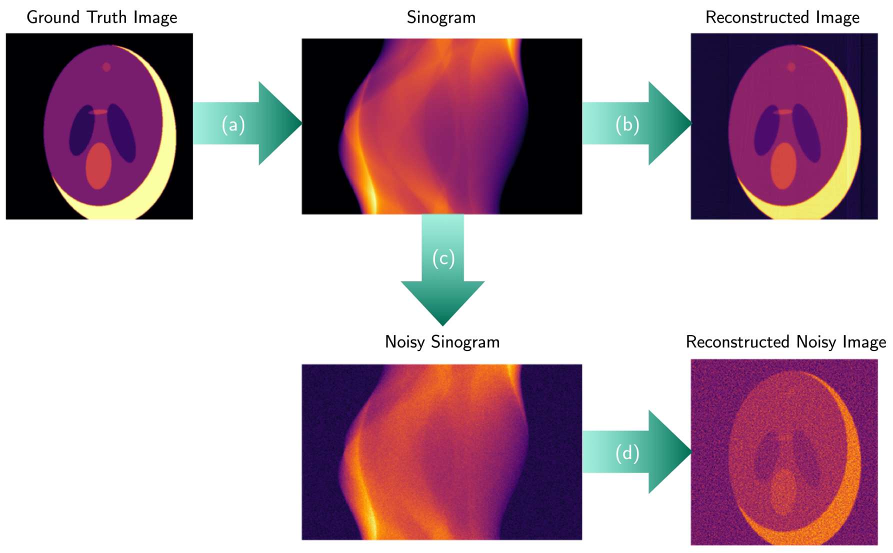

Figure 3.1: A flowchart of the artificial data generation pipeline used in this work. A Shepp-Logan phantom is used as groundtruth image. **(a)** The Radon transform is used to generate a sinogram, corresponding to measurement data. **(b)** In the noiseless case, this sinogram is directly used to train our model, which produces a reconstructed image of the ground truth. Note that even though the sinogram is noiseless, the reconstruction is not perfect. **(c)** In a more realistic scenario, Gaussian white noise is added to the sinogram. **(d)** This noisy sinogram data is then fed into the model, which produces a reconstructed image of the ground truth, of lower quality that produced in the noiseless case.

3.2 Classical CT Reconstruction

Five classic CT image reconstruction methods – FBP, CGLS, EM, SART, and SIRT – described in Appendix B, were used to establish a baseline for reconstruction performance. These were implemented with the `TomoPy ASTRA` package [77]. Their results are reported in Section 4.1.

3.3 INRs with Uncertainty for Medical Imaging

In the context of CT medical imaging, the INR model should quantify uncertainty in its reconstruction, $\hat{f}$, of the attenuation coefficient function, f. Recall, however, that the model only has access to potentially noisy projective measurement data, $p_\theta(r)$, which is related to f by the Radon transform, as described in Section 2.1.4. Hence, uncertainty quantification of INRs for medical imaging requires some modifications to the previously described MLP formulation.

3.3.1 Encoding CT Image Reconstruction in INRs

While the CT measurement data comes in the form of a sinogram, $p_\theta(r)$, the goal of reconstruction is to generate a photo of the cross-section of attenuation coefficients, $f(x, y)$. In Section 4.1, we experimentally demonstrate how classic reconstruction performance improves as a function of increased measurements views, as predicted by the projection slice theorem. The recent CoIL work [28] leverages this by using an INR to learn a functional form of the sinogram. The model input is sinogram location (θ, r) and the output is projection measurement $p_\theta(r)$. The resulting sinogram

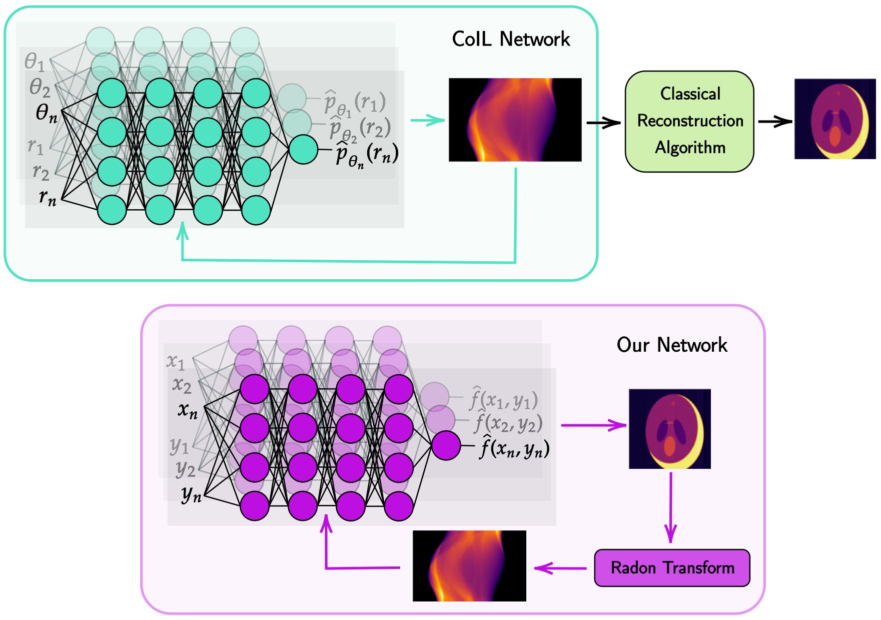

Figure 3.2: The CoIL [28] approach to image reconstruction learns a functional form of the sinogram and relies on an external, classical reconstruction algorithm to generate the desired CT cross-section image. Our approach instead leverages an end-to-end architecture, directly learning a functional form of the desired reconstruction image. This is achieved by applying a Radon transform to the fully sampled image in each training epoch, in order to retrieve the reconstructed sinogram for training.

is then passed to a classical reconstruction algorithm, which reconstructs image $\hat{f}$. By generating a functional sinogram, the CoIL INR enables sampling from more view angles than the original measurement data. This enables the classical algorithm to achieve improved reconstruction quality.

In this work, we consider instead, an **end-to-end** approach to image reconstruction. Rather than predicting the sinogram measurement field, our model directly predicts the final cross-section attenuation coefficient function, as illustrated in Figure 3.2. This is mostly for two reasons. First, experience with NNs has shown that end-to-end predictions are generally more accurate than those produced by combinations of handcrafted modules. Second, the uncertainty quantification must be presented at the pixel level to be of use for most applications. The network must thus generate an output image and associated uncertainty map. For 2D Shepp-Logan phantoms, the model input is a pixel coordinate (x, y) and its output is the corresponding predicted attenuation coefficient value $\hat{f}(x, y)$. The sinogram data is incorporated in the model via the training loss function. Given a groundtruth sinogram $p_\theta(r)$, the loss of the INR output $\hat{f}(x, y)$ is defined as

$$\mathcal{L}\big(p_\theta(r), \hat{f}(x,y)\big) = \frac{1}{2|\Theta \times \mathcal{R}|} \sum_{\theta \in \Theta} \sum_{r \in \mathcal{R}} \left(p_\theta(r) - \int_{\mathcal{Y}} \int_{\mathcal{X}} \hat{f}(x,y)\, \delta(x \cos\theta + y \sin\theta - r)\, dxdy \right)^2, \tag{3.3.1.1}$$

where $\Theta = \{\theta_1, ..., \theta_n\}$ is the set of view angles, $\mathcal{R} = \{r_1, ..., r_n\}$ the set of discrete detector measurement locations, $\mathcal{X} \times \mathcal{Y}$ the set of image pixels (x, y), and the integral surrounding $\hat{f}$ is the Radon transform. For sinograms derived from an artificial ground truth image, $f(x, y)$, the projective measurements are defined as

$$p_\theta(r) = \mathcal{N}(0, \sigma^2) + \int_{\mathcal{Y}} \int_{\mathcal{X}} f(x,y)\, \delta(x \cos\theta + y \sin\theta - r)\, dxdy, \tag{3.3.1.2}$$

where $\mathcal{N}(0, \sigma^2)$ is an optional Gaussian noise term. Since the loss is calculated directly on the desired output, end-to-end training minimizes the propagation of error. However, it has a cost in terms of training complexity. Each training iteration requires sampling the model $|\mathcal{X} \times \mathcal{Y}|$ times, once per image pixel, and a Radon transform must be calculated for all $|\Theta \times \mathcal{R}|$ points in the sinogram. For a 256×256 image with 20 view angles (θ) and detector length of 256 (r), this adds up to $335,544,320$ operations. However, given the relatively small nature of INRs by deep learning standards, we did not find this computationally barring, with networks taking no more than a few minutes to train.

3.3.2 Uncertainty Quantification of INRs for CT Image Reconstruction

In the traditional NN setting, weights are treated and trained as point estimates. For uncertainty quantification, we convert the NN into a BNN, in which each weight has a distribution of values. Weight W_{ij} is assigned a Gaussian prior, $p(W_{ij}) = \mathcal{N}(\mu_{ij}, \sigma_{ij})$, and the weight posterior distribution $p(W_{ij}|h)$ and the posterior predictive distribution $p(\hat{f}|h)$ are given by (2.2.2.1) and (2.2.2.2), respectively. In practice, it is not feasible to reconstruct the entire posterior predictive distribution. Instead, we sample from the distribution N times to generate an approximation. The exact implementation of training and sampling depend on the uncertainty quantification method.

BBB approximates the posterior weight distribution via the variational inference approach described in Section 2.2.2. After training, each weight is characterized by the distribution $q(W_{ij}|\psi)$. To approximate the predictive posterior, a set of N weight matrices, $\{W^{(k)}\}_{k=1}^{N}$, is sampled from $q(W|\psi)$. For each matrix $W^{(j)}$, the network is treated as a point-wise INR of weights $W^{(j)}$ to synthesize a noisy image, $K^{(j)} = \{\hat{f}_{W^{(j)}}(x, y)\}_{\mathcal{X}\times\mathcal{Y}}$. The N reconstructed image samples are used for uncertainty quantification, producing average image, $\bar{K}$, with variance image, K^*, defined as

$$\bar{K} = \frac{1}{N}\sum_{j=1}^{N} K^{(j)}\ ; \quad K^* = \frac{1}{N}\sum_{j=1}^{N}\left(K^{(j)} - \bar{K}^{(j)}\right)^2, \tag{3.3.2.1}$$

in which all operations are performed pixel-wise. The average image $\bar{K}$ would be presented to a doctor and is used for peak signal-to-noise ratio (PSNR) calculation. Meanwhile, the variance image K^* could be used by a doctor to gain better understanding of uncertainty in the reconstructed image and is used for negative log-likelihood (NLL) calculation. The computations of the PSNR and NLL calculation are described in Appendix E.

For MCD, practical uncertainty quantification follows closely from the theory discussed in Section 2.2.2. A standard point-wise INR is trained. At each evaluation, N forward-passes are performed, in which all parameters are held constant, except for a subset of each layer's weights, which are set to zero according to the probability of dropout. The final network output of each forward-pass is saved as image sample $K^{(n)}$. As in (3.3.2.1), these samples are used to calculate the average and variance images. Implementing a DE over MCD baselearner architectures simply requires generating $\frac{N}{M}$ image samples per baselearner and aggregating all of them, using

$$\bar{K}_{\mathrm{DE}} = \frac{1}{N}\sum_{m=1}^{M}\sum_{j=1}^{N/M} K_m^{(j)}\ ; \quad K_{\mathrm{DE}}^* = \frac{1}{N}\sum_{m=1}^{M}\sum_{j=1}^{N/M}\left(K_m^{(j)} - \bar{K}_{\mathrm{DE}}^{(j)}\right)^2. \tag{3.3.2.2}$$

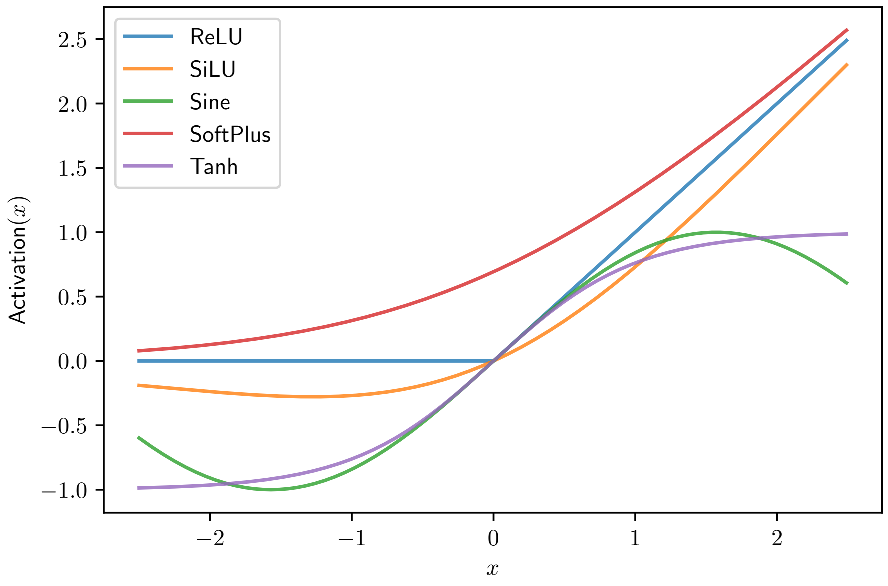

Figure 3.3: The five different activation functions – ReLU, SiLU, Sine, SoftPlus, and Tanh – used in our experiments.

Finally, in HMC, the problem of sampling parameters, $\{W^{(1)}, ..., W^{(N)}\}$, from the BNN weight posterior, $p(W|h)$, is reformulated in terms of physics-inspired dynamics. This is achieved by mapping the problem into an artificial Hamiltonian, with system potential energy governed by the parameter values and a system kinetic energy artificially designed to preserve overall energy, ensuring the algorithm samples points within the typical set. A detailed description of the HMC sampling algorithm is included in Appendix F. After the HMC algorithm is run, T weight parameter samples are generated, excluding the B prior burn-in samples. Rather than saving all T samples, we only save a subset, $N << T$. Saved samples are equally spaced among the T in an effort to minimize their correlation, in the case of poor HMC chain mixing. These N weight samples are then fed into the INR to generate image samples $\{K^n\}_{n=1}^{N}$, which are used to calculate average and variance images as described in (3.3.2.1). Note that in practice $N = 50$ was used for all uncertainty quanitification approaches.

3.4 Tuneable MLP Model Parameters

There are several degrees of freedom in designing an MLP, including its size, embeddings, activation functions, and optimizer. These design choices are critical in determining model performance, but there is little theoretical understanding of how to best select most of these model parameters. In this work we tuned over network width, depth, and activation function. Network sizes were kept fairly small, as is common in the INR literature. Activation functions considered include the **rectified linear unit (ReLU)**, **sigmoid-weighted linear unit (SiLU)**, **Sine**, **SoftPlus**, and **Tanh** functions, plotted in Figure 3.3. Given recent work indicating that **random Fourier features (RFFs)** assist network learning of higher-frequency image components [60], all networks were initialized with an RFF embedding of tuneable frequency Ω_0. Finally, all networks were optimized using the **adaptive moment estimation (Adam)** optimizer or **adaptive moment estimation with**

weight decay (Adam-W) optimizer (with an added weight decay hyperparameter). We provide detailed descriptions of these tuneable model parameters, as well as known insights for how they affect model performance, in Appendix D.

3.5 Model Performance and Uncertainty Metrics

Several metrics are commonly used to evaluate model performance and uncertainty calibration. Well known are **peak-signal-to-noise Ratio (PSNR)** and **negative log likelihood (NLL)**, for which we refer the reader to Appendix E. In this section we primarily focus on calibration and coverage.

3.5.1 Calibration and Coverage

Calibration is a metric that assesses a model's ability to predict the probabilities of its outcomes, gauging the reliability of the model's confidence in its predictions. For example, a model performing class predictions is considered calibrated if it assigns a class 50% probability and that class actually appears 50% of the time in prediction. For further information on calibration of class prediction models, we refer the reader to [78, 79]. Since this work focuses on regression models, the remaining discussion is centered on calibrated regression [80].

Let F be the cumulative distribution function (CDF) of model predictions $\hat{f}(x,y)$, that seek to approximate ground truth image $f \in \mathcal{F}$, where $\mathcal{F}$ denotes the functional space of possible images. Letting $\hat{f}_x \triangleq \hat{f}(x,y)$ We denote the corresponding quantile function as

$$F_x^{-1}(\tilde{p}) = \inf\{f_x : \tilde{p} \leq F(f_x)\}, \tag{3.5.1.1}$$

where F^{-1} performs the mapping $F^{-1} : [0,1] \to \mathcal{F}$ and $\tilde{p}$ is a confidence interval. For calibrated regression, ground truth pixel $f(x,y)$ should fall in a, say, 90% confidence interval 90% of the time. Thus, the regression model is calibrated for confidence interval $\tilde{p}$ if

$$\lim_{X\to\infty} \frac{1}{X} \sum_{x=1}^{T} \mathbb{I}\{f_x \leq F_x^{-1}(\tilde{p})\} \ = \ \tilde{p} \ , \ \forall \tilde{p} \in [0,1], \tag{3.5.1.2}$$

as the number of pixel samples approaches infinity, $X \to \infty$. If f_X denotes the ground truth value for i.i.d. random pixel $(x,y) \in X$, a sufficient condition for calibrated regression is

$$p\big(f_X \leq F_X^{-1}(\tilde{p})\big) = \tilde{p} \ , \ \forall \ \tilde{p} \in [0,1]. \tag{3.5.1.3}$$

Since practical dataset sizes are finite, preventing perfect calibration, different metrics have been developed to assess empirical model calibration.

Reliability diagrams serve as a visual representation of model calibration, plotting expected sample accuracy as a function of average model confidence. Ideally these would be continuous plots, but, in practice, samples are binned into M bins according to their prediction confidence. Let B_m be the set of indices, i, of samples with prediction confidence, $\hat{p}_i$ in the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$. The expected accuracy (acc) and confidence (conf) are approximations to the terms of (3.5.1.3), namely

$$p\big(f_X \leq F_X^{-1}(\tilde{p})\big) \approx \mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \mathbb{I}\{f_i \leq F_i^{-1}(\tilde{p})\} \tag{3.5.1.4}$$

$$\tilde{p} \approx \mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \hat{p}_i. \tag{3.5.1.5}$$

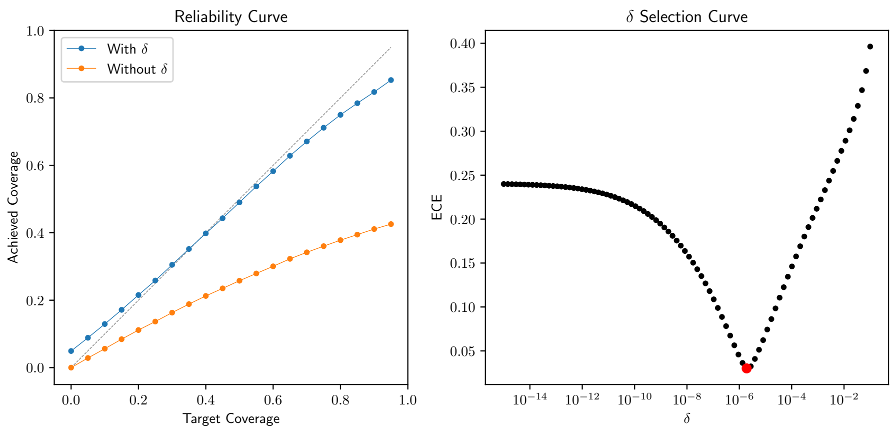

Figure 3.4: Both plots were made with an MCD model trained on 20 views, achieving a PSNR of 19. **Left)** A plot of the model reliability curves, with the grey dashed line indicating a perfectly calibrated model. The blue curve is the empirical reliability curve of the model when a small δ term is added symmetrically to the target coverage, in order to slightly widen the quantile ranges. Although this δ term has nearly negligible magnitude, it significantly improves the model reliability curve, as illustrated by the orange curve of reliability without the added δ term. **Right)** The added δ term was not chosen arbitrarily, but selected to minimize ECE.

The **calibration error (CE)** is the discrepancy

$$\mathrm{CE}(\tilde{p}) = \mid p\big(f_X \leq F_X^{-1}(\tilde{p})\big) - \tilde{p} \mid \approx \mid \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \mid = \mathrm{CE}(B_m). \tag{3.5.1.6}$$

It can be measured on a reliability diagram as the difference between the expected accuracy curve and the ideal $\mathrm{acc}(B_m) = \mathrm{conf}(B_m)$ line. The **expected calibration error (ECE)** quantifies the calibration error of the full distribution as

$$\mathrm{ECE}(x, f) = \frac{1}{M} \sum_{m=1}^{M} \mid \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \mid. \tag{3.5.1.7}$$

The model is considered calibrated if $\mathrm{ECE}(x, f) = 0$.

In practice, modifications were made to the previously described theory of reliability curves and ECE. You may notice in Figure 3.4 that, instead of plotting *accuracy* and *confidence*, we instead plot analogous *target coverage* and *achieved coverage*. Typically, a **coverage** value, $\bar{p}$, refers to a quantile of data points lying within $\pm\frac{\bar{p}}{2}\%$ of the median (50% quantile). Specifically, in our setup, we use different uncertainty quantification methods (BBB, MCD, HMC, and DE) to sample N different model weights for our INR, each set of weights corresponding to a different model output. Given that each output corresponds to an image, for each pixel, (x, y), we have a distribution of N predicted values, $K_N(x, y)$. Ideally, the median of the pixel distribution would be equivalent to the ground truth pixel value, $I(x, y)$. However, this is unrealistic to expect in practice. Thus, we check whether the ground truth pixel lies within the predicted pixel distribution quantile, Q_n, specified by coverage value, $\bar{p}$,

$$Q_{50-\frac{\bar{p}}{2}}\big(K_N(x, y)\big) \leq I(x, y) \leq Q_{50+\frac{\bar{p}}{2}}\big(K_N(x, y)\big). \tag{3.5.1.8}$$

If a model is perfectly calibrated, $\bar{p}\%$ of reconstructed pixels distributions will contain the ground truth pixel in their $\bar{p}\%$-th quantile, corresponding to the grey dashed line in Figure 3.4. Thus, model

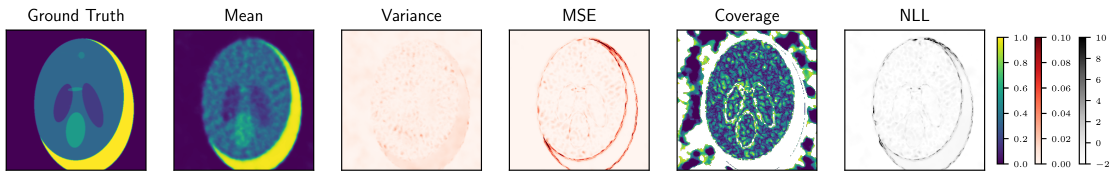

Figure 3.5: The ground truth image is plotted alongside different metrics for assessing the reconstructed predicted distribution generated by an HMC INR model. From left to right, we show the ground truth, predicted mean image, predicted variance image, mean squared error, coverage quantile of each pixel, and negative log-likelihood of each pixel. Note that PSNR is calculated using the ground truth and predicted mean image. In a real medical setting, the ground truth is unknown, the doctor would be given the predicted mean image and the predicted variance image could be provided as supplementary information to help the doctor reach a diagnosis. Further note that white regions in the coverage image denote that the ground truth pixel value did not fall in the range of the predicted distribution.

confidence can be seen as a pre-selected quantile for each pixel (target coverage), p, while accuracy is the percentage of pixel distributions containing the ground truth that quantile (achieved coverage),

$$\text{achieved coverage}(I, K_N, \bar{p}) = \frac{1}{|\mathcal{X}|}\frac{1}{|\mathcal{Y}|}\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\mathbb{I}\left\{Q_{50-\frac{\bar{p}}{2}}\big(K_N(x,y)\big) \le I(x,y) \le Q_{50+\frac{\bar{p}}{2}}\big(K_N(x,y)\big)\right\}. \tag{3.5.1.9}$$

The ECE is thus implemented as,

$$\text{ECE}(I, K_N) = \frac{1}{|\mathcal{P}|}\sum_{\bar{p}\in\mathcal{P}} |\text{achieved coverage}(I, K_N, \bar{p}) - \bar{p}| \cdot i, \tag{3.5.1.10}$$

where $\mathcal{P}$ is a finite set of percentages evenly spaced in $[0, 1]$, separated by percentage interval $i << 1$. The fifth image of Figure 3.5, plots the smallest quantile of each pixel containing the corresponding ground truth pixel, for an example HMC reconstruction with $N = 50$. Note that white regions indicate that the ground truth value does not fall within the minimum and maximum predicted pixel values.

There is one final modification made in implementing the reliability curves, in order to effectively assess model calibration. As described in Appendix C.1, the final layer of all the MLPs used for the INR have a sigmoid activation, ensuring that the model output is in the range $(0, 1)$. However, the sigmoid function only approaches 0 and 1 in the infinite limit, meaning that in practice our model will never output 0 or 1 exactly. However, our images contain a large percentage of pixels with exactly 0 value, especially for noiseless artificial data, which in the context of medical imaging is regions containing air and no tissue. This is problematic for calibration, since all of predicted pixel values will be near-zero, but will not actually contain the ground truth value of 0. This is illustrated by the orange reliability curve in Figure 3.4, for which only 40% of pixels contain the ground truth in their full range of predicted pixel values, for an MCD model trained on 20 views with $N = 50$. Our proposed solution to this issue is slightly widening the quantile range by adding a negligible δ term. Thus, for coverage value $\bar{p}$, we now check if the ground truth pixel lies in,

$$Q_{50-\frac{\bar{p}}{2}}\big(K_N(x,y)\big) - \delta \le I(x,y) \le Q_{50+\frac{\bar{p}}{2}}\big(K_N(x,y)\big) + \delta, \tag{3.5.1.11}$$

where $0 < \delta << 1$. In this case, if our predicted pixel values are slightly larger than 0, the δ offset can widen the quantile range to include 0, enabling these pixels to contribute to the achieved calibration (this also applies to pixels with exact value of 1). The improvement in using a delta offset is illustrated by the blue reliability curve in Figure 3.4, which is much closer to the ideal grey dashed line than the orange curve with δ. It should be noted that the value of δ is not assigned arbitrarily, but instead optimized to minimize overall ECE. For too small a δ, the quantiles will not be widened sufficiently to capture ground truth 0 pixels. However, for too large of δ, the quantiles will be widened too much, reducing overall calibration, as achieved coverage is much higher than target coverage for low coverage values. Thus, ECE as a function of δ is expected to have a unique minima, as illustrated by the example in Figure 3.4.

For some of the experimental discussion, it must be mentioned that before implementing ECE, a similar, but non-standard coverage metric was implemented. This metric, which we refer to as **coverage mean squared error (CMSE)** is mathematically defined as,

$$\text{CMSE}(I, K_N) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} (\text{achieved coverage}(I, K_N, p) - p)^2, \tag{3.5.1.12}$$

where $\mathcal{P}$ is a finite set of percentages evenly spaced in $[0, 1]$. It is analogous to ECE, but sums over the square of the coverage errors, rather than the areas. Thus, CMSE can be thought of as a proxy for ECE, but is not exactly the same, as the relative impact of larger calibration errors is reduced by the fact that it is squared. Note that the CMSE was also implemented without the δ correction term.

3.5.2 Assessing Model Quality

All of these metrics provide different means of assessing model quality. PSNR quantifies image reconstruction quality, coverage metrics such as ECE gauge the uncertainty calibration, and NLL encapsulates both. Ideally, both image reconstruction and calibration quality are optimized, meaning naively NLL would be the best metric. However, there is often a trade-off between calibration and prediction quality. In particular, classificaton networks can overfit to NLL without overfitting to 0/1 loss. Interestingly, this is typically beneficial to classification accuracy, at the expense of calibration. Thus, for deep neural networks, overfitting to NLL manifests in probabilistic error rather than reconstruction error [81].

Furthermore, while this work focuses on uncertainty quantification of INRs for medical imaging, little prior work has addressed this problem. Most existing techniques only quantify reconstruction PSNR. Hence, for fair comparison, we will assess and optimize our models primarily according to PSNR. However, in the case of similarly performing models, we use NLL, CMSE, and ECE as secondary selectors for the best model.

It should be noted that initial attempts at optimizing models according to coverage metrics resulted in preference for the lowest capacity models possible. This indicates that optimal performance according to coverage favors blurry image reconstruction, with as little certainty as possible in the final image. This is to be expected for brain images with sparse edges, where high-confidence, especially in an edge, results in a large calibration penalty.

3.6 MLP Model Selection Procedures

In this section, we describe the model selection process used for each of the four types of uncertainty quantification discussed in Section 2.2, for both the 5- and 20-view cases. Large hyperparameter sweeps are used to find the optimal parameters for BBB and MCD. The best performing MCD model is used as the base model for training DEs. Due to the high computational demands of HMC, we did not have the time to run hyperparameter sweeps for this approach. However, we developed metrics to assess HMC performance and convergence. Analyses of the trends and results of these model selection experiments are reported in Sections 4.2-4.3.

3.6.1 Hardware, Software, and Contribution Notes

The following experiments were computationally intensive, requiring hundreds of compute hours on parallelized **graphical processing units (GPUs)**. Specifically, experiments were run on the `ziz` GPU cluster of the University of Oxford Department of Statistics. This cluster consists of four GPU nodes with 8 GPUs each, consisting of a mixture of GTX 1080, GTX 1080Ti, and GeForce RTX 2080 Ti cards. The project codebase was developed in `Python`, using `Pytorch` [82] to implement the NN functionality and `Blitz` [83] for the the BNN functionality. `Hamiltorch` [84] was additionally used to implement HMC. PhD candidate Bobby He started the project using the SIREN [63] codebase. He added `Hydra` [85], `Weights and Biases (W&B)` [86], and `Simple Linux Utility Resource Management (SLURM)` [87] functionality, so that experiments could be conducted efficiently on the `ziz` cluster. He also implemented the base CT image reconstruction pipeline using INRs, metrics (NLL and PSNR), and uncertainty quantification procedures (MCD, BBB, DE, and HMC). All remaining work, such as further modifications and improvements to He's work, the hyperparameter sweep code, coverage metrics, integration of classical reconstruction approaches, and HMC convergence metrics were implemented by the thesis author, who also conducted all experiments, visualizations, and analysis of the results discussed in the thesis. Given that the project Github repo has not yet been made publicly available, key project code was submitted with the thesis.

3.6.2 BBB and MCD

For both BBB and MCD, MLP design choices had a large effect on INR reconstruction performance. Carefully designed hyperparameter sweeps were thus run to strategically search the MLP parameter space for four different settings: (1) MCD 5-view, (2) MCD 20-view, (3) BBB 5-view, and (4) BBB 20-view. This section presents a detailed discussion of the procedure used and choices made in reaching the final MCD 20-views model which, as described in Section 4.5, is the overall best performing final model across all four settings. Similar search pipelines and methodologies were used in the remaining three cases, but discussion is omitted for the sake of conciseness.

The model selection process began with a coarse grid search to efficiently prune across the wide range of possible parameter combinations. Specifically, we considerd model activation type, depth, width, RFF embedding frequency Ω_0, and dropout probability. Among these, activation type was the only categorical parameter, using the five activation types of Figure 3.3: Tanh, SoftPlus, Sine, SiLU, and ReLU. For the remaining parameters, this initial coarse grid search was used to get a sense of orders of magnitude, for the sake of computational feasibility. We

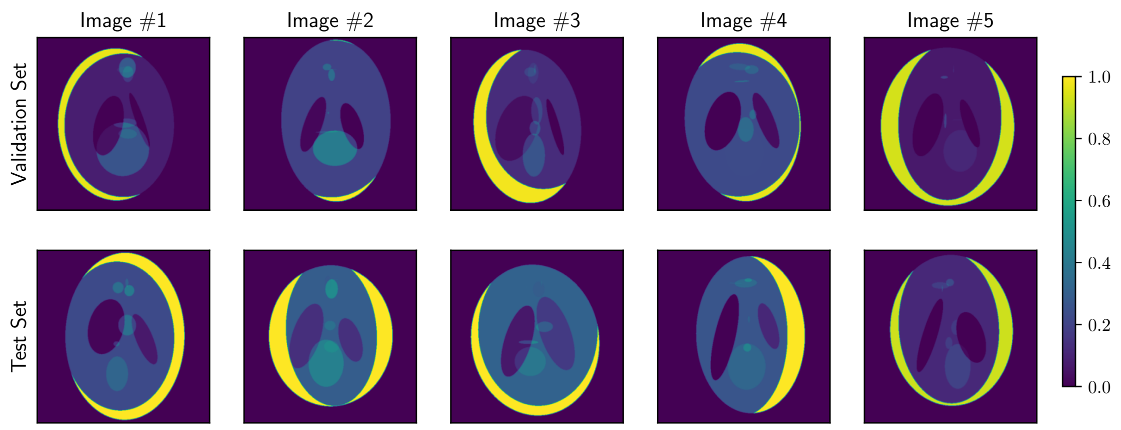

Figure 3.6: The ground truth images used in tuning and assessing our BBB, MCD, and DE models. The five validation set images were used to optimized model hyperparameters, while test set images were used to assess the finalized models.

swept over model depths of 3, 6, and 9; widths of 16, 64, 256, and 1024; and RFF Ω_0's of 1, 5, 10, and 15. Three values - 0.2, 0.5, and 0.8 - were considered for the final parameter, dropout probability, which is specific to MCD. For BBB, we instead swept over the Gaussian prior standard deviation (values 10, 100, and 1000)[1] and KL factor (values 1e-10, 1e-5, and 1e-1)[2]. For these coarse grid searches, all networks were trained using the Adam optimizer with no weight decay and the default learning rate of 3e-4. For each set of parameters, three individual INRs were trained, one for each of the three validation images, Image #1-#3, shown in Figure 3.6[3]. All reported metrics are averaged across the three test images, in an effort to ensure model generalization and prevent overfitting to a particular image. For both the 5- and 20-view experiments, 2,160 (2,512) models were trained and tested for the MCD (BBB) coarse grid sweeps. This resulted in a total of 9,344 models trained and tested during these initial grid searches. Since the performance metrics are calculated from a distribution of predictions, sampled according to the uncertainty method, all hyperparameter sweeps used 50 prediction samples to enable uncertainty quantification, mean output prediction, and metric calculation.

MCD 20-view coarse grid search sweep results are visualized in Figure 3.7, for the three different metrics of interest: PSNR, NLL, and CMSE[4]. These visualizations, generated by `W&B`, show that different MLP configurations perform best under each of the metrics. This is further demonstrated quantitatively by the results presented in Tables 3.1 and 3.2. Table 3.1 contains the relative importance of each model parameter in determining model performance for each metric. The importances were calculated by `W&B`, via the relative feature importance of a random forest trained with all the hyperparameter values as input and their corresponding metric value as output. Table 3.2 contains the correlations of each hyperparameter value with model performance,

[1]We originally swept over BBB standard deviation (values 0.2, 0.5, and 0.8), which are more on par with theoretical expectations. However, we found that increasing prior standard deviation significantly improved final model performance.

[2]Given the added BBB had uncertainty hyperparameter, we reduced relative number of search values for the remaining sweep parameters.

[3]Again, for the sake of computationally efficiency, only validation Images #1 and #2 were used for BBB.

[4]In retrospect, ECE would have been desired as a metric over CMSE. However, it was not being considered or implemented at the time these hyperparameter sweeps were run. Given the similarities between CMSE and ECE, described in Section 3.5.1, CMSE is treated as a proxy for ECE.

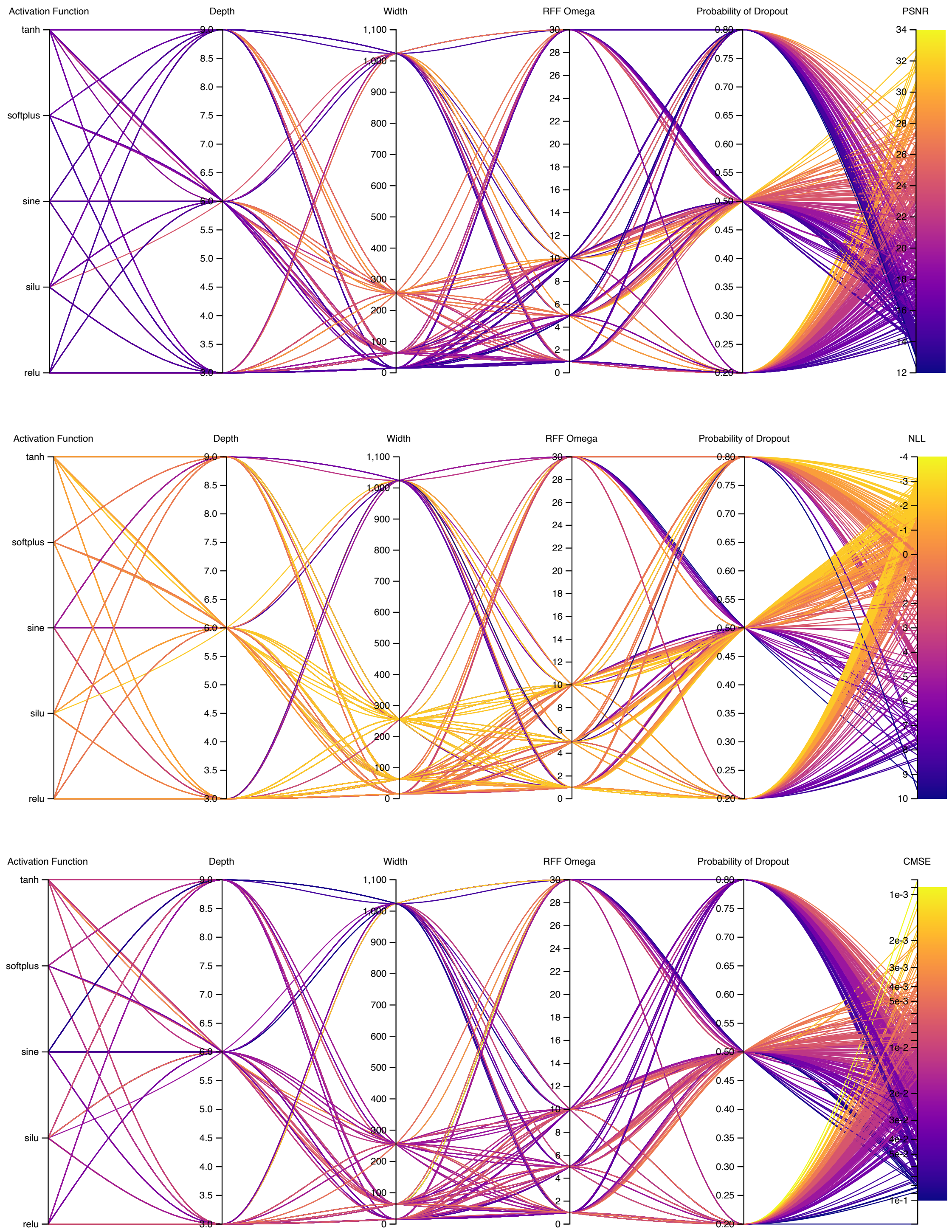

Figure 3.7: Visualization of the MCD 20-view coarse grid search hyperparameter sweep, according to three different metrics – PSNR, NLL, and CSME (averaged across three validation images). These are interactive plots generated by `W&B`, which enabled us to explore optimal hyperparameter configurations.

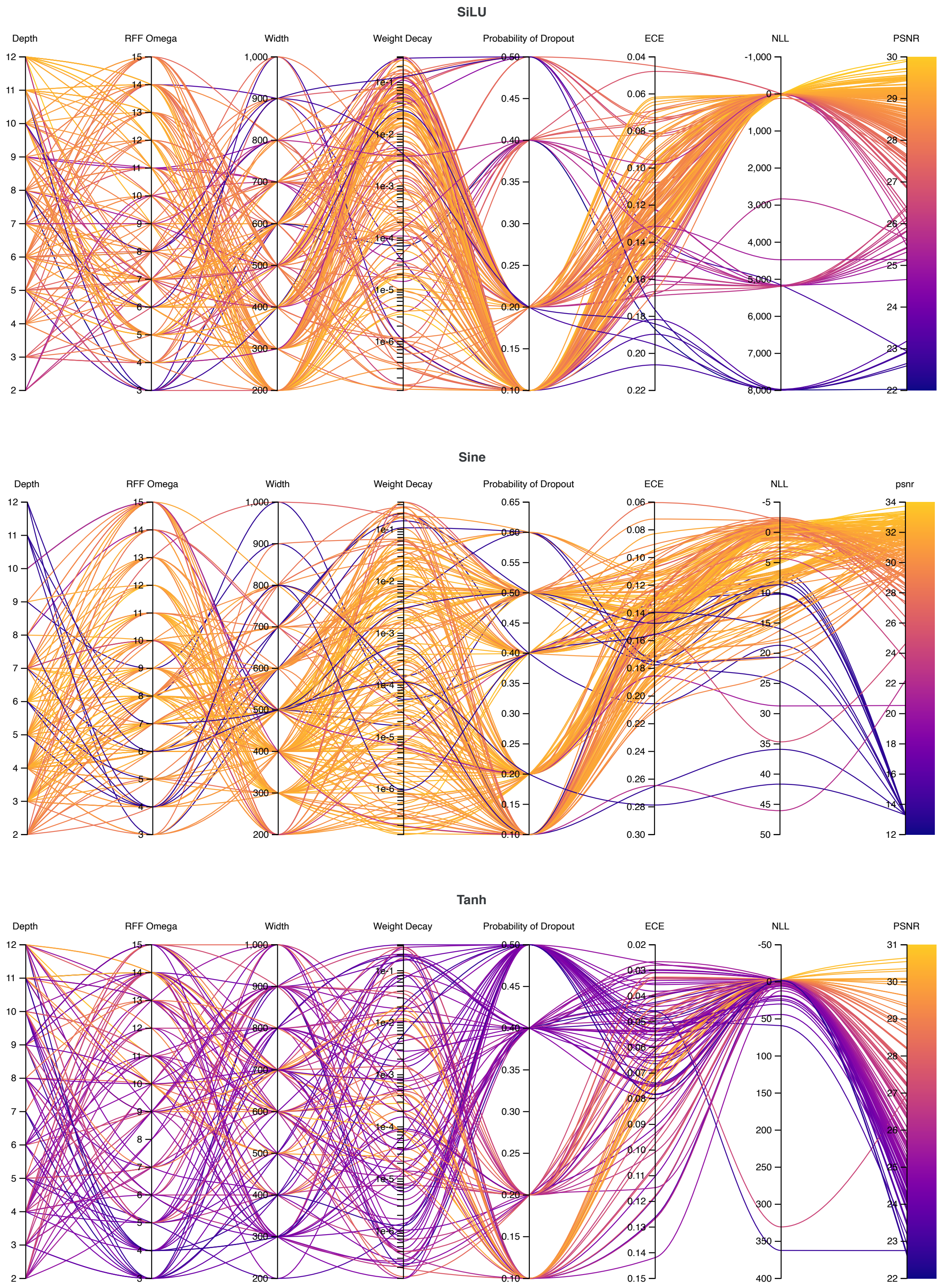

Figure 3.8: Visualization of the three MCD 20-view fine Bayesian search hyperparameter sweeps, for each activation function type – SiLU, Sine, and Tanh (averaged across five validation images). These are interactive plots generated by `W&B`, which enabled us to explore optimal hyperparameter configurations.

Metric	Activation Function					Depth	Width	RFF Ω	ℙ(Dropout)
	ReLU	SiLU	Sine	Softplus	Tanh				
PSNR	0.017	0.035	0.063	0.053	0.034	0.116	0.316	0.134	0.231
NLL	0.000	0.022	0.001	0.092	0.001	0.049	0.125	0.030	0.678
CMSE	0.025	0.083	0.088	0.054	0.031	0.139	0.232	0.135	0.213

Table 3.1: Importance, reported by `W&B`, of different model parameters in determining model performance according to each metric, for the 20-view MCD coarse grid hyperparameter sweep. Importance values were determined by feature importance in a random forest, trained with the hyperparameter values as input and the metric as the target output. Importance is a relative quantity, with larger importance indicating the hyperparameter has a larger effect on model performance, under the metric.

Metric	Activation Function					Depth	Width	RFF Ω	ℙ(Dropout)
	ReLU	SiLU	Sine	Softplus	Tanh				
PSNR	-0.055	0.206	-0.070	-0.249	0.155	-0.240	0.130	-0.022	-0.470
NLL	-0.052	0.034	-0.075	0.142	-0.045	0.100	0.228	-0.028	-0.084
CMSE	-0.047	-0.076	0.244	0.115	-0.235	0.218	0.305	0.077	0.028

Table 3.2: Correlations, reported by `W&B`, of different model parameters with each model performance metric, for the 20-view MCD coarse grid hyperparameter sweep. Since we aim to maximize PSNR, *positive* correlation indicates that increasing or using the parameter improves model performance. On the other hand, since we aim to minimize NLL and CMSE, *negative* correlation indicates that increasing or using the parameter improves model performance. Parameters that improve performance are highlight in green, while those that decrease performance are highlighted in red.

as calculated by `W&B`, according to each metric. Recall that the aim is to maximize PSNR while minimizing NLL and CMSE. A positive correlation means that the presence or increased magnitude of a hyperparameter contributes positively to PSNR, but negatively to NLL and CSME. For the latter, negative correlations are desired. Using the interactive visualizations and quantitative summaries, we gained insight as to how different parameters affected model performance in the four uncertainty-view settings. An in-depth analysis of these trends is presented in Section 4.2.

For now, we conclude our discussion of methodology, with the second hyperparameter sweep – a fine Bayesian search used to generate the final models. This Bayesian sweep leveraged a reduced search space, informed by the first coarse grid search. It was found that Bayesian sweeps do not perform well with categorical variables, so independent sweeps of ∼200 runs were performed for each of the three best performing activation functions from the grid search. Since only 600 models were trained in total in each uncertainty-view setting, all five validation images of Figure 3.6 were used as the validation set, to improve model generalization ability. Further, AdamW was used to optimize the models, with a weight decay hyperparameter added to the sweep. In the case of MCD with 20-views, three sweeps were performed, one for each of the three activation functions: Sine, SiLU, and Tanh. A log uniform distribution, in the range 1e-16 to 1e-1, was swept for the weight decay, while uniform distributions $U(\text{min}, \text{max}, q)$, where q is the discrete interval, were generated and swept over for the remaining numerical parameters: depth $\in U(2, 12, 1)$, width $\in U(200, 1000, 100)$, $\Omega_0 \in U(3, 15, 1)$, and $p(\text{dropout}) \in U(0.1, 0.6, 0.1)$. The results of these sweeps are visualized in Figure 3.8. The top performing model according to PSNR, across all three Bayesian sweeps, was selected as the final MCD 20-view model, as reported in Section 4.5.

3.6.3 Deep Ensembles

Given the robust and computationally intensive nature of DEs, we did not perform large-scale hyperparameter sweeps, as was the case for BBB and MCD. As described in Section 2.2.3, DEs combine the outputs of multiple base learner models to improve uncertainty calibration. Assuming each base learner makes a reasonable prediction, even if not optimized, adding more base learners should only maintain or improve uncertainty calibration. In order to create a DE of size M (DE-M), the top-M performing models identified by the MCD hyperparameter sweeps were used as base learners. If N total samples were desired for uncertainty quantification, each of the MCD baselearners was sampled repeatedly to generate $\frac{N}{M}$ predictions. These predictions were pooled together to create a sample of size N, from which uncertainty was quantified, the mean prediction generated, and model metrics calculated. In order to remain consistent with the BBB and MCD experiments, we used $N = 50$.

3.6.4 Hamiltonian Monte Carlo

HMC was implemented via the `Hamiltorch` python package [84], using a leapfrog integrator and No-U-Turn sampler [88]. There are two key challenges in optimizing HMC performance: (1) the large number of hyperparameters to be tuned and (2) the large computational demands of each HMC run[5]. This makes the previous large-scale hyperparameter sweep approaches unfeasible. However, given the physical intuition behind HMC, described in Section 3.3.2, there are some governing principles for hyperparameter selection. We used these intuitions to find hyperparameters that enabled HMC to sample from the desired posterior. We also developed metrics to verify HMC convergence, as discussed in Section 4.4. We next discuss the physical intuitions behind HMC.

The `Hamiltorch` HMC implementation contains several hyperparameters that govern HMC convergence[6]: burn-in length (B), number of iterations after burn-in (T), number of leapfrog integrator steps per iteration (L), number of samples to save (N), mass matrix (M), step size (Δt), and prior precision (τ). One of the most important parameters is the step-size, with smaller Δt giving rise to better Hamilton's equations solutions and, thus, better proposals. However, for constant L, too small a step-size reduces HMC to a random walk, failing to explore the full extent of the typical set (the main motivation for using HMC over other MCMC algorithms). Increasing L, however, results in longer computation times. Hence, there is a trade-off between proposal accuracy, amount of exploration, and computation time. To address this, the No-U-Turn sampler is used to adaptively adjust step-size during burn-in, optimizing according to the selected number of leap-frog steps L. The prior precision, τ, dictates the BNN parameter initializations. Given that NNs are typically initialized with values in the range $[-1, 1]$, $\tau \geq 1$ was used to ensure a Gaussian variance smaller than 1. Testing a few different τ values, we found $\tau = 100$ to work well generally. In total, $T' = 1500$ HMC iterations were performed per run, with $B = 500$ burn-in iterations. For consistency with the other approaches, 50 samples were saved from the eligible $T = 1000$ HMC iterations. The mass matrix plays an important role in defining the system's

[5]With our GPU setup, training a single BBB or MCD INR took on the order of a few minutes, as compared to hours per HMC run.

[6]Note that all these HMC convergence hyperparameters only affect uncertainty modeling (like probability of dropout for MCD and the KL factor as well as prior standard deviatiion of BBB). Beyond them, there still remain the network hyperparameters – width, depth, RFF Ω_0, and activation function – to be tuned.

kinetic energy and initializing the momentum values, P_W, at the beginning of each HMC iteration. Given the interplay between the position and momentum parameters in the Hamiltonian and leapfrog updates, its diagonal entries are typically initialized to have the same order of magnitude of the position parameter values. Since the weights are initialized to small quantities around 0, we simply set the mass matrix to have diagonal value 1. This was found to enable HMC convergence. Finally, a predefined set of network hyperparameters (width= 256, depth= 3, activation=ReLU, and RFF variance $\Omega_0 = 10$) was used for all HMC sweeps. This helped eliminate potentially confounding factors in the search for the hypermeters that led to best HMC convergence.

Although we did find hyperparameters that enabled HMC convergence, convergence was not guaraneteed for all runs. To assess convergence during an HMC run, we implemented the **potential scale reduction factor (PSRF)** metric [89], commonly known as $\hat{R}$. This metric has been used in prior work [73] to assess HMC performance in sampling from large-scale BNN posteriors (i.e. ImageNet network BNNs). The motivation behind $\hat{R}$ is that, rather than performing one long HMC run, several HMC **chains** are run simultaneously with different initializations. $\hat{R}$ estimates the ratio between the out-of-chain and the in-chain sample variance. Ideally, $\hat{R} = 1$, indicating that all chains are sampling across the distribution in an un-correlated fashion. In practice, however, different chains may fail to mix, causing the out-of-chain variance to be greater than the in-chain variance and $\hat{R} > 1$. In more detail, $\hat{R}$ is defined with respect to a scalar function, $\varphi(W^{(t)})$, of the HMC parameter samples $\{W_c^{(t)} \mid c \in \{1, ..., C\}, t \in \{0, ..., T\}\}$, where $W_c^{(t)}$ is the sample of the cth chain at the tth iteration after burn-in. Letting $\varphi_{ct} \triangleq \varphi(W_c^{(t)})$, $\hat{R}$ is defined as

$$\bar{\varphi}_{c\cdot} \triangleq \frac{1}{T}\sum_t \varphi_{ct}\ ;\quad \bar{\varphi}_{\cdot\cdot} \triangleq \frac{1}{CT}\sum_{c,t}\varphi_{ct} \tag{3.6.4.1}$$

$$\varrho \triangleq \frac{1}{C-1}\sum_c (\bar{\varphi}_{c\cdot} - \bar{\varphi}_{\cdot\cdot})^2\ ;\ \eta \triangleq \frac{1}{C(T-1)}\sum_{c,t}(\varphi_{ct} - \bar{\varphi}_{c\cdot})^2 \tag{3.6.4.2}$$

$$\zeta \triangleq \frac{T-1}{T}\eta + \varrho \tag{3.6.4.3}$$

$$\hat{R} \triangleq \frac{C+1}{C}\frac{\zeta}{\eta} - \frac{T-1}{CT}. \tag{3.6.4.4}$$

In our experiments, we calculated $\hat{R}$ for two scalar functions: $\varphi^{(1)}(W^{(t)}) = W^{(t)}$ and $\varphi^{(2)}(W^{(t)}) = \hat{f}(W^{(t)})$, where $\hat{f}$ is the function encoded by the INR. Calculating the $\hat{R}$ of $\varphi^{(1)}$ enables us to understand convergence in parameter space, while that of $\varphi^{(2)}$ enables us to understand convergence in output image space. Analysis of these two metrics is presented for each run in Section 4.4.

Given the physics-inspired nature of the HMC, another simple check of HMC performance is via the conservation of energy principle. Since HMC embeds the sampling problem into a Hamiltonian, with kinetic and potential energy terms (and no outside forces), the system energy should remain constant. Any large fluctuations in Hamiltonian likely indicate that the Metropolis-Hastings acceptance probability step did not filter a bad proposal and HMC may no longer sample from the typical set. Beyond this, the training loss of (3.3.1.1) is computed after each HMC iteration. If the HMC is converging, this is expected to decrease. However, if the sampler has left the typical set, training loss will likely increase. Finally, there are various guidelines for how high the acceptance rate should be, ranging from 30 to 60%. In general, however, too high of an acceptance rate indicates that the chain is not exploring enough, while too low of a rate generally indicates that the step-size is too large and HMC is frequently attempting to sample regions outside the typical set. Thus, in addition to $\hat{R}$, we also track Hamiltonian energy, proposed Hamiltonian energy, training loss, and acceptance rate, to monitor the convergence of individual HMC chains. It should also be noted that, due to the increased computation cost of HMC, we did not use any of the validation or test set images for HMC assessment. Instead, we used a singular Shepp-Logan phantom image, depicted as the ground truth image in Figure 3.5.

4 Experimental Results and Analysis

Contents

4.1 Classic CT Reconstruction 35
4.2 MCD & BBB Model Analysis 36
4.2.1 MCD Hyperparameter Analysis 36
4.2.2 BBB Hyperparameter Analysis 41
4.3 DE Performance Analysis 43
4.4 HMC Run Analysis 46
4.4.1 20-View Best Performing Single-Chain 46
4.4.2 20-View Best Performing Multi-Chain 47
4.4.3 5-View Best Performing Multi-Chain 50
4.5 Uncertainty Quantification with INRs 51

4.1 Classic CT Reconstruction

We begin our experimental analysis by evaluating the performance of the classical reconstruction algorithms described in Appendix B. For each method, the reported PSNR is the average reconstruction PSNR across the five ground truth test images of Figure 3.6. Figure 4.1 shows the PSNR as a function of view angle, for each of the five reconstruction methods. Also shown are the values below which PSNR is usually considered unacceptable (20dB), at which reconstruction is considered lossy (30dB), and above which has high quality (40dB). FBP has the worst performance, which is particularly poor in the low-view regime ($<$ 30 views). The iterative reconstruction algorithms perform better. CGLS has slow convergence to high quality image reconstruction. EM, SIRT, and SART all converge with far fewer views, achieving lossy compression with only $\sim$20 views. EM levels out and requires around 180 views to achieve high-quality reconstruction. SIRT and SART have near identical performance, passing the high-quality reconstruction threshold with only $\sim$75 views and SIRT performing slightly better for increased views.

Table 4.1 reports the PSNR values obtained for 5, 20, and 180 views. EM is the best performer for 5 views, SART for 20, and SIRT for 180. However, while the performance of the three algorithms

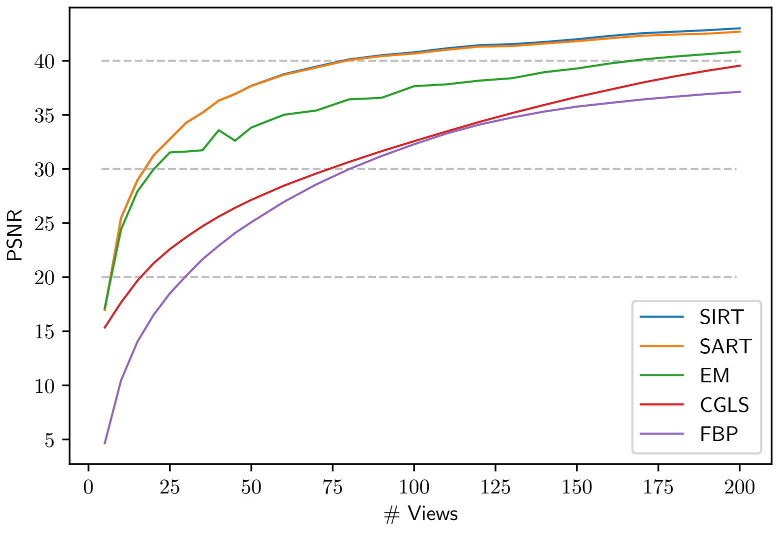

Figure 4.1: PSNR as a function of number of views for the classical reconstruction algorithms.

# Views	FBP	CGLS	EM	SART	SIRT
5	7.68	16.38	**21.39**	21.12	21.12
20	17.35	21.85	30.22	**31.98**	31.97
180	36.74	38.6	40.46	42.51	**42.76**

Table 4.1: Classical reconstruction PSNR, averaged across the five validation set images, in the 5-, 20-, and 180-view cases. The best achieved PSNR for each view-# is bolded.

is similar for 5, SART and SIRT perform better than EM for larger number of views. In the low-view regime, roughly an order of magnitude is required to achieve a 10dB improvement in average PSNR. Figure 4.2 shows a reconstructed image for each of these algorithm-view combinations. With 5 views the reconstruction algorithms are able to capture low-frequency object structure, but the image would not be useful for medical diagnosis. With 20 views it is clear that the algorithms are already capturing high-frequency components of the object, but the images have many artifacts. By 180 views, the reconstructed images are nearly identical to the ground truth images of Figure 3.6, with any discrepancies in PSNR due mostly to minor image reconstruction artifacts or imprecisions.

4.2 MCD & BBB Model Analysis

As discussed in Section 3.6.2, large hyperparameter sweeps were performed to tune the MCD and BBB models in both the 5- and 20-view cases. From these sweeps, we can gain valuable insights about the importance and robustness of different model parameters for training a CT image reconstruction INR with uncertainty quantification.

4.2.1 MCD Hyperparameter Analysis

For MCD hyperparameter sweeps, metrics were averaged across the INR model reconstruction of the first three validation set images of Figure 3.6. However, we also recorded the PSNR of the INRs trained on each individual image. Box plots for the individual distributions of PSNR for validation

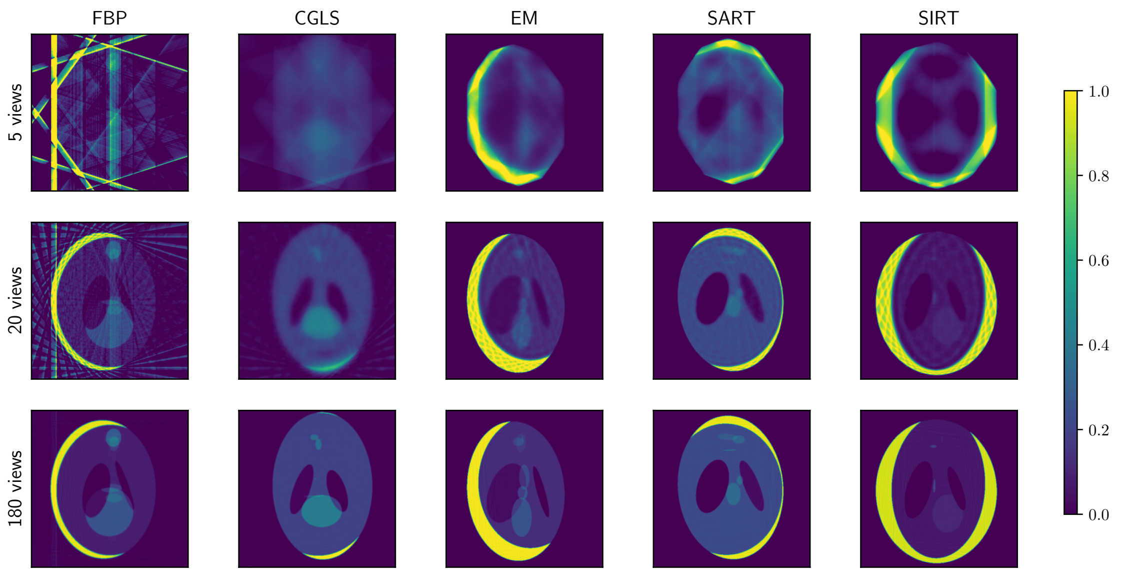

Figure 4.2: Reconstructions by the five classical algorithms with different number of views on the validation set images.

Images #1, #2, and #3 are shown in Figure 4.3. Ideally, PSNR should be consistent across the three images. This is roughly the case (barring a few outlier points) for models using the Tanh, SoftPlus, Sine, and Silu activation functions. Notice, however, that in all cases the distribution is broadest for Image #3. This effect is exacerbated for models using the ReLU activation function, for which PSNR of Image #3 ranges all the way from approximately 0dB to nearly the maximum achieved PSNR, in both the 5- and 20-view cases. This indicates that ReLU in particular, but all the all activation functions to some extent, struggle to capture features of Image #3.

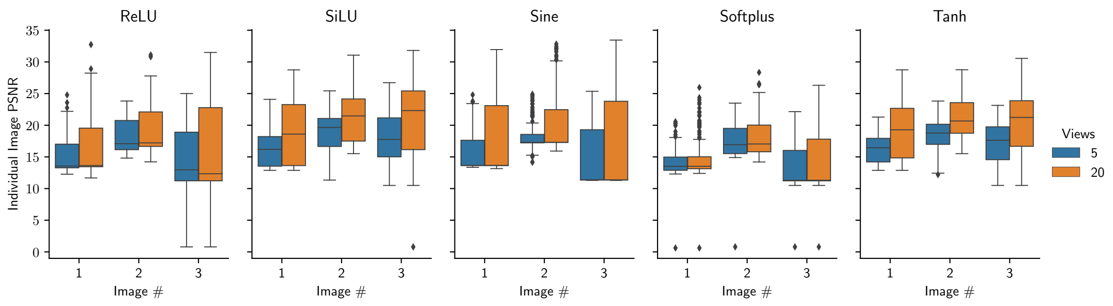

Figure 4.3: Reconstruction PSNR for Images #1-3 of Figure 4.3, for MCD INRs with different activations and different numbers of views.

To understand what is unique about Image #3, the image reconstructions of the best overall performing model are shown, for each activation function and 20-views, in Figure 4.4. It is apparent that Sine and SiLU produce smoother images, while Tanh, Softplus, and ReLU produce spottier images. Except for SoftPlus, all activation functions manage to capture low-frequency image information and strong edges. However, all activations struggle to capture the small, low-intensity ellipses in the center of Image #3. Overall, it is clear that, irrespective of activation function, 20-views are insufficient to robustly capture fine CT image details. This individual image analysis also suggests that the Sine activation performs the best for MCD, achieving the highest overall PSNR.

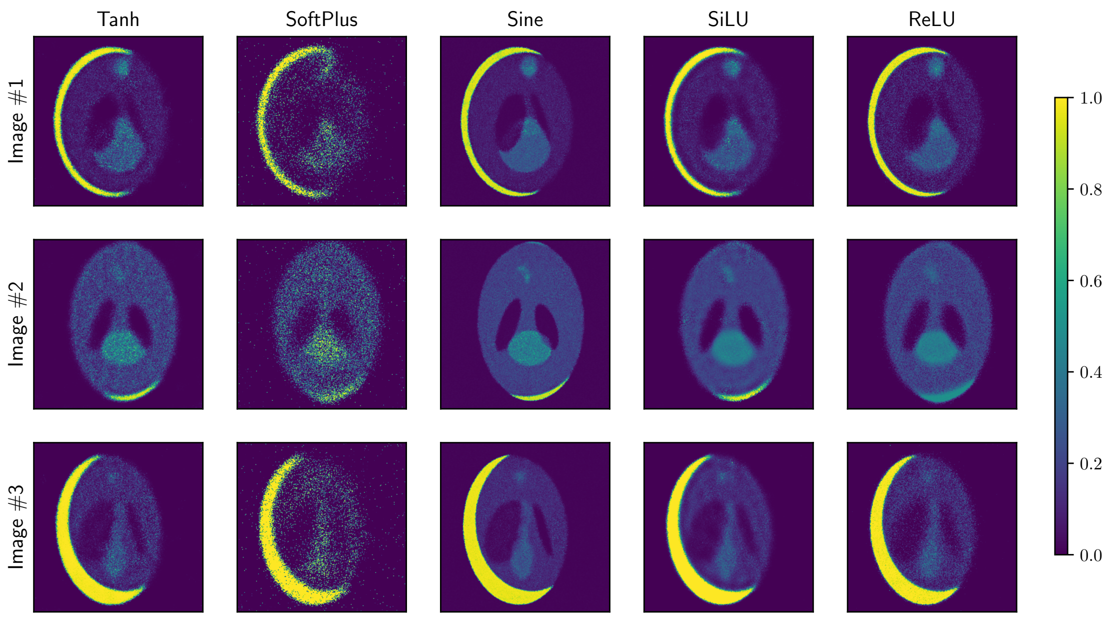

Figure 4.4: Best image reconstructions obtained with each activation function, for 20-view MCD.

Figure 4.5 shows boxplots of the average PSNR distributions for MCD models trained in the 5- and 20-view cases. Each column contains all the models trained with one of the five activation functions, while each row shows how the PSNR distribution changes as a function of hyperparameter – depth, width, probability of dropout, or RFF frequency Ω_0. Ideally, PSNR would be consistently large across hyperparameter values, indicating that the architecture is robust and does not require much tuning. In practice, however, we find that the activation functions are either consistent or high-performing, but not both. As previously mentioned, Sine achieves the best overall PSNR. However, it is also the least consistent activation function, with its highest performing models typically being outliers (indicated by diamonds in Figure 4.5). Softplus, on the other extreme, is very consistent across hyperparameter values, but performs consistently poorly. Tanh, Silu, and ReLU have less extreme variations. Their top models perform slightly worse than the best Sine models, but they perform much more consistently across hyperparameter values (ReLU is a bit inconsistent in width and probability of dropout). This suggests that the Sine network is potentially the best reconstruction network, but significant tuning (in terms of hyperparameter search) effort may be needed to achieve that solution. For practitioners inclined to perform less tuning, the Tanh and SiLU networks may be a preferred solution, due to their robustness and competitive top performance.

For ReLU, Tanh, SiLU, and Sine, it should also be noted that the PSNR distributions behave similarly in the 5- and 20-view cases (with 5 views performing consistently worse than 20 views, as expected) for all hyperparameters except RFF Ω_0. This is consistent with recent results [60] suggesting that RFF embeddings enable NNs to learn higher frequency image information. In the 5-view case, where the available data is insufficient for the INR to confidently learn high-frequency image features, performance is poor for increasing Ω_0. In the 20-view case, the increased data enables the network to learn higher frequency image components. However, because a higher-frequency Ω_0 is required to ensure the network can actually learn those frequencies, best performance tends to occur for larger Ω_0. This effect can also be observed in Figure 4.6. For 5 views, the reconstruction is blurry for $\Omega_0 = 1$ but the network starts to learn smaller ellipses for $\Omega_0 = 5$. By $\Omega_0 = 10$, however, the network is trying to learn higher-frequencies than the data cannot specify, resulting in artifacts, which are exacerbated for $\Omega_0 = 30$. In the 20-view case, a similar trend of reduced blurriness and increasingly sharper images can be seen between $\Omega_0 = 1$ an $\Omega_0 = 10$. However, the reconstructed images have much more recognizable details and less artifacts than those obtained with 5 views. It is only for $\Omega_0 = 30$ that artifacts start to appear.

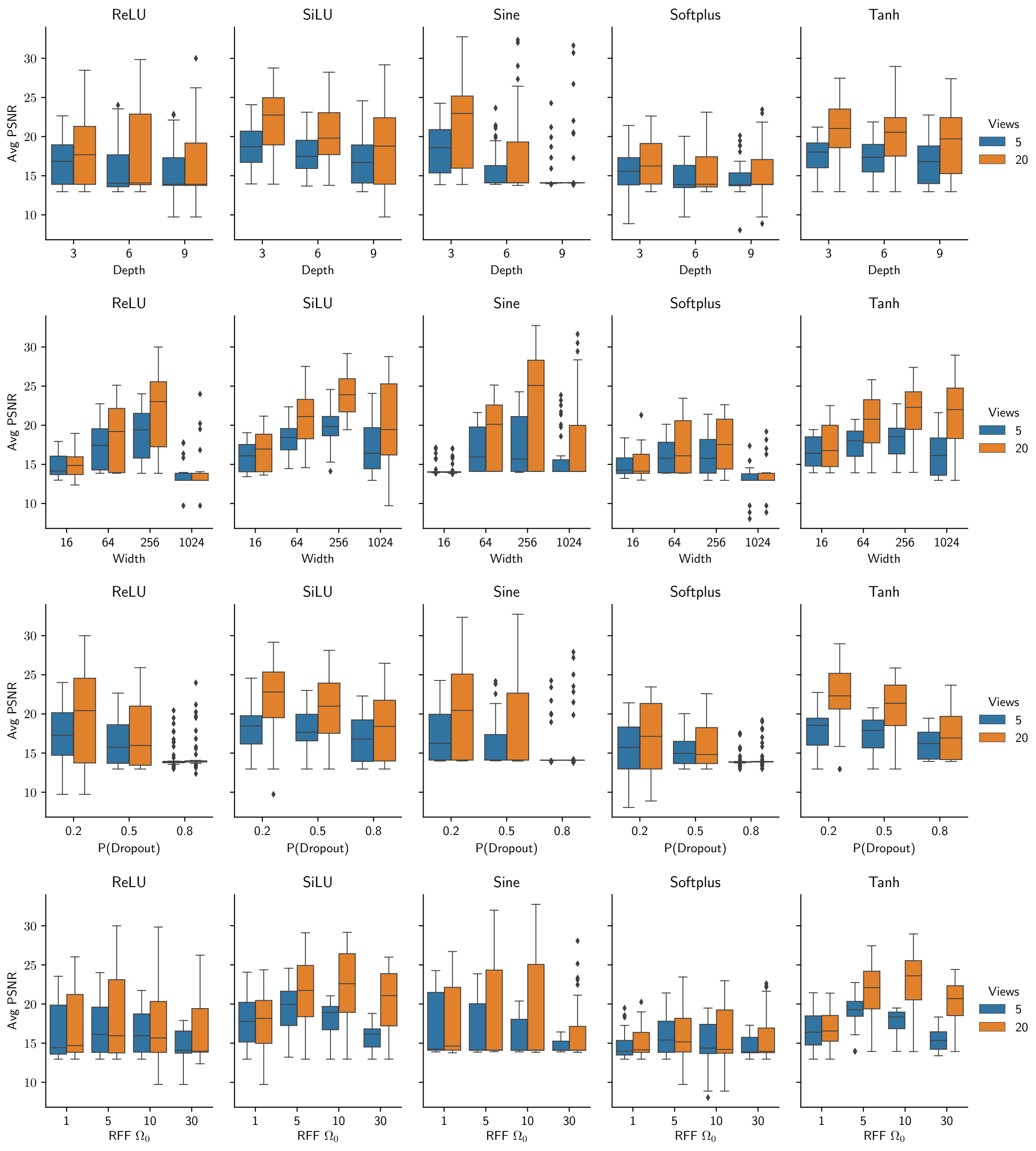

Figure 4.5: Boxplots of the average PSNR of MCD models trained in the coarse grid search hyperparameter sweep, for both 5 and 20 views. Each column corresponds to a different activation function and each row to a sweep over one of the remaining hyperparameters - depth, width, probability of dropout, and RFF frequency Ω_0. Individual diamond points are outliers.

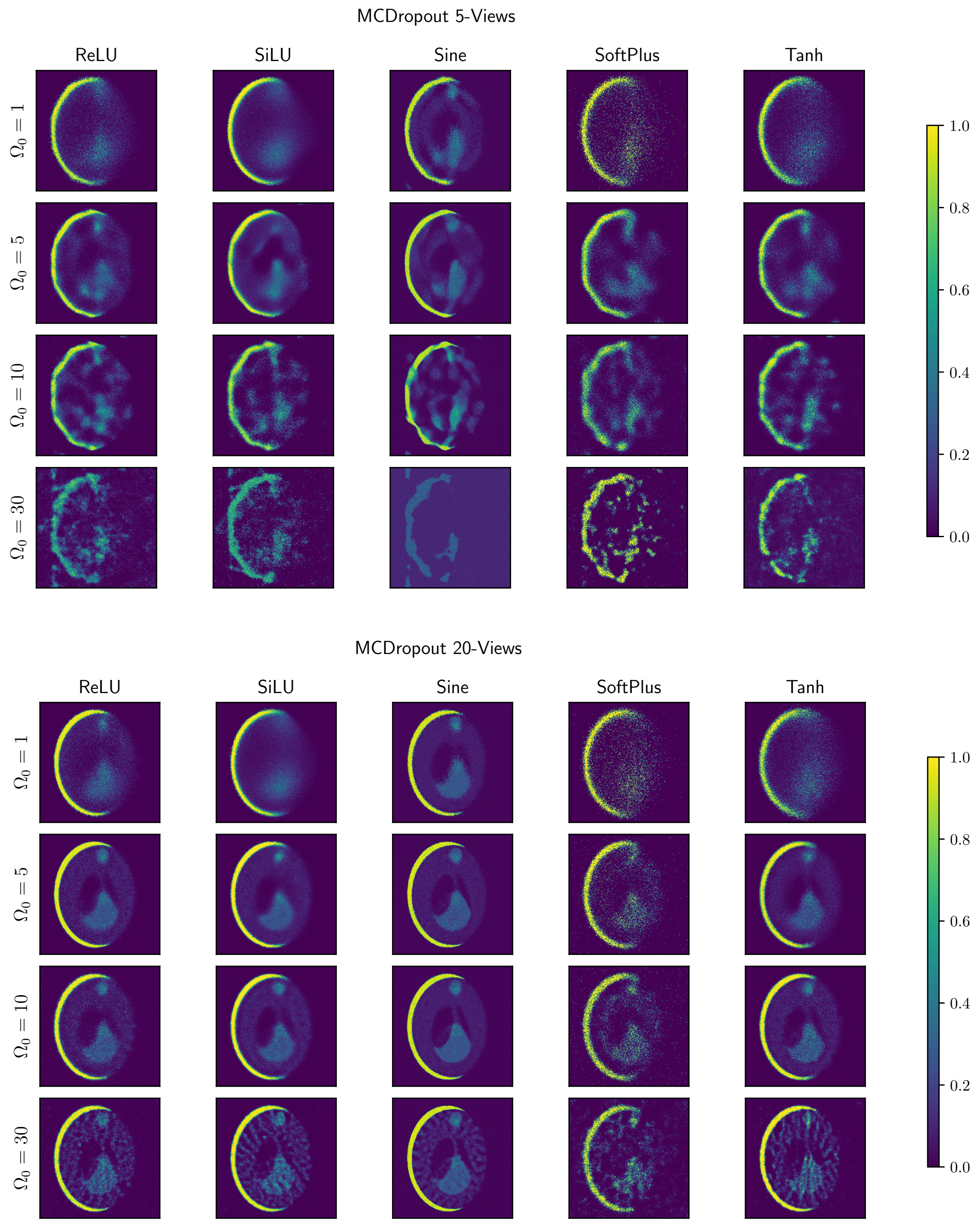

Figure 4.6: MCD image reconstruction, in both the 5- and 20-view cases, for each activation function and RFF frequency Ω_0 value. Note that in the 5-view case, $\Omega_0 = 5$ enables the network to learn low-frequency image features, without many artifacts. Smaller Ω_0 causes the network to produce overly simple output, whereas larger Ω_0 induces high-frequency artifacts. In the 20-view case, similar observations are made, but the optimal $\Omega_0 = 10$. In this case, the reconstruction has higher frequencies without significant artifacts, for most activation functions.

MCD Key Observations: MCD INRs struggle to capture low-intensity image details and edges, even for 20-views. In this regime, the Sine activation function outperforms Tanh, SoftPlus, SiLU, and ReLU according to PSNR, but is inconsistent across hyperparameter configurations, indicating it may be difficult to tune. Tanh and SiLU achieve competitive performance and are more robust with respect to hyperparameter tuning. Finally, as view-# increases, the optimal RFF freuqency Ω_0 also increases, suggesting that larger frequencies enable the network to learn higher frequency image features.

4.2.2 BBB Hyperparameter Analysis

The BBB model selection analysis is presented in a similar fashion to that of MCD. The main differences to MCD are that, in the BBB grid search, metrics are averaged over only the first two validation set images; the width and RFF search spaces are reduced; and the uncertainty parameters are the KL factor and Gaussian prior standard deviation (not probability of dropout). Figure 4.7 shows boxplots of average PSNR as a function of different hyperparameter values. Unlike MCD, model performance is extremely consistent across activation functions for every hyperparameter, except for width. We note that Tanh has some variation for RFF Ω_0, following trends similar to those discussed in the case of MCD. Overall, SiLU performs the best, with ReLU and Tanh following closely behind. Interestingly, Sine performs the worst, failing to reach even 20db. However, greater performance consistency across hyperparameter configurations comes at the price of weaker top-performing models, which achieve lower PSNR values than those of MCD. Because the image sets are not identical, these comparisons across approaches should be taken with some reservation.

To understand why there is so much variation in BBB performance as a function of width, consider Figure 4.8, where the average PSNR obtained for each width is plotted as a function of prior standard deviation. It can be seen that the PSNR values are extremely consistent for each network width, across prior standard deviations. From a Bayesian perspective, the fact that the prior does not affect inference suggests that the latter is dominated by the model likelihood. However, mean performance decreases as a function of model width, indicating that the model becomes increasingly misspecified for larger widths. Given the Gaussian assumptions made in the variational inference specifications of BBB, described in Section 2.2.2, this suggests that the true posterior distribution becomes less Gaussian as network width increases, and the variational approximation deteriorates. When combined with the low performance of BBB relative to other uncertainty quantification methods, reported in Table 4.4, this indicates that BBB may not be well suited for uncertainty quantification of INRs in the medical imaging context.

BBB Key Observations: BBB INRs perform extremely consistently across all activations for each hyperparameter, except width. The SiLU activation function performs best, with ReLU and Tanh close behind. Sine performs the worst. BBB appears insensitive to RFF Ω_0 for all activation functions except Tanh. For a given width, model performance as a function of prior standard deviation is extremely consistent, indicating that the likelihood term dominates the inference. Consistently decreasing performance with width then suggests that the Gaussian approximation of the variational posterior is less accurate for models of larger widths.

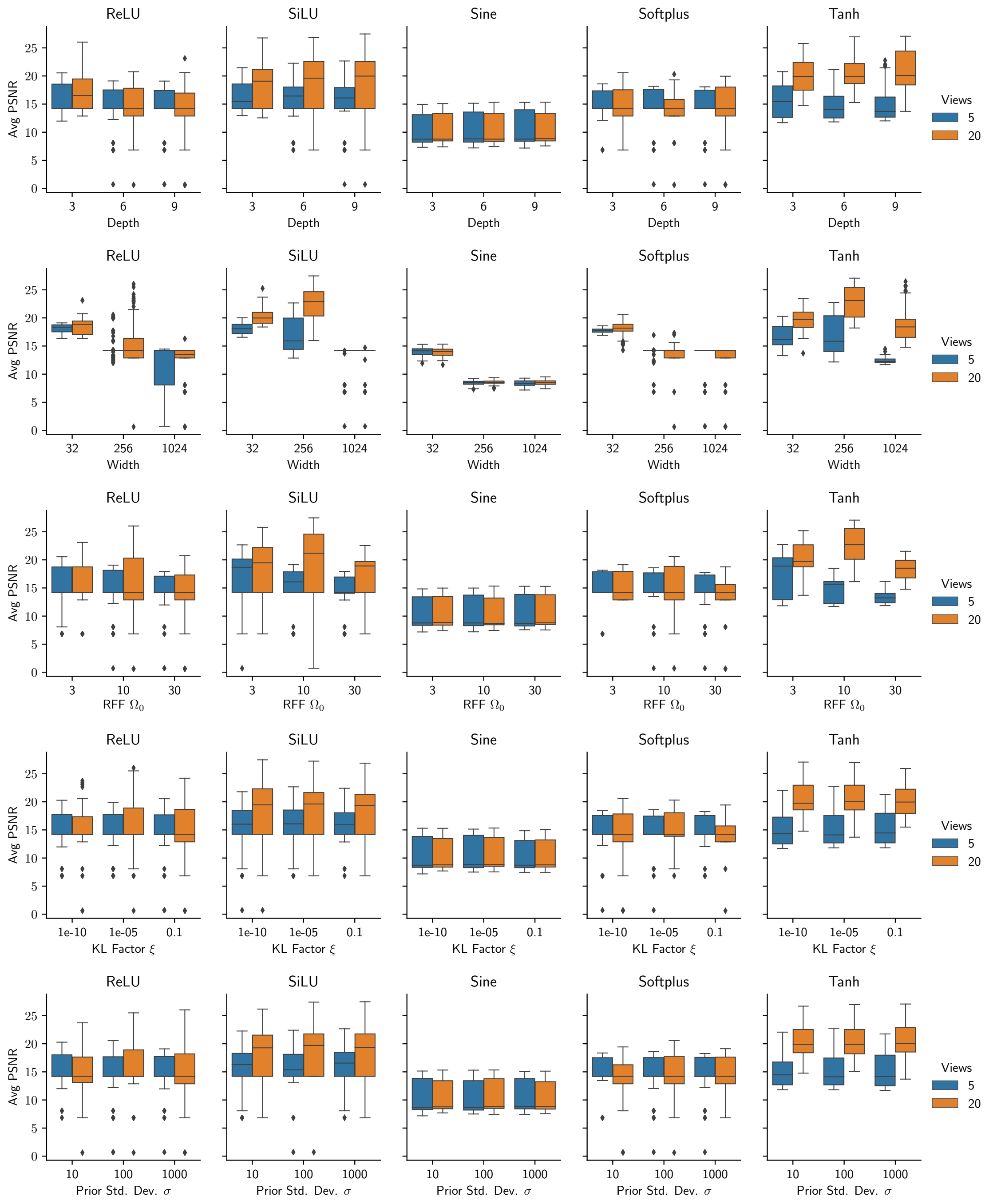

Figure 4.7: Boxplots of the average PSNR of BBB models trained in the coarse grid search hyperparameter sweep, for both 5 and 20 views. Each column corresponds to a different activation function and each row to a sweep over one of the remaining hyperparameters - depth, width, RFF Ω_0, KL factor ξ, and prior standard deviation σ.

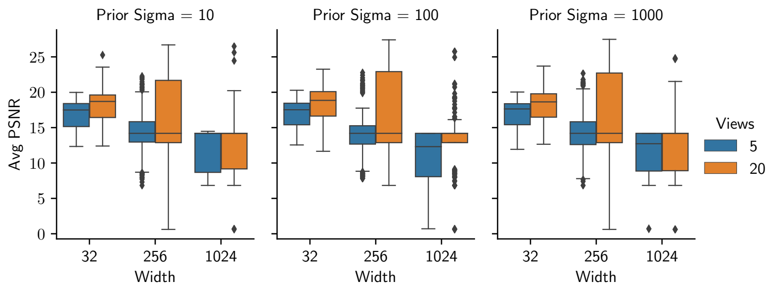

Figure 4.8: Boxplots of average PSNR of BBB models of varying width for each value of prior standard deviation σ, for both 5- and 20-view cases.

4.3 DE Performance Analysis

As described in Section 3.6.3, DEs of M base learners were created by combining the top-M performing NNs, according to average PSNR. As reported in Section 4.5, the best MCD model outperforms the best BBB model significantly, with at least a 2dB increase in PSNR, as well as reduced NLL and ECE, for both the 5- and 20-view cases. Thus, we ensembled the top MCD models produced by the second fine Bayesian hyperparameter sweeps of Section 3.6.2. The parameterizations and performance of the 10 best performing models for both the 5- and 20-view cases are listed in Tables 4.2 and 4.3, respectively. Note that, in both cases, the model architectures vary greatly across models. While the Sine activation function is fairly consistent, the remaining parameters vary greatly. For example, in the 5-view case, model depths range from 3 to 8, widths from 300 to 800, RFF Ω_0 from 2 to 8, and probability of dropout (PD) from 0.2 to 0.7, all with a variety of weight decays. These variations increase base learner diversity well beyond different weight values.

DE combines varied base learners to improve uncertainty calibration. In principle, if all base learners achieved the same PSNR, model performance should only increase (or plateau) and variance should only decrease (or plateau) as more base-learners are added. In our case, however, each new added base learner had a slightly lower PSNR on the validation set. To verify how this affected model performance and calibration, we generated plots of each metric as a function

Rank	Activation	Depth	Width	RFF Ω_0	PD	W. Decay	PSNR	NLL	ECE
1	Sine	4	800	2	0.4	0.001	26.15	-1.437	0.122
2	Sine	3	600	4	0.4	0.157	25.79	-1.799	0.008
3	Sine	3	600	8	0.7	0.366	25.76	-1.579	0.009
4	Sine	3	700	2	0.4	4.46e-4	25.75	-1.533	0.117
5	Sine	4	700	3	0.5	9.36e-4	25.72	-1.390	0.011
6	Sine	3	500	5	0.7	0.077	25.63	2.622	0.161
7	Sine	5	700	3	0.6	0.012	25.62	-1.149	0.123
8	Sine	3	500	5	0.7	0.068	25.62	-1.643	0.007
9	SiLU	8	300	3	0.2	2.68e-7	25.62	0.219	0.389
10	Sine	3	500	4	0.7	0.170	25.59	-1.79	0.083

Table 4.2: Top 10 performing MCD models and their performances, for the **5-view** case. Models are ranked by PSNR, but NLL and ECE are also reported.

Rank	Activation	Depth	Width	RFF Ω_0	PD	W. Decay	PSNR	NLL	ECE
1	Sine	3	400	9	0.4	2.06e-5	33.74	0.701	0.134
2	Sine	4	300	12	0.4	2.63e-7	33.41	1.076	0.149
3	Sine	3	500	8	0.5	0.006	33.37	-0.502	0.137
4	Sine	5	500	11	0.4	3.74e-6	33.29	-0.288	0.137
5	Sine	6	400	10	0.2	5.85e-6	33.28	4.654	0.152
6	Sine	6	500	8	0.2	1.117e-4	33.24	2.622	0.161
7	Sine	4	400	9	0.5	0.001	33.21	-0.798	0.143
8	Sine	3	500	10	0.5	2.53e-4	33.20	-0.716	0.129
9	Sine	5	500	11	0.2	7.87e-7	33.18	1.973	0.163
10	Sine	4	500	8	0.5	4.56e-5	33.17	-1.257	0.114

Table 4.3: Top 10 performing MCD models and their performances, for the **20-view** case. Models are ranked by PSNR, but NLL and ECE are also reported.

of # of DE base learners. As shown in Figure 4.9, both PSNR and MSE improve significantly as the first base learners are added to the ensemble, but begin to worsen for larger ensembles. Considering that models of worse PSNR are being added with each increase in # of base learners, it is unsurprising that performance eventually degrades. However, ensembling never reduces performance below that of using the single best model. NLL and ECE are more sensitive to the number of base learners. NLL demonstrates the overall best performance gain as a function of baselearners. ECE, however, does not change consistently with DE size, with large ensembles sometimes even harming performance relative to the single best model. In all, it is clear that ensembling improves model performance and calibration. However, larger ensembles have no gains over smaller ensembles and are far more computationally expensive. Thus, for the final results presented in Section 4.5, only ensembles of sizes 2, 5, and 10 are considered.

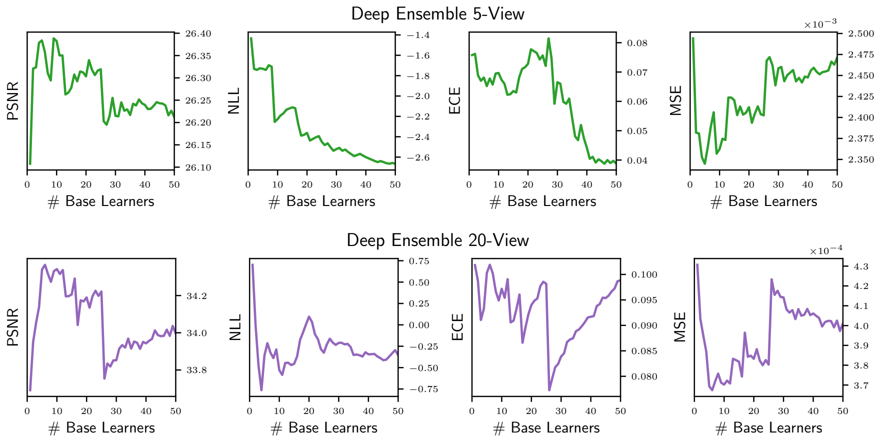

Figure 4.9: PSNR, NLL, ECE, and MSE plotted as a function of the number of base-learners used in DEs, for both the 5- and 20-view cases. Note that each added base learner performs slightly worse in terms of image reconstruction quality than the network preceding it.

Figures 4.10 and 4.11 illustrate image reconstruction performance for the different ensemble types on test set Image #3. In the 5-view case, DEs achieve impressive performance improvements, with a PSNR increase of over 1.5dB for DE-5 and an ECE reduction of 0.011 for DE-2. In the

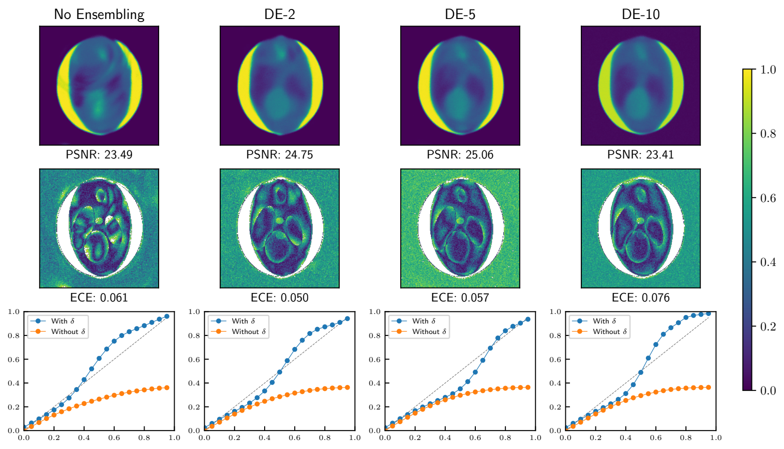

Figure 4.10: Image reconstruction and calibration performance of the final DE models for test set Image #3 and 5-views. Different columns show the results of different ensemble sizes, ranging from 1 to 10. Top row shows the reconstructed image, middle row the pixelwise coverage, and bottom row the reliability curve.

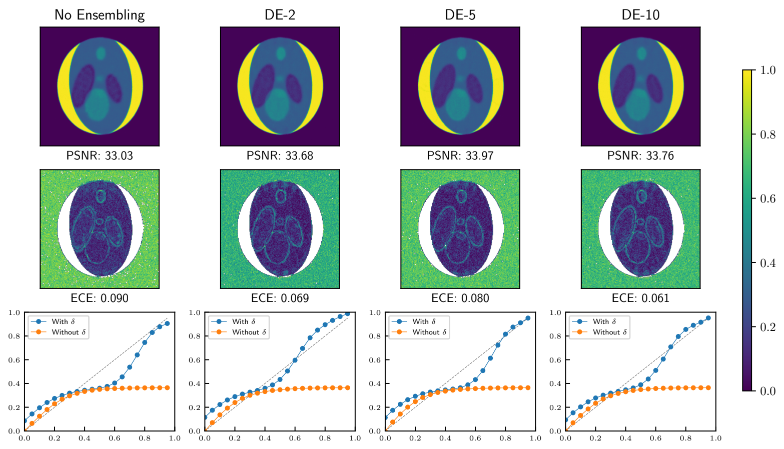

Figure 4.11: Image reconstruction and calibration performance of the final DE models for test set Image #3 and 20-views. Different columns show the results of different ensemble sizes, ranging from 1 to 10. Top row shows the reconstructed image, middle row the pixelwise coverage, and bottom row the reliability curve.

20-view case the baseline is much better performing. Hence, although there are gains in PSNR, these are not very noticeable in the reconstructed image. However, the improvements in calibration are larger, with an ECE drop of 0.2 for DE-2 and a clear improvement in the image reliability curve.

DE Key Observations: Ensembling MCD architectures can improve image reconstruction quality and model calibration. However, performance and calibration can decrease as weaker models are added to the ensembles. We found that smaller ensemble sizes of 2, 5, and 10 achieved the best trade-off between improving performance while remaining computationally feasible.

4.4 HMC Run Analysis

In this section, we utilize the HMC convergence metrics discussed in Section 3.6.4 to analyze the best performing HMC runs in both the 5- and 20-view cases. Note that all runs were performed using 3 independent chains to learn the ground truth image of Figure 3.5.

4.4.1 20-View Best Performing Single-Chain

We begin by assessing the HMC run of best overall performance, according to the PSNR of a single chain. Figure 4.13 shows the results produced by the each chain, as well as the combination of all three chains, for 20 views. It is clear that chains 0 and 1 did not properly sample from the BNN distribution, while chain 2 produced reasonable samples. These very different chain performances can be predicted by monitoring the HMC convergence and performance metrics tracked throughout the run, which are show in Figure 4.12.

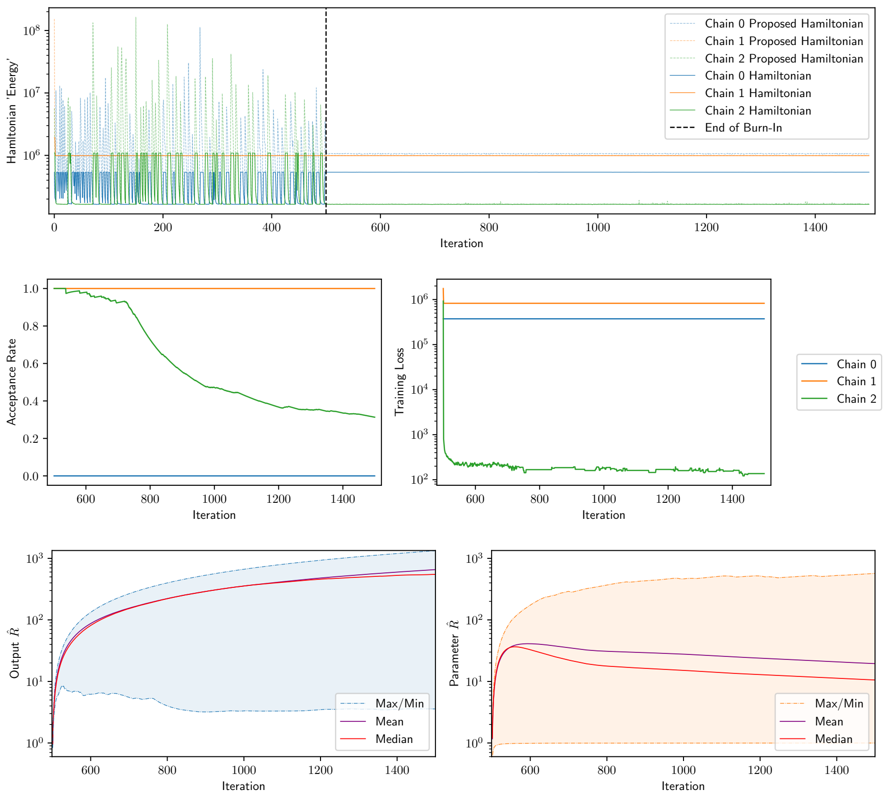

Figure 4.12: HMC convergence metrics for the top performing 20-view single-chain run. Top: Hamiltonian energy. Center: acceptance rate (left) and training loss (right). Bottom: $\hat{R}$ of both network output (left) and parameters (right). The different post burn-in values of Hamiltonian energy for the different chains, acceptance rates of 0 and 1 for chains 0 and 1, large training losses for these chains, and large $\hat{R}$ values for parameters and output, suggest that the HMC runs of chains 0 and 1 were unsuccessful.

The top figure shows the Hamiltonian and proposed Hamiltonian values for each chain, as a function of run iteration. As described in Section 3.6.4, we expect the Hamiltonian value to remain constant, by conservation of energy. The plots are split between the burn-in and post burn-in

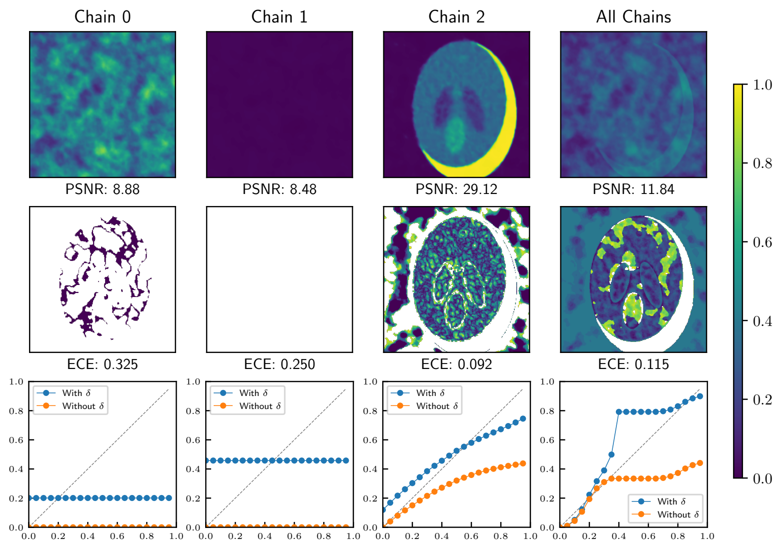

Figure 4.13: Results produced by three HMC chains, for the top performing 20-view single-chain run, as well as their combination. Top: reconstructed image. Middle: pixelwise coverage. Bottom: reliability curve. Chains 0 and 1 failed to produce samples from true BBN distribution.

periods. The large Hamiltonian oscillations during burn-in are caused by step-size modifications by NUTS. In principle, if all chains were sampling from the same posterior distribution, we would expect them to have similar overall Hamiltonian energy after burn-in, but vary in terms of relative potential and kinetic energy values across samples, as they are traversing the typical set. In this run, however, each chain has a distinct Hamiltonian energy after burn-in. The middle row of the figure shows the acceptance rate and training loss for each iteration. The plot of acceptance rates indicates that chains 0 and 1 are problematic, respectively rejecting and accepting every proposal. In the full-rejection case of chain 0, each sample produced by HMC is identical to the sample produced at the end of burn-in. This can be see in Figure 4.13, by the fact that the ultimate output of chain 0 is simply noise. Meanwhile, the acceptance of every proposal by chain 1 indicates that the leap-frog updates are not exploring the distribution space. Hence, it is unsurprising that, in Figure 4.13, image reconstructed by chain 1 consists entirely of zero values. The chain likely got trapped in a regime where weights only allow the network to produce zero values. The constant, large training loss values of chains 0 and 1 further emphasize these difficulties in sampling the true distribution. The third row of the figure shows the $\hat{R}$ of the output and network parameters as a function of training iteration. The fact that chains 0 and 1 did not properly sample from the distribution is reflected by the fact that both plots have large values, with mean and median values far larger than 1.

4.4.2 20-View Best Performing Multi-Chain

We next assess a model in which all three chains sampled well from the distribution, but none individually performed as well as the best single-chain. It should be noted that it was rare for all three chains to sample well during our HMC runs. In general, only one generated reasonable samples, if any. This can be attributed to our suboptimal HMC hyperparameters. Figure 4.14 shows that, unlike the previous example, all chains have similar Hamiltonian energies following

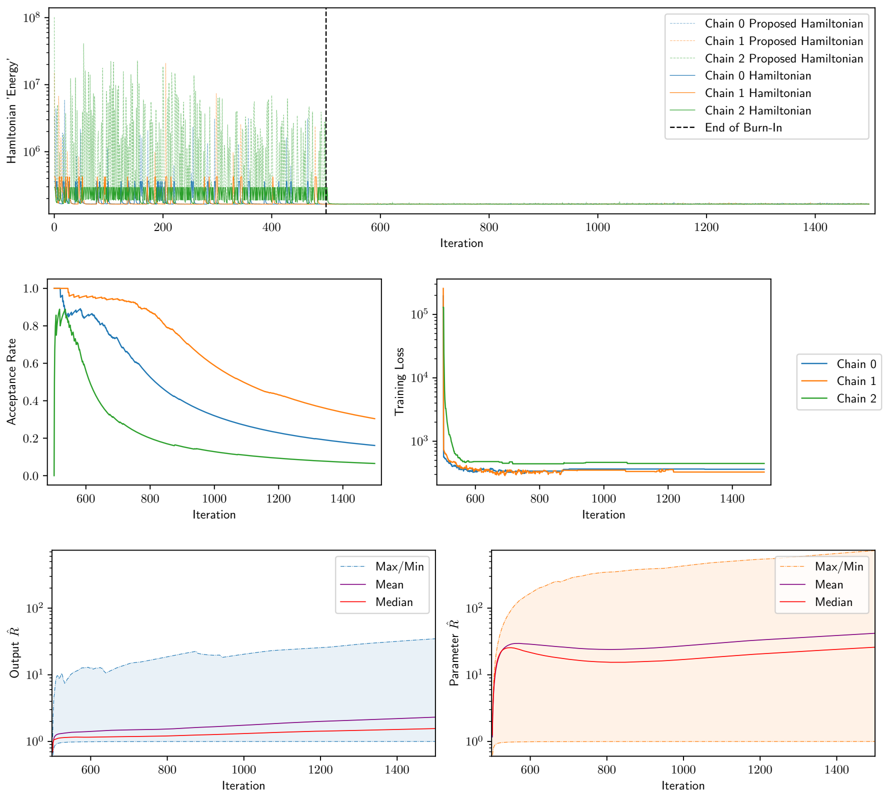

Figure 4.14: HMC convergence metrics for the best performing 20-view multi-chain run. Top: Hamiltonian energy. Center: acceptance rate (left) and training loss (right). Bottom: $\hat{R}$ of both network output (left) and parameters (right). The similar post burn-in values of Hamiltonian energy for the different chains, acceptance rates that gradually decrease from 1 to 0, significant training loss drops for all chains, and output $\hat{R}$ values centered at 1, suggest that all chains were successful.

burn-in, all three chains both accept and reject proposals, all three training losses drop significantly during the run, and the output $\hat{R}$ distribution is closely centered to 1. Although there is only one correct network output, several network parameterizations can achieve that same output, due to the large numbers of symmetries in neural networks. This explains why the parameter $\hat{R}$ distribution is not distributed about the ideal value of 1. This does not necessarily raise concerns in terms of image reconstruction performance.

The reconstructed image and calibration plots of all three HMC chains are presented in Figure 4.15. It is apparent that all three chains sampled from the true BNN weight distribution posterior. However, the reconstructed images seem to have high-frequency artifacts. This could be attributed, in part, to the un-tuned RFF Ω_0 model hyperparameter. Furthermore, when samples from all three chains are combined, the output image quality improves, with an approximately 1-2dB increase in PSNR over each chain, and a *significant* improvement in model calibration. Not only does ECE decrease by two orders of magnitude, but the reliability curve is significantly improved over that of each individual chain. To understand this, we analyze the plots of Figure 4.16.

The top plot of the figure presents counts of the value of $\hat{R}$ for the network output, at the end of the HMC run. While HMC seems to be sampling well for most outputs (corresponding to individual pixels in the reconstructed image), there are a non-negligible number of pixels with

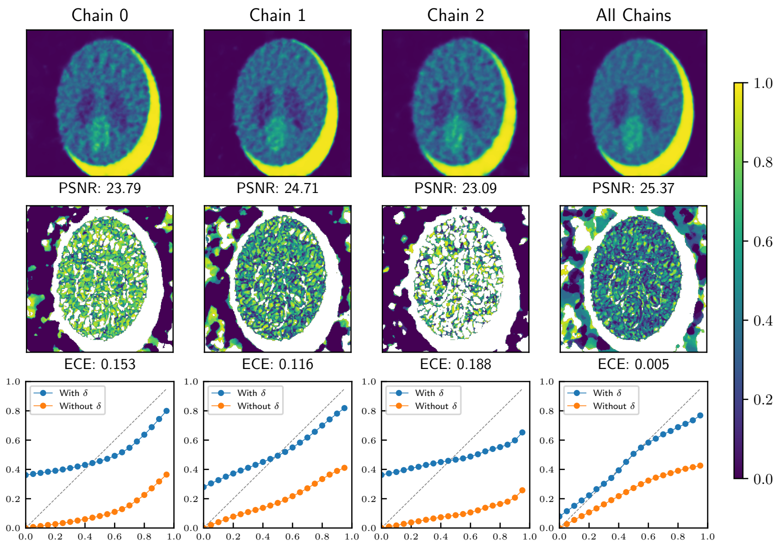

Figure 4.15: Results produced by three HMC chains, as well as their combination, for the top-performing multi-chain 20-view run. Top: reconstructed image. Middle: pixelwise coverage. Bottom: Reliability curve. All chains seem to produce samples from the true BBN distribution.

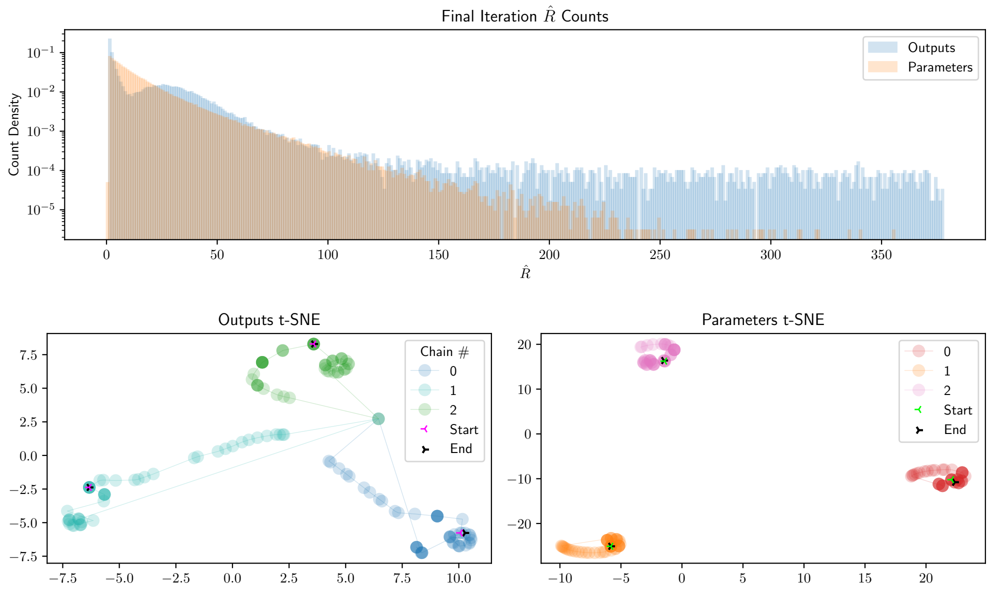

Figure 4.16: Top: Counts of $\hat{R}$ value in the final iteration of HMC for both output and parameters in the top performing 20-view multi-chain. Bottom: t-SNE visualizations of network output (left) and parameters (right).

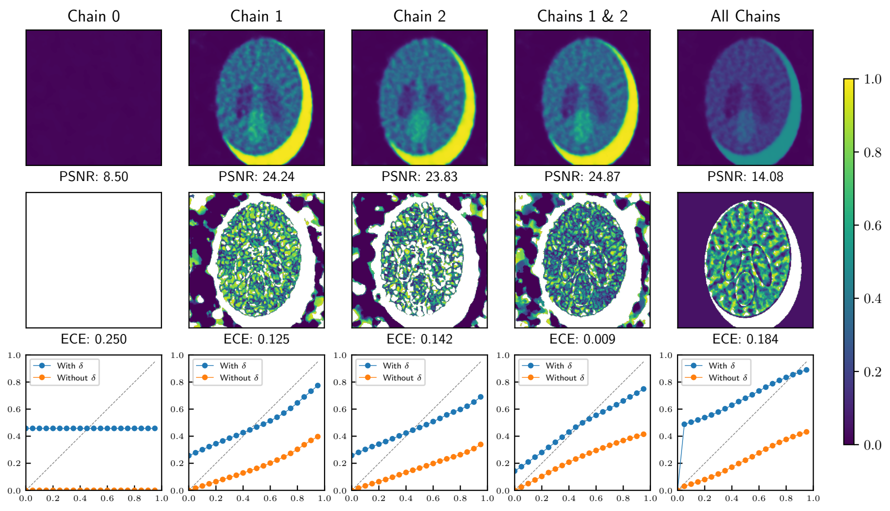

Figure 4.17: Results produced by three HMC chains, as well as their combination, for the top-performing 5-view multi-chain run. Top: reconstructed image. Middle: pixelwise coverage. Bottom: Reliability curve. Chain 0 failed to produce samples from true BBN distribution.

$\hat{R} > 1$. This suggests the chains are not fully mixing and are sampling from distinct regimes of the predictive posterior, $p(\hat{f}|h)$. This behaviour is further illustrated by the t-SNE[1] plot shown in the left of the bottom row of the figure. While individual chains appear to sample from a common component of the predictive posterior distribution at some point in the run, they also appear to spend most iterations sampling different components of the distribution. The t-SNE plot of $\hat{R}$ values observed for the network parameters suggests that the three chains sample networks with quite different weights. However, as mentioned above, this is not necessarily a problem, since these different networks could produce a similar output. Overall, it appears that, even though all chains managed to sample from the true BNN posterior and predictive posterior, the chains did not fully mix. The fact that combining samples from the different chains is equivalent to using samples from a single chain with better mixing, explains the improved PSNR and calibration when the chains are combined.

4.4.3 5-View Best Performing Multi-Chain

All previous observations hold for the 5-view best performing HMC run. In this case, two out of the three chains produced reasonable samples. The run results are shown in Figure 4.17, from which it is apparent that chains 1 and 2 sampled well, but this was not the case for chain 0. Combining the samples from chains 1 and 2 improved PSNR and calibration, analogously to the previous three chain example. However, incorporating samples from chain 0 as well significantly worsened performance relative to the best chains.

Since these results are similar to those discussed above, we omit further discussion for the sake of brevity. We only present the HMC run t-SNE plots in Figure 4.18, to note that while the parameter sample t-SNE manifolds are all quite distinct, the chain 1 and chain 2 output sample

[1]**t-distributed stochastic neighbor embedding (t-SNE)** [90, 91] is a non-linear dimensionality reduction technique well suited for visualizing high-dimensional data in a low-dimensional space. Although our network has hundreds of thousands of parameters and tens of thousands of outputs, t-SNE enables us to visualize the sampled output and parameter space distributions in 2D.

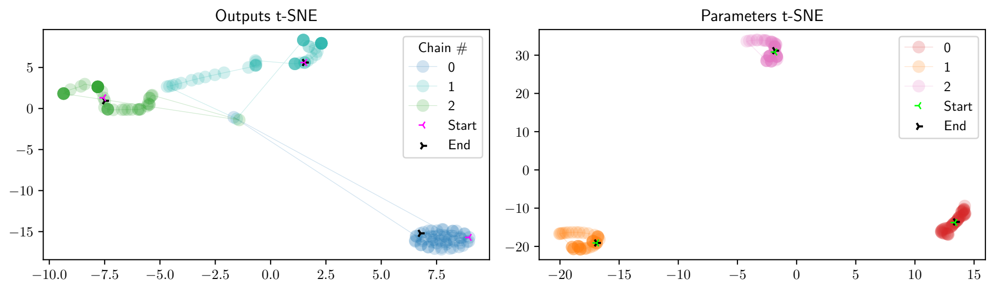

Figure 4.18: t-SNE visualizations of network output (left) and parameters (right) for the top-performing 5-view multi-chain run.

t-SNE manifolds are far closer to one another than the chain 0 manifold. This reiterates the fact that chains 1 and 2 are sampling from similar regions of the predictive posterior distribution. Further note than the samples within chains 1 and 2 are much more scattered than those within the chain 0 manifold. This indicates that chain 0 was trapped sampling in a local region of the predictive posterior, likely due to too small of a step-size. In this case, we would expect a very high acceptance rate for chain 0. Referring to the run acceptance metrics (not shown), chain 0 did in fact have a constant acceptance rate of 1, analogous to chain 1 in Figure 4.12. This all indicates that t-SNE plots of the output samples can provide insights on HMC performance.

HMC Key Observations: Hamiltonian energy, acceptance rate, training loss, output $\hat{R}$, and parameter $\hat{R}$ are all informative quantities to track during a multi-chain BNN HMC run, in order to gauge relative chain performance, chain mixing, and overall convergence. t-SNE visualizations of the output and parameter samples can be used to obtain additional insights. If all chains converge, the final output $\hat{R}$ distribution should be tightly distributed around 1, while final parameter $\hat{R}$ will likely be broadly distributed. This can be attributed to the wide range of symmetries that exist in NNs, which enable vastly different weight parameterizations to produce the same model output. Finally, combining samples from multiple chains that produce reasonable samples can further boost reconstructed image quality and significantly improve calibration, by compensating for lack of mixing within individual chains.

4.5 Uncertainty Quantification with INRs

The primary goal of this work was to develop a CT image reconstruction technique which not only achieves good reconstruction quality, but also generates calibrated uncertainty estimates over the reconstructed image. Several approaches were proposed to achieve this goal. Table 4.4 presents a comparison of their results.

Classical reconstruction results are incorporated as a baseline of comparison for PSNR, but do not report uncertainty[2]. This work primarily focused on using INRs to learn the image reconstruction mapping. Four different uncertainty quantification procedures – HMC, BBB, MCD,

[2]Note that a Bayesian formulation of the iterative reconstruction techniques can be used to generate uncertainty values via HMC sampling on different prior distributions. This, however, is not standard in the field and was not readily available in any existing open-source medical image reconstruction packages. Thus, we leave such a comparison as future work.

and DE – were implemented and assessed. As shown in the table, these methods consistently outperformed the classical reconstruction techniques in terms of image quality, while producing reasonably well calibrated uncertainty estimates. BBB was found to be the worst performing uncertainty quantification approach, generally producing the poorest calibrated uncertainty estimates and worse image reconstruction than classical techniques in the 20-view regime. MCD consistently outperformed classical approaches and was generally better calibrated than BBB. Ensembling over MCD base learners, however, was the most successful approach, outperforming the best classical approach by $\sim$4dB in the 5-view case and $\sim$3dB in the 20-view case, as well as achieving the lowest overall NLL and ECE values. Finally, despite computational challenges in optimizing HMC, even the suboptimal HMC configurations performed quite well. In the 5-view regime, HMC achieved a PSNR nearly equivalent to the average PSNR of the best DE. Although relative performance was worse in the 20-view case, HMC still outperformed BBB.

One further remark regarding to Table 4.4 is that although the classical reconstruction procedures did not use the validation set images as a validation set (since no hyperparameters were tuned), image reconstruction quality still deteriorated in the test set. This indicates that the test set data is actually more challenging than the validation set. Given that the performance of BBB, MCD, and DEs did not significantly decline on the test set, we have strong reason to believe that none of these approaches were over-fitted to the validation set.

Figures 4.19 and 4.20, visually illustrate the difference in the uncertainty quantification of the different methods, for the 5- and 20-view cases respectively. Beginning with the 5-view case, the mean predicted image was fairly blurry for all methods other than HMC, only capturing low-frequency image components and exhibiting high uncertainty surrounding edges. HMC, however, managed to capture image edges and high-frequency components very well, especially given the limited data available. While some high-frequency artifacts are present, we attribute these to the lack of optimization of the RFF Ω_0 parameter and believe that further HMC tuning could improve performance. This proves to be a promising next step for this work. Finally, note that all reliability curves are very well calibrated in this 5-view case (after a δ-selection adjustment), except for BBB. Similar trends are present in the 20-view case, but it is visually harder to distinguish differences in the image output due to the higher quality of all reconstructions. In this case, the mean reconstruction by HMC appears noisier than those of most other approaches.

# Views	Recon Type	Validation Set			Test Set		
		PSNR	NLL	ECE	PSNR	NLL	ECE
5	FBP	7.68	–	–	5.15	–	–
	CGLS	16.38	–	–	14.62	–	–
	EM	21.39	–	–	19.88	–	–
	SART	21.12	–	–	19.75	–	–
	SIRT	21.12	–	–	21.12	–	–
	HMC*	–	–	–	24.87*	-1.616*	0.090*
	BBB	23.26	-1.190	0.152	22.52	0.138	0.203
	MCD	26.15	-1.473	0.111	24.45	-1.572	0.083
	DE-2	26.31	-1.730	0.091	24.49	-1.774	0.069
	DE-5	**26.44**	-1.737	0.085	**24.88**	-1.751	0.067
	DE-10	26.36	**-2.226**	**0.075**	24.67	**-1.969**	**0.068**
20	FBP	17.35	–	–	15.71	–	–
	CGLS	21.85	–	–	20.82	–	–
	EM	30.22	–	–	29.11	–	–
	SIRT	31.98	–	–	30.44	–	–
	SART	31.97	–	–	30.45	–	–
	HMC*	–	–	–	29.12*	-1.676*	0.009*
	BBB	28.25	1.650	0.121	28.16	0.562	0.119
	MCD	33.74	0.701	0.135	33.08	1.093	0.113
	DE-2	33.96	0.005	0.136	33.44	-0.372	0.102
	DE-5	34.31	-0.364	0.134	**34.02**	-0.625	0.101
	DE-10	**34.38**	**-0.529**	**0.131**	33.86	**-0.774**	**0.096**

Table 4.4: INR accuracy and calibration results in both the 5- and 20-view cases are presented for all four uncertainty quantification approaches. Classical approaches are separated by a dashed line and do not uncertainty quantification. For BBB, MCD, and DEs results are presented as an average over all five validation images and all five test set images, depicted in Figure 3.6. Due to computational limitations, no validation set was used to tune optimal hyperparameters for HMC. A star is placed next to all HMC results to further demarcate that HMC was trained on a singular image, outside the validation and test sets. The best result for each metric – PSNR, NLL, and ECE – is bolded in each subcategory.

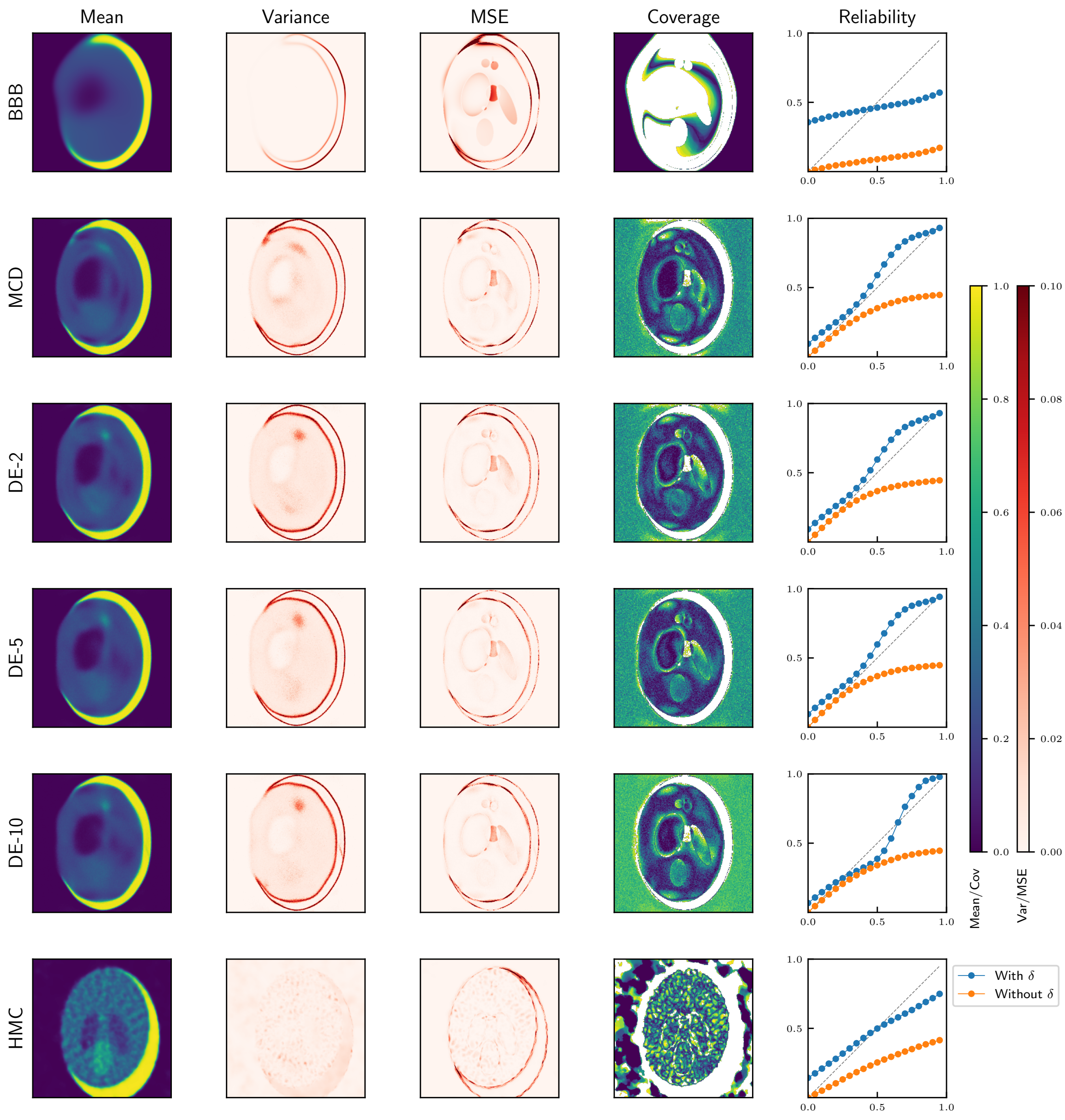

Figure 4.19: Validation results of all approaches in the 5-view case. From left to right: mean image reconstruction, variance, MSE, coverage, and reliability diagram.

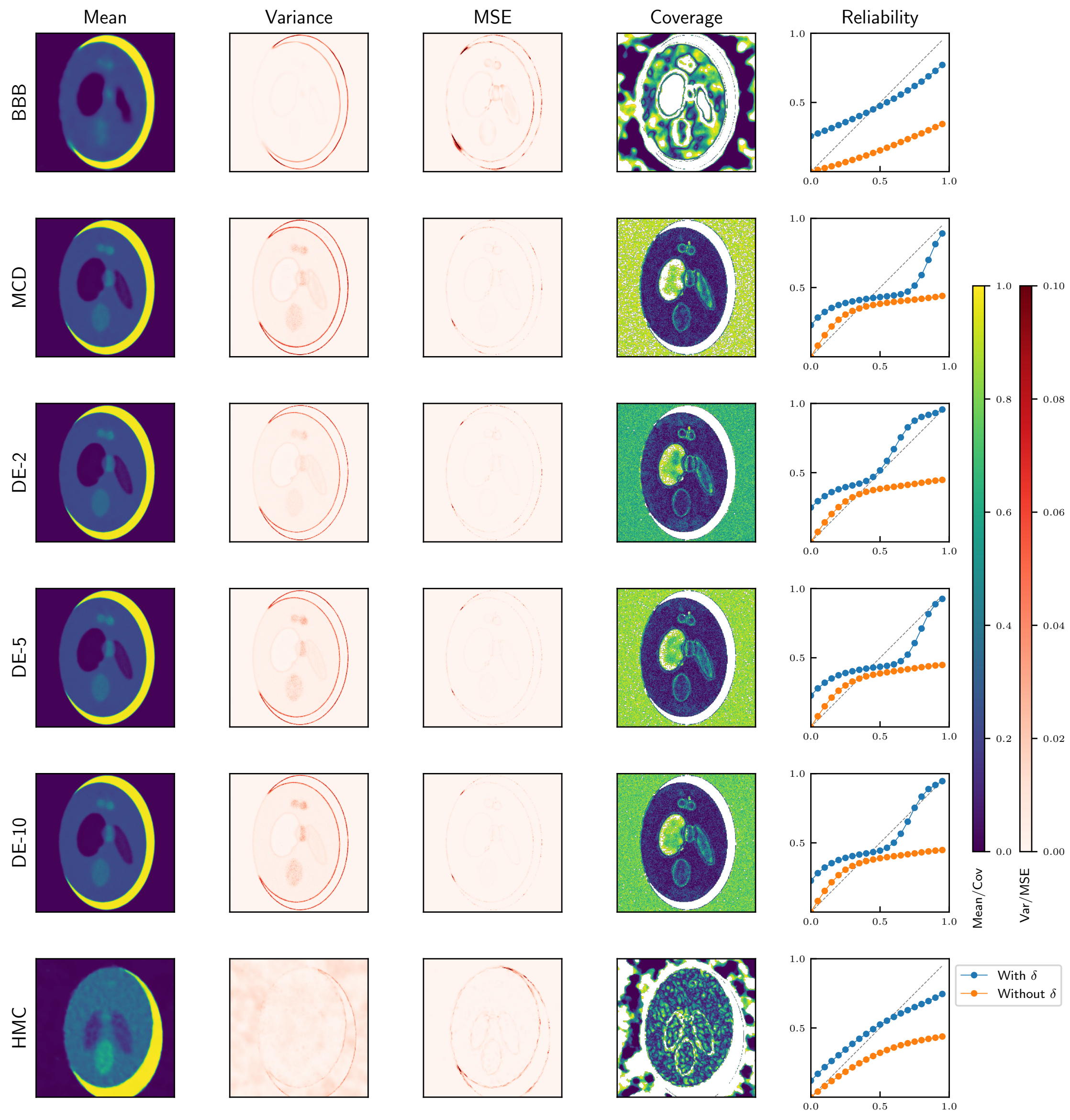

Figure 4.20: Validation results of all approaches in the 20-view case. From left to right: mean image reconstruction, variance, MSE, coverage, and reliability diagram.

5 Conclusions

Medical imaging plays a crucial role in the modern healthcare system, enabling doctors to make informed diagnoses based on the visualization of problematic organs and damaged tissues within a patient. However, many imaging platforms, such as CT scanners, emit radiation which is harmful to the patient and increase the risk of pathologies like cancer. This has created interest in approaches that minimize radiation exposure, while still providing the doctor with sufficient information for a proper diagnosis. In this work, we aimed to address both these challenges through the introduction of a novel CT image reconstruction approach, based on INRs. We found that our approach outperforms traditional image reconstruction quality in the low-measurement regime and is able to provide reasonably well calibrated uncertainty estimates over its output. Calibration is highly desirable, for several reasons. First, it addresses the lack-of-interpretability concerns that could bar the use of NN-based approaches in healthcare. Given estimates of uncertainty, doctors can better understand artifacts in the reconstructed images. Second, they provide a novel source of information for doctors to make their diagnoses. If a generated image has a lot of variance in a region of interest, for example an area where there appears to be a tumor, the doctor could order another scan to be taken to ensure proper diagnosis before prescribing cancer treatment. More generally, if the uncertainty estimates are accurate enough, it becomes possible to even develop fully automated triage systems, where computers organize medical images by their perceived complexity of analysis and can even assign them to healthcare providers of different levels of specialization. Finally, calibrated uncertainty quantification is a first step towards active learning. In future work, we aim to leverage these calibrated uncertainty estimates to inform real-time optimized measurement collection, which would reduce overall radiation exposure. The results above already show that, with just 20 equally spaced measurement views, the proposed INR approach produces image reconstructions that meet the typical definition of acceptable in the literature (PSNR $>$ 30 dB) for the phantoms used in this work. Further work will need to be done to ensure that reconstruction performance is similar on actual medical images and that the algorithms are robust to noisy measurements.

Although restricted to phantoms and noiseless data, this work presents the first large-scale study of model parameterization for INRs with uncertainty quantification. We found that MCD outperforms BBB and that hyperparameter selection plays an important role in model performance. Specifically, for MCD we found that activation function choice has a large affect on image reconstruction quality. As argued in the SIREN [63] work, the top performing MCD models used the Sine activation. However, we found the Sine activation to be inconsistent in performance, with Silu, Tanh, and Relu achieving similar, slightly lower performance, but greater consistency. This implies that networks with Sine activations can require substantial further tuning effort, during training. Further, we confirmed previous findings that RFF embeddings enable NNs to learn high-frequency image components better [60]. However, we found that the frequency choice used in the RFF embedding must be consistent with the amount of data used in training the INR. Too low of an RFF Ω_0 prohibits higher frequency learning, while too high of an RFF Ω_0 results in high-frequency image artifacts. Furthermore, consistent with previous work [75], we found that ensembling MCD architectures can improve image reconstruction quality and model calibration, achieving significant improvements even for small numbers of base learners. Finally, while we did not not optimize HMC hyperparameters, we implemented and verified the validity of several metrics for assessing HMC convergence. We found Hamiltonian energy, acceptance rate, training loss, and the $\hat{R}$ distribution for model output to be informative of whether the chain is sampling from the true BNN posterior. Despite previous discussion of parameter $\hat{R}$ in the BNN HMC literature [73], we found it to be less useful, given the symmetries that exist in NN parameterizations. We further verified that t-SNE visualizations of the HMC output samples provide insight into the relative sampling performance and mixing of different chains. We found that combining samples from chains that produced reasonable samples but did not fully mix slightly improved image reconstruction quality and drastically improved calibration. In all, DEs of the top 5 or 10 performing MCD methods achieved the best results in terms of image reconstruction and calibration. However, we have indications that, when subject to a search for optimal hyperparameters, HMC may achieve better results, especially in lower data regimes.

Finally, while this work proposes the first use of uncertainty quantification for INRs, it is not the first use case of INRs for CT image reconstruction. As recently as 2020, the CoIL architecture was demonstrated to improve image reconstruction pipelines by learning a functional form of the measurement sinogram [28]. This, however, still necessitates the use of classical image reconstruction to reconstruct the ultimate desired image cross-section. It also lacks support for uncertainty estimation, since the relation between the uncertainty of sinogram values and image pixels is not immediately evident. Hence, training a network to produce calibrated estimates of sinogram values does not automatically produce the estimates of pixel value uncertainty that are much more important for most medical applications. Our approach instead proposes an end-to-end image reconstruction pipeline, in which the network output is the desired image cross-section. This is achieved by sampling the full image and performing a Radon transform in the network loss at each training epoch. Given the small size of INRs, we found this to be computationally tractable. The end-to-end approach is likely to increase performance and provides seamless support for uncertainty estimation, as shown in this work. In future work, it would be interesting to compare the performance of our end-to-end approach to that of the CoIL architecture, as well as consider approaches to increase the training efficiency of the proposed end-to-end solution.

Appendices

A
X-Ray Matter Interactions

The typical energy of X-ray photons generated for medical CT is in the range 20-140keV. X-rays in this energy range interact with matter via the photoelectric effect, the Compton effect, and coherent scattering.

In the **photoelectric effect**, the X-ray photon relinquishes all its energy to liberate a lower-energy, deep-shell electron in an atom. The X-ray photon thus ceases to exist, while the liberated electron becomes a photoelectron. Meanwhile, outer-shell electrons fall inward to fill the photoelectron's vacancy in the atom, emitting a photon of characteristic radiation in the process. Since the resultant characteristic radiation photon has less energy than the original X-ray photon, it is attenuated by matter within the body and might not make it to the detector. The probability of photoelectric absorption is proportional to the cube of the atomic number of the matter, meaning that different tissues in the body will have different rates of x-ray absorption.

The **Compton effect** is similar to the photoelectric effect, except that the incoming X-ray photon has far more energy than the binding energy of the electron. Thus, when the X-ray photon strikes and frees the electron from the atom, it only loses a small amount of energy and may be deflected at any angle from $0 - 180°$. Low-energy X-ray photons typically backscatter, whereas higher energy photons have a high probability of forward scattering. Because of the wide deflection angle, such scattered photons provide little information about the interaction. Further, the probability of a Compton interaction is purely based on the electron density of the material, not the atomic number. Thus, Compton scattering does not help in distinguish different tissues in the body and is typically seen as a source of noise in medical imaging.

The final interaction type, **coherent** or **Rayleigh scattering**, is the least important to CT imaging. In this case, the incident X-ray has far lower energy than the electron binding energy, meaning no electrons are liberated and ionization does not occur. Instead, the oscillating electric field of the X-ray causes the electrons to oscillate and emit radiation of the same wavelength, similarly to the operation of a radio station transmitter.

B
Classic CT Reconstruction Algorithms

Contents

B.1 Filtered Back-Projection (FBP) 63
B.2 Algebraic and Iterative Reconstruction 64
B.3 Simultaneous Iterative Reconstruction Technique (SIRT) 65
B.4 Simultaneous Algebraic Reconstruction Technique (SART) 65
B.5 Conjugate Gradient Least Squares (CGLS) 66
B.6 Expectation Maximization (EM) 66

In this appendix, we provide brief descriptions of the classical approaches implemented in this work via the `TomoPy Astra` [77] software package. These algorithms are clinically approved and widely used in medical imaging, serving as a basis of comparison for the methods developed in this work.

B.1 Filtered Back-Projection (FBP)

Filtered back-projection (FBP) [92] is an analytic algorithm which calculates a stable, discretized version of the inverse Radon transform. As the name implies, there are two key steps: filtering and back-projection.

The forward-projection of (2.1.4.5) describes how X-rays passing through the object domain create a measurement. In **back-projection (BP)**, this measurement is integrated back along the X-ray path across the object domain. This is done over all projection angles θ, using

$$\hat{f}_{\mathrm{BP}}(x, y) = \int p_\theta(x \cos\theta + y \sin\theta)\, d\theta \tag{B.1.0.1}$$

to reconstruct the object attenuation coefficient image. As the number of projection angles, θ, increases, the image reconstruction improves. However, as shown in Figure B.1, this back-projection is insufficient to guarantee a clear image. While information about the low frequencies of the object are captured in measurements at several view angles, that of high frequencies may only be captured in a few view-angles. Thus, the low frequencies are sampled far more densely than the higher frequencies, resulting in a blurry image. This can be corrected by suppressing the lower frequencies with filtering, by applying to each projective measurement, $p_\theta(r)$, the sequence of a Fourier transform (FFT), high-pass filter, and an inverse Fourier transform (iFFT). While several high-pass filters can be used, a popular choice is the Ram-Lak filter, which generates the filtered projective measurement

$$\tilde{p}_\theta(r) = \int \mathcal{P}_\theta(\omega)|\omega|e^{i2\pi\omega r}d\omega, \tag{B.1.0.2}$$

where $\mathcal{P}_\theta(\omega)$ is the Fourier transform of $p_\theta(r)$ and $|\omega|$ the frequency response of the filter. Performing back-projections of all the filtered projective measurements,

$$\hat{f}_{\text{FBP}}(x,y) = \int \tilde{p}_\theta(x\cos\theta + y\sin\theta)\,d\theta, \tag{B.1.0.3}$$

results in a sharper object attenuation coefficient image. Figure B.1 visualizes the difference in reconstruction performance of BP and FBP for increasing measurement view angles, θ. Although the analytic FBP algorithm is fast and numerically stable, it suffers from poor resolution-noise trade-off.

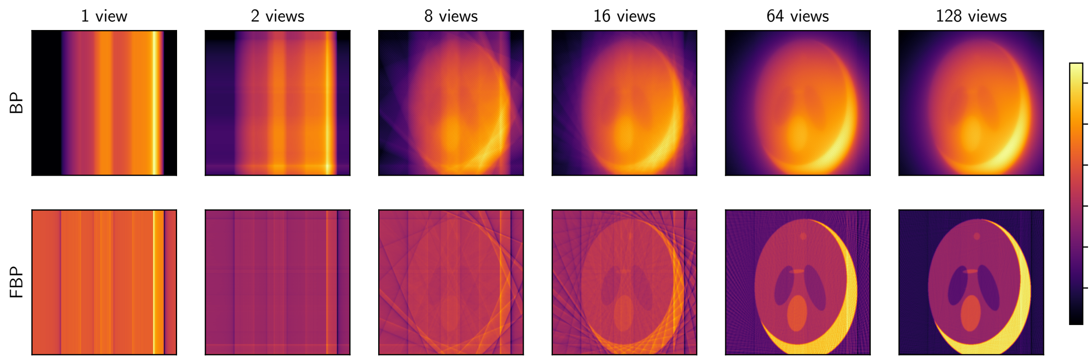

Figure B.1: Comparison of the reconstruction quality, as a function of the number of views, of the BP (top) and FBP (bottom) algorithms.

B.2 Algebraic and Iterative Reconstruction

The reconstruction problem can be formulated as a system of linear equations

$$W\vec{f} = \vec{p}, \tag{B.2.0.1}$$

where $\vec{p}$ is an $m \times 1$ vector of the m projective measurement values in the sinogram; $\vec{f}$ is an $n \times 1$ vector of the n attenuation coefficient pixel values in the reconstruction image; and W is a, typically sparse, $m \times n$ weight matrix representing the contribution of each of the m sinogram values to each of the n image pixel values. Given the vector $\vec{p}$, the goal is to solve for $\vec{f}$. If W were invertible, $\vec{f}$ would simply be $W^{-1}\vec{p}$. However, because n is usually much larger than m, the system of equations of (B.2.0.1) is underconstrained. In **algebraic reconstruction**, this problem is addressed by using **iterative algorithms** that pose the reconstruction of $\vec{f}$ as the solution of a constrained optimization problem,

$$\vec{f}^{\,*} = \arg\min_{\vec{f}} ||\vec{p} - W\vec{f}||, \text{ subject to } f_i \geq 0\ \forall i.$$

Several families of iterative solvers can be used to solve this optimization, such as Landweber, Krylov subspaces, and expectation maximization (EM). The key benefit of iterative methods is that prior system knowledge can be integrated, via the cost function and initialization of W. Their down-side is that they are not necessarily stable, may not converge, and are much slower than analytic techniques, such as FBP.

B.3 Simultaneous Iterative Reconstruction Technique (SIRT)

The **simultaneous iterative reconstruction technique (SIRT)** [93, 94] is a Landweber iterative method that updates the image reconstruction using all available sinogram projection data, $\vec{p}$, simultaneously. The optimization update at step k is defined as

$$\vec{f}^{(k+1)} = \vec{f}^{(k)} + CW^T R\big(\vec{p} - W\vec{f}^{(k)}\big),$$

where $R \in \mathbb{R}^{m\times m}$ is a diagonal matrix containing the inverse row sums, $r_{ii} = (\sum_{j=0}^{n-1} w_{ij})^{-1}$, and $C \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing the inverse column sums, $c_{ii} = (\sum_{i=0}^{m-1} w_{ij})^{-1}$. The weighted projection difference, $R\big(\ \vec{p} - W\vec{f}^{(k)}\ \big)$, corresponds to the inverse of the length each X-rays passes through the volume. Shorter rays have a higher contribution, with the weighting required to guarantee convergence. This difference is forward-passed back to the image domain, using the weighted back-projection term, CW^T, where it can be used to update the reconstruction. These updates iteratively solve the problem

$$\vec{f}^{\,*} = \arg\min_{\vec{f}} ||\,\vec{p} - W\vec{f}\,||_R = \arg\min_{\vec{f}} \big(\vec{p} - W\vec{f}\big)^T R\,\big(\vec{p} - W\vec{f}\big),$$

converging to a weighted least-squares solution, with weights given by the inverse row sums of W.

B.4 Simultaneous Algebraic Reconstruction Technique (SART)

The **algebraic reconstruction technique (ART)** [95] was one of the first proposed algebraic iterative algorithms for CT image reconstruction. It is a Landweber technique almost identical to the SIRT algorithm. However, a single projective measurement is used to update the reconstruction image per update step. Generally, the ART algorithm reaches a solution much faster than SIRT, but does not have stable convergence if the system of equations is inconsistent, for example due to measurement noise.

The **simultaneous algebraic reconstruction technique (SART)** [96] was proposed in 1984, as an improvement to ART, and is also a Landweber algebraic iterative algorithm. It combines the reduced runtimes of ART with the improved convergence of SIRT, by using all the projective measurements from a single view angle in each optimization iteration. The update of image vector index i at step n is defined as

$$f_i^{(n+1)} = f_i^{(n)} + \frac{\lambda_n}{\sum_{j=0}^{n-1} W_{ij}} \sum_{j=\theta_n L+1}^{\theta_n L+L} W_{ij} \frac{p_j - \hat{p}_j}{\sum_{g=0}^{m-1} W_{gj}}, \qquad \text{(B.4.0.1)}$$

where $\lambda_n << 1$ is a, potentially dynamic, relaxation parameter; θ_n is the $(n \mod N)^{\text{th}}$ measurement angle of the sinogram, assuming N total measurement angles; and L is the number of projective measurements taken at each angle. SART typically converges to a good reconstruction within a few iterations.

B.5 Conjugate Gradient Least Squares (CGLS)

The **conjugate gradient least squares (CGLS)** [97] algorithm is a Krylov subspace iterative method. Since it requires a positive-definite system matrix, the CT image reconstruction problem is reformulated in terms of the set of normal equations

$$W^T W \vec{f} = W^T \vec{p}. \tag{B.5.0.1}$$

Due to the positive-definiteness of $W^T W$, there exists a set of conjugate normal vectors $\mathcal{Q} = \{\vec{q}_1, ..., \vec{q}_n\}$, where $\vec{q}_i^T W^T W \vec{q}_j = 0, \forall\, i \neq j \in (1, n)$. Since $\mathcal{Q}$ forms a basis for $\mathbb{R}^n$, the image vector $\vec{f}$ can be rexpressed as a linear combination of these conjugate normal vectors,

$$\vec{f} = \sum_{i=1}^{n} \alpha_i \vec{q}_i. \tag{B.5.0.2}$$

Thus, solving for $\vec{f}$ becomes a problem of solving for the conjugate normal basis vector directions, $\vec{q}_i$, and their corresponding weights, α_i. This can be achieved iteratively by expressing the problem as a quadratic least-squares minimization of the function

$$L(\vec{f}) = \frac{1}{2} \vec{f}^T W^T W \vec{f} - \vec{f}^T W^T \vec{p}, \tag{B.5.0.3}$$

which has gradient $\nabla L(\vec{f}) = W^T W \vec{f} - W^T \vec{p}$ and a guaranteed unique minimizer because the Hessian $\nabla^2 L(\vec{f}) = W^T W$ is symmetric positive-definite. The name conjugate gradient least squares comes from the fact that, in each iteration, a conjugate basis vector and its weight are found by taking a gradient step in the direction that minimizes the least-squares function, $L(\vec{f})$, as

$$\vec{f}^{(k+1)} = \vec{f}^{(k)} + \alpha_k \vec{q}_k \tag{B.5.0.4}$$

$$\vec{r}_k = W^T \vec{p} - W^T W \vec{f}^{(k)} \tag{B.5.0.5}$$

$$\vec{q}_k = \vec{r}_k - \sum_{i<k} \frac{\vec{q}_i^T W^T W \vec{r}_k}{\vec{q}_i^T W^T W \vec{q}_i} \vec{q}_i \tag{B.5.0.6}$$

$$\alpha_k = \frac{\vec{q}_k^T \vec{r}_k}{\vec{q}_k^T W^T W \vec{q}_k} \tag{B.5.0.7}$$

where $\vec{r}_k$ is the residual at step k. Thus, the main difference between SIRT/SART and CGLS is that the search direction in SIRT/SART is determined only by the projection difference at that point, while in CGLS the search directions of all the previous iterations are also taken into account. CGLS typically converges much faster than SIRT, but has a large memory footprint.

B.6 Expectation Maximization (EM)

The final classical approach to CT image reconstruction that we consider is a statistical iterative method known as **expectation maximization (EM)** [98]. This technique explicitly encodes prior knowledge about the X-ray physics at hand. Each projective measurement, p_j is modeled as a Poisson distribution,

$$p_j \sim \mathcal{P}_j = \text{Poisson}(\lambda_j) = \frac{\lambda_j^{p_j} e^{-\lambda_j}}{p_j!}, \tag{B.6.0.1}$$

where the distribution mean $\lambda_j = \mathbb{E}[\mathcal{P}_j]$ is the function

$$\lambda_j = \sum_i W_{ij} f_i \tag{B.6.0.2}$$

of the probability $W_{i,j}$ that an X-ray photon penetrating image pixel i was measured at detector location j; and the underlying attenuation coefficient function f to reconstruct.

The measurement sinogram is modeled as the likelihood

$$\mathbb{P}(\vec{p}|\vec{f}) = \prod_j \frac{\lambda_j^{p_j} e^{-\lambda_j}}{p_j!} = \prod_j \frac{(\sum_i W_{ij} f_i)^{p_j} e^{-(\sum_i W_{ij} f_i)}}{p_j!}. \tag{B.6.0.3}$$

The EM algorithm computes the maximum likelihood estimate of f,

$$\hat{f}_{ML} = \max_f \left[\log \left(\mathbb{P}(p|f) \right) \right], \tag{B.6.0.4}$$

by alternating between expectation and maximization steps. These can be combined into the update-step

$$\hat{f}_i^{(k+1)} = \frac{\hat{f}_i^{(k)}}{\sum_j W_{ij}} \sum_j \frac{W_{ij} p_j}{\sum_i W_{ij} \hat{f}_i^{(k)}}. \tag{B.6.0.5}$$

The EM algorithm is computationally intensive, but guaranteed to converge to a local optimum of the likelihood. Further, although a Poisson distribution was assumed for p_j in this discussion, further knowledge of the detector noise can be easily incorporated into the model.

C
Neural Networks

Contents

C.1 Neural Network Mathematical Formulation 69
C.2 Neural Network Training and Prediction 71
C.2.1 Hyperparameter Sweeps 72

In this work, we investigate an alternative to the CT image reconstruction problem based on artificial neural networks (ANNs). Frequently referred to as **neural networks (NNs)**, these are computational models that have recently achieved large popularity in machine learning. This is, in large part, due to their strength for **representation learning**, enabling the automated discovery, from raw data, of representations needed for classification and detection. Deep NNs have several layers, learning data representations with multiple levels-of-abstraction, referred to as **deep learning** [99]. Algorithmic advances (the development of backpropagation [100] and stochastic gradient descent [101]), hardware improvements (the widespread availability of graphical processing units (GPUs) [102]), and access to large scale datasets (such as the ImageNet dataset [103]), have propelled deep learning beyond the academic sphere. Over the last decade, deep learning has achieved state-of-the-art performance in several fields, including natural language processing [104], speech recognition [105], and image recognition [106]. More recently, NNs have also proven useful for other forms of learning, such as generative modeling [107] and reinforcement learning [108]. In this section, we will discuss the mathematical formulation of multilayer perceptron (MLP) networks, to provide a notational basis for our discussion of one of the newest and most exciting applications of NNs, implicit neural representations (INRs).

C.1 Neural Network Mathematical Formulation

A NN is a computational graph composed of **neurons**. As depicted in Figure C.1(a), a neuron takes a vector, $\vec{x} = (x_1, x_2, ..., x_n)^T$, and scalar **bias**, b, as input, outputting a scalar, z. This

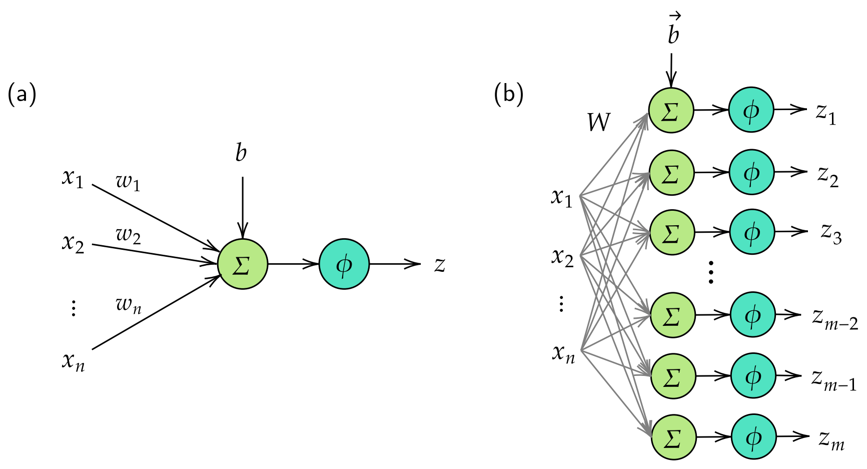

Figure C.1: Computation of a) a NN neuron, and b) a NN layer. In these graphs, Σ denotes a summation and ϕ a non-linear activation function.

mapping consists of a linear combination of the inputs, with **weights** $\vec{w} = (w_1, w_2, ..., w_n)^T$ and a non-linear **activation function** ϕ,

$$z = \phi\left(b + \sum_n w_n \cdot x_n\right) = \phi\big(b + \vec{w}^T\vec{x}\big). \tag{C.1.0.1}$$

Modern NNs are composed by thousands of neurons, organized into network **layers**. Each layer consists of m neurons, each with one output, z_i. The value m is known as the network **width**. Each neuron has its own set of weights, $\vec{w}_i$, and bias b_i, which are used to compute the output z_i, according to

$$z_i = \phi\left(b_i + \sum_l W_{il} \cdot x_l\right) = \phi\big(b_i + W_i^T\vec{x}\big). \tag{C.1.0.2}$$

where W is a $m \times n$ matrix containing the layer weights, W_{ik} is the weight between input k and output i and $\vec{b}$ is a vector of dimension $(m, 1)$ containing the biases b_i. A NN layer is denoted fully-connected if there is a weight between each of its inputs and outputs, i.e. the matrix W is dense. While it is possible to design a NN layer in which each neuron has a different activation function, only networks with a common activation function across neurons are considered in this work.

A NN is typically comprised of multiple layers, as illustrated in Figure C.2. When the layers are fully connected, the network is denoted as a **multilayer perceptron (MLP)**. The total number of layers, k, is known as the network **depth**, resulting in the notion of 'deep' learning. In a fully-connected network, each layer j takes as input the output vector of layer $j-1$, $\vec{z}^{\,j-1}$, and outputs the vector $\vec{z}^{\,j}$, using a distinct weights matrix, W^j, and bias vector, $\vec{b}^{\,j}$. Thus, the i^{th} output of the j^{th} layer is expressed as,

$$z_i^j = \phi\left(b_i^j + \sum_n W_{in}^j \cdot z_n^{j-1}\right) = \phi\big(b_i^j + (W_i^j)^T\vec{z}^{\,j-1}\big). \tag{C.1.0.3}$$

While it is possible to vary the width of each layer, we only consider networks of uniform width, m. In this work, we consider MLPs that output a normalized scalar quantity, $\hat{f} \in (0, 1)$. This is guaranteed by feeding the last layer's output into a single neuron with a **sigmoid** activation function,

$$\sigma(z_i) = \frac{1}{1 + e^{-z_i}} = \frac{e_i^z}{e_i^z + 1}, \tag{C.1.0.4}$$

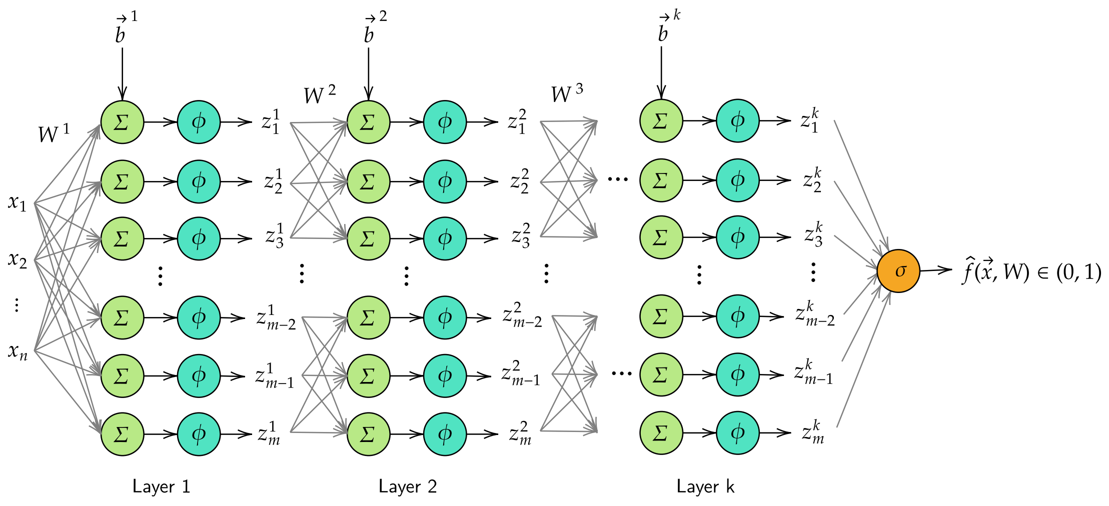

Figure C.2: An MLP composed of k layers of width m.

depicted as the orange node in Figure C.2. As illustrated in Figure C.3, this function is centered around $\sigma(0) = \frac{1}{2}$, approaching 1 as $x \to \infty$ and -1 as $x \to -\infty$. The overall mapping implemented by all the operations of Figure C.2 is denoted as $\hat{f}(\vec{x}, W)$, where $W = \{W^j, b^j\}_{j=1}^k$ contains all network parameters.

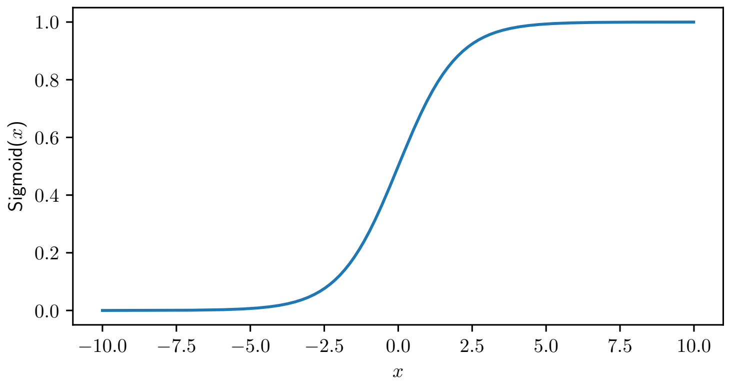

Figure C.3: Sigmoid function.

C.2 Neural Network Training and Prediction

While activation function type, network width, and network depth are all design choices, the weights and biases of a NN must be learned in order for the network to perform useful computation. Typically, NNs are trained in a **supervised learning** setting, in which the training data has labels, i.e. it consists of input-output pairs, (x, f). A loss function, $\mathcal{L}$, is used to assess the network performance, by measuring the difference between the groundtruth output, f, and predicted network output, $\hat{f}$, for each training example x. The **mean-squared error (MSE)**,

$$\mathcal{L}_{\text{MSE}} = \sum_i (f_i - \hat{f}(\vec{x}_i, W))^2, \tag{C.2.0.1}$$

is a common loss function for regression tasks. The NN is trained by updating its weights and biases so as to minimize this loss. This is achieved by performing gradient descent on this loss with respect to the NN parameter weights and biases. Gradient descent is an iterative procedure, where the network parameters are updated according to

$$W^{i+1} = W^i - \eta \nabla_W \mathcal{L}_{\text{MSE}}, \tag{C.2.0.2}$$

where i is the gradient descent iteration, $\nabla_W \mathcal{L}_{\text{MSE}}$ the gradient of the loss with respect to parameters W and η a **learning rate** that controls the step size of each iteration. For neural networks, the gradient descent updates can be efficiently calculated using an algorithm known as **backpropagation** [100]. The optimization is iterative, typically converging towards a local minima of the loss. Once the optimization has converged, the network weights are frozen and the function $\hat{f}(\vec{x}, W)$ can be used to predict labels for unlabeled data $\vec{x}$.

C.2.1 Hyperparameter Sweeps

Hyperparameter sweeps can be used to test and optimize NN parameterization. The hyperparameter sweeps discussed in the following section were implemented with `W&B`, which offers a variety of search strategies. Two search strategies were used: grid and Bayesian. In a **grid search** a finite set of values is chosen per hyperparameter and every combination of parameters is tested. This is ideal for understanding how individual parameters affect performance, but is not computationally feasible for large or continuous parameter spaces. **Bayesian search**, alternatively, enables search over large, continuous parameter spaces by narrowing the search space according to a model goodness criterion, such as maximizing PSNR. This is achieved by using a Gaussian process to model the relationship between parameters and the goodness metric and choosing the parameters that maximize the probability of model improvement.

D Tuneable MLP Model Parameters

Contents

D.1 Width and Depth 73
D.2 Fourier Feature Mappings 73
D.3 Activation Functions 74
D.4 Optimizer 75

In this appendix, we layout the different parameters we considered in designing our MLPs and provide any known insights as to how they affect model performance. These insights informed the hyperparameter sweeps described in the following section.

D.1 Width and Depth

The size of a neural network is determined by both its width (number of nodes per layer) and depth (number of layers), as described in Appendix C. Universal approximation theorems [109] have been derived in both the arbitrary-width [110, 111] and arbitrary-depth [112–114] cases, demonstrating that NNs are theoretically guaranteed universal function approximators in the infinite limit. In practice, however, neural networks have finite width and depth. Recent work has empirically demonstrated and theoretically suggested that, in this regime, increased-depth networks generally perform better than increased-width networks [115, 116]. It is also known that, while neural networks are overparametrized relative to the amount of training data, this overparametrization is key for their generalization ability [117, 118]. However, there are no exact guidelines on how to choose the width or depth of a network. In this work, the best performing width and depth were found empirically for each network, via the hyperparameter sweeps described in Section 3.6.

D.2 Fourier Feature Mappings

Random Fourier features (RFF) were first introduced in 2007 [119] as a means of accelerating kernel methods. The key idea is to map the input data to a randomized low-dimensional feature space, while maintaining the kernel of the original data. Given input $\vec{x} \in \mathbb{R}^n$, the RFF mapping takes the form

$$\gamma_{\text{RFF}}(\vec{x}) = [\cos(2\pi B\vec{x}), \sin(2\pi B\vec{x})]^T, \tag{D.2.0.1}$$

where B is an $m \times n$ matrix, with each entry sampled from $\mathcal{N}(0, \Omega_0^2)$. The standard deviation, Ω_0, is a tuneable hyperparameter, but remains static after initialization – i.e. it is not modified with NN weights during the MLP training. There exist other types of Fourier feature mappings, such as **positional encodings**, in which

$$\gamma_{\text{PE}}(\vec{x}) = [..., \cos(2\pi\Omega_0^{j/m}\vec{x}), \sin(2\pi\Omega_0^{j/m}\vec{x}), ...]^T, \tag{D.2.0.2}$$

for $j = 0, ..., m-1$.

In 2018, it was theoretically demonstrated that NNs can be approximated by kernel regression via the **neural tangent kernel (NTK)** [118]. Using this intuition, in 2020, it was argued that applying a simple Fourier feature mapping to input data enables MLPs to learn high-dimensional functions rapidly, even in low-dimensional problem domains [60], making the technique particularly well-suited for INRs. In fact, positional encodings have been shown to have key importance in the success of NeRF [16] and Fourier feature mappings have been shown to boost the performance of the CoIL network for medical image reconstruction [28]. In this work, all networks apply an RFF mapping, γ_{RFF}, to the input data. The standard deviation, Ω_0, is tuned among other hyperparameters in the sweeps described in Section 3.6.

D.3 Activation Functions

Activation functions are key to the success of neural networks, transforming what would otherwise be simple linear systems into complex, non-linear universal function representers. We performed hyperparameter sweeps with five activations widely used in the MLP and INR literature – ReLU, SiLU, Sine, SoftPlus, and Tanh. We now briefly review these activation functions, as well as their use in deep learning.

The **rectified linear unit (ReLU)**, plotted in blue in Figure 3.3, was introduced as early as the 1960s for visual feature extraction in hierarchical NNs [120]. The ReLU is defined as

$$\text{ReLU}(x) = \max\{0, x\}\ , \tag{D.3.0.1}$$

returning its input if greater than zero and otherwise returning zero. Despite its hard non-linearity at zero, non-differentiability at zero, and vanishing gradient challenge [121], the ReLU was shown in 2011 to enable better training than previously used activation functions, such as Sigmoid and Tanh, by inducing sparse representations [122]. As of 2017, the ReLU was the most popular activation function for deep NNs [123].

The **sigmoid-weighted linear unit (SiLU)**, plotted in orange in Figure 3.3, is a specific instance of the **Swish** activation function family and was proposed in 2017 as a continuous, 'undershooting' version of the ReLU [123]. The Swish family, parameterized by β, is defined as

$$\text{Swish}_\beta(x) = x \cdot \sigma(\beta x) = \frac{x}{1 + e^{-\beta x}}\ , \tag{D.3.0.2}$$

where $\sigma(x)$ is the sigmoid function. By setting β to different values in $[0,\infty)$, Swish$_\beta$ non-linearly interpolates smooth functions between the linear function and ReLU. In 2017, Swish was empirically shown to outperform ReLU, a result theoretically attributed to its bounded, smooth, and non-monotonic nature [123]. More recently, Swish has been shown to outperform both ReLU and Sine in the context of CT image reconstruction via Automatic Integration (AutoInt) [67]. The SiLU is the specific instance of Swish where $\beta = 1$,

$$\mathrm{SiLU}(x) = x \cdot \sigma(x) = \frac{x}{1+e^{-x}}\ . \tag{D.3.0.3}$$

The **Sine** activation function, plotted in green in Figure 3.3, is the sinusoid

$$\mathrm{Sine}_{\omega_0} = \sin(\omega_0 \cdot x). \tag{D.3.0.4}$$

In the 2020 SIREN paper [63], INRs with sinusoidal activation functions and random Fourier features were empirically demonstrated to outperform ReLU-based INRs. Theoretically, it was argued that these periodic activations are better suited to capturing naturally complex signals and their derivatives. However, the performance of these activations depends strongly on the choice of frequency, ω_0, which needs to be tuned.

The **SoftPlus** activation function, plotted in red in Figure 3.3, has continuous and differentiable form

$$\mathrm{Softplus}(x) = \ln(1+e^x). \tag{D.3.0.5}$$

It was introduced in 2001 [124] as the primitive of the sigmoid function. It is primarily used as a smooth approximation to the ReLU activation and to constrain to positive outputs, since $\mathrm{Softplus}(x) \in (0,\infty)$.

The hyperbolic tangent **Tanh**, plotted in purple in Figure 3.3, has form

$$\mathrm{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{D.3.0.6}$$

and is both differentiable and monotonic. It has a form similar to the sigmoid function, with $\mathrm{Tanh}(x) = 2\sigma(2x) - 1$, but lies in the range (-1,1) instead of (0,1), meaning it does not constrain to positive values. Before the ReLU became popular, the sigmoid and Tanh were two of the most common activation functions. Tanh was easier to train and typically outperformed the sigmoid as an activation function. However, because these sigmoidal activation functions saturate for large inputs, their derivatives vanish for these inputs, leading to slow convergence of learning algorithms. This has motivated the increased use of ReLU-like activation functions [125], which ameliorate the vanishing derivative problem.

D.4 Optimizer

As described in Section C.2, the weights and biases of a neural network are optimized by gradient descent of the network loss with respect to its weights and biases. There are several possible choices of **optimizer**, which implement different variations on gradient descent. The **gradient descent** procedure of (C.2.0.2) is usually difficult to implement, because each iteration requires the evaluation of the gradient of the loss of (C.2.0.1) over the entire training set. This is very inefficient in terms of the number of computations per parameter update. **Stochastic gradient descent (SGD)** addresses this problem by using a subset of the training data per descent step. This drastically improves optimization time, although a descent direction is not guaranteed at each iteration. Empirically, SGD has proven to work quite well. **Adaptive moment estimation (Adam)** [126] further improves on SGD's convergence by adding a momentum component and rescaling. Each weight and bias parameter has a unique, variable learning rate, dependent on its gradient values in recent optimization iterations. Finally, **adaptive moment estimation with weight decay (Adam-W)** [127] incorporates a weight decay regularization in the loss function. This is an L_2 penalty that reduces model complexity and improves generalization ability by shrinking the weight vector. In this work, Adam and Adam-W are used to train the INR models, with the default learning rate of 3e-4 and a tuneable weight decay term for Adam-W. Attempts at tuning learning rate proved worse or of similar performance to using the default value.

E
Model Performance and Uncertainty Metrics (Extended)

Contents

E.1 Peak Signal-to-Noise Ratio (PSNR) 77
E.2 Negative Log Likelihood (NLL) 77

E.1 Peak Signal-to-Noise Ratio (PSNR)

Peak signal-to-noise ratio (PSNR) is a metric frequently used to quantify reconstruction quality of images and videos. Given an $m \times n$ ground truth image I and a noisy reconstruction image K, the PSNR (in dB) is defined as the ratio between the maximum pixel value of I and the MSE between I and K,

$$\text{PSNR}(I,K) = 10 \cdot \log_{10}\left(\frac{\max(I)^2}{\text{MSE}(I,K)}\right) \text{ where } \text{MSE}(I,K) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I_{ij} - K_{ij}]^2. \quad \text{(E.1.0.1)}$$

The higher the PSNR, the better the reconstructed image matches the original. In the absence of any noise, I and K are identical, $\text{MSE}(I,K) = 0$, and PSNR is infinite. For lossy images, PSNR is typically between 30-50dB, with values over 40dB considered very good, and values below 20dB considered unacceptable [128].

E.2 Negative Log Likelihood (NLL)

Negative log likelihood (NLL) is a commonly used metric of probabilistic model quality. In this work, we assume an independent Gaussian model, where pixel i is sampled from a Gaussian distribution of mean f_i, the groundtruth pixel value, and variance σ^2, estimated by the sample variance, $\hat{\sigma}^2$, of the network pixel responses

$$\hat{\mu} = \frac{1}{|\mathcal{X} \times \mathcal{Y}|} \sum_{x,y} \hat{f}(x,y) \tag{E.2.0.1}$$

$$\hat{\sigma}^2 = \frac{1}{|\mathcal{X} \times \mathcal{Y}|} \sum_{x,y} (\hat{f}(x,y) - \hat{\mu})^2. \tag{E.2.0.2}$$

The NLL is the negative of the log-likelihood of the pixel values under this model, assuming independent sampling,

$$\mathrm{NLL}(\hat{f}, f) = \sum_i \left(\frac{1}{2\hat{\sigma}^2} (\hat{f}_i - f_i)^2 + \frac{1}{2} \log(2\pi\hat{\sigma}^2) \right) \tag{E.2.0.3}$$

A good probabilistic model will maximize the likelihood. Thus, the NLL is minimized iff $f_i = \hat{f}_i \, \forall i$.

Although NLL can be used to indirectly measure model calibration, recent work has shown that there is a disconnect between NLL and accuracy. For example, classification neural networks can overfit to NLL without overfitting to 0/1 loss. Surprisingly, this is beneficial to classification accuracy, at the expense of well-modeled probabilities. Thus, for deep neural networks, overfitting to NLL manifests in probabilistic error rather than classification error [81].

F
HMC Sampling Algorithm

Our goal in using HMC is to sample weight parameters, $\{W^{(1)}, ..., W^{(N)}\}$, from the BNN weight posterior, $p(W|h)$. This problem is reformulated in terms of physics-inspired dynamics. These dynamics are governed by Hamiltonian

$$\mathcal{H}(W, P_W) = U(W) + \frac{1}{2} P_W^T M^{-1} P_W, \tag{F.0.0.1}$$

where W are the 'position' terms, P_W are the 'momentum' terms, $U(W) = -\ln p(W|h)$ is the system potential energy, $\frac{1}{2} P_W^T M^{-1} P_W$ is the system kinetic energy, and M is a symmetric positive definite mass matrix. HMC starts by initializing parameters, $W^{(0)} \sim \mathcal{N}(0, \frac{1}{\tau})$, by sampling from the Gaussian distribution of prior precision τ. At HMC iteration n, the parameters are initialized to $W^{(n)}(0) = W^{(n)}$ while the momentum is initialized by sampling from the normal distribution $P_W^{(n)}(0) \sim \mathcal{N}(0, M)$ of variance defined by the mass matrix. The leapfrog iterative algorithm is then used to simulate system dynamics for time $L\Delta t$, where L is the number of leapfrog steps and Δt the step size. In each step, the leapfrog algorithm alternates between momentum and position updates, using

$$P_W^{(n)}\left(t + \frac{\Delta t}{2}\right) = P_W^{(n)}(t) - \frac{\Delta t}{2} \nabla U(W)|_{W=W^{(n)}(t)} \tag{F.0.0.2}$$

$$W^{(n)}(t + \Delta t) = W^{(n)}(t) + \Delta t M^{-1} P_W^{(n)}\left(t + \frac{\Delta t}{2}\right) \tag{F.0.0.3}$$

$$P_W^{(n)}(t + \Delta t) = P_W^{(n)}\left(t + \frac{\Delta t}{2}\right) - \frac{\Delta t}{2} \nabla U(W)|_{W=W^{(n)}(t+\Delta t)}, \tag{F.0.0.4}$$

so as to solve Hamilton's equations

$$\frac{dW}{dt} = \frac{\partial \mathcal{H}}{\partial P_W} \tag{F.0.0.5}$$

$$\frac{dP_W}{dt} = \frac{\partial \mathcal{H}}{\partial W}. \tag{F.0.0.6}$$

Since the leapfrog iterative algorithm is a discretized numerical approximation to the true integral, the limit $\Delta t \to 0$ would be needed to solve Hamilton's equations exactly. To correct for bad proposals from the leapfrog method, an HMC Metropolis-Hastings acceptance ratio is defined as

$$\alpha_{\mathrm{HMC}}(W^{(n)}(0), W^{(n)}(L\Delta t)) = \min\left\{1, \frac{\exp[-\mathcal{H}(W^{(n)}(L\Delta t), P_W^{(n)}(L\Delta t))]}{\exp[-\mathcal{H}(W^{(n)}(0), P_W^{(n)}(0))]}\right\}, \tag{F.0.0.7}$$

where $\mathcal{H}$ is the Hamiltonian defined in (F.0.0.1). In result, the parameter sample returned by iteration n, which is also the parameter initialization of iteration $n+1$, is

$$W^{(n+1)}(0) | W^{(n)}(0), W^{(n)}(L\Delta t) = \begin{cases} W^{(n)}(L\Delta t), & \text{with probability } \alpha_{\mathrm{HMC}}(W^{(n)}(0), W^{(n)}(L\Delta t)) \\ W^{(n)}(0), & \text{otherwise} \end{cases}. \tag{F.0.0.8}$$

This process is repeated for T' HMC iterations. Since the prior weight initialization does not usually lie within the typical set of the distribution, the samples initially output by HMC are poor indicators of the typical set region. To address this problem, a burn-in period of B initial iterations is defined and all burn-in samples are discarded after the algorithm is complete.

References

[1] Eugene C. Lin. “Radiation Risk From Medical Imaging”. In: *Mayo Clinic Proceedings* 85.12 (2010), pp. 1142–1146.

[2] Eugenio Picano. “Sustainability of Medical Imaging”. In: *Bmj* 328.7439 (2004), pp. 578–580.

[3] Amy Berrington de Gonzalez and Sarah Darby. “Risk of Cancer from Diagnostic X-Rays: Estimates for the UK and 14 Other Countries”. In: *The Lancet* 363.9406 (2004), pp. 345–351.

[4] Saiprasad Ravishankar, Jong Chul Ye, and Jeffrey A. Fessler. “Image Reconstruction: From Sparsity to Data-Adaptive Methods and Machine Learning”. In: *Proceedings of the IEEE* 108.1 (2020), pp. 86–109.

[5] L. A. Feldkamp, L. C. Davis, and J. W. Kress. “Practical Cone-Beam Algorithm”. In: *J. Opt. Soc. Am. A* 1.6 (1984), pp. 612–619.

[6] Anders H. Andersen. “Algebraic Reconstruction in CT from Limited Views”. In: *IEEE Transactions on Medical Imaging* 8.1 (1989), pp. 50–55.

[7] David L. Donoho. “Compressed Sensing”. In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306.

[8] Alexandra Shchukina et al. “Pitfalls in Compressed Sensing Reconstruction and How to Avoid Them”. In: *Journal of Biomolecular NMR* 68.2 (2017), pp. 79–98.

[9] Tim W. Nattkemper and Axel Wismüller. “Tumor Feature Visualization with Unsupervised Learning”. In: *Medical Image Analysis* 9.4 (2005), pp. 344–351.

[10] Khalid Raza and Nripendra Kumar Singh. “A Tour of Unsupervised Deep Learning for Medical Image Analysis”. In: *arXiv preprint arXiv:1812.07715* (2018).

[11] Dinggang Shen, Guorong Wu, and Heung-Il Suk. “Deep Learning in Medical Image Analysis”. In: *Annual Review of Biomedical Engineering* 19 (2017), pp. 221–248.

[12] Ge Wang et al. “Image Reconstruction is a New Frontier of Machine Learning”. In: *IEEE transactions on Medical Imaging* 37.6 (2018), pp. 1289–1296.

[13] Quanshi Zhang and Song-Chun Zhu. “Visual Interpretability for Deep Learning: A Survey”. In: *arXiv preprint arXiv:1802.00614* (2018).

[14] Biraja Ghoshal and Allan Tucker. “Estimating Uncertainty and Interpretability in Deep Learning for Coronavirus (COVID-19) Detection”. In: *arXiv preprint arXiv:2003.10769* (2020).

[15] Feng-Lei Fan et al. “On Interpretability of Artificial Neural Networks: A Survey”. In: *IEEE Transactions on Radiation and Plasma Medical Sciences* (2021).

[16] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–421.

[17] Michael Niemeyer et al. “Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3504–3515.

[18] Shunsuke Saito et al. “PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2304–2314.

[19] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. “Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations”. In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 1121–1132.

[20] Zhiqin Chen and Hao Zhang. “Learning Implicit Fields for Generative Shape Modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948.

[21] Boyang Deng et al. “NASA: Neural Articulated Shape Approximation”. In: *arXiv preprint arXiv:1912.03207* (2019).

[22] Kyle Genova et al. “Learning Shape Templates with Structured Implicit Functions”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7154–7164.

[23] Kyle Genova et al. “Local Deep Implicit Functions for 3D Shape”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4857–4866.

[24] Chiyu Jiang et al. “Local Implicit Grid Representations for 3D Scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6001–6010.

[25] Jeong Joon Park et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[26] Philipp Henzler, Niloy J. Mitra, and Tobias Ritschel. “Learning a Neural 3d Texture Space from 2D Exemplars”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8356–8364.

[27] Michael Oechsle et al. “Texture Fields: Learning Texture Representations in Function Space”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4531–4540.

[28] Yu Sun et al. “CoIL: Coordinate-based Internal Learning for Imaging Inverse Problems”. In: *arXiv preprint arXiv:2102.05181* (2021).

[29] Albert W. Reed et al. “Dynamic CT Reconstruction from Limited Views with Implicit Neural Representations and Parametric Motion Fields”. In: *arXiv preprint arXiv:2104.11745* (2021).

[30] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. “Active Learning with Statistical Models”. In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 129–145.

[31] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *arXiv preprint arXiv:1612.01474* (2016).

[32] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *International Conference on Machine Learning*. PMLR. 2016, pp. 1050–1059.

[33] Charles Blundell et al. “Weight Uncertainty in Neural Network”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1613–1622.

[34] Simon Duane et al. “Hybrid Monte Carlo”. In: *Physics Letters B* 195.2 (1987), pp. 216–222.

[35] Radford M. Neal et al. “MCMC using Hamiltonian Dynamics”. In: *Handbook of Markov Chain Monte Carlo* 2.11 (2011), p. 2.

[36] Michael Betancourt. “A Conceptual Introduction to Hamiltonian Monte Carlo”. In: *arXiv preprint arXiv:1701.02434* (2017).

[37] Toufique A. Soomro et al. “Artificial Intelligence (AI) for Medical Imaging to Combat Coronavirus Disease (COVID-19): A Detailed Review with Direction for Future Research”. In: *Artificial Intelligence Review* (2021), pp. 1–31.

[38] David J. Brenner and Eric J. Hall. “Computed Tomography — An Increasing Source of Radiation Exposure”. In: *New England Journal of Medicine* 357.22 (2007), pp. 2277–2284.

[39] Care Quality Commission et al. “Radiology Review: A National Review of Radiology Reporting within the NHS in England”. In: *Care Quality Commission* (2018).

[40] Amy Berrington De González et al. “Projected Cancer Risks from Computed Tomographic Scans Performed in the United States in 2007”. In: *Archives of Internal Medicine* 169.22 (2009), pp. 2071–2077.

[41] C.A. Roobottom, G. Mitchell, and G. Morgan-Hughes. “Radiation-Reduction Strategies in Cardiac Computed Tomographic Angiography”. In: *Clinical Radiology* 65.11 (2010), pp. 859–867.

[42] Jennifer C O’Daniel, Donna M Stevens, and Dianna D Cody. “Reducing radiation exposure from survey CT scans”. In: *American Journal of Roentgenology* 185.2 (2005), pp. 509–515.

[43] Thomas R Goodman, Adel Mustafa, and Erin Rowe. “Pediatric CT radiation exposure: where we were, and where we are now”. In: *Pediatric radiology* 49.4 (2019), pp. 469–478.

[44] Rebecca Smith-Bindman et al. "Radiation Dose Associated with Common Computed Tomography Examinations and the Associated Lifetime Attributable Risk of Cancer". In: *Archives of Internal Medicine* 169.22 (2009), pp. 2078–2086.

[45] J.H. Hubbell and S.M. Seltzer. *Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients.* (version 1.4) Availabile: `http://physics.nist.gov/xaamdi` [2021, 08, 22]. Originally published as NISTIR 5632, National Institute of Standards and Technology, Gaithersburg, MD (1995). National Institute of Standards and Technology, Gaithersburg, MD, 2004.

[46] M.J. Berger et al. *ESTAR, PSTAR, and ASTAR: Computer Programs for Calculating Stopping-Power and Range Tables for Electrons, Protons, and Helium Ions.* (version 1.2.3) Availabile: `http://physics.nist.gov/Star` [2021, 08, 22]. Originally published as: Berger, M.J., NISTIR 4999, National Institute of Standards and Technology, Gaithersburg, MD (1993). National Institute of Standards and Technology, Gaithersburg, MD, 2004.

[47] Jiang Hsieh. *Computed Tomography: Principles, Design, Artifacts, and Recent Advances.* Vol. 114. SPIE Press, 2003.

[48] Stanley R. Deans. *The Radon Transform and Some of Its Applications.* Courier Corporation, 2007.

[49] Sigurdur Helgason. *Groups & Geometric Analysis: Radon Transforms, Invariant Differential Operators and Spherical Functions: Volume 1.* Academic Press, 1984.

[50] Ronald N Bracewell. "Strip Integration in Radio Astronomy". In: *Australian Journal of Physics* 9.2 (1956), pp. 198–217.

[51] Nikos Paragios, Mikael Rousson, and Visvanathan Ramesh. "Matching Distance Functions: A Shape-To-Area Variational Approach for Global-To-Local Registration". In: *European Conference on Computer Vision.* Springer. 2002, pp. 775–789.

[52] Mikael Rousson and Nikos Paragios. "Shape Priors for level Set Representations". In: *European Conference on Computer Vision.* Springer. 2002, pp. 78–92.

[53] Tony Chan and Wei Zhu. "Level Set Based Shape Prior Segmentation". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).* Vol. 2. 2005, pp. 1164–1170.

[54] Lars Mescheder et al. "Occupancy Networks: Learning 3D Reconstruction in Function Space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 4460–4470.

[55] Matan Atzmon and Yaron Lipman. "SAL: Sign Agnostic Learning of Shapes from Raw Data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 2565–2574.

[56] Mateusz Michalkiewicz et al. "Implicit Surface Representations as Layers in Neural Networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 4743–4752.

[57] Amos Gropp et al. "Implicit Geometric Regularization for Learning Shapes". In: *arXiv preprint arXiv:2002.10099* (2020).

[58] Songyou Peng et al. "Convolutional Occupancy Networks". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16.* Springer. 2020, pp. 523–540.

[59] Rohan Chabra et al. "Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction". In: *European Conference on Computer Vision.* Springer. 2020, pp. 608–625.

[60] Matthew Tancik et al. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in Neural Information Processing Systems.* Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 7537–7547.

[61] Frank Dellaert and Lin Yen-Chen. "Neural Volume Rendering: NeRF And Beyond". In: *arXiv preprint arXiv:2101.05204* (2020).

[62] Ricardo Martin-Brualla et al. "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021, pp. 7210–7219.

[63] Vincent Sitzmann et al. "Implicit Neural Representations with Periodic Activation Functions". In: *Advances in Neural Information Processing Systems* 33 (2020).

[64] Matthew Tancik et al. “Learned Initializations for Optimizing Coordinate-Based Neural Representations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2846–2855.

[65] Vincent Sitzmann et al. “MetaSDF: Meta-learning Signed Distance Functions”. In: *arXiv preprint arXiv:2006.09662* (2020).

[66] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. “SDF-SRN: Learning Signed Distance 3D Object Reconstruction from Static Images”. In: *arXiv preprint arXiv:2010.10505* (2020).

[67] David B. Lindell, Julien NP Martel, and Gordon Wetzstein. “AutoInt: Automatic Integration for Fast Neural Volume Rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14556–14565.

[68] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or Epistemic? Does It Matter?” In: *Structural Safety* 31.2 (2009), pp. 105–112.

[69] Alex Kendall and Yarin Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *arXiv preprint arXiv:1703.04977* (2017).

[70] Andrew Gordon Wilson and Pavel Izmailov. “Bayesian Deep Learning and a Probabilistic Perspective of Generalization”. In: *arXiv preprint arXiv:2002.08791* (2020).

[71] Siddhartha Chib and Edward Greenberg. “Understanding the Metropolis-Hastings Algorithm”. In: *The american statistician* 49.4 (1995), pp. 327–335.

[72] Walter R. Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov Chain Monte Carlo in Practice*. CRC Press, 1995.

[73] Pavel Izmailov et al. “What Are Bayesian Neural Network Posteriors Really Like?” In: *arXiv preprint arXiv:2104.14421* (2021).

[74] Jeremy Nixon, Balaji Lakshminarayanan, and Dustin Tran. “Why Are Bootstrapped Deep Ensembles Not Better?” In: *”I Can’t Believe It’s Not Better!”NeurIPS 2020 workshop*. 2020.

[75] Sheheryar Zaidi et al. “Neural Ensemble Search for Uncertainty Estimation and Dataset Shift”. In: *arXiv preprint arXiv:2006.08573* (2020).

[76] Lawrence A Shepp and Benjamin F Logan. “The Fourier Reconstruction of a Head Section”. In: *IEEE Transactions on Nuclear Science* 21.3 (1974), pp. 21–43.

[77] Daniël M Pelt et al. “Integration of TomoPy and the ASTRA Toolbox for Advanced Processing and Reconstruction of Tomographic Synchrotron Data”. In: *Journal of Synchrotron Radiation* 23.3 (2016), pp. 842–849.

[78] Chuan Guo et al. “On Calibration of Modern Neural Networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1321–1330.

[79] Jeremy Nixon et al. “Measuring Calibration in Deep Learning”. In: *CVPR Workshops*. Vol. 2. 7. 2019.

[80] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. “Accurate Uncertainties for Deep Learning using Calibrated Regression”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2796–2804.

[81] Chuan Guo et al. “On Calibration of Modern Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1321–1330.

[82] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[83] Piero Esposito. *BLiTZ - Bayesian Layers in Torch Zoo (a Bayesian Deep Learing library for Torch)*. `https://github.com/piEsposito/blitz-bayesian-deep-learning/`. 2020.

[84] Adam D Cobb and Brian Jalaian. “Scaling Hamiltonian Monte Carlo Inference for Bayesian Neural Networks with Symmetric Splitting”. In: *Uncertainty in Artificial Intelligence* (2021).

[85] Omry Yadan. *Hydra - A Framework for Elegantly Configuring Complex Applications*. Github. 2019. URL: `https://github.com/facebookresearch/hydra`.

[86] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: `https://www.wandb.com/`.

[87] Andy B. Yoo, Morris A. Jette, and Mark Grondona. “SLURM: Simple Linux Utility for Resource Management”. In: *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer. 2003, pp. 44–60.

[88] Matthew D. Hoffman, Andrew Gelman, et al. “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo”. In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1593–1623.

[89] Andrew Gelman and Donald B. Rubin. “Inference from Iterative Simulation using Multiple Sequences”. In: *Statistical Science* 7.4 (1992), pp. 457–472.

[90] Geoffrey Hinton and Sam T Roweis. “Stochastic neighbor embedding”. In: *NIPS*. Vol. 15. Citeseer. 2002, pp. 833–840.

[91] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[92] Xiaochuan Pan, Emil Y. Sidky, and Michael Vannier. “Why Do Commercial CT Scanners Still Employ Traditional, Filtered Back-Projection for Image Reconstruction?” In: *Inverse Problems* 25.12 (2009), p. 123009.

[93] S.E. Pryse et al. “Tomographic Imaging of the ionospheric Mid-Latitude Trough”. In: *Annales Geophysicae*. Vol. 11. 2-3. 1993, pp. 144–149.

[94] Gary S. Bust and Cathryn N. Mitchell. “History, Current State, and Future Directions of Ionospheric Imaging”. In: *Reviews of Geophysics* 46.1 (2008).

[95] Richard Gordon, Robert Bender, and Gabor T. Herman. “Algebraic Reconstruction Techniques (ART) for Three-Dimensional Electron Microscopy and X-Ray Photography”. In: *Journal of Theoretical Biology* 29.3 (1970), pp. 471–481.

[96] Anders H. Andersen and Avinash C. Kak. “Simultaneous Algebraic Reconstruction Technique (SART): A Superior Implementation of the ART Algorithm”. In: *Ultrasonic Imaging* 6.1 (1984), pp. 81–94.

[97] Jin-Yun Yuan and Alfredo Noel Iusem. “Preconditioned Conjugate Gradient Method for Generalized Least Squares Problems”. In: *Journal of Computational and Applied Mathematics* 71.2 (1996), pp. 287–297.

[98] Bao-Yu Dong. “Image Reconstruction using EM Method in X-Ray CT”. In: *2007 International Conference on Wavelet Analysis and Pattern Recognition*. Vol. 1. IEEE. 2007, pp. 130–134.

[99] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[100] David E. Rumelhart et al. “Backpropagation: The Basic Theory”. In: *Backpropagation: Theory, Architectures and Applications* (1995), pp. 1–34.

[101] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[102] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25 (2012), pp. 1097–1105.

[103] Jia Deng et al. “Imagenet: A Large-Scale Hierarchical Image Database”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[104] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[105] Steffen Schneider et al. “wav2vec: Unsupervised Pre-Training for Speech Recognition”. In: *arXiv preprint arXiv:1904.05862* (2019).

[106] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[107] Ian Goodfellow et al. “Generative Adversarial Networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[108] David Silver et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016), pp. 484–489.

[109] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks are Universal Approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

[110] George Cybenko. “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314.

[111] Kurt Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257.

[112] Zhou Lu et al. “The Expressive Power of Neural Networks: A View from the Width”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[113] Boris Hanin. “Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations”. In: *Mathematics* 7.10 (2019).

[114] Patrick Kidger and Terry Lyons. “Universal Approximation with Deep Narrow Networks”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 2306–2327.

[115] Zhou Lu et al. “The Expressive Power of Neural Networks: A View from the Width”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6232–6240.

[116] Maithra Raghu et al. “On the Expressive Power of Deep Neural Networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2847–2854.

[117] Behnam Neyshabur et al. “Towards Understanding the Role of Over-Parametrization in Generalization of Neural Networks”. In: *arXiv preprint arXiv:1805.12076* (2018).

[118] Arthur Jacot, Franck Gabriel, and Clément Hongler. “Neural Tangent Kernel: Convergence and Generalization in Neural Networks”. In: *arXiv preprint arXiv:1806.07572* (2018).

[119] Ali Rahimi, Benjamin Recht, et al. “Random Features for Large-Scale Kernel Machines”. In: *NIPS*. Vol. 3. 4. Citeseer. 2007, p. 5.

[120] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. “Analysis of Function of Rectified Linear Unit used in Deep Learning”. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, pp. 1–8.

[121] Lu Lu et al. “Dying ReLU and Initialization: Theory and Numerical Examples”. In: *arXiv preprint arXiv:1903.06733* (2019).

[122] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[123] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. “Searching for Activation Functions”. In: *arXiv preprint arXiv:1710.05941* (2017).

[124] Charles Dugas et al. “Incorporating Second-Order Functional Knowledge for Better Option Pricing”. In: *Advances in Neural Information Processing Systems* (2001), pp. 472–478.

[125] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[126] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[127] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[128] David R. Bull. “Chapter 4 - Digital Picture Formats and Representations”. In: *Communicating Pictures*. Ed. by David R. Bull. Oxford: Academic Press, 2014, pp. 99–132.