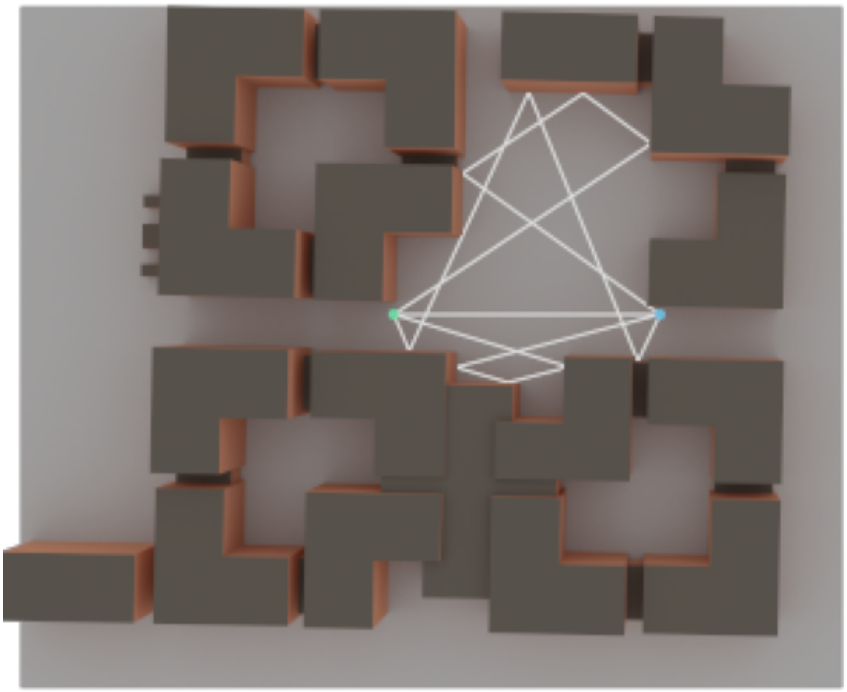
Interpolating Channel Gains Using Fourier Neural Operators

Semester Project
MatTek8 2.109b



Aalborg University
Mathematics-Technology

This report is written in the typesetting language LaTeX using the text editor Overleaf. Experiments and data processing are conducted using Python 3.11.8. Illustrations are created using the TikZ package in LaTeX or Python 3.11.8. References are cited using the IEEE method.

Department of Mathematical Sciences
Skjernvej 4A
DK - 9220 Aalborg
http://www.math.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Interpolating Channel Gains Using Fourier Neural Operators

Theme:
Signal/data processing systems

Project Period:
Spring Semester 2024

Project Group:
MatTek8 2.109b

Participants:
Stine Byrjalsen
Christian Dausel Jensen
Laurits Randers
Julie Timmermann Werge

Supervisors:
Martin Voigt Vejling
Christophe Biscio
Petar Popovski

Page Numbers: 66

Date of Completion:
May 21, 2024

Abstract:

This project aims to investigate the use of Fourier neural operators to interpolate channel gains and explore whether the operator is discretisation-invariant. Data simulation is performed using Sionna Ray Tracing in a scene resembling a real location in Aalborg. To test the performance of the Fourier neural operator, it has been compared to a baseline model, which was found to outperform the Fourier neural operator. A sensitivity analysis, involving the addition of Gaussian noise to the input, was conducted to assess the model's robustness to noise. It revealed that the Fourier neural operator is sensitive to noise. To investigate the Fourier neural operator's discretisation-invariance, either the output or input resolution was varied while keeping the other fixed. Results indicated some discretisation-invariance for the output resolution but not for the input resolution. Thus, the main conclusion is that while the Fourier neural operator displayed promising characteristics, it was outperformed by the baseline and exhibited limitations in noise sensitivity and discretisation-invariance.

Preface

This project is written by group 2.109b in the 8th semester 2024, studying mathematics and technology at the Department of Mathematical Sciences at Aalborg University. The project is written from February 1, 2024, to May 21, 2024. The topic of the project is *Interpolating Channel Gains Using Fourier Neural Operators*.

The Fourier neural operator has been implemented using Python 3.11.8 and trained on an NVIDIA GeForce RTX 4070 SUPER GPU. Synthetic data used for training, validating, and testing the Fourier neural operator has been generated through the creation of a scene in Blender version 3.6.9.

The group would like to thank the three supervisors, Martin Voigt Vejling, Christophe Biscio, and Petar Popovski for their guidance throughout the project period.

Aalborg University, May 21, 2024

Stine Byrjalsen
sbyrja20@student.aau.dk

Christian Dausel Jensen
cdje20@student.aau.dk

Laurits Randers
lrande18@student.aau.dk

Julie Timmermann Werge
jwerge20@student.aau.dk

Nomenclature

Acronym	Meaning
CDI	Channel Distribution Information
FNO	Fourier Neural Operator
LOS	Line-Of-Sight
NLOS	Non-Line-Of-Sight
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
RT	Ray Tracing
GELU	Gaussian Error Linear Unit
ReLU	Rectified Linear Unit

Notation	Description
V	Vector space
$\mathbb{F}$	Field that is either $\mathbb{R}$ or $\mathbb{C}$
$\mathbb{Q}$	The set of rational numbers
$\mathbf{0}$	Zero vector
$\mathcal{P}$	Set of polynomials
$\|\cdot\|$	Norm
$\mathcal{L}$	The set of linear operators
L^p	The space of Lebesgue measurable functions with the L_p-norm
$\mathbb{R}^+$	The set of non-negative real numbers
$\mathcal{F}$	Fourier transform
$\mathcal{F}^{-1}$	Inverse Fourier transform
$G^\dagger$	Solution operator
i	Imaginary number; $\sqrt{-1}$
$\mathcal{N}$	Normal distribution
$\mathbb{E}$	Expected value
$\boldsymbol{\nabla}$	Del operator

Contents

1 Introduction

In communication systems, the signal received at a certain location is determined by the transmitted signal and the communication channel between the transmitter and the receiver [33, p. 55]. Within wireless communication channels, the signal can propagate along multiple paths, which can be either direct or indirect, depending on whether the signal is reflected by objects. As the signal travels from transmitter to receiver, it can be subject to phenomena such as reflection, diffraction, scattering, and diffusion, depending on the characteristics of the transmission environment. This gives rise to multipath propagation, which alters the phase and amplitude of the received signals [18, p. 3] [10, p. 27]. These alterations, along with path loss, shadowing, noise, and interference, contribute to the random behaviour of a wireless channel [8, p. 1478]. This random behaviour of the wireless channel impacts the reliability of the received signal.

Obtaining measurements for all characteristics of a communication channel at every location of interest can be expensive and, in some cases, impossible. Therefore, it is desirable to estimate channel characteristics at locations without conducting actual measurements [39, p. 1]. However, the random variations make accurate estimation and characterisation of the channel properties in a wireless communication system challenging. Therefore, to ensure reliable and efficient communication performance, these technical challenges must be addressed [10, p. 7]. Maxwell's equations have been historically used as the foundation for understanding radio wave propagation. Computational limitations, however, restricted their application to simple environments. The development of finite-element analysis and ray-tracing techniques made it possible to approximate the solutions of Maxwell's equations in more complex environments. However, their computational complexity and dependency on 3D modules of all objects limit their applicability. Therefore, to address the challenges of characterising the propagation of radio waves across space, the concept of radio map estimation was proposed [33, p. 54].

Radio map estimation is the process of constructing a map that represents the distribution of wireless communication channel parameters across a given area. This involves utilising spatially distributed measurements from sensors alongside their locations and applying interpolation techniques to estimate channel parameters at locations where direct measurements are unavailable. This approach does not require physical modelling of the propagation environment, thus serving as a data-driven alternative to the earlier mentioned techniques [33, pp. 53–54]. Recent papers have used deep learning algorithms, such as convolutional neural networks [25] [39], deep neural networks [23], and transfer learning [17], to estimate radio maps.

Understanding radio maps and how they can be estimated is essential for comprehending the challenges associated with interpolating the channel characteristics of points in the channel where no measurements are available. An introduction to radio maps will therefore be provided, along with related work and applications for it.

1.1 Radio Maps

A radio map is a spatial representation of the characteristics of a wireless communication channel across a geographical area. The characteristics can be the received signal power, power spectral density, or channel gain [33, p. 53] [35, p. 1].

Types of Radio Maps

There are different types of radio maps, depending on the application and the characteristics of the wireless environment. The most common types of radio maps are based on the received signal strength or the propagation channel effects, called signal strength maps and propagation maps, respectively [33, p. 55]. Some selected signal strength maps are presented in Table 1.1.

Type of Map	**Example of Application**
Coverage map	Locate coverage holes
Power map	Locate areas with high interference
Power spectral density map	Analyse frequency usage

Table 1.1: Three selected types of the signal strength maps.

A coverage map provides insights into the coverage areas of wireless networks, identifying areas with weak or no coverage and allowing for the determination of suitable locations for deploying new base stations. A power map is used to map the transmitted power levels, providing information about coverage patterns, signal propagation properties, and possible areas of interference. By far, power maps are the radio maps that garnered the most interest [34, p. 55]. A power spectral density map distributes the power across the frequency domain, as opposed to power maps that just obtain it across space. This helps to understand the frequency-domain characteristics of the wireless environment [33, pp. 55–56].

Radio maps that only focus on the channel are called propagation maps. Each specific aspect of the propagation environment gives rise to a different kind of propagation map. It could, for example, be the channel gain. Channel gain maps use the transmitter and receiver locations to provide the channel gain. With a channel gain map and information about the locations and transmit powers within a specific area, it is possible to obtain a power map [33, p. 56].

Changes in the locations and transmit powers affect only signal strength maps; however, alterations in the scattering environment, such as the construction of new buildings, affect both propagation and signal strength maps [33, p. 56].

Radio maps can also leverage statistical knowledge of the communication channel by incorporating spatial correlations of channel statistics based solely on location. This is known as a channel distribution information (CDI) radio map [19, p. 1].

Estimating Radio Maps

Radio map estimation is based on a collection of measurements acquired by spatially distributed sensors along with their locations. These sensors capture metrics such as signal strength and propagation patterns. Interpolation techniques are then employed to address spatial gaps in the dataset and generate a complete map. Estimating radio maps can help adapt to variables such as signal strength, transmission speed, and other factors affecting wireless communication performance [18, p. 4].

Numerous approaches exist for estimating radio maps, roughly grouped into two categories: data-driven methods and model-based methods [33, p. 54]. Data-driven methods rely on empirical measurements collected from specific locations to estimate values at non-measured locations. This is done by using signal processing approaches such as Kriging, K-nearest neighbours, tensor completion, support vector regression, or matrix completion [25, p. 2]. Kriging is a common spatial interpolation technique that is extensively applied to radio map estimation [23, p. 1]. Data-driven methods are simple to implement and computationally efficient; however, they may not always capture the full range of environmental factors that influence signal propagation [15, p. 1]. Model-based methods use mathematical models of the transmission environment to predict channel characteristics at different locations. These models may include ray-tracing simulations or the dominant path model [25, p. 3]. Model-based methods are computationally costly and complex to implement since they rely on the accuracy of available maps representing the environment; however, they generalise well to new scenarios [15, p. 1].

A common method for estimating radio maps involves the use of machine learning, as it can be used either as a data-driven method, a model-based method, or a combination of the two. This method is often more accurate than the data-driven methods and more computationally efficient than the model-based methods [15, p. 1]. However, it requires a sufficient amount of training data and can demand a large amount of time during the training process [16, p. 1]. Moreover, due to variations in channel characteristics across different geographical areas, re-training the network is necessary for each specific area.

Methods employing deep learning have intensified the work within the field of estimating radio maps [33, p. 55], with many approaches leveraging deep neural networks [23]. Specifically, convolutional neural networks [39], alongside architectures like UNet [25] [16]. Transfer learning has also emerged as a valuable tool [17], as it can help tackle the issues of developing a new radio map for each new geographical area. Transfer learning does that by using the knowledge acquired in the source environment for the learning task and using it in the target environment when both the source and target environments are sufficiently similar. Transfer learning can also mitigate the need for extensive data by pre-training models on simulated data and fine-tuning them with real-measured data [15, p. 1].

1.2 Applications and Related Work

Radio maps have numerous applications, including the improvement of cellular network coverage, localisation of user equipment using signal strength measurements, managing signal beams in advanced wireless technologies like 5G and 6G, network planning, autonomous vehicle path planning, aerial traffic management in unmanned aerial systems, fingerprinting localisation, and so forth [35, p. 1] [16, p. 1].

The paper by Kallehauge et al. [19] introduces CDI radio maps as an approach to estimating CDI based solely on location. The study validates the effectiveness of CDI maps in predicting small-scale fading statistics relevant to ultra-reliable low-latency communication. Their findings showed that the CDI radio map was able to meet the desired network reliability with high confidence, a crucial requirement in ultra-reliable low-latency communication.

The recent study [25] from 2020 has introduced a deep learning method for simulating radio maps in urban environments given city geometry and transmitter locations. The method they developed was called RadioUNet. Unlike previous approaches that require re-training for different city maps, RadioUNet learns to approximate the outcome of the underlying physical phenomenon, enabling it to produce radio maps for any given transmitter location and city map. The method outperforms previous deep learning approaches at that time and achieves good performance in terms of both accuracy and runtime. The study also developed a new dataset called RadioMapSeer for others to use in the development of deep learning methods for path loss prediction.

The introduction to UNet models leads to the new study [16] from 2024 that aims to present a new radio map dataset generated through ray-tracing simulations based on city geometries derived from real-world urban areas. In contrast to existing datasets, the dataset in this study is the first to feature directional antennas at the transmitter and to include vegetation and realistic building heights with approximated roof shapes. The study also added aerial images of the same locations in order to train models for situations where exact knowledge of the city geometry is not available, thus enabling the prediction of radio maps from images. The study actually makes a comparison to the RadioUNet dataset, which is only simulated in 2D with buildings as the only environmental data.

A deep learning algorithm based on spatial signal strength prediction is proposed in [23]. The method differs from previous approaches in three aspects: it doesn't require transmitter location knowledge, it avoids dependency on the knowledge of shadowing and pathloss model parameters, and it integrates 3D environment mapping. The 3D maps allow the algorithm to predict scattering and blockage due to buildings. The study demonstrates that deep learning can be used to predict radio signal propagation in urban settings. This leads to the use of radio maps and 3D models in the prediction of blockages in wireless communication systems. Blockages, caused by obstacles such as buildings, walls, and vegetation, cause variations in the transmitted signals. Thus, blockage prediction helps to identify areas where signal blockages are likely to occur, which allows for more accurate modelling of signal propagation and better predictions of signal strengths in the presence of obstacles. A recent study on

blockage prediction in millimetre-wave and terahertz communication links employed meta-learning and recurrent neural networks to predict blockages earlier and with fewer observed data samples [20]. The study highlights the importance of proactive countermeasures to reduce the impact of blockages, such as implementing reactive or preventive mechanisms in communication systems. Given the absence of mathematical models for blockages, predictors are often based on machine learning models, which rely on large amounts of training data. To address this problem, the study uses meta-learning, which enables neural networks to be trained for new deployments with significantly fewer samples than typically needed.

Since countless approaches exist for estimating radio maps, a recent paper by Shrestha et al. compared the performance of eight radio map estimators using real data collected with an unmanned aerial vehicle [35]. The compared algorithms include three traditional estimators, namely K-nearest neighbour, Kriging, and kernel ridge regression; four deep neural networks, where two of them are the UNet algorithms from [25] and [23]; and a hybrid estimator proposed by the authors themselves. They considered two power map estimation problems and found that the traditional estimators, such as K-nearest neighbours and Kriging, had surprisingly good overall performance. They also found that the traditional estimators outperform the deep neural network estimators in situations where the amount of training data is insufficient. However, since each estimator excels in specific situations, the study came to the conclusion that no single estimator consistently outperforms the others. The authors also proposed an algorithm that combines deep neural networks and traditional estimators. The result was that the hybrid estimator outperformed all the other estimators in nearly all cases.

1.3 Problem Statement

In this project, the focus is on estimating a radio map by interpolating channel characteristics in locations where direct measurements are unavailable. We have decided to focus on interpolating the channel gain since it can provide information about how the signal travels between the transmitter and receiver. This is crucial in, among other, ultra-reliable low-latency communication [19].

Traditionally, interpolation techniques have relied on methods such as kernel-based learning or Kriging [34, p. 1], but recent algorithms in channel estimation and interpolation have relied on methods such as transfer learning [17], deep neural networks [23], and convolutional neural networks [25]. However, the rapid growth in deep learning has given rise to new classes of neural networks, such as the Fourier neural operator (FNO) [26]. This network is specifically designed to map between function spaces within a bounded open set. Such a neural network has the useful property known as discretisation-invariance. This means that FNOs can be trained on one discretisation and evaluated on an arbitrary discretisation, regardless of the discretisation of the input and output function spaces [27, p. 1]. FNOs have not yet been utilised to estimate radio maps. Hence, this motivates our project to use FNOs to predict the channel gain at desired locations within a geographical area.

This project therefore aims to answer:

How can a Fourier neural operator be utilised for interpolating channel gains, and can a discretisation-invariant operator be achieved?

1.4 Data

In this project, we want to consider both synthetic and real-measured data. The real-measured data was collected at Fredrik Bajers Vej 7, 9000 Aalborg. The dataset contains measured channel frequency responses obtained from a channel where a transmitter is systematically moved to 127 different locations while the receiver remains stationary. Thus, forming a uniform grid of equilateral triangles with a side length of 5m; see Figure 1.2. The measurements were done in a courtyard enclosed by buildings, creating a rich scattering environment. The receiver was positioned to provide a combination of line-of-sight (LOS) and non-line-of-sight (NLOS) channels during the measurements; see Figure 1.1.

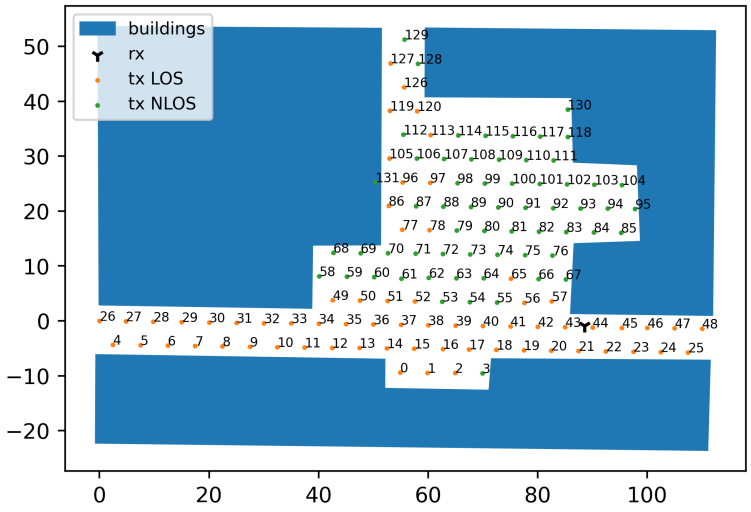


Figure 1.1: The red dots are the LOS scenarios, and the green dots are the NLOS scenarios. The abbreviations "tx" and "rx" stand for transmitter and receiver. The setup is provided by [19].

Measurements were conducted with and without a metal plate in front to evaluate the impact of blockage prediction on signal propagation. This is illustrated in Figure 1.2.

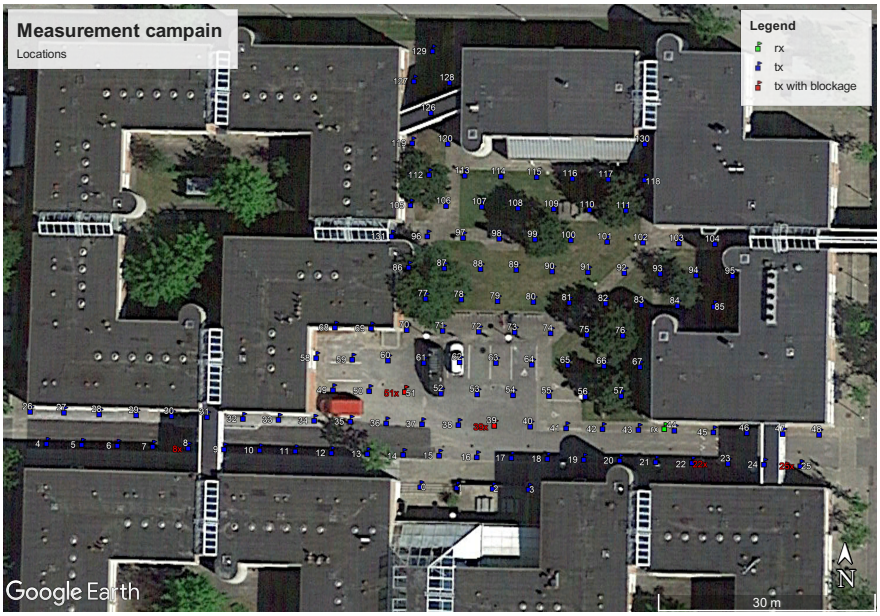


Figure 1.2: The green square is the location of the receiver, the blue squares are the location of the transmitter without the metal plate in front, and the red squares are the transmitter locations with a metal plate in front. The figure is from [19].

The measurements were conducted across a wide frequency range, spanning from 2 GHz to 16 GHz, with additional measurements from 28 GHz to 30 GHz. The measurements were taken with a 1 MHz interval. For more information on the components in the measurement system, see [19, pp. 3–4].

The synthetic data used in this project is generated using Sionna Ray Tracing (RT), a TensorFlow-based open-source library tailored for simulating the physical layer of wireless and optical communication systems [14, p. 1]. To generate the data, a 3D model of the area where the real measurements were taken is created and imported into Sionna RT [13, p. 2]. This will allow us to simulate the real measurement environment, which gives us a larger dataset to work with. It is important to note potential discrepancies between the synthetic and real-measured data, as variations in environmental factors, such as vehicles and trees, and material properties, can affect the signal propagation characteristics.

1.5 Project Delimitation

In this project, we will focus on the courtyard of Fredrik Bajers Vej 7, initially with the aim of incorporating real-measured data into the analysis. Hence, the synthetic data is generated on the basis of having to implement the real-measured data. However, the project will only rely on the synthetic data generated from simulated channel gains in the courtyard of Fredrik Bajers Vej 7. To generate the synthetic data, we will employ Sionna RT, a deterministic simulation method. To match the real-measured data, a systematically fixed set of transmitter positions within the courtyard serves as the measurement points. To generate diverse

datasets, potential receiver locations are then varied within a region of the courtyard. For each receiver position, we will simulate the channel gain for every transmitter location at both measurement and interpolation points.

As the aim of this project is to use FNOs to interpolate the channel gain between measured data points within a channel, it is reasonable to consider a simple starting point. Thus, the channel environment is assumed to be static. We also assume that environmental factors such as vehicles and trees are omitted. Furthermore, the medium through which signals propagate is assumed to be homogeneous. Additionally, only the reflection phenomena are considered, avoiding diffraction, scattering, and diffusion. Furthermore, the waves are limited to one reflection with objects in the scene. Finally, our model is assumed to be a narrowband model with a frequency of 2 GHz.

2 Operators in Banach Spaces

This chapter serves as an introduction to operator theory which is a prerequisite for comprehending FNOs. Banach spaces are introduced as they provide a framework for studying and understanding the properties of infinite-dimensional vector spaces. Subsequently, an overview of general operator theory is provided.

2.1 Vector Spaces

A vector space is a collection of elements, called vectors, which can be added together and multiplied by scalars, while still satisfying certain conditions. Vector spaces are fundamental mathematical structures used to study the properties of vectors and their algebraic operations. The notation $\mathbb{F}$ denotes a field and can be replaced with either $\mathbb{R}$ or $\mathbb{C}$.

Definition 2.1 (Vector Space)
A vector space is a set V along with an addition $+$ on V and a scalar multiplication $\cdot$ on V such that the following properties hold:

- Commutativity: $\mathbf{v} + \mathbf{u} = \mathbf{u} + \mathbf{v}$ for all $\mathbf{v}, \mathbf{u} \in V$.
- Associativity: $(\mathbf{v} + \mathbf{u}) + \mathbf{w} = \mathbf{v} + (\mathbf{u} + \mathbf{w})$ and $(ab)\mathbf{v} = a(b\mathbf{v})$ for all $\mathbf{v}, \mathbf{u}, \mathbf{w} \in V$ and for all $a, b \in \mathbb{F}$.
- Additive identity: There exists an element $0 \in V$ such that $\mathbf{v} + 0 = \mathbf{v}$ for all $\mathbf{v} \in V$.
- Additive inverse: For every $\mathbf{v} \in V$, there exists $\mathbf{w} \in V$ such that $\mathbf{v} + \mathbf{w} = 0$.
- Multiplicative identity: $1\mathbf{v} = \mathbf{v}$ for all $\mathbf{v} \in V$.
- Distributive properties: $a(\mathbf{v} + \mathbf{u}) = a\mathbf{v} + a\mathbf{u}$ and $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$ for all $a, b \in \mathbb{F}$ and all $\mathbf{v}, \mathbf{u} \in V$.

[2, p. 12]

The elements in a vector space can be functions. Proposition 2.2 states that a function space is a vector space [2, pp. 13–14]. A function space is a set of functions sharing certain properties or characteristics.

Proposition 2.2 (Function Space)
Let V be a vector space over $\mathbb{F}$ and $D \subseteq \mathbb{F}^n$ be a nonempty set. The set of functions from D to $\mathbb{F}$ is denoted as $V^D = \{f : D \to \mathbb{F}\}$. For $f, g \in V^D$ and $f + g \in V^D$, addition is defined as

$$(f + g)(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \quad \text{for } \mathbf{x} \in D,$$

and for $\lambda \in V$, $f \in V^D$, and $\lambda f \in V^D$, multiplication is defined as

$$(\lambda f)(\mathbf{x}) = \lambda f(\mathbf{x}) \quad \text{for } \mathbf{x} \in D.$$

Then V^D is a vector space over $\mathbb{F}$ with the additive identity $0 : D \to \mathbb{F}$ defined by

$$0(\mathbf{x}) = 0 \quad \text{for } \mathbf{x} \in D$$

and additive inverse $-f : D \to \mathbb{F}$ defined as

$$(-f)(\mathbf{x}) = -f(\mathbf{x}) \quad \text{for } \mathbf{x} \in D.$$

[2, pp. 13–14]

A finite-dimensional vector space is defined by a finite basis set. They have efficient computations and geometric interpretations, particularly in low-dimensional spaces like $\mathbb{R}^2$ and $\mathbb{R}^3$. Moreover, finite-dimensional vector spaces yield fundamental results in linear algebra, for example, the theorem stating that every operator on a finite-dimensional nonzero complex vector space has an eigenvalue [2, p. 132]. In addition, operations like matrix multiplication, solving linear systems, and computing determinants are well-defined in finite-dimensional spaces [2, p. 332]. Furthermore, the definition of a finite-dimensional vector space is required in order to define infinite-dimensional vector spaces. These spaces become relevant when we define FNOs, which operate between infinite-dimensional vector spaces.

Definition 2.3 (Finite-dimensional Vector Space)
A vector space is finite-dimensional if it is spanned by a finite set of vectors.

[2, p. 30]

For every positive integer n, it can be shown that $\mathbb{F}^n$ is a finite-dimensional vector space. Another example of a finite-dimensional vector space is the set of all polynomials in $\mathbb{F}$ at degree most m, denoted $\mathcal{P}_m(\mathbb{F})$, where m is a non-negative integer [2, pp. 30–31]. Let $p(x)$ and $q(x)$ be the two polynomials of degree at most m

$$p(x) = a_0 + a_1 x + \cdots + a_m x^m \quad \text{and} \quad q(x) = b_0 + b_1 x + \cdots + b_m x^m.$$

Then, $\mathcal{P}_m(\mathbb{F})$ is a vector space defined with the following addition and scalar multiplication

$$\begin{aligned} p(x) + q(x) &= (a_0 + b_0) + (a_1 + b_1)x + \cdots + (a_m + b_m)x^m, \\ \lambda p(x) &= (\lambda a_0) + (\lambda a_1)x + \cdots + (\lambda a_m)x^m, \end{aligned}$$

where $\lambda \in \mathbb{F}$.

Definition 2.4 (Infinite-dimensional Vector Space)
A vector space is called infinite-dimensional if it is not finite-dimensional. [2, p. 31]

The vector space that encompasses all polynomials over $\mathbb{F}$ is denoted $\mathcal{P}(\mathbb{F})$. Since no list spans $\mathcal{P}(\mathbb{F})$, it must be an infinite-dimensional vector space. Consider any list of elements of $\mathcal{P}(\mathbb{F})$. Let m be the highest degree of the polynomials in the list. It follows that any polynomial in the span of this list has a degree at most m. Thus, the polynomial z^{m+1} is not in the span of our list [2, p. 31]. Another example of an infinite-dimensional vector space is the function space of continuous functions defined on an interval $[a, b]$, denoted by $C([a, b])$. This space consists of all continuous functions $f : [a, b] \to \mathbb{F}$. The dimension of $C([a, b])$ is infinite because there are infinitely many linearly independent functions that can be used to span the space.

A vector space with a notion of distance called a metric between its elements is called a metric space.

Definition 2.5 (Metric Space)
Let V be a vector space on $\mathbb{F}$. A metric space (V, d) is a vector space along with a metric $d : V \times V \to \mathbb{R}$ such that the following conditions are satisfied for all $\mathbf{v}, \mathbf{u}, \mathbf{w} \in V$:

- Non-negativity: $d(\mathbf{v}, \mathbf{u}) \geq 0$, where $d(\mathbf{v}, \mathbf{u}) = 0$ if and only if $\mathbf{v} = \mathbf{u}$.
- Symmetry: $d(\mathbf{v}, \mathbf{u}) = d(\mathbf{u}, \mathbf{v})$.
- Triangle inequality: $d(\mathbf{v}, \mathbf{u}) \leq d(\mathbf{v}, \mathbf{w}) + d(\mathbf{w}, \mathbf{u})$.

[7, p. 86]

A metric space defines a distance in a vector space with specific properties. A normed vector space introduces a norm as a measure of magnitude for vectors within the space.

Definition 2.6 (Normed Vector Space)
Let V be a vector space over $\mathbb{F}$. A function $\|\cdot\| : V \to \mathbb{R}$ on V is called a norm if the following conditions are satisfied for all $\mathbf{v}, \mathbf{u} \in V$ and every scalar $\lambda \in \mathbb{F}$:

- Non-negativity: $\|\mathbf{v}\| \geq 0$.
- Positive definiteness: $\|\mathbf{v}\| = 0$ if and only if $\mathbf{v} = \mathbf{0}$.
- Absolute homogeneity: $\|\lambda\mathbf{v}\| = |\lambda|\|\mathbf{v}\|$.
- Triangle inequality: $\|\mathbf{v} + \mathbf{u}\| \leq \|\mathbf{v}\| + \|\mathbf{u}\|$.

A normed vector space is denoted $(V, \|\cdot\|)$. [7, p. 84]

From Definition 2.5 and Definition 2.6 it can be shown that the normed vector space $(V, \|\cdot\|)$ is a metric space with metric

$$d(\mathbf{v}, \mathbf{u}) = \|\mathbf{v} - \mathbf{u}\| \quad \text{for all } \mathbf{v}, \mathbf{u} \in V$$

[7, p. 86]. Convergence in a normed vector space is defined as convergence in norm.

Definition 2.7 (Convergence in Norm)
Let $\{\mathbf{v}_n\}_{n\in\mathbb{N}}$ be a sequence in $(V, \|\cdot\|)$. Then $\{\mathbf{v}_n\}_{n\in\mathbb{N}}$ is said to converge in norm to $\mathbf{v}$ if

$$\lim_{n\to\infty} \|\mathbf{v}_n - \mathbf{v}\| = 0.$$

[9, p. 72]

Convergence in norm reveals how sequences approach specific vectors in normed vector spaces. Cauchy sequences focus on how elements within a sequence become arbitrarily close to each other as the sequence progresses.

Definition 2.8 (Cauchy Sequence)
A sequence $\{\mathbf{v}_n\}_{n\in\mathbb{N}}$ in $(V, \|\cdot\|)$ is called a Cauchy sequence if

$$\forall\, \epsilon > 0\, \exists\, N \in \mathbb{N}\, \forall\, m, n \in \mathbb{N} : \quad m, n \geq N \Rightarrow \|\mathbf{v}_n - \mathbf{v}_m\| < \epsilon.$$

[9, p. 73]

The following proposition establishes a property for convergent sequences in normed vector spaces.

Proposition 2.9
Every convergent sequence in a normed vector space is a Cauchy sequence. [7, pp. 86–87]

Proof
Let $\mathbf{v}$ be the limit of a convergent sequence $\{\mathbf{v}_n\}_{n\in\mathbb{N}}$ in $(V, \|\cdot\|)$. Then, for any $\epsilon > 0$, there exists $N \in \mathbb{N}$ such that $\|\mathbf{v}_n - \mathbf{v}\| < \frac{\epsilon}{2}$ and $\|\mathbf{v}_m - \mathbf{v}\| < \frac{\epsilon}{2}$. Using the triangle inequality from Definition 2.5

$$\|\mathbf{v}_n - \mathbf{v}_m\| \leq \|\mathbf{v}_n - \mathbf{v}\| + \|\mathbf{v} - \mathbf{v}_m\| < \epsilon. \qquad \blacksquare$$

Note that every Cauchy sequence in a normed vector space is not necessarily convergent. This leads to the definition of a Banach space.

Definition 2.10 (Banach Space)
A normed vector space $(V, \|\cdot\|)$ in which every Cauchy sequence converges is complete, also referred to as a Banach space. [7, p. 87]

In a $L^p(D)$ space, functions are Lebesgue measurable almost everywhere on D such that $\int_D |f(\mathbf{x})|^p \mathrm{d}\mathbf{x} < \infty$ for a scalar-valued function $f \in L^p(D)$. The $L^p(D)$ space is an example of a Banach space, where $D \subseteq \mathbb{R}^n$, with the norm

$$\|f\|_p = \left(\int_D |f(\mathbf{x})|^p \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}},$$

for $1 \leq p < \infty$ [1, p. 50].

FNOs are defined as an operator between seperable Banach spaces. To establish the definition of a separable space, it is essential to first introduce the definition of a dense subset.

Definition 2.11 (Dense)
A subset $S \subset V$ of a normed vector space $(V, \|\cdot\|)$ is dense if and only if for every $r > 0$ and all $\mathbf{v} \in V$ there is an element $\mathbf{s} \in S$ that satisfies $\|\mathbf{v} - \mathbf{s}\| < r$. [7, p. 86]

The set of rational numbers $\mathbb{Q}$ is a dense subset of $\mathbb{R}$, meaning that between any two real numbers, there exists a rational number [7, p. 86]. For instance, between any two distinct real numbers a and b where $a < b$, there exists a rational number q such that $a < q < b$.

A separable space is defined by the existence of a countable, dense subset within it.

Definition 2.12 (Separable)
A normed vector space $(V, \|\cdot\|)$ is separable if it contains a countable dense subset.

[7, p. 86]

As an example of a separable space, take the $L^p(D)$ space from the example of a Banach space. If D is assumed to be a separable measure space, then $L^p(D)$ is separable for any p in $1 \leq p < \infty$ [6, p. 98].

2.2 Operators

This section introduces operators within vector spaces, focusing on linear bounded operators. Linear operators are mappings between vector spaces preserving linearity properties.

Definition 2.13 (Linear Operator)
Let V and W be vector spaces over $\mathbb{F}$. The mapping $A : V \to W$ is called a linear operator if the following properties hold

- Additivity: $A(\mathbf{v} + \mathbf{w}) = A\mathbf{v} + A\mathbf{w}$ for all $\mathbf{v}, \mathbf{w} \in V$.
- Homogeneity: $A(\lambda\mathbf{v}) = \lambda A\mathbf{v}$ for every $\lambda \in \mathbb{F}$ and all $\mathbf{v} \in V$.

The set of linear operators from V to W is denoted as $\mathcal{L}(V, W)$, and the set of linear operators from V to V is denoted $\mathcal{L}(V)$. [7, p. 105]

An example of a linear operator is the Fourier transform. The Fourier transform satisfies the additivity and homogeneity properties from Definition 2.13. Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ be vector-valued functions, and a and b be constants

$$\begin{aligned}\mathcal{F}(a\mathbf{f} + b\mathbf{g})_j(\boldsymbol{\omega}) &= \int_{\mathbb{R}^n} (af_j(\mathbf{x}) + bg_j(\mathbf{x}))\, \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}^\intercal \mathbf{x}}\, \mathrm{d}\mathbf{x} \\ &= a\int_{\mathbb{R}^n} f_j(\mathbf{x})\mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}^\intercal \mathbf{x}}\, \mathrm{d}\mathbf{x} + b\int_{\mathbb{R}^n} g_j(\mathbf{x})\mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}^\intercal \mathbf{x}}\, \mathrm{d}\mathbf{x} \\ &= a\mathcal{F}(\mathbf{f})_j(\boldsymbol{\omega}) + b\mathcal{F}(\mathbf{g})_j(\boldsymbol{\omega})\end{aligned}$$

for $\boldsymbol{\omega} \in \mathbb{R}^n$ and $j = 1, \ldots, m$. See Appendix A for definitions regarding the Fourier transform. A linear operator is continuous if and only if it is bounded [7, p. 106].

Definition 2.14 (Bounded Linear Operator)
Let V and W be normed vector spaces over $\mathbb{F}$. A linear operator $A : V \to W$ is bounded if there exists a positive constant $M \in \mathbb{F}$ such that

$$\|A\mathbf{v}\|_W \leq M\|\mathbf{v}\|_V \quad \text{for all } \mathbf{v} \in V,$$

where $\|\cdot\|_V$ and $\|\cdot\|_W$ denote the norms in the spaces V and W, respectively. [7, p. 106]

The Fourier transform is a bounded linear operator on $L^2(\mathbb{R})$ with the norm $\|\cdot\|_2$. Let $f \in L^2(\mathbb{R})$ then, by the Plancherel theorem

$$\|\mathcal{F}(f)\|_2 = \|f\|_2$$

[9, p. 222]. This implies that the Fourier transform preserves the norm of the function. Since the norm is preserved, the Fourier transform is indeed a bounded operator on $L^2(\mathbb{R})$ with a bound of $M = 1$. The Fourier transform is also an integral operator.

Proposition 2.15 (Kernel Integral Operator)
Let $D \subset \mathbb{R}^n$ and $\tilde{D} \subset \mathbb{R}^m$ be Lebesgue measurable domains, where n and m are positive integers. Assume $1 < p < \infty$ and $1 < q < \infty$ where $\frac{1}{p} + \frac{1}{q} = 1$, and let $K : D \times \tilde{D} \to \mathbb{F}$ be a Lebesgue measurable function. Suppose the norm of K is finite, i.e.

$$\|K\| = \left(\int_D \left(\int_{\tilde{D}} |K(x,y)|^p \, \mathrm{d}y \right)^{\frac{q}{p}} \, \mathrm{d}x \right)^{\frac{1}{q}} < \infty.$$

Then, for any function $f \in L^q(\tilde{D})$, the operator T defined by

$$(Tf)(x) = \int_{\tilde{D}} K(x,y) f(y) \, \mathrm{d}y,$$

defines a bounded linear integral operator T from $L^q(\tilde{D})$ into $L^p(D)$. The function $K(x,y)$ is referred to as the integral kernel of the operator T. [1, pp. 149–150]

The proof for Proposition 2.15 is omitted and can be found in [1, p. 150].

By Proposition 2.15 the Fourier transform is an integral operator expressed as

$$(Tf)(\boldsymbol{\omega}) = \int_{\tilde{D}} K(\boldsymbol{\omega}, \mathbf{x}) f(\mathbf{x}) \mathrm{d}\mathbf{x} \quad \text{for } \boldsymbol{\omega} \in \tilde{D},$$

where $K(\boldsymbol{\omega}, \mathbf{x}) = \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}^\intercal \mathbf{x}}$.

3 Neural Operators

In this chapter, the architecture of neural operators will be introduced. Neural operators are methods of learning a mapping between two infinite-dimensional spaces using only finite observation pairs of input and output from the mapping [26, p. 3].

3.1 Learning Operators

In this section, the theory of approximating operators is presented. Let $D \subset \mathbb{R}^d$ be a bounded, open subset of the spatial domain, $\mathcal{A} = \{\mathbf{a} : D \to \mathbb{R}^{d_a}\}$ be a separable Banach space of functions consisting of input functions of the operator taking values in $\mathbb{R}^{d_a}$, and $\mathcal{U} = \{\mathbf{u} : D \to \mathbb{R}^{d_u}\}$ be a separable Banach space of functions consisting of output functions of the operator taking values in $\mathbb{R}^{d_u}$. See Chapter 2 for definitions regarding operators in Banach spaces. The operator that is being learned is denoted $G^\dagger : \mathcal{A} \to \mathcal{U}$. The purpose of neural operators is to approximate this operator using observed data. Suppose that N input and output pairs $\{\mathbf{a}^{(i)}, \mathbf{u}^{(i)}\}_{i=1}^N$ are observed of the true operator, then $G^\dagger(\mathbf{a}^{(i)}) = \mathbf{u}^{(i)}$. It is assumed that $\mathbf{a}^{(i)} \sim \mu$ are i.i.d. samples from a probability measure μ with support on $\mathcal{A}$ [26, p. 3].

The goal is to approximate $G^\dagger$ using a parameterised operator

$$G_{\boldsymbol{\theta}} : \mathcal{A} \to \mathcal{U}, \quad \boldsymbol{\theta} \in \Theta,$$

where Θ is a finite-dimensional parameter space. Hence, the problem of finding the approximation can be stated as the following minimisation problem

$$\min_{\boldsymbol{\theta} \in \Theta} \mathbb{E}_{\mathbf{a} \sim \mu}[C(G_{\boldsymbol{\theta}}(\mathbf{a}), G^\dagger(\mathbf{a}))],$$

where $C : \mathcal{U} \times \mathcal{U} \to \mathbb{R}^+$ is a loss function and $\mathbb{E}$ is the expectation operator with respect to probability measure μ [26, p. 3].

One family of parameterised operators is the neural operator.

3.2 Neural Operators

In this section, the general architecture of the neural operator is presented, which was first proposed in [27]. The neural operator $G_{\boldsymbol{\theta}} : \mathcal{A} \to \mathcal{U}$ is a neural network consisting of three parts. First, a lifting layer P maps the input function $\mathbf{a} \in \mathcal{A}$ to the first hidden representation $\mathbf{v}^{(0)}$. Next, an iterative kernel layer that maps from one hidden representation $\mathbf{v}^{(t)}$ to the next $\mathbf{v}^{(t+1)}$ is applied T times for $t = 0, \ldots, T-1$. Lastly, there is a projection layer Q that maps from the last hidden representation $\mathbf{v}^{(T)}$ to the solution function $\mathbf{u} \in \mathcal{U}$ [26, p. 4].

The architecture of the neural network is then given by

$$\mathbf{a} \mapsto \mathbf{v}^{(0)} \mapsto \mathbf{v}^{(1)} \mapsto \cdots \mapsto \mathbf{v}^{(T)} \mapsto \mathbf{u}.$$

The structure of the individual layers will be elaborated on in the following subsections.

Lifting Layer

The lifting layer is a point map $P : \mathbb{R}^{d_a} \to \mathbb{R}^{d_v}$ that maps the input function $\mathbf{a} \in \mathcal{A}$ to the first hidden representation $\mathbf{v}^{(0)} \in \mathcal{V} = \{\mathbf{v} : D \to \mathbb{R}^{d_v}\}$. The hidden layer dimension d_v is a hyperparameter and is chosen such that $d_v > d_a$. The lifting layer is a local mapping, meaning the action is pointwise [22, p. 9].

The lifting layer is implemented as a feedforward neural network with one hidden layer [26, p. 4]. This means that the kth entry in $\mathbf{v}^{(0)}$ is given by

$$v_k^{(0)}(\mathbf{x}) = \sum_{j=1}^{M_P} w_{kj}^{(2)} \sigma \left(\sum_{i=1}^{d_a} w_{ji}^{(1)} a_i(\mathbf{x}) + b_j^{(1)} \right) + b_k^{(2)} \quad \text{for } k = 1, \ldots, d_v,$$

where $w_{ji}^{(1)}, w_{kj}^{(2)}, b_j^{(1)}, b_k^{(2)}$ denotes the trainable weights and biases, respectively, M_P is the number of neurons in the hidden layer, $a_i(\mathbf{x})$ is the ith entry of $\mathbf{a}(\mathbf{x})$, $v_k^{(0)}(\mathbf{x})$ is the kth entry of $\mathbf{v}^{(0)}(\mathbf{x})$, and σ are non-linear activation functions [5, p. 228].

Iterative Kernel Layer

The iterative kernel layer maps from one hidden representation $\mathbf{v}^{(t)} \in \mathcal{V}$ to the next $\mathbf{v}^{(t+1)} \in \mathcal{V}$ and is applied for $t = 0, \ldots, T-1$. The iterative kernel layer consists of a local linear operator, a non-local kernel integral operator, and a bias, which are summed and passed to a pointwise nonlinear activation function

$$\mathbf{v}^{(t+1)}(\mathbf{x}) = \sigma \left(\mathbf{W}^{(t)} \mathbf{v}^{(t)}(\mathbf{x}) + \mathcal{K}^{(t)}(\mathbf{v}^{(t)})(\mathbf{x}) + \mathbf{b}^{(t)} \right) \quad \mathbf{x} \in D, \quad t = 0, \ldots, T-1,$$

where σ is a non-linear activation function whose action is defined component-wise, $\mathbf{W}^{(t)} \in \mathbb{R}^{d_v \times d_v}$ is a trainable weight matrix, $\mathbf{b}^{(t)} \in \mathbb{R}^{d_v}$ is a trainable bias, and $\mathcal{K}^{(t)}(\mathbf{v}^{(t)}) : \mathcal{V} \to \mathcal{V}$ is a parameterised kernel integral operator [22, p. 10]. The kernel integral operator is given by

$$\mathcal{K}^{(t)}(\mathbf{v}^{(t)})(\mathbf{x}) = \int_D \boldsymbol{\kappa}^{(t)}(\mathbf{x}, \mathbf{y}) \mathbf{v}^{(t)}(\mathbf{y}) \mathrm{d}\mathbf{y} \quad \text{for } \mathbf{x} \in D, \tag{3.1}$$

where $\boldsymbol{\kappa}^{(t)} : D \times D \to \mathbb{R}^{d_v \times d_v}$ denotes a continuous kernel that parameterises the operator [22, pp. 10–11]. The kernel integral operator is non-local because it depends on values of $\mathbf{v}^{(t)}$ that are not specifically located at the point $\mathbf{x}$. This means that the operator's output at a given point is influenced by the values of $\mathbf{v}^{(t)}$ across the entire domain, rather than just $\mathbf{x}$.

Projection Layer

The projection layer Q maps the last hidden representation $\mathbf{v}^{(T)} \in \mathcal{V}$ to the solution function $\mathbf{u} \in \mathcal{U}$. Analogously with the lifting layer, $d_v > d_u$ and Q is also a local map [22, p. 9].

The projection layer is implemented the same way as the lifting layer, using a feedforward neural network with one hidden layer. So each entry in the output function $\mathbf{u}(\mathbf{x})$ is given

by

$$u_k(\mathbf{x}) = \sum_{j=1}^{M_Q} \tilde{w}_{kj}^{(2)} \sigma \left(\sum_{i=1}^{d_v} \tilde{w}_{ji}^{(1)} v_i^{(T)}(\mathbf{x}) + \tilde{b}_j^{(1)} \right) + \tilde{b}_k^{(2)} \quad \text{for } k = 1, \dots, d_u,$$

where $\tilde{w}_{ji}^{(1)}, \tilde{w}_{kj}^{(2)}$, $\tilde{b}_j^{(1)}, \tilde{b}_k^{(2)}$ denotes the trainable weights and biases, respectively, M_Q is the number of neurons in the hidden layer, $u_k(\mathbf{x})$ is the kth entry of $\mathbf{u}(\mathbf{x})$, $v_i^{(T)}(\mathbf{x})$ is the ith entry of $\mathbf{v}^{(T)}(\mathbf{x})$, and σ are nonlinear activation functions [5, p. 228].

3.3 Fourier Neural Operators

There exist multiple different ways to parameterise the kernel integral operator (3.1). In [26], the FNO was proposed as the parameterised kernel integral operator in Fourier space.

This is done by first letting $\boldsymbol{\kappa}^{(t)}(\mathbf{x}, \mathbf{y}) = \boldsymbol{\kappa}^{(t)}(\mathbf{x} - \mathbf{y})$ then $\boldsymbol{\kappa}^{(t)} : D \to \mathbb{R}^{d_v \times d_v}$ and further assuming that $\boldsymbol{\kappa}^{(t)}$ is a periodic function. Then the kernel integral operator (3.1) becomes

$$\mathcal{K}^{(t)}(\mathbf{v}^{(t)})(\mathbf{x}) = \int_D \boldsymbol{\kappa}^{(t)}(\mathbf{x} - \mathbf{y}) \mathbf{v}^{(t)}(\mathbf{y}) \mathrm{d}\mathbf{y} \quad \text{for } \mathbf{x} \in D,$$

which is recognised as a convolution operator. Using the convolution theorem for matrix- and vector-valued functions Theorem A.5, the integral can be expressed as

$$\mathcal{K}^{(t)}(\mathbf{v}^{(t)})(\mathbf{x}) = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{\kappa}^{(t)}) \mathcal{F}(\mathbf{v}^{(t)}))(\mathbf{x}) \quad \text{for } \mathbf{x} \in D,$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively. The Fourier transform is taken element-wise on the matrix- and vector-valued functions, see Definition A.2.

Using this, the kernel $\boldsymbol{\kappa}^{(t)}$ can be parameterised in the Fourier domain by using its Fourier transform $\mathbf{R}^{(t)} : D \to \mathbb{C}^{d_v \times d_v}$. For a given frequency mode $\mathbf{k} \in D$, we have that $\mathcal{F}(\mathbf{v}^{(t)})(\mathbf{k}) \in \mathbb{C}^{d_v}$ and $\mathbf{R}^{(t)}(\mathbf{k}) \in \mathbb{C}^{d_v \times d_v}$. The kernel transform becomes

$$\mathcal{K}^{(t)}(\mathbf{v}^{(t)})(\mathbf{x}) = \mathcal{F}^{-1}(\mathbf{R}^{(t)} \mathcal{F}(\mathbf{v}^{(t)}))(\mathbf{x}) \quad \text{for } \mathbf{x} \in D.$$

This is called spectral convolution. Using this kernel integral operator, the iterative kernel layers in the FNO are given by

$$\mathbf{v}^{(t+1)}(\mathbf{x}) = \sigma \left(\mathbf{W}^{(t)} \mathbf{v}^{(t)}(\mathbf{x}) + \mathcal{F}^{-1}(\mathbf{R}^{(t)} \mathcal{F}(\mathbf{v}^{(t)}))(\mathbf{x}) + \mathbf{b}^{(t)} \right) \quad \mathbf{x} \in D, \quad t = 0, \dots, T-1, \tag{3.2}$$

where $\mathbf{W}^{(t)}$, $\mathbf{R}^{(t)}$, and $\mathbf{b}^{(t)}$ are the trainable parameters of each layer.

The assumption that $\boldsymbol{\kappa}^{(t)}$ is a periodic function implies that $\boldsymbol{\kappa}^{(t)}$ has a Fourier series expansion. Since the Fourier series coefficients completely describe the function, discrete frequency modes will be used $\mathbf{k} \in \mathbb{Z}^d$ instead of continuous Fourier mode $\mathbf{k} \in \mathbb{R}^d$.

Since $\boldsymbol{\kappa}^{(t)}$ is a real-valued function, the Fourier coefficients must be conjugate symmetric

$$\mathbf{R}^{(t)}(-\mathbf{k})_{j,i} = \left(\mathbf{R}^{(t)}(\mathbf{k}) \right)^*_{j,i} \quad \text{for } \mathbf{k} \in \mathbb{Z}^d \quad j = 1, \dots, d_v, \quad i = 1, \dots, d_v.$$

Hence, only the positive Fourier coefficients need to be parameterised [22, p. 21]. The parameterisation is further restricted by only concerning a maximal number of Fourier coefficients that are given in the set

$$Z_{k_{\max}} = \{\mathbf{k} \in \mathbb{Z}^d : |k_j| \leq k_{\max,j} \text{ for } j = 1, \ldots, d\}.$$

This simplifies the implementation by limiting the number of parameters and has been shown to improve the performance [26, p. 5]. The cardinality of $Z_{k_{\max}}$ is denoted $k_{\max} = |Z_{k_{\max}}|$. Using this, $\mathbf{R}^{(t)}$ can be parameterised as a complex-valued $(k_{\max} \times d_v \times d_v)$ tensor, where all the included Fourier coefficients are stacked along the first axis.

Figure 3.1 illustrates the entire architecture of the FNO.

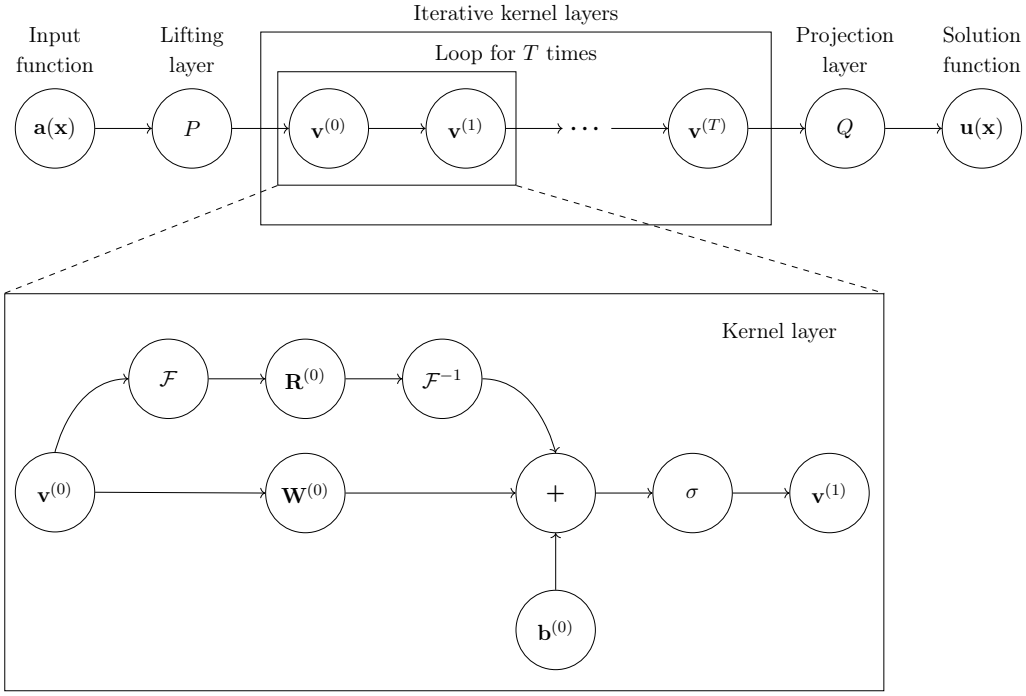


Figure 3.1: The architecture of the Fourier neural operator.

3.4 Discretisation

Until now, we have worked with an infinite-dimensional space of functions. However, in order to work with the functions $\mathbf{a}^{(i)}$ and $\mathbf{u}^{(i)}$ numerically, they are discretised.

Let $D^{(i)} = \{\mathbf{x}_\ell^{(i)}\}_{\ell=1}^n \subset D$ be an n-point discretisation of the domain D. Then the discretised observation is given as

$$\left\{ \mathbf{a}^{(i)}\Big|_{D^{(i)}}, \mathbf{u}^{(i)}\Big|_{D^{(i)}} \right\}_{i=1}^N,$$

where $\mathbf{a}^{(i)}\big|_{D^{(i)}} \in \mathbb{R}^{n \times d_a}$ and $\mathbf{u}^{(i)}\big|_{D^{(i)}} \in \mathbb{R}^{n \times d_u}$. Note that the data samples can have different discretisations $D^{(i)}$. The hidden representations $\mathbf{v}^{(t)}$ are likewise discretised on D as

$\mathbf{v}^{(i)}\big|_{D^{(i)}} \in \mathbb{R}^{n\times d_v}$. Hence, the Fourier transform in (3.2) is performed by using the discrete Fourier transform (DFT) $\hat{\mathcal{F}}(\mathbf{v}^{(t)}) \in \mathbb{C}^{n\times d_v}$. Where $\hat{\mathcal{F}}(\mathbf{v}^{(t)})$ denotes the DFT of $\mathbf{v}^{(i)}\big|_{D^{(i)}}$. The discretisation is assumed to be uniform on D. Hence, the more computationally efficient fast Fourier transform (FFT) can be used to compute the DFT.

The multiplication $\mathbf{R}^{(t)}\mathcal{F}(\mathbf{v}^{(t)})$ in (3.2) can then be defined as

$$\left(\mathbf{R}^{(t)}\hat{\mathcal{F}}(\mathbf{v}^{(t)})\right)_{k,l} = \sum_{j=1}^{d_v} \mathbf{R}^{(t)}_{k,l,j}\left(\hat{\mathcal{F}}(\mathbf{v}^{(t)})\right)_{k,j} \quad \text{for} \quad k = 1,\dots,k_{\max}, \quad l = 1,\dots,d_v.$$

Hence, $\mathbf{R}^{(t)}\hat{\mathcal{F}}(\mathbf{v}^{(t)}) \in \mathbb{C}^{k_{\max}\times d_v}$. It is assumed that $n > k_{\max}$. Therefore, it is only necessary to compute the first $k_{\max}$ Fourier modes of $\hat{\mathcal{F}}(\mathbf{v}^{(t)})$. However, to evaluate the FNO kernel in all $\mathbf{x} \in D^{(i)}$ the matrix $\mathbf{R}^{(t)}\hat{\mathcal{F}}(\mathbf{v}^{(t)}) \in \mathbb{C}^{k_{\max}\times d_v}$ is zero-padded in the first dimension such that $\mathbf{R}^{(t)}\hat{\mathcal{F}}(\mathbf{v}^{(t)}) \in \mathbb{C}^{n\times d_v}$, resulting in $\hat{\mathcal{F}}^{-1}(\mathbf{R}^{(t)}\hat{\mathcal{F}}(\mathbf{v}^{(t)})) \in \mathbb{R}^{n\times d_v}$. This is valid because it is equivalent to setting all Fourier coefficients in $\mathbf{R}^{(t)}$ index above $k_{\max}$ equal to zero.

The FNO is invariant to discretisation meaning that the operator can be evaluated with a different discretisation than it is trained on. This is also called zero-shot super-resolution. Zero-shot means that the operator generalises to discretisation which was not a part of the training set.

4 Sionna Ray Tracing

Sionna RT by Nvidia is a fusion of machine learning techniques and traditional ray tracing, designed for radio propagation modelling [13, p. 317]. It is a differentiable ray tracer which can compute the gradients of generated field components such as channel impulse response with respect to parameters such as radio materials [13, p. 317]. Furthermore, Sionna RT is a deterministic ray tracer. In this project, we will use Sionna RT to calculate the channel impulse response, which will be used to generate synthetic data consisting of channel gains to train our FNO network. Therefore, in this chapter, we will primarily focus on how Sionna RT computes the channel impulse response.

The process of generating channel impulse responses using Sionna RT involves computing the paths that the electromagnetic waves travel from the transmitter to the receiver within the scene. This involves emitting rays in all directions from the transmitter, tracking their paths as they interact with objects in the environment, and only recording the paths that end up at the receiver [14, p. 4].

Sionna RT accounts for various propagation phenomena, including reflection, scattering, and diffraction [13, p. 2]. When a ray encounters a surface, it may be reflected, scattered in different directions, or diffracted around the obstacle, depending on the material properties and geometry of the scene. As the rays propagate through the scene, their characteristics, such as path length, angle of incidence, and time delay, are recorded. These characteristics are used for calculating the channel impulse response [32].

In this chapter, we will delve into the components necessary for generating channel impulse responses using Sionna RT, such as the scene, rays, radio materials, and electromagnetic fields. Additionally, we will introduce Maxwell's equations, which describe the behaviour of electromagnetic fields.

4.1 Electromagnetic Plane Wave

Electromagnetic waves are combinations of electric and magnetic fields. An electric field describes the influence or force that electric charges exert on other charges within the surrounding space. The electric field is a vector quantity varying from point to point [11, p. 61]. A stationary electric charge produces only an electric field, while a moving electric charge generates, in addition, a magnetic field. The magnetic field is also a vector quantity describing the magnetic influence in a region around a magnet or moving electric charge [11, p. 211].

An electromagnetic plane wave is a wave where the electric field and magnetic field are perpendicular to each other and the direction of propagation [11, pp. 395–397]. An illustration of an electromagnetic plane wave can be seen in Figure 4.1.

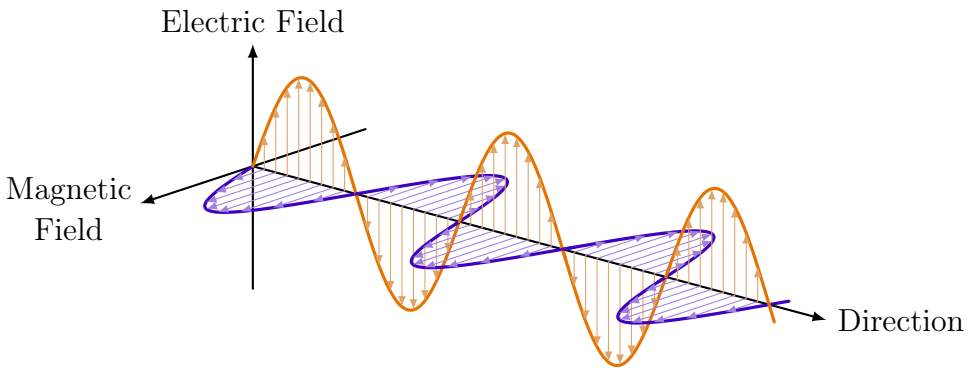


Figure 4.1: Electromagnetic plane wave.

4.2 Electromagnetic Wave Phenomena

The electromagnetic wave can be affected by the environment it propagates through, as explained by three phenomena: reflection, diffraction, and scattering. The phenomena can be seen in Figure 4.2. Sionna RT currently supports all three phenomena.

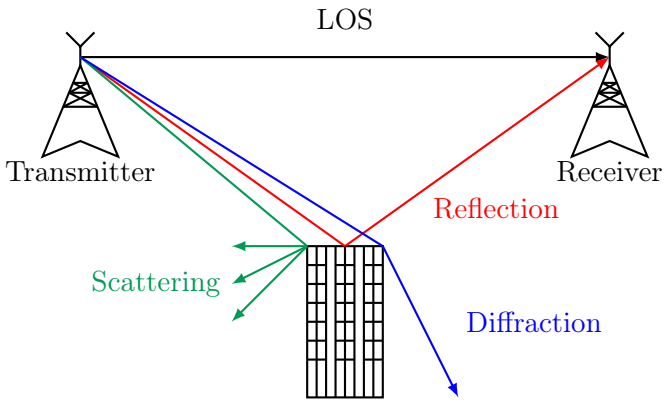


Figure 4.2: Illustration of LOS, reflection, diffraction, and scattering.

Reflection occurs when an electromagnetic wave encounters a radio material. A part of the wave's energy is absorbed, known as shadowing, and the other part is reflected [30, p. 284]. The direction of the reflected wave is equal to the angle of incidence. This concept is known as Snell's law of reflection [3, p. 743]. As the signal continues to travel, the power is attenuated due to travelling large distances. This is known as path loss [30, p. 279]. Besides the distance between transmitter and receiver, path loss can also be influenced by factors such as reflection, absorption, diffraction, and vegetation [39, p. 1].

Diffraction occurs when an electromagnetic wave bends around edges, especially sharp edges called wedges [3, p. 761]. Whether diffraction occurs depends on the wavelength of the electromagnetic wave and the size of the obstacle [30, p. 284].

Scattering happens when an electromagnetic wave encounters an object that is comparable in size to its wavelength. In this case, the energy of the wave is scattered, resulting in multiple propagating waves [30, p. 284]. Scattering can also occur in the reflections of very rough objects or even in the atmosphere [37, p. 31]. In urban areas, there is a high potential for scattering [30, p. 345].

Reflection, diffraction, and scattering all give rise to multipath propagation, which can result in interference, either constructive or destructive. Constructive interference enhances the signal strength, while destructive interference attenuates or cancels out the signal [30, p. 284].

4.3 Ray Tracing Components

The fundamental component of Sionna RT is the scene, which is required to generate channel impulse responses [14, p. 4]. A scene is a digital twin of a physical environment, which can be rendered with a software called Blender to resemble real places in the world [14, pp. 1–2]. Rendering a scene involves defining the medium the ray will travel through and the radio materials it will interact with.

A ray is an affine line specified by its origin and direction. Mathematically, we will denote the parametric form of a ray $\mathbf{r}$ as a function of a scalar value t

$$\mathbf{r}(t) = \mathbf{o} + t\boldsymbol{d}, \quad 0 \leq t < \infty,$$

where the origin is a point $\mathbf{o}$, the direction is a vector $\boldsymbol{d}$, and the scalar value t is the position along the ray. The ray is determined by its starting point $\mathbf{o}$ and the points it passes through. In Sionna RT, the ray's origin is at the transmitter's location or its reflection point, and its direction is the vector from the transmitter or reflection point towards a radio material or receiver [29, p. 7]. Hence, a ray is a geometric concept used in Sionna RT to describe the path in which an electromagnetic wave travels.

The electromagnetic wave will propagate through a medium influenced by encountering radio materials [14, p. 2]. Sionna RT allows users to specify three parameters that describe how radio materials interact with electromagnetic waves. The first parameter is relative permittivity ε_r which is a measure of how much a material responds to an electric field compared to a perfect vacuum [3, p. 39] [36, p. 10]. The second parameter is relative permeability, μ_r. This quantity measures the extent to which the material allows magnetic fields to penetrate through it compared to the ease of penetration in a perfect vacuum [38, p. 3]. The last parameter of radio materials is electrical conductivity σ, which measures the ability of a material to conduct electrical current [24, p. 13]. Sionna RT assumes a homogeneous medium, which means the permittivity and permeability remain constant throughout the entire medium [31]. Hence, the medium is assumed to be homogeneous throughout this chapter. These parameters are essential in Maxwell's equations.

4.4 Prerequisite for Maxwell's Equations

Maxwell's equations are a set of fundamental equations in electromagnetism that describe the behaviour of electric and magnetic fields. These equations describe the relationship between electric charges and currents and the electric and magnetic fields they produce [11, p. 338]. The electric and magnetic fields are described as the vector fields, $\mathbf{E}$ and $\mathbf{B}$, respectively. These fields are expressed through differential equations, which involve vector derivatives such as divergence and curl [11, pp. 16–17, 61, 211].

Before defining Maxwell's equations, the del operator will be introduced. The del operator is a vector differential operator which performs spatial differentiation with respect to the Cartesian coordinates. If the del operator is applied to a one-dimensional function, it denotes the gradient. When applied to a vector field, it can denote the gradient, the divergence, or the curl, depending on how it is applied [11, p. 14].

In a three-dimensional Cartesian coordinate system, the del operator $\boldsymbol{\nabla}$ is defined as

$$\boldsymbol{\nabla} = \mathbf{e}_x \frac{\partial}{\partial x} + \mathbf{e}_y \frac{\partial}{\partial y} + \mathbf{e}_z \frac{\partial}{\partial z},$$

where $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are the unit vectors with respect to each axis [11, p. 16]. When applying the del operator to a vector field $\mathbf{v}$, the divergence, denoted as $\boldsymbol{\nabla} \cdot \mathbf{v}$, is obtained by computing the dot product. The divergence measures how much a vector field spreads out from a specified point. Mathematically, the divergence is defined as follows

$$\boldsymbol{\nabla} \cdot \mathbf{v} = \left(\mathbf{e}_x \frac{\partial}{\partial x} + \mathbf{e}_y \frac{\partial}{\partial y} + \mathbf{e}_z \frac{\partial}{\partial z} \right) \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z},$$

where v_x, v_y, and v_z represent the x, y, and z components of the vector field $\mathbf{v}$. This expression yields a scalar value for the divergence [11, p. 17].

The curl can be obtained by the cross product, $\boldsymbol{\nabla} \times \mathbf{v}$, and is a measure of how much a vector swirls around a specified point [11, p. 19]. Mathematically, the curl is defined as follows

$$\boldsymbol{\nabla} \times \mathbf{v} = \begin{vmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ v_x & v_y & v_z \end{vmatrix} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z} \right) \mathbf{e}_x + \left(\frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x} \right) \mathbf{e}_y - \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right) \mathbf{e}_z$$

[11, p. 18].

4.5 Differential Form of Maxwell's Equations

Maxwell's equations in differential form will be introduced using the notation from the prerequisites. Starting with Gauss' law of electricity, then Gauss' law of magnetism, Faraday's law, and finally, Ampere's law with Maxwell's modification.

Maxwell's electric field equation is expressed as

$$\boldsymbol{\nabla} \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \tag{4.1}$$

where ρ is the source charge density, and ϵ_0 is the permittivity in a perfect vacuum [11, p. 241]. This equation is known as Gauss' law of electricity, and describes how the electric field behaves in the presence of electric charges. Specifically, the equation determines the electric field when ρ is given. If the charge is positive, then the electric field diverges away from that point [11, p. 241].

Maxwell's magnetic field equation is expressed as

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0. \tag{4.2}$$

[11, p. 241]. This equation is known as Gauss' law for magnetism. There are no magnetic point sources, as there are for electric fields. Magnetic point sources create magnetic charges, and do not exist in nature, hence why the equation is equal to zero [11, pp. 241–242].

The third of Maxwell's equations is Faraday's law, which expresses the curl of an electric field

$$\boldsymbol{\nabla} \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{4.3}$$

where t represents time [11, p. 332]. The negative sign indicates the direction of the induced electric field. This equation states that a changing magnetic field induces an electric field [11, p. 334].

Ampere's law states that magnetic fields relate to electric currents. Maxwell added the displacement magnetic current density term

$$\mathbf{J}_d = \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}$$

to state that a changing electric field produces a magnetic field. Ampere's law, with Maxwell's addition, is expressed as

$$\boldsymbol{\nabla} \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \mathbf{J}_d, \tag{4.4}$$

where μ_0 is the permeability in a perfect vacuum. The term $\mathbf{J}$ is the current density and adds both the magnitude and direction of current flow at a specific point in space [11, p. 334]. This modification to Ampere's law was introduced by Maxwell to ensure the consistency of electromagnetic theory and to account for phenomena like electromagnetic waves [11, p. 334].

4.6 Far Field Assumption

In wireless communication, a transmitter emits an electromagnetic field uniformly in all directions, creating what is known as a spherical wave [3, p. 283]. In Sionna RT, it is assumed that the receiver observes a time-harmonic electromagnetic plane wave [14]. Time-harmonic implies that the amplitude of the plane wave varies sinusoidally with time [3, p. 21]. Additionally, the transmitter is assumed to be in the far field. The far field is a region where the emitted electromagnetic wave is far enough away to appear as plane waves. This is

because the curvature of the wavefront becomes negligible, and the waves propagate as if they were planar [4, p. 72]. Furthermore, the far field region is where the field patterns of an antenna become independent of the distance. The far field region is defined to exist at distances greater than $\frac{2D^2}{\lambda}$ from the antenna, where D is the largest dimension of the antenna and λ is the wavelength of the electromagnetic wave [4, pp. 34–35]. The near field region is defined at distances less than $0.62\sqrt{\frac{D}{\lambda}}$. Unlike the far field region, the near field region is dependent on distance [4, p. 34].

Assuming that the transmitter emits an electromagnetic field uniformly combined with the far-field condition, suggests that the direction of propagation of the plane wave radiates outward uniformly from the antenna [4, p. 72]. Additionally, the receiver is assumed to observe a time-harmonic uniform plane wave travelling through a homogeneous medium [3, p. 123].

4.7 Channel Impulse Response

This section delves into the concepts needed for Sionna RT to generate a channel impulse response. A time-harmonic plane electric wave $\mathbf{E}_\mathbf{p}(z) \in \mathbb{C}^3$ travelling in a homogeneous medium can be described at position $\mathbf{p} \in \mathbb{R}^3$ at time t, travelling in the z direction

$$\mathbf{E}_\mathbf{p}(z) = \underbrace{\mathbf{E}_0^+ \mathrm{e}^{-\mathrm{i}\beta z}}_{\mathbf{E}_\mathbf{p}^+} + \underbrace{\mathbf{E}_0^- \mathrm{e}^{+\mathrm{i}\beta z}}_{\mathbf{E}_\mathbf{p}^-}.$$

Here, $\beta = \omega\sqrt{\mu\epsilon}$ is the phase constant, where $\omega = 2\pi f$ is the angular frequency, f is the frequency of the wave in Hz, and μ and ϵ are the magnetic permeability and electric permittivity, respectively [31]. The terms $\mathbf{E}_0^+$ and $\mathbf{E}_0^-$ represent the amplitudes of the travelling waves in the positive and negative z direction, respectively [3, p. 126].

The associated time-harmonic plane magnetic wave, travelling in the y direction, can be determined by

$$\mathbf{B}_y^\pm = \pm \frac{1}{\sqrt{\frac{\mu}{\varepsilon}}} \mathbf{E}_\mathbf{p}^\pm,$$

where the wave impedance η is denoted as the ratio of electric and magnetic fields

$$\eta = \frac{\mathbf{E}_\mathbf{p}^+}{\mathbf{B}_y^+} = \sqrt{\frac{\mu}{\varepsilon}}.$$

The reciprocal value $\frac{1}{\eta}$ represents how the properties of the medium influence the relationship between electric and magnetic fields [3, p. 126].

A useful measure to describe the performance of an antenna is the radiation intensity. Radiation intensity describes how much power is radiated by the antenna in different directions [4, p. 40]. Sionna RT assumes that a spherical wave emitted from the centre of the antenna can characterise the electric far field of an antenna

$$\mathbf{E}(r, \theta, \phi) = \mathbf{E}_0(\theta, \phi) \frac{\mathrm{e}^{-\mathrm{i}\beta r}}{r},$$

where $\mathbf{E}_0(\theta, \phi)$ is the electric field phasor, r is the distance from the antenna to the observation point where the electric field is being measured, θ is the zenith angle, and ϕ is the azimuth angle [31]. By assuming a spherical wave, the Poynting vector is given by

$$\mathbf{S}(r, \theta, \phi) = \frac{1}{2\eta}\|\mathbf{E}(r, \theta, \phi)\|^2 \hat{\mathbf{r}},$$

where $\hat{\mathbf{r}}$ is the radial unit vector. The Poynting vector describes the flow of energy in an electromagnetic wave. It points in the direction that energy is moving and its magnitude represents the rate of energy flow per unit area. It simplifies for an ideal isotropic antenna

$$\mathbf{S}_{\text{iso}}(r, \theta, \phi) = \frac{P_T}{4\pi r^2}\hat{\mathbf{r}},$$

where P_T is the input power to the antenna.

The antenna pattern $\mathbf{F}(\theta, \phi)$ describes the spatial distribution of the radiation intensity as a function of direction. The antenna pattern is defined as

$$\mathbf{F}(\theta, \phi) = \frac{\mathbf{E}_0(\theta, \phi)}{\max_{\theta,\phi} \|\mathbf{E}_0(\theta, \phi)\|},$$

where the maximum value $\max_{\theta,\phi} \|\mathbf{E}_0(\theta, \phi)\|$ corresponds to the direction in which the antenna radiates the most power [31].

Another useful measure is the antenna gain G. The antenna gain is defined by how much power the antenna radiates in a specific direction compared to that of an ideal isotropic source, hence

$$G = \frac{\max_{\theta,\phi}\|\mathbf{S}(r, \theta, \phi)\|}{\|\mathbf{S}_{\text{iso}}(r, \theta, \phi)\|} = \frac{2\pi}{\eta P_T}\max_{\theta,\phi}\|\mathbf{E}_0(\theta, \phi)\|^2$$

[31]. The antenna gain can also be defined with a directional dependency as

$$G(\theta, \phi) = \frac{2\pi}{\eta P_T}\|\mathbf{E}_0(\theta, \phi)\|^2 = G\|\mathbf{F}(\theta, \phi)\|^2.$$

Since $G(\theta, \phi)$ contains no information about the phase and $\mathbf{F}(\theta, \phi)$ contains no information about the maximum gain G, Sionna RT combines G and $\mathbf{F}(\theta, \phi)$ such that the new antenna pattern $\mathbf{C}(\theta, \phi)$ has both information about the phase and maximum gain

$$\mathbf{C}(\theta, \phi) = \sqrt{G}\mathbf{F}(\theta, \phi),$$

such that $G(\theta, \phi) = \|\mathbf{C}(\theta, \phi)\|^2$ [32].

When the electromagnetic wave travels along each transmission path, it encounters radio materials and phenomena that transform the wave. To model this cascade of interaction, Sionna RT composites all these transformations into a matrix, denoted as $\tilde{\mathbf{T}}$, which captures the changes to the wave as it travels. It is assumed that the matrix includes necessary coordinate transformations [32]. Consider the path delay $\tau_m = \frac{r_m}{c_m}$, where $m = 1, \ldots, N$ denotes the mth path out of the total N paths, r_m denotes the total path length, and c_m

represents the average propagation speed. To include the phase shift to each corresponding path, the transformation matrix can be modified as follows

$$\mathbf{T}_m = \tilde{\mathbf{T}}_m \mathrm{e}^{\mathrm{i}2\pi f \tau_m},$$

where f is the frequency [31].

The channel frequency response in Sionna RT is computed as a summation over all paths, incorporating antenna patterns from both the transmitting and receiving antennas. Hence, the expression for the channel frequency response calculated in Sionna RT is

$$H(f) = \sum_{m=1}^{N} \underbrace{\frac{\lambda}{4\pi} \mathbf{C}_R(\theta_{R,m}, \phi_{R,m})^H \mathbf{T}_m \mathbf{C}_T(\theta_{T,m}, \phi_{T,m})}_{a_m} \mathrm{e}^{-\mathrm{i}2\pi f \tau_m},$$

where a_m is the amplitude of each path, $\mathbf{C}_R(\theta_{R,m}, \phi_{R,m})$ is the antenna pattern for the receiver, and $\mathbf{C}_T(\theta_{T,m}, \phi_{T,m})$ is the antenna pattern for the transmitter [31]. The channel impulse response is obtained by using the inverse Fourier transform on the channel frequency response

$$h(\tau) = \sum_{m=1}^{N} a_m \delta(\tau - \tau_m),$$

which is the formula used in Sionna RT [31].

5 Generating Synthetic Data with Sionna RT

In this project, we will use synthetic data based on the real-measured data obtained at Aalborg University, as described in Section 1.4. The objective of this chapter is therefore to outline the process for generating channel gains in Sionna RT.

5.1 Loading the Scene

The synthetic data is generated using Fredrik Bajers Vej 7 as the scene, aiming to replicate the environment where the real-measured data was obtained. A satellite image of the location can be seen in Figure 1.2. The resulting scene can be seen in Figure 5.1.

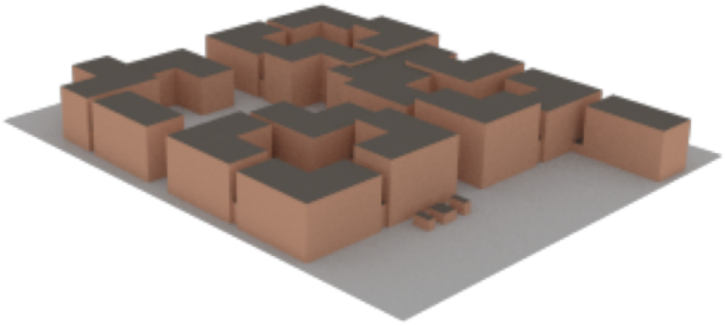


Figure 5.1: The scene generated of Fredrik Bajers Vej 7.

This scene is created using data from OpenStreetMap, which imports the 3D representations of buildings and terrain into the 3D program Blender. Notably, elements such as trees, vehicles, pedestrians, and bicycle racks are not included in the Blender scene, thereby introducing a divergence from the real-measured data. Subsequently, radio materials are applied within Blender, with Sionna RT providing a selection of basic radio materials, each with their respective properties. The three parameters describing the properties of the radio material are described in Section 4.3. In Sionna RT, the materials are assumed to be non-ionised and non-magnetic, thus the relative permeability is $\mu_r = 1$. The relative permittivity ε_r and the electrical conductivity σ are calculated as

$$\varepsilon_r = a f_{\text{GHz}}^b,$$
$$\sigma = c f_{\text{GHz}}^d,$$

where f_{GHz} is the frequency of the wave in GHz, and the constants a, b, c and d characterise the material [28, pp. 23–24]. The chosen radio materials for the scene and their properties can be seen in Table 5.1.

Type	**Material**	**Relative Permitivity ε_r**		**Conductivity σ**	
		a	b	c	d
Ground	Concrete	5.24	0	0.0462	0.7822
Walls	Brick	3.91	0	0.0238	0.16
Roof	Metal	1	0	10^7	0

Table 5.1: All the selected materials are valid for a frequency range of $1 - 100$ GHz. The values are obtained from [28, p. 24].

5.2 Configuration of Transmitters and Receiver

After establishing the initial scene, the transmitters and the receiver are configured and added to the scene. All transmitters and the receiver are equipped with the same configured antenna in the simulation. We configure the transmitters and the receiver in our scene as single antennas.

Antenna Pattern

The antenna is characterised by its pattern, which describes how it radiates or receives electromagnetic energy. We have chosen to use a dipole antenna pattern due to its simplicity [4, p. 151]. A dipole antenna is composed of two conductive elements, see Figure 5.2. These elements, typically straight wires or rods, are aligned parallel to each other with a small gap between them [4, p. 4]. The length of each wire is crucial in determining the antenna's performance characteristics, including its radiation pattern. Generally, longer antennas result in narrower beam patterns. This phenomenon leads to increased directivity as the antenna lengthens [4, p. 173].

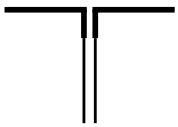


Figure 5.2: A simple illustration of a dipole antenna.

A dipole antenna exhibits omnidirectional patterns [4, p. 55]. An omnidirectional pattern is defined as a non-directional pattern in one plane and a directional pattern in any orthogonal plane. This classifies the omnidirectional pattern as a special type of directional pattern. Directional patterns refer to the antenna's ability to radiate or receive electromagnetic waves more efficiently in specific directions [4, p. 33].

The omnidirectional pattern can be seen in Figure 5.3 for an infinitesimal dipole. In this scenario, the dipole is represented by an infinitesimally small linear wire, where the length of the wire is much smaller than the wavelength of the electromagnetic waves it interacts with. This idealised model aids in simplifying theoretical calculations in antenna analysis. Positioned symmetrically at the origin of the coordinate system, the infinitesimal dipole serves as a point source for radiating or receiving electromagnetic waves [4, p. 151].

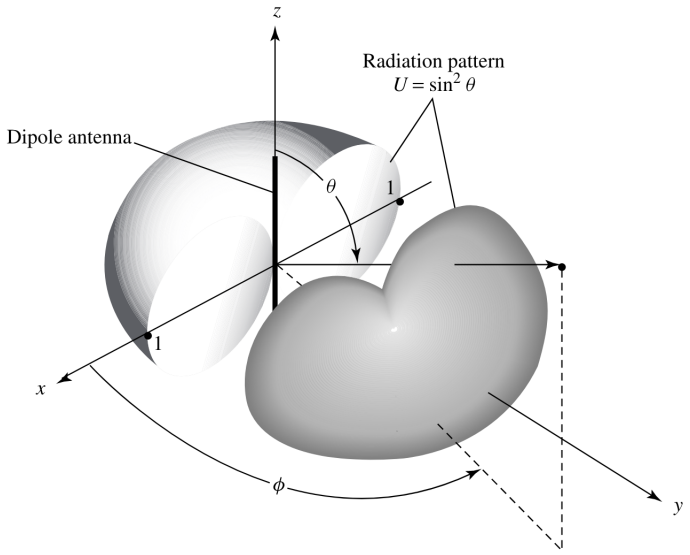
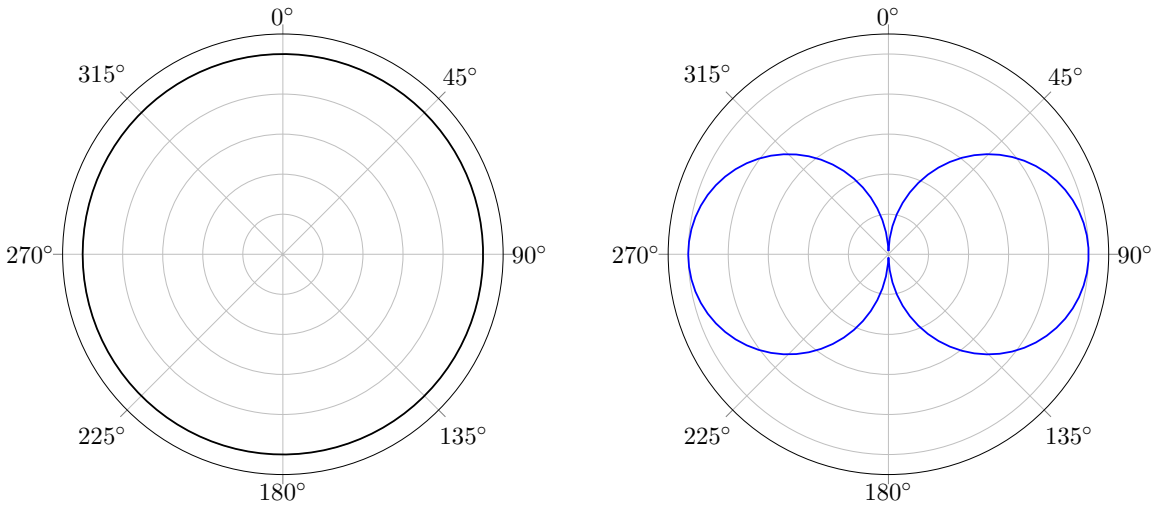


Figure 5.3: Omnidirectional beampattern for an infinitesimal dipole antenna. The figure is from [4, p. 161].

The antenna pattern comprises principal E and H plane patterns, each defining specific orientations of electromagnetic field vectors and the direction of maximum radiation. The E plane refers to the plane containing the electric field vector, and the H plane contains the magnetic field vector [4, p. 33]. In Figure 5.3, the E plane corresponds to the elevation plane, aligning with the xz-plane, while the H plane corresponds to the azimuth plane, aligning with the xy-plane. The omnidirectional pattern exhibits non-directional characteristics in the azimuth plane and directional properties in the elevation plane [4, p. 33].

The patterns for the azimuth and elevation planes can be seen in Figure 5.4a and Figure 5.4b, respectively. The figures illustrate how the azimuth plane is omnidirectional, while the elevation plane exhibits directional properties.



(a) Azimuth pattern.

(b) Elevation pattern.

Figure 5.4

Polarisation

Antenna polarisation is the orientation of the transmitted wave [4, p. 71]. We have chosen vertical polarisation, where the electric field oscillates in a vertical plane perpendicular to the direction of propagation, see Figure 4.1.

5.3 Propagation Paths

After setting up the scene and adding the configured transmitters and the receiver, the propagation paths are calculated. A propagation path starts at a transmitter and ends at a receiver. The path is described by the channel coefficient a and delay τ, as well as the angles of departure (θ_T, ϕ_T) and arrival (θ_R, ϕ_R), see Chapter 4 for details. The path computation is performed in two main steps.

Firstly, the trajectories of the paths are determined through path tracing, a process independent of radio materials, antenna patterns, and radio device orientations. This independence is advantageous for studying the impact of various radio materials without needing to retrace propagation paths, making it particularly useful for calibrating scene parameters. Initially, ray directions are organised in a Fibonacci lattice on the unit sphere, which is a simple way to produce an even point distribution [32]. Subsequently, paths originating from the transmitter and reaching the receiver are traced. To generate candidate paths, the system traces a total of 10^6 random rays.

Secondly, the electromagnetic fields corresponding to the traced paths are computed, taking into account the radio materials, antenna patterns, and radio device orientations [32]. An example of computed propagation paths can be seen in Figure 5.5.

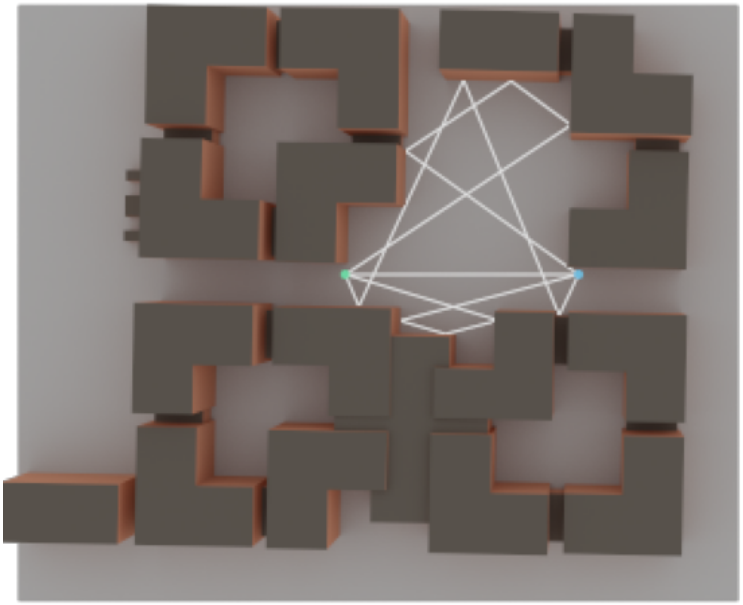


Figure 5.5: Paths created in Sionna RT. Allowed with a maximum of 3 interactions with objects within the scene, focusing solely on identifying LOS and reflection paths. The recorded paths are illustrated as white lines generated by casting rays from the transmitter to the receiver, which are the green and blue circles, respectively.

5.4 From Paths to Channel Gains

The derivation of the channel impulse response is provided in depth in Section 4.7. In Sionna RT, the resulting channel impulse response provides the path coefficients a and the path delays τ for each path. These coefficients and delays are components within the narrowband model. Employing a narrowband model simplifies our approach by focusing on a specific frequency range, reducing computational complexity. Furthermore, it is less susceptible to distortion caused by delay spread, ensuring a more accurate representation of the received signal's characteristics [10, p. 84]. The narrowband model can be expressed as

$$y = hx + n.$$

Here y is the received signal, x is the transmitted signal, n is additive white complex Gaussian noise, and h is the complex channel coefficient given by

$$h = \sum_{m=1}^{N} a_m \mathrm{e}^{-\mathrm{i}2\pi f \tau_m},$$

where a_m represents the amplitude of each path m and τ_m is the path delay. The channel gain $|h|$ is determined by taking the absolute value of h [19, p. 3] [37, p. 36]. The channel gain from our narrowband model forms our synthetic dataset for training FNOs. The architecture of the model is presented in the next chapter.

6 Architecture of Fourier Neural Operator to Interpolate Channel Gains

The purpose of this chapter is to walk through the setup, from the preparation of data to the architecture of the FNO. Our goal is to approximate the operator $G^\dagger$ by a parameterised operator as presented in Chapter 3.

6.1 Data Generation

This section will outline the methodology for generating data used in training, validating, and testing the FNO. The data generation process involves computing the channel gain between a stationary set of transmitter locations and a mobile receiver using Sionna RT.

We are generating the data using the digital twin of Fredrik Bajers Vej 7, as described in Section 5.1. The transmitter positions are confined within the green square measuring 10×10 m, centred in the courtyard of Fredrik Bajers Vej 7, as depicted in Figure 6.1. Furthermore, the potential receiver locations form the yellow square frame measuring 20×20 m, centred on the same coordinates as the green square, as depicted in Figure 6.1. Note that the green square and the yellow square frame do not overlap.

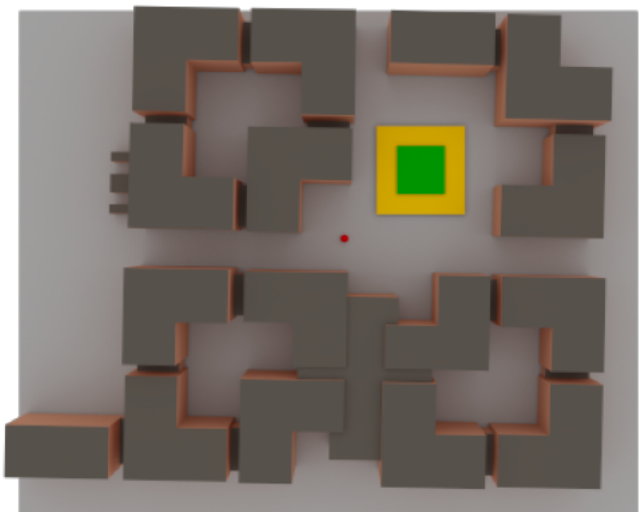


Figure 6.1: A digital twin of Fredrik Bajers Vej 7. The green square covers a 10×10 m area of the transmitter positions. The yellow square frame covers a 20×20 m area of the receiver positions. Origo is illustrated by the red dot.

The channel gains are measured in a 10×10 grid, resulting in 100 deterministic measured points. The distance between the measured points is 1.11 m. The interpolated points are

likewise generated in a 20×20 grid, giving a total of 400 interpolated points. The distance between the interpolated points is 0.53 m. See Figure 6.2.

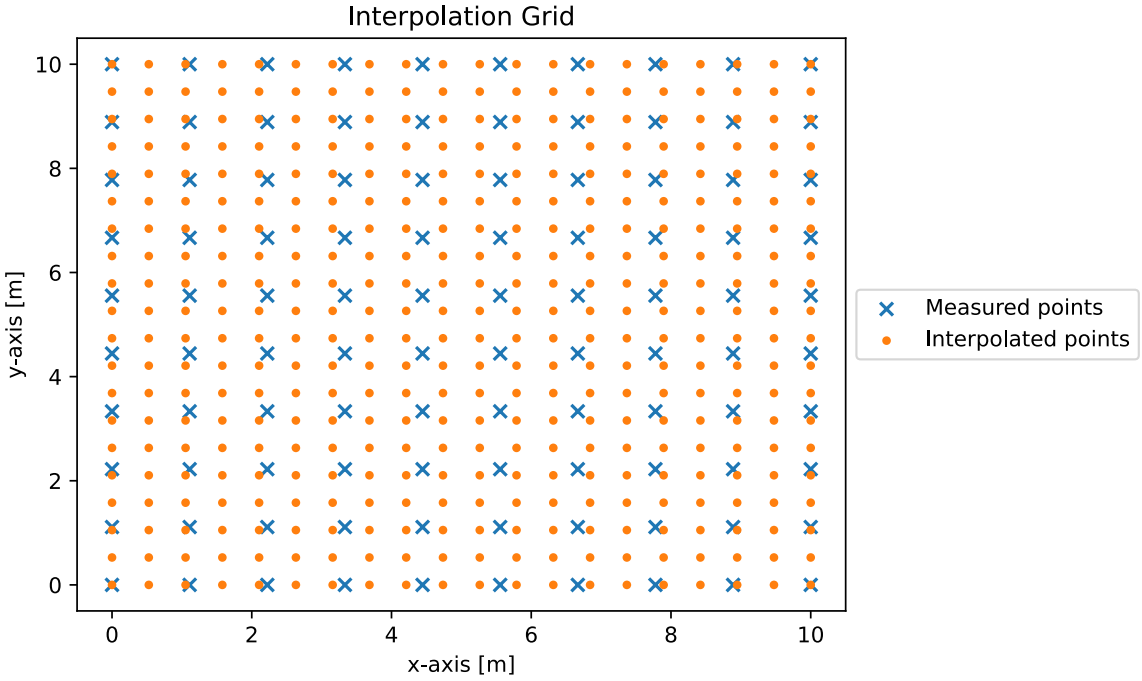


Figure 6.2: The 10×10 measured points and 20×20 interpolated points plot.

In order to generate different realisations of the data, the receiver is moved inside the yellow square frame depicted in Figure 6.1. The receiver positions are drawn i.i.d. from a uniform distribution within the yellow square frame. For each receiver coordinate, the channel gain is simulated for each transmitter location both in the 10×10 measurement points and in the 20×20 interpolation points. This is done for 1600 receiver positions, resulting in 1600 datasets. The measurement points will be stored in the input tensor called $\mathbf{A}$, and the interpolation points will be stored in the output tensor called $\mathbf{U}$. The tensor structure will be described in the next section.

The Input and Output Tensors

The input to the FNO is a tensor $\mathbf{A} \in \mathbb{R}^{5 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}}$, where N_{u_x} and N_{u_y} denote the number of interpolated points in the x and y axes, respectively. Likewise, N_{a_x} and N_{a_y} denote the number of measured points in the x and y axes, respectively. The four last dimensions will be referred to as the spatial dimensions. The first dimension has a length of 5 and is referred to as the channel dimension, given by

$$\mathbf{A}[:, i, j, k, l] = [|h(x_{a_k}, y_{a_l})|, x_{u_i}, y_{u_j}, x_{a_k}, y_{a_l}]^\intercal.$$

The first entry in the channel dimension is the channel gain of the measurement points (x_{a_k}, y_{a_l}) denoted $|h(x_{a_k}, y_{a_l})|$. The second and third entries are the x and y values of the

interpolated points, denoted as $\{x_{u_i}\}_{i=1}^{N_{ux}}$ and $\{y_{u_j}\}_{j=1}^{N_{uy}}$, respectively. The fourth and fifth entries are the x and y values of the measurement points, denoted as $\{x_{a_k}\}_{k=1}^{N_{ax}}$ and $\{y_{a_l}\}_{l=1}^{N_{ay}}$, respectively. Note that the channel gains are repeated in the last two spatial dimensions, i.e., the fourth and fifth dimensions.

The output of the FNO is a tensor $\mathbf{U} \in \mathbb{R}^{1\times N_{ux}\times N_{uy}\times N_{ax}\times N_{ay}}$ with one channel dimension

$$\mathbf{U}[:, i, j, k, l] = |h(x_{u_i}, y_{u_j})|.$$

The output tensor has one channel, the channel gain, corresponding to the interpolation points (x_{u_i}, y_{u_j}) denoted $|h(x_{u_i}, y_{u_j})|$. Again, note that the channel gains are repeated this time in the first two spatial dimensions, i.e., the second and third dimensions.

The reason for repeating the channel gains in both the input and output tensors is in order to facilitate the discretisation-invariance in both the number of measurement points and interpolated points.

6.2 Architecture of the Fourier Neural Operator

The FNO is based on the model proposed in [26], which is also described in Chapter 3. A breakdown of the model layers and their number of learnable parameters is presented in Table 6.1.

Neural Network Architecture			
Layer	**Type**	**Output Dimensionality**	**Learnable Parameters**
1	Fully Connected Layer	$64 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	384
2	Fully Connected Layer	$32 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	2080
3	Fourier Layer	$32 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	769056
4	Fourier Layer	$32 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	769056
5	Fourier Layer	$32 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	769056
6	Fourier Layer	$32 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	769056
7	Fully Connected Layer	$64 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	2112
8	Fully Connected Layer	$1 \times N_{u_x} \times N_{u_y} \times N_{a_x} \times N_{a_y}$	65

Table 6.1: The neural network architecture for the FNO. The model includes a total of 3,080,865 learnable parameters.

Activation Function

The same activation function is used both in the Fourier layers and the fully connected layers. The activation function used in each layer is the Gaussian Error Linear Unit (GELU), defined as

$$\text{GELU}(x) = x\Phi(x),$$

where $\Phi(x)$ is the cumulative distribution function for a standard Gaussian distribution. The GELU function is estimated as

$$\text{GELU}(x) = \frac{1}{2}x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right).$$

The GELU activation function is a smooth approximation of the Rectified Linear Unit (ReLU) function. It smoothly transitions between zero and the input value, allowing for a more continuous activation compared to the ReLU. See Figure 6.3.

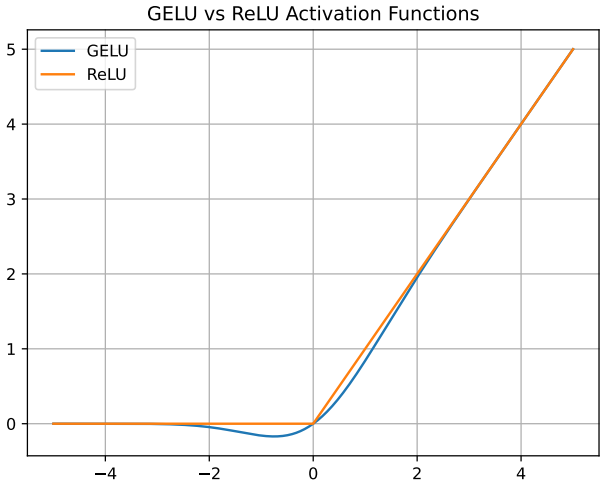


Figure 6.3: Plot of the GELU and ReLU activation functions.

Furthermore, GELU has been found to improve performance across various domains, including computer vision, natural language processing, and speech tasks [12].

Layers

The input data is first processed by the lifting layer, which transforms the input from having 5 channels to a higher-dimensional representation with 32 channels. The lifting layer is a multi-layer perceptron with one hidden layer of dimension 64.

The lifted representation is then passed through the four Fourier layers. All the Fourier layers have the same input and output channel dimensions of 32. Each Fourier layer consists

of a spectral convolution and a skip connection. The spectral convolution is implemented by taking a 4D FFT on the spatial dimensions. Only the Fourier coefficients in the set

$$Z_{k_{\max}} = \{\mathbf{k} \in \mathbb{Z}^4 : |k_j| \leq 5 \text{ for } j = 1, \ldots, 4\}$$

are kept. Hence, only the first 5 Fourier coefficients in each spatial dimension are saved. The coefficients are then multiplied by a trainable complex matrix and zero-padded to the original size before the inverse 4D FFT is applied.

The skip connection multiplies each spatial dimension with a 32×32 weight matrix and adds a bias. The skip connection facilitates the flow of gradients during training, helping to improve the stability and convergence of the network.

The outputs from the spectral convolution and skip connection are added, and a GELU activation function is applied to each element individually.

The output from the Fourier layers is processed by the projection layer, which is another multi-layer perceptron with one hidden layer with dimension 64. The multi-layer perceptron transforms the output of the last Fourier layers back into the channel output dimension of 1.

Training Procedure and Loss Functions

The network is trained by back propagation using the Adam optimiser. The Adam optimiser is initialised with an initial learning rate of 0.001. Furthermore, the learning rate is decreased by a factor of 0.7 every 50 epoch.

The loss function is used to compute the loss between the predicted output $\hat{\mathbf{U}}$ and the ground truth output $\mathbf{U}$. The loss function used is the relative ℓ_2 loss

$$\ell_{2\text{ loss}}(\hat{\mathbf{U}}, \mathbf{U}) = \frac{\left\|\hat{\mathbf{U}} - \mathbf{U}\right\|_2}{\|\mathbf{U}\|_2}.$$

The validation loss is calculated as the average loss over the validation data set and is given by

$$\ell_{2\text{ val}} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \frac{\left\|\hat{\mathbf{U}}_i - \mathbf{U}_i\right\|_2}{\|\mathbf{U}_i\|_2},$$

where N_{val} is the number of samples in the validation set and $\hat{\mathbf{U}}_i$ and $\mathbf{U}_i$ are the ith prediction and ground truth output, respectively.

The test loss is calculated in the same way as

$$\ell_{2\text{ test}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\left\|\hat{\mathbf{U}}_i - \mathbf{U}_i\right\|_2}{\|\mathbf{U}_i\|_2},$$

where N_{test} is the number of samples in the test set.

Hyperparameter Overview

Below in Table 6.2 is an overview of the hyperparameters used in our FNO.

Parameter	**Value**
Number of Fourier layers	4
Fourier modes	(5, 5, 5, 5)
Hidden channels	32
Lifting hidden layers dimension	64
Projection hidden layers dimension	64
Initial learning rate	0.001
Learning rate decay factor	0.7
Period of learning rate decay	50
Epochs	350

Table 6.2: Model hyperparameters.

7 Experiments

This chapter focuses on presenting the experiments conducted to evaluate the performance of the FNO in interpolating channel gains. The performance entails a comparison between the predicted channel gains and the ground truth channel gains. The FNO will also be compared to a simple baseline model. Furthermore, a sensitivity analysis will be conducted.

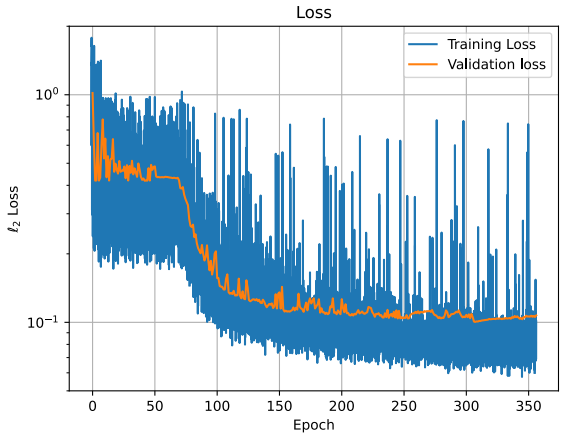
7.1 Results

The experiment is constructed based on the configuration detailed in Section 6.1. The architecture of the developed model is described in Chapter 6. The data is split into the sets seen in Table 7.1. The input and output resolution in all the sets are fixed at 10×10 and 20×20, respectively.

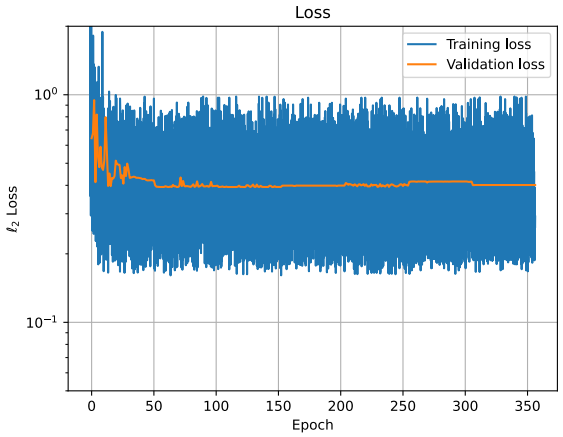
Set	Size in %
Training	70
Validation	15
Test	15

Table 7.1

In Figure 7.1, the training and validation loss as a function of epochs is shown for two models with the same hyperparameters as shown in Table 6.2. The only difference is the weight initialisation. We have observed that the model does not converge every time. The model in Figure 7.1a converges, while the model in Figure 7.1b does not.



(a) Training and validation loss per epoch for a model that converges. The test loss is 0.224.

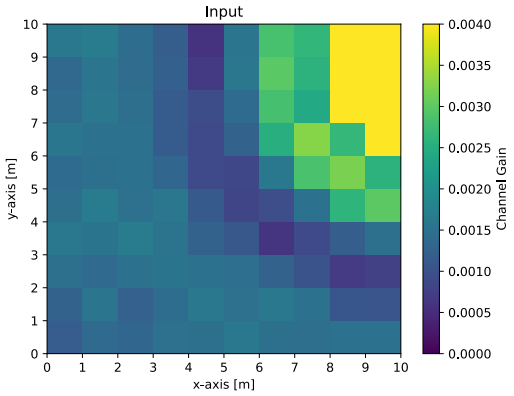


(b) Training and validation loss per epoch for a model that does not converge. The test loss is 0.448.

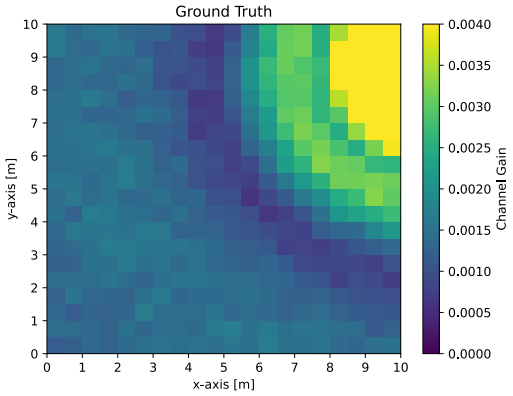
Figure 7.1

For the model that converges, both the training and validation loss decrease as the number of epochs increases and drops around the 70th epoch. Towards the end of training, the validation loss diverges slightly from the training loss, suggesting potential overfitting. Additionally, there are some spikes in the training loss. For the rest of the section, the model, whose loss is depicted in Figure 7.1a will be used.

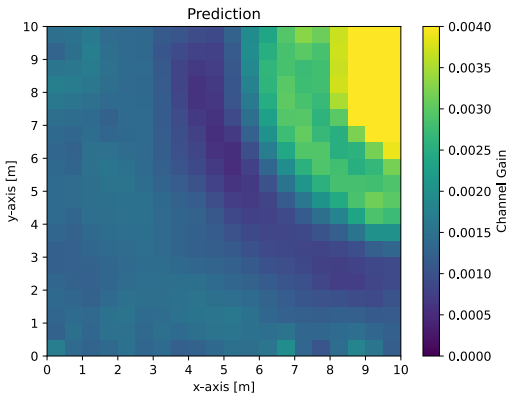
The results from one of the test sets are shown in Figure 7.2. The input to the model can be seen in Figure 7.2a. The ground truth of channel gains in a higher resolution can be seen in Figure 7.2b, and the predicted output from the model can be seen in Figure 7.2c. The model captures details such as the curve-like structure in the upper right corner. The residuals are calculated as the absolute difference between the ground truth and predicted output, see Figure 7.2d. The FNO predicts the output for each input point. It has been observed that the predictions do not vary a lot in the output tensor. Hence, only the first prediction is shown.



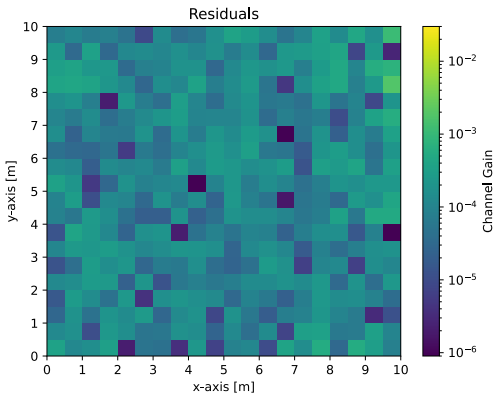
(a) The input to the FNO.



(b) The ground truth channel gains.



(c) The predicted channel gains from the FNO.

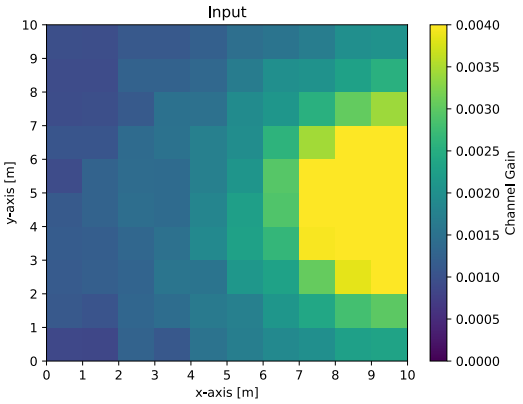


a

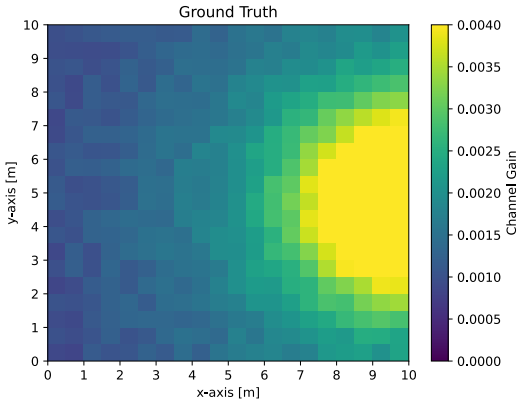
(d) The residuals. Note that the scaling on the z-axis differs from the first three plots. The scaling is logarithmic, facilitating clearer observation of the differences.

Figure 7.2: The receiver is placed at coordinates (25.10, 20.82, 1.15) and the test loss is 0.0953.

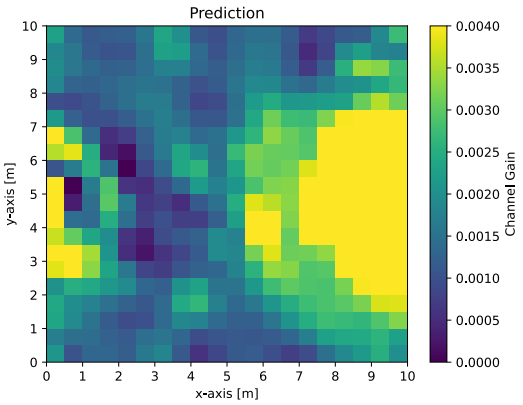
The results from one of the test sets with a higher test loss are shown in Figure 7.3.



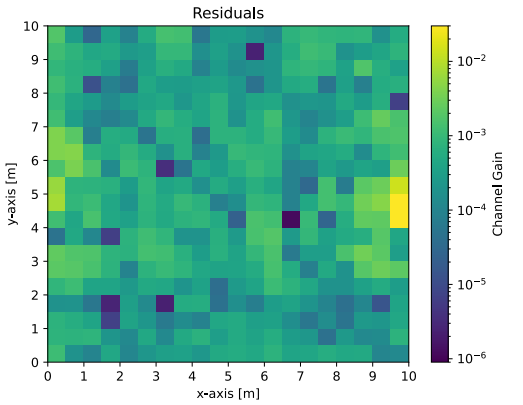
(a) The input to the FNO.



(b) The ground truth channel gains.



(c) The predicted channel gains from the FNO.



(d) The residuals. Note that the scaling on the z-axis differs from the first three plots. The scaling is logarithmic, facilitating clearer observation of the differences.

Figure 7.3: The receiver is placed at coordinates (24.07, 16.68, 1.15) and the test loss is 0.427.

The test loss for this test set is 0.427, which is higher compared to the test loss of 0.0953 obtained in Figure 7.2. This may be explained by comparing the ground truth channel gains in Figure 7.3b with the predicted channel gains in Figure 7.3c. It can be seen that the model predicts higher channel gains at locations where low channel gains are present. This entails larger residuals, as can be seen in Figure 7.3d.

7.2 Baseline Model

To investigate the performance of the FNO, we will compare it to a baseline model. The baseline is a simple model that uses bilinear interpolation. This interpolation method interpolates values within a two-dimensional rectangular grid based on the values of its surrounding grid points by considering linear interpolation in two dimensions [21, pp. 261–262]. The purpose of comparing to this baseline is to investigate how a simple model performs compared to our FNO. It should be noted that the baseline model is not a state-of-the-art algorithm. Therefore, the baseline model cannot verify whether the FNO performs well compared to other existing models.

Initially, the baseline model was tested on the original test set described in Section 7.1. The baseline model obtained a test loss of 0.104 compared to the test loss of 0.224 obtained from the FNO.

To further investigate and compare the performance of the FNO and the baseline model, we have conducted experiments varying either input or output resolution in the test data while keeping the other fixed. The reason to do this was to see how well the two models perform when faced with different input or output resolutions. Especially how the performance of the FNO behaves with different input or output resolutions from those seen during training. Recall that the FNO is only trained on a 10×10 input resolution and a 20×20 output resolution.

Output Resolution

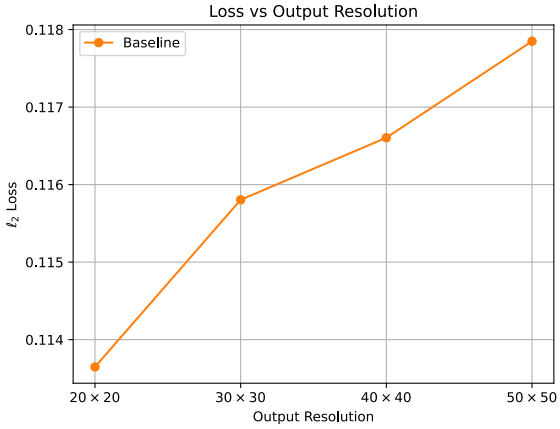
The first experiment was conducted by varying the output resolution within the test data while keeping the input resolution fixed. Hence, the test data consists of 50 data samples for each output resolution ranging from 20×20 to 50×50, with the input resolution remaining fixed at 10×10, see Table 7.2. The same receiver locations are used between data sets, meaning that only the output resolution varies between the sets.

Test Set	Input Resolution	Output Resolution
1	10×10	20×20
2	10×10	30×30
3	10×10	40×40
4	10×10	50×50

Table 7.2: Test sets for different output resolutions.

The results of these experiments are depicted in Figure 7.4, where the test loss is plotted as a function of four different output resolutions for both the FNO and the baseline model.

(a) Test loss as a function of output resolution for the FNO.

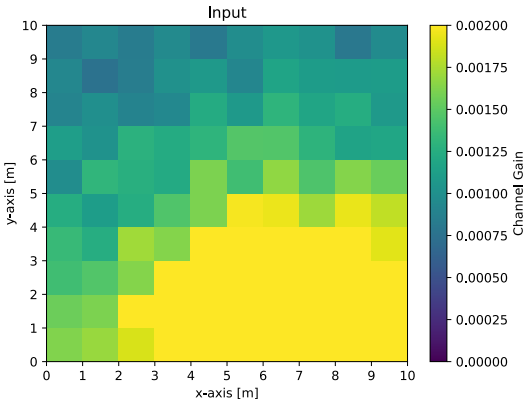


(b) Test loss as a function of output resolution for the baseline model.

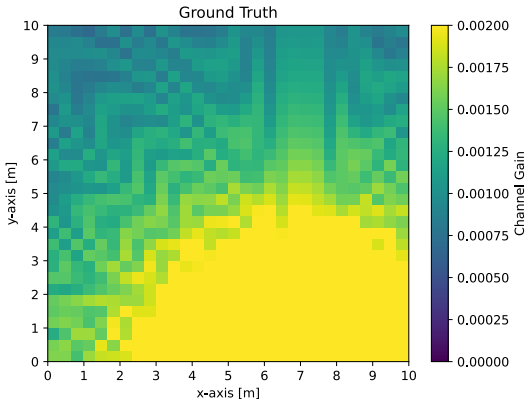
Figure 7.4

For both plots, it is observed that as the size of the output resolution increases, the test loss increases as well. The smallest test loss for both models is when the output resolution is 20×20, which for the FNO is the same resolution it was trained on. This suggests that the FNO performs best when the output resolution aligns with the resolution it has been trained on. Additionally, the increase in the test loss for the FNO is notable as the output resolution increases from 20×20 to 30×30; however, it is less pronounced as the output resolution increases from 30×30 to 40×40, and similarly from 40×40 to 50×50. In contrast, the baseline model shows minimal variation in test loss across all the different output resolutions.

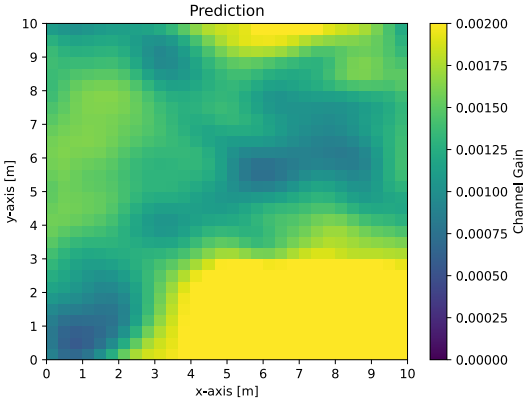
The results from one of the test sets are shown in Figure 7.5, where the input resolution is 10×10 and the output resolution is 30×30. The input to the FNO and the baseline model can be seen in Figure 7.5a. The ground truth of channel gains can be seen in Figure 7.5b. The predicted output from the FNO and baseline model can be seen in Figure 7.5c and Figure 7.5d, respectively. The residuals for the FNO and baseline model can be seen in Figure 7.5e and Figure 7.5f, respectively.



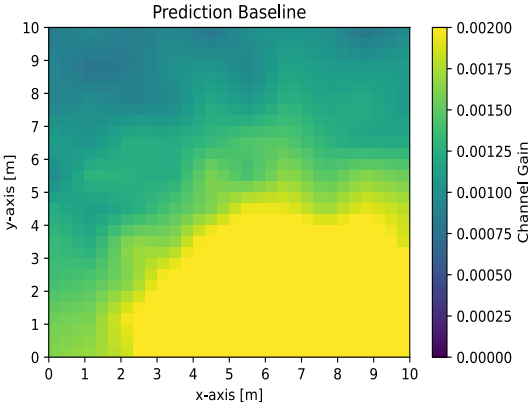
(a) The input to the FNO and baseline model.



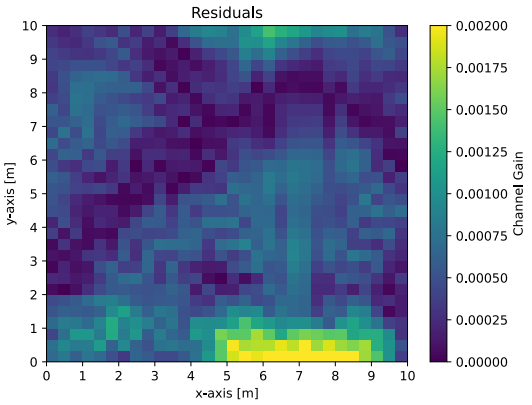
(b) The ground truth channel gains.



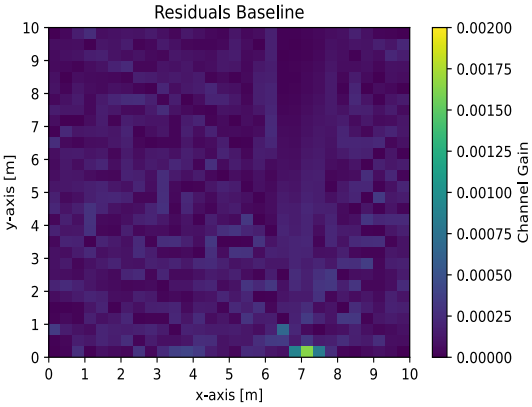
(c) The predicted channel gains for the FNO.



(d) The predicted channel gains for the baseline model.



(e) The residuals for the FNO.



(f) The residuals for the baseline model.

Figure 7.5: The input resolution is 10×10 and the output resolution is 30×30. The receiver is placed at coordinates (21.12, 11.02, 1.15) and the test loss for the FNO and baseline model are 0.295 and 0.070, respectively.

The test loss is higher for the FNO compared to the baseline model. Visually, the baseline's prediction has a closer resemblance to the ground truth than the prediction from the FNO.

This is also confirmed by looking at the residuals, where the residuals in general are higher valued for the FNO than for the baseline.

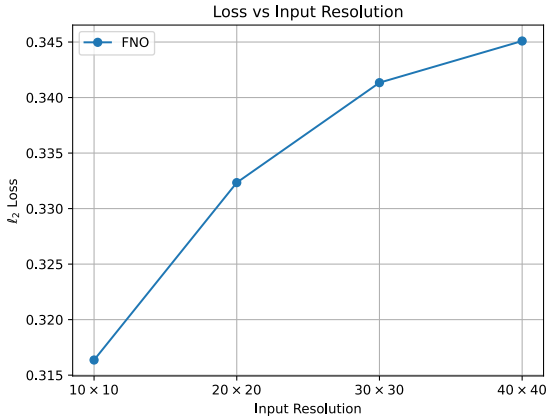
Input Resolution

The second experiment was conducted by varying the input resolution in the test data while keeping the output resolution fixed. Hence, the test data consists of 50 data samples for each input resolution ranging from 10×10 to 40×40, with the output resolution remaining fixed at 50×50, see Table 7.3. Again, the same receiver locations are used between data sets, meaning that only the output resolution varies between the sets.

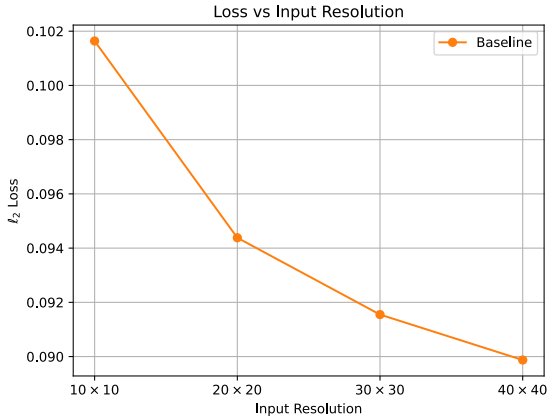
Test Set	Input Resolution	Output Resolution
1	10×10	50×50
2	20×20	50×50
3	30×30	50×50
4	40×40	50×50

Table 7.3: Test sets for different input resolutions.

The results of these experiments are depicted in Figure 7.6, where the test loss is plotted as a function of four different input resolutions for both the FNO and the baseline model.



(a) Test loss as a function of input resolution for the FNO.

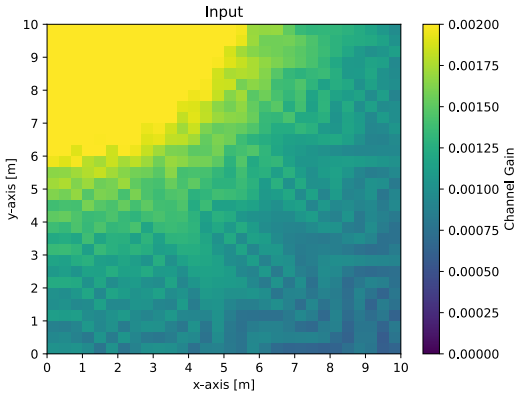


(b) Test loss as a function of input resolution for the baseline model.

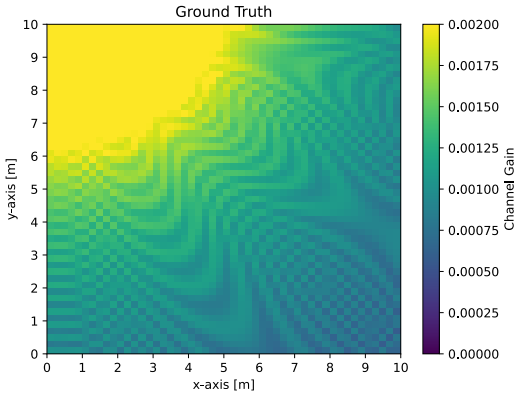
Figure 7.6

In Figure 7.6a, it is observed that as the size of the input resolution increases, the test loss increases as well. The smallest test loss is when the input resolution is 10×10, which is the same resolution the FNO was trained on. Conversely, in Figure 7.6b, it is observed that as the size of the input resolution increases, the test loss decreases. This was expected, as higher input resolution provides more information to the model, resulting in decreased test loss.

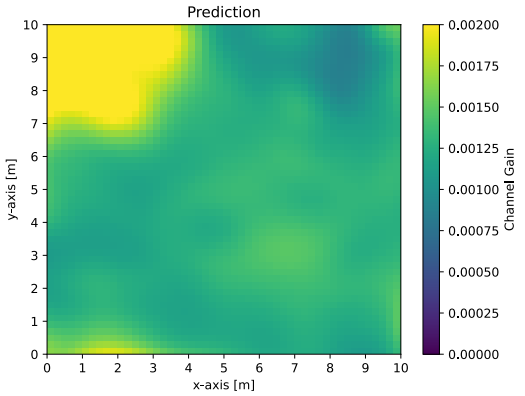
The results from one of the test sets are shown in Figure 7.7, where the input resolution is 30×30 and the output resolution is 50×50. The input to the FNO and the baseline model can be seen in Figure 7.7a. The ground truth of channel gains can be seen in Figure 7.7b. The predicted output from the FNO and baseline model can be seen in Figure 7.7c and Figure 7.7d, respectively. The residuals for the FNO and baseline model can be seen in Figure 7.7e and Figure 7.7f, respectively.



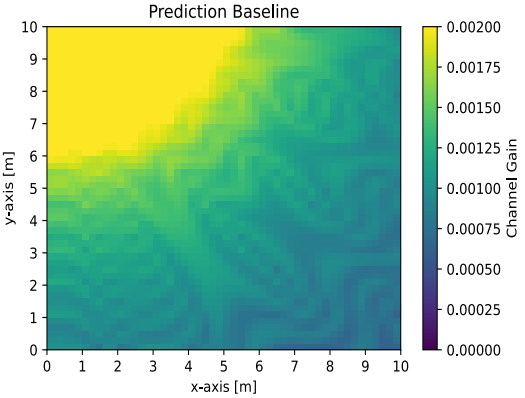
(a) The input to the FNO and the baseline model.



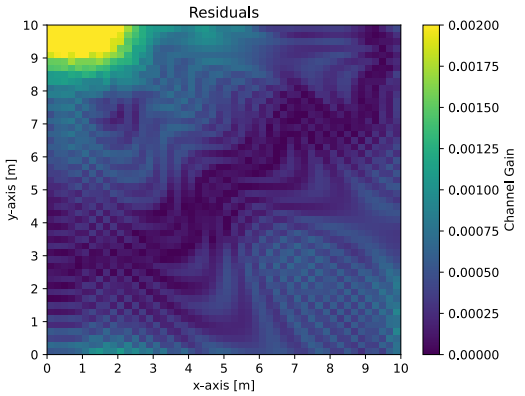
(b) The ground truth channel gains.



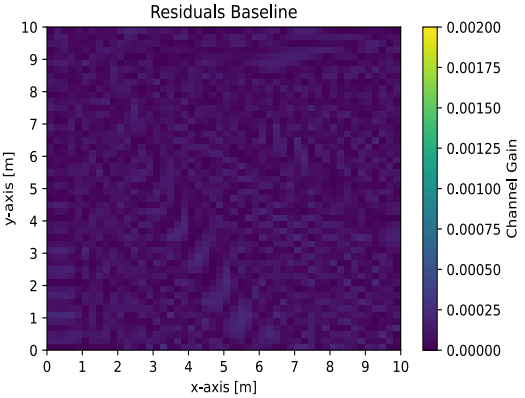
(c) The predicted channel gains from the FNO.



(d) The predicted channel gains from the baseline model.



(e) The residuals for the FNO.



(f) The residuals for the baseline model.

Figure 7.7: The input resolution is 30×30 and output resolution is 50×50. The receiver is placed at coordinates (14.26, 23.32, 1.15) and the test loss for the FNO and baseline are 0.352 and 0.055, respectively.

The test loss is again higher for the FNO compared to the baseline model. Visually, the baseline's prediction captures finer details from the ground truth than the prediction from the FNO. This is also confirmed by looking at the residuals, where the residuals, in general, are

of higher value for the FNO compared to the baseline model. Furthermore, the finer details emerge in the residual plot, suggesting that the FNO encounters challenges in accurately predicting them.

7.3 Sensitivity Analysis

In this section, the aim is to investigate how sensitive the input given to the FNO is to noise. Specifically, Gaussian noise is added to the channel gains in the existing test data. Thus, an analysis of how the test loss changes as the variance in the noise increases is made.

Impact of Gaussian Noise Variance on Test Loss

Gaussian noise with a mean of 0 and varying variances is added to the input data, simulating noisy measurements. The variance ranges from 10^{-7} to $9 \cdot 10^{-7}$ with a step size of 10^{-7}. The step size is chosen such that there are 9 equidistant steps from the beginning to the end of the variance range. To avoid obtaining infeasible negative channel gain measurements, a new sample is drawn from the Gaussian distribution until the resulting channel gain is no longer negative. Consequently, this process introduces skewness to the dataset but is chosen to align with real-world constraints.

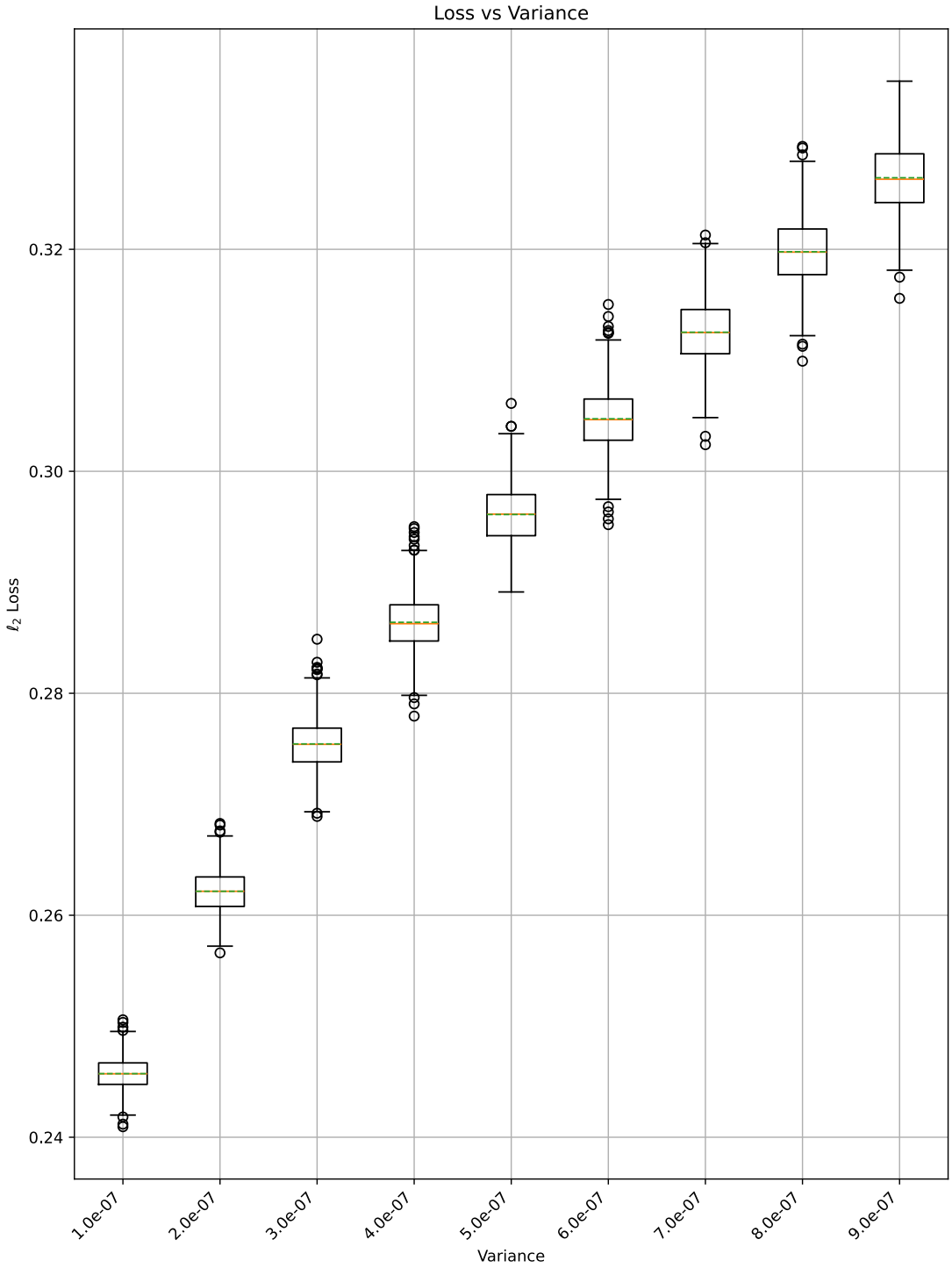


Figure 7.8: The variance ranges from 10^{-7} to $9 \cdot 10^{-7}$, where each step is a simulation of 1000 instances to create its own boxplot. The orange line in the boxplot is the median, and the dashed green line is the mean value.

Figure 7.8 illustrates the test loss as a function of variance in the noise added to the input data. The box in the boxplot represents the interquartile range, which is the middle 50% of the data. The median divides the data into two halves, with 50% of the observations falling below it and 50% above it. The bottom edge of the box represents the first quartile. This is the value below which 25% of the data falls. The top edge of the box represents the third quartile. This is the value below which 75% of the data falls. The whiskers extend to the furthest data points within 1.5 times the interquartile range, which is the range between the first and the third quartile. Any data points beyond the whiskers are considered outliers and are illustrated as dots.

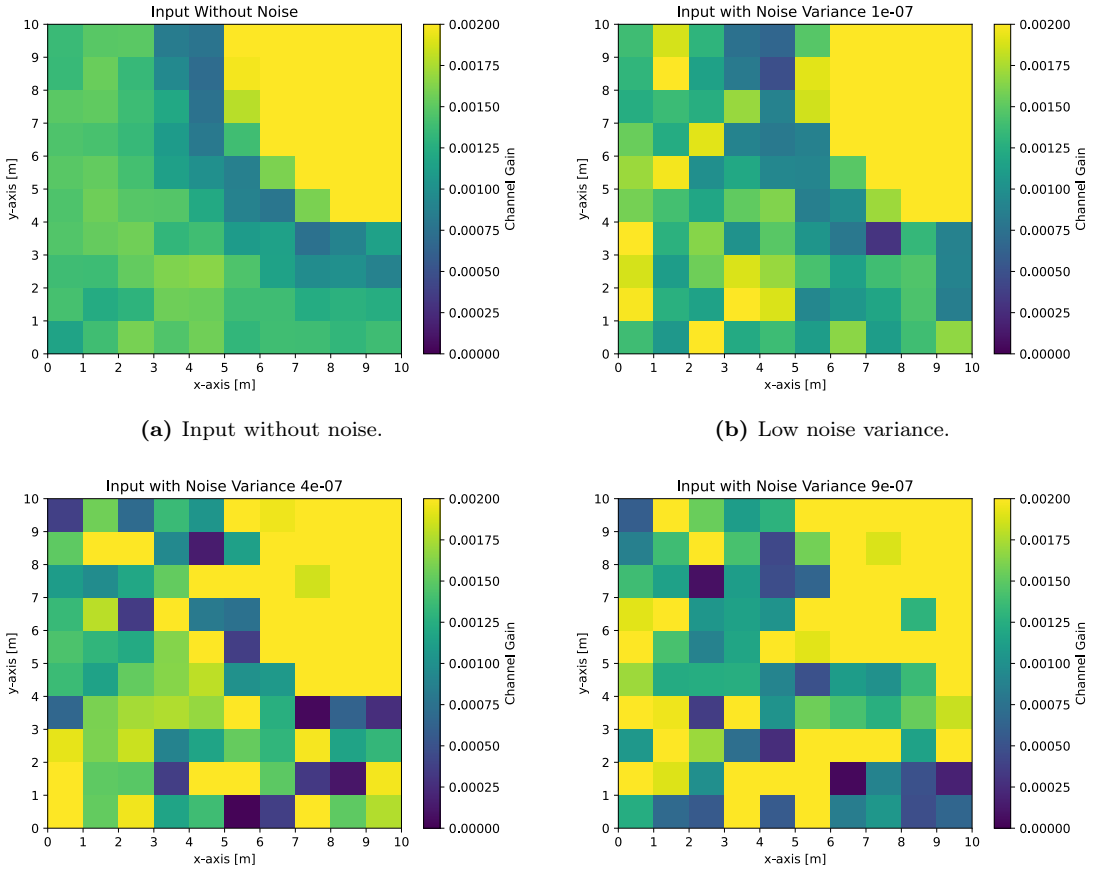
Initially, with no added noise, the test loss for the dataset is 0.22. As the variance increases, the test loss generally increases as well. The interquartile range tends to increase with higher variances, indicating more variability in the loss values at higher variances.

A Visual Representation of the Influence of Noise

To illustrate the sensitivity analysis, we focus on three specific examples. The three variance levels are $10^{-7}, 4{\cdot}10^{-7}$, and $9{\cdot}10^{-7}$, and will be referred to as low, medium, and high variance, respectively. Additionally, the noiseless input data, along with its corresponding prediction and residuals, will be plotted to provide a reference point for the sensitivity analysis. We analyse how the test loss changes when Gaussian noise is added to the input at a specific receiver coordinate. We consider the receiver positioned at coordinates $(24.75, 21.21, 1.15)$.

Given the coordinates, the receiver is placed at the top right corner of the orange square, as described in Section 6.1. Consequently, high channel gain values are received in the top right corner of the input, output, and prediction.

By adding noise to the input for the different levels of variances, the corresponding channel gains can be seen in Figure 7.9.



(a) Input without noise.

(b) Low noise variance.

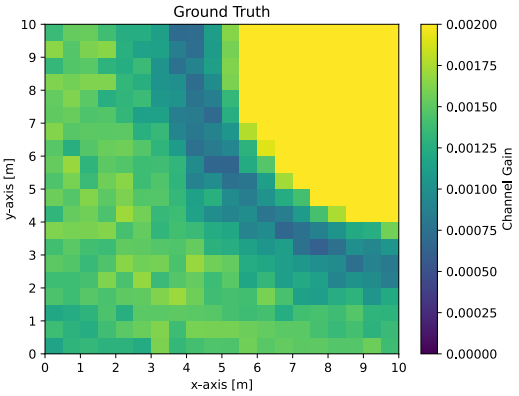
(c) Medium noise variance.

(d) High noise variance.

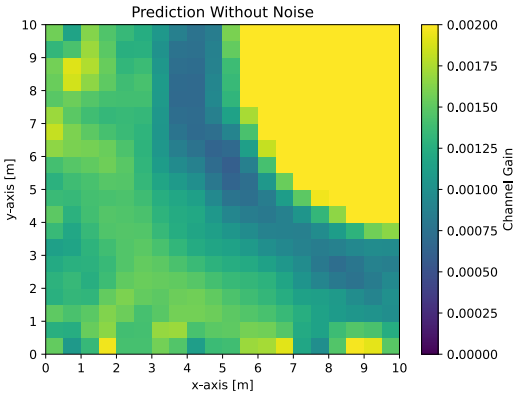
Figure 7.9: Effect on the input for different levels of noise variance. The input resolution is 10×10.

The added noise visually alters the input, increasing multiple channel gains as the noise variance is increased. This can be seen in Figure 7.9b, Figure 7.9c, and Figure 7.9d for the low, medium, and high variance of noise, respectively.

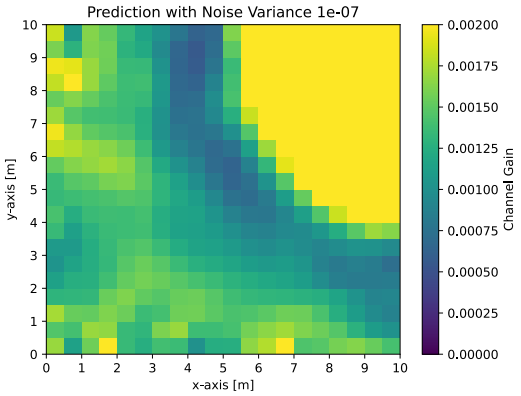
When each noisy input from Figure 7.9 is input to the FNO, it generates a 20×20 resolution prediction, as illustrated in Figure 7.10.



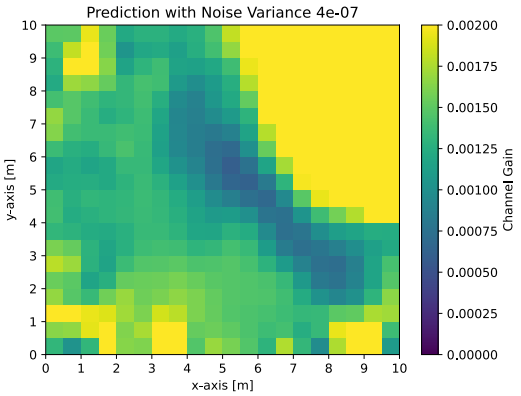
(a) Ground Truth.



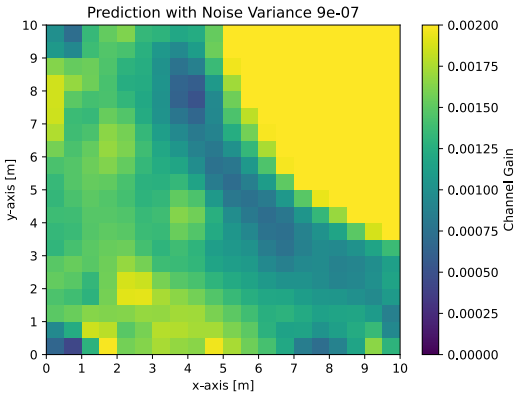
(b) Without noise with a test loss of 0.138.



(c) Low noise variance with a test loss of 0.160.



(d) Medium noise variance with a test loss of 0.184.

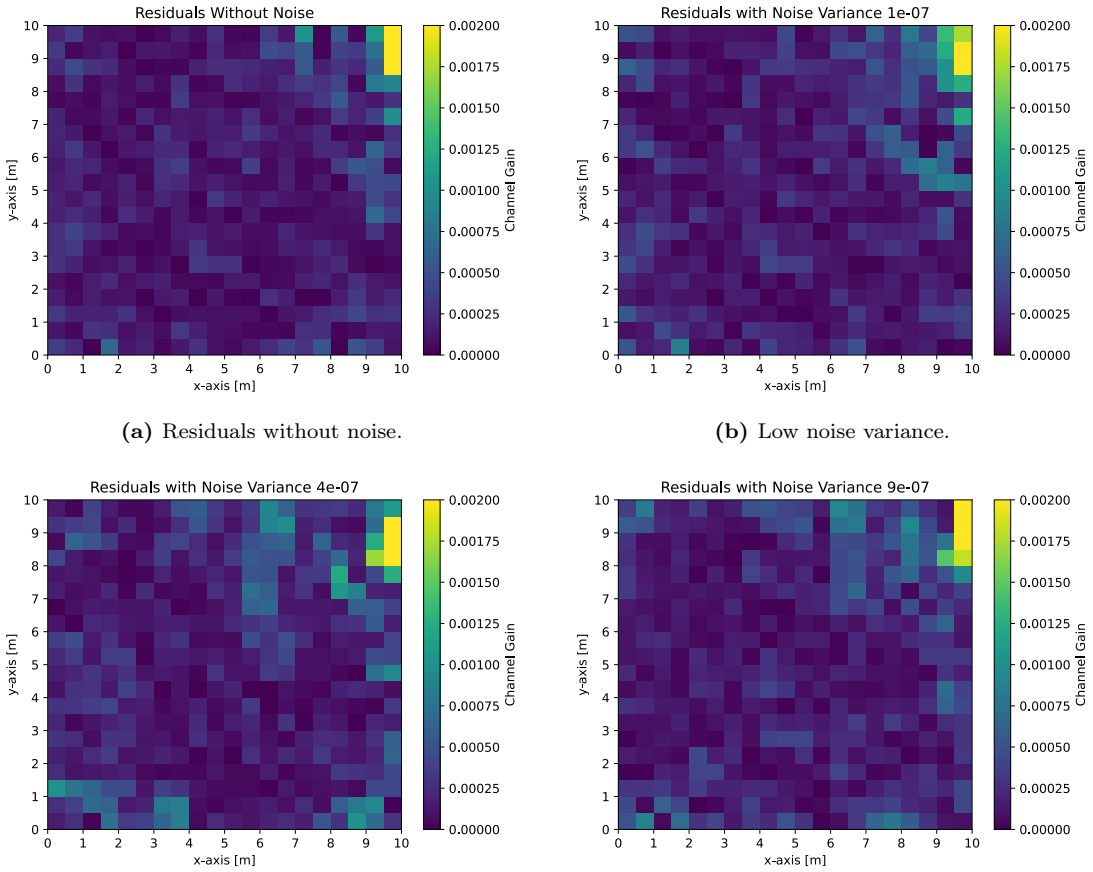


(e) High noise variance with a test loss of 0.202.

Figure 7.10: Effect on the prediction for different levels of noise variance. The output resolution is 20×20.

The main impact is observed at the edges, where the channel gains are predicted as higher values. However, when the noise with high variance is added in Figure 7.10e some of the high channel gains at the edges are decreased.

To further evaluate the prediction against the ground truth, the residuals are plotted in Figure 7.11, showing the difference between predicted and ground truth channel gains.



(a) Residuals without noise.

(b) Low noise variance.

(c) Medium noise variance.

(d) High noise variance.

Figure 7.11: Residuals between predictions from the noisy input data and ground truth.

For all residual plots, the highest values are located in the upper right corner. It can be observed that as the noise variance increases, the residuals' values do too.

8 Discussion

In this project, a model for interpolating channel gains with arbitrary discretisation in both the number of measurements and the number of interpolated points has been developed. The model was proposed in Chapter 6. The model was trained and tested on simulated channel gains generated using Sionna RT, as described in Chapter 7. The generated training data has an input resolution of 10×10 and an output resolution of 20×20. The model was trained and obtained a test loss of 0.22. A baseline model was also proposed in Chapter 7 using bilinear interpolation. The baseline has a test loss of 0.10. Hence, the baseline model outperforms the proposed FNO model on the test set. Furthermore, it is observed that the FNO model sometimes produces artefacts in the prediction where it produces high channel gains. This can be seen in Figure 7.3 with a test loss of 0.427. However, in other samples from the test set, the model performs better with no artefacts present. This can be seen in Figure 7.2 with a test loss of 0.0953. Hence, the test loss varies a lot between samples in the test set, which could lead to an overall higher test loss. This could potentially be mitigated by having more data and, hence, a larger test set.

Invariance to Discretisation

One feature of the proposed FNO model is that it is discretisation-invariant both in the number of measurements and the number of interpolated points. To validate this claim, different test sets were generated with the same input discretisation of 10×10 and different output discretisations. The results of the tests can be seen in Figure 7.4. It was found that the test loss increases as the output resolution increases for the FNO and baseline models, as expected. It is also worth noting that the loss is much lower when the input and output discretisation are the same as in the training data. However, test loss does not increase as rapidly between the output resolutions, which the FNO is not trained on. The FNO model is to some extent discretisation-invariant in the output resolution, however, it performs best on data it has seen in training. A way to mitigate this could be to include different discretisations in the training data set.

The FNO model's ability to be input discretisation-invariant was also tested by generating different test sets with different input discretisations and the same output discretisation of 50×50. The results of the tests can be seen in Figure 7.6. The test loss increases as the input resolution increases for the FNO model, which is not what is expected. The FNO performs the best when the input resolution is 10×10, which is the same input resolution as the FNO was trained on. Since the loss increases as the input resolution increases, the FNO is not discretisation-invariant in the input. Again, this could potentially be mitigated by training with multiple different input discretisations.

Training Process

Achieving convergence during training proved challenging in our experiments, as seen in Figure 7.1. Even though the two plots have the same hyperparameters, the training loss only converged for one of them. This inconsistency suggests that we need to further investigate techniques to improve the convergence. Hyperparameter tuning could have potentially addressed this issue by optimising the model parameters for better convergence. Data augmentation could also have been employed. There are several methods in which data augmentation may be employed; for instance, introducing noise to the data or training in various environments may contribute to better performance. Additionally, expanding the training data set by incorporating different discretisations could address the convergence problem as well as the discretisation-invariance problem.

Due to the structure of the output tensor $\mathbf{U}$, the FNO predicts the same output for each point in the input. This means that the FNO needs to learn in the training process to predict the same output for all the input points. This also leads to a higher number of parameters in the model than are required. This could be one of the explanations for the difficulty in training the FNO. The trained FNO does not predict the same in all the outputs.

Hyperparameter Tuning

The hyperparameters were initially chosen to match the ones in [26]. We decided to use 5 Fourier modes in each dimension instead of 12. The reason for this choice was that we could use data at a lower resolution. However, there is a trade-off since less information is used from the input to predict the output. Hence, a higher number of Fourier modes could lead to better performance. This could also improve the performance when predicting output with a discretisation that has not been seen before since more of the input information is used.

We have experimented with learning rates, as it is the most important hyperparameter for tuning networks. We have tried experimenting with a fixed learning rate, however, that did not converge. Consequently, the majority of our tuning focused on adjusting the learning rate decay factor and decay period. Hence, we found that decreasing the learning rate from the initial value of 0.001 by a factor of 0.7 every 50 epochs, similar to what is done in [26], was most effective.

Following a similar approach as [26], we have chosen to have 32 hidden channels. This is another hyperparameter we could have experimented with.

Sensitive to Noise

The sensitivity analysis conducted has yielded insights into the model's behaviour under varying noise conditions. Notably, a sensitivity to small changes in the variance of the added noise was observed in Figure 7.8. The box plots representing test loss for each noise variance yielded test losses widely spread on the y-axis. This indicates that the model is highly sensitive to even small fluctuations in noise variance. The model was not trained on noisy data, which may contribute to its observed sensitivity to noise. Moreover, it is plausible to

think that datasets closely resembling those used for model training exhibit lower sensitivity to noise. This prompts considerations for refining our training data to include datasets that better represent real-world scenarios, thus potentially enhancing the model's robustness to noise perturbations.

9 Conclusion

In this project, we investigated how to utilise FNOs to interpolate channel gains. Furthermore, we aimed to achieve a discretisation-invariant FNO, which is desirable as it would enable the model to be applied to different discretisations beyond those it was trained on. This led to the following problem statement:

How can a Fourier neural operator be utilised for interpolating channel gains, and can a discretisation-invariant operator be achieved?

To answer this, we first presented the theory of FNOs. Afterwards, we introduced Sionna RT, which was used to simulate data at a scene made to digitalise Fredrik Bajers Vej 7. Maxwell's differential equations were presented as they provide the underlying theory of propagating electromagnetic waves. These equations yield the underlying differential equations that our FNO aims to solve to predict the behaviour of electromagnetic waves. The FNO model was constructed based on [26] with our own adaptations. The model was trained on a dataset with input resolution 10×10 and output resolution 20×20. The test loss of the model was compared to that of a baseline model using bilinear interpolation, with selected predictions and their residuals presented for evaluation. To investigate the model's discretisation-invariance, we varied the output resolution for a fixed input resolution and the input resolution for a fixed output resolution. This process was also applied to the baseline model for comparison. Additionally, a sensitivity analysis was conducted by adding Gaussian noise with varying levels of variance to the input, assessing the model's robustness to noise.

In conclusion, we successfully developed a FNO to interpolate channel gains. However, our model achieved a test loss of 0.224, while a baseline model using bilinear interpolation achieved a lower test loss of 0.104. Therefore, it can be concluded that the baseline model outperforms the FNO we trained.

The FNO demonstrated a degree of discretisation-invariance when the output resolution was varied for a fixed input resolution. It achieved the lowest test loss on the output resolution it was trained on, but the differences in the test loss for previously unseen output resolutions were minimal. Therefore, it can be concluded that the FNO shows signs of being discretisation-invariant in terms of output resolution. However, the model was not discretisation-invariant when the input resolution was varied for a fixed output resolution. In fact, the test loss increased as the input resolution increased, which was contrary to our expectations. Additionally, the FNO showed sensitivity to noise, which further impacted its performance.

Overall, while our FNO displayed some promising characteristics, it was outperformed by the baseline model and exhibited limitations in terms of discretisation-invariance and noise sensitivity.

10 Future Work

The proposed FNO model is only trained on a limited area of the larger environment at Fredrik Bajers Vej 7. It would be interesting to train the model on data generated at multiple locations in the environment. A proposed segmentation of the courtyard can be seen in Figure 10.1. This would lead to a larger training set with more variation, which could increase the performance of the model.

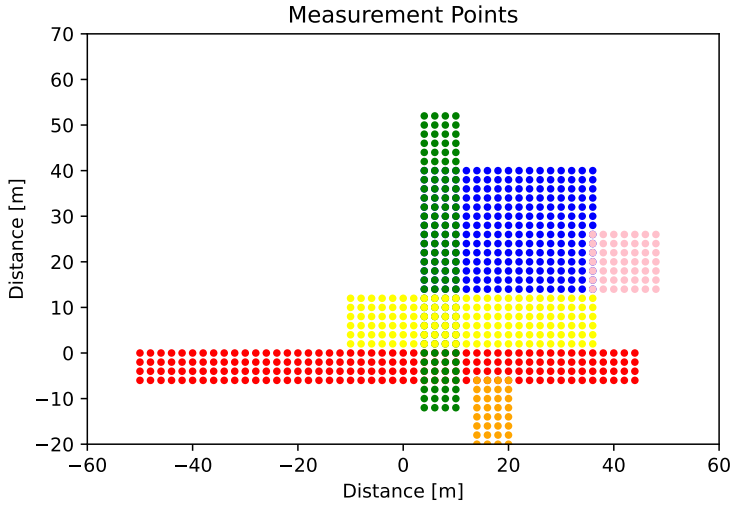


Figure 10.1: A proposed segmentation of the courtyard at Fredrik Bajers Vej 7.

It would also be interesting to investigate whether the model learns parameters specific to the environment it is trained on, such as material properties and building locations. This is feasible due to the inclusion of the location of each measured data point in the input tensor given to the FNO.

This also allows for potentially employing the real-world data described in Section 1.4 for transfer learning purposes. In this scenario, the model would first be trained on simulated data from Sionna RT and then fine-tuned on real-world data. This approach is advantageous because it requires less real data to train the model initially. Furthermore, the real-world data captures details in the environment that simulations do not encompass; for example, the presence of trees, which are absent in the simulation.

It could also be interesting to investigate using other deep learning architectures to address the problem. For instance, one could consider employing a convolutional neural network architecture, such as UNet, which was proposed in [25]. This enables the possibility of incorporating additional input data beyond measured channel gains. For instance, one could include the locations of buildings or a satellite image of the environment alongside the measured channel gains as inputs to the convolutional neural network. However, such an architecture is not discretisation-invariant.

Bibliography

[1] Hans Wilhelm Alt. *Linear Functional Analysis*. Universitext. London: Springer, 2016. ISBN: 978-1-4471-7279-6. DOI: 10.1007/978-1-4471-7280-2. URL: http://link.springer.com/10.1007/978-1-4471-7280-2 (visited on 03/20/2024).

[2] Sheldon Axler. *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Cham: Springer International Publishing, 2024. ISBN: 978-3-031-41025-3. DOI: 10.1007/978-3-031-41026-0. URL: https://link.springer.com/10.1007/978-3-031-41026-0 (visited on 03/01/2024).

[3] Constantine A. Balanis. *Advanced Engineering Electromagnetics*. 2nd edition. Hoboken, NJ: Wiley, Jan. 24, 2012. 1040 pp. ISBN: 978-0-470-58948-9.

[4] Constantine A. Balanis. *Antenna theory: analysis and design*. 3rd ed. Hoboken, NJ: John Wiley, 2005. 1117 pp. ISBN: 978-0-471-66782-7.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, Aug. 17, 2006. 738 pp. ISBN: 978-0-387-31073-2.

[6] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. New York, NY: Springer, 2011. ISBN: 978-0-387-70913-0. DOI: 10.1007/978-0-387-70914-7. URL: https://link.springer.com/10.1007/978-0-387-70914-7 (visited on 03/20/2024).

[7] Donald L. Cohn. *Measure Theory: Second Edition*. Birkhäuser Advanced Texts Basler Lehrbücher. New York, NY: Springer, 2013. ISBN: 978-1-4614-6955-1. DOI: 10.1007/978-1-4614-6956-8. URL: https://link.springer.com/10.1007/978-1-4614-6956-8 (visited on 02/28/2024).

[8] Tao Feng, Timothy R. Field, and Simon Haykin. "Stochastic Differential Equation Theory Applied to Wireless Channels". In: *IEEE Transactions on Communications* 55.8 (Aug. 2007). Conference Name: IEEE Transactions on Communications, pp. 1478–1483. ISSN: 1558-0857. DOI: 10.1109/TCOMM.2007.902531. URL: https://ieeexplore.ieee.org/abstract/document/4291820 (visited on 04/10/2024).

[9] Gerald B. Folland. *Fourier analysis and its applications*. Pure and applied undergraduate texts. OCLC: 913062399. Providence, Rhode Island: American Mathematical Association, 2009. 433 pp. ISBN: 978-0-8218-4790-9.

[10] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, Aug. 8, 2005. 676 pp. ISBN: 978-0-521-83716-3.

[11] David J. Griffiths. *Introduction to electrodynamics*. Fourth edition. Boston: Pearson, 2013. 599 pp. ISBN: 978-0-321-85656-2.

[12] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. June 5, 2023. DOI: 10.48550/arXiv.1606.08415. arXiv: 1606.08415[cs]. URL: http://arxiv.org/abs/1606.08415 (visited on 05/01/2024).

[13] Jakob Hoydis et al. "Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling". In: *2023 IEEE Globecom Workshops (GC Wkshps)*. 2023 IEEE Globecom Workshops (GC Wkshps). Dec. 2023, pp. 317–321. DOI:

10.1109/GCWkshps58843.2023.10465179. URL: https://ieeexplore.ieee.org/document/10465179 (visited on 04/16/2024).

[14] Jakob Hoydis et al. *Sionna: An Open-Source Library for Next-Generation Physical Layer Research*. Mar. 20, 2023. DOI: 10.48550/arXiv.2203.11854. arXiv: 2203.11854[cs,math]. URL: http://arxiv.org/abs/2203.11854 (visited on 02/29/2024).

[15] Masahiro Iwasaki et al. "Transfer Learning-Based Received Power Prediction with Ray-tracing Simulation and Small Amount of Measurement Data". In: *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall). ISSN: 2577-2465. Nov. 2020, pp. 1–6. DOI: 10.1109/VTC2020-Fall49728.2020.9348557. URL: https://ieeexplore.ieee.org/document/9348557 (visited on 04/10/2024).

[16] Fabian Jaensch, Giuseppe Caire, and Begüm Demir. *Radio Map Estimation – An Open Dataset with Directive Transmitter Antennas and Initial Experiments*. Jan. 12, 2024. DOI: 10.48550/arXiv.2402.00878. arXiv: 2402.00878[cs,eess]. URL: http://arxiv.org/abs/2402.00878 (visited on 03/07/2024).

[17] Rahul Jaiswal et al. *Leveraging Transfer Learning for Radio Map Estimation via Mixture of Experts*. Oct. 25, 2023. DOI: 10.36227/techrxiv.24354250.v1.

[18] Tobias Kallehauge et al. "Delivering Ultra-Reliable Low-Latency Communications via Statistical Radio Maps". In: *IEEE Wireless Communications* 30.2 (Apr. 2023). Conference Name: IEEE Wireless Communications, pp. 14–20. ISSN: 1558-0687. DOI: 10.1109/MWC.002.2200372. URL: https://ieeexplore.ieee.org/document/10105152 (visited on 04/10/2024).

[19] Tobias Kallehauge et al. *Experimental Study of Spatial Statistics for Ultra-Reliable Communications*. Feb. 17, 2024. DOI: 10.48550/arXiv.2402.11356. arXiv: 2402.11356[eess]. URL: http://arxiv.org/abs/2402.11356 (visited on 03/07/2024).

[20] Anders E. Kalør, Osvaldo Simeone, and Petar Popovski. "Prediction of mmWave/THz Link Blockages Through Meta-Learning and Recurrent Neural Networks". In: *IEEE Wireless Communications Letters* 10.12 (Dec. 2021). Conference Name: IEEE Wireless Communications Letters, pp. 2815–2819. ISSN: 2162-2345. DOI: 10.1109/LWC.2021.3118269. URL: https://ieeexplore.ieee.org/document/9562750 (visited on 04/10/2024).

[21] Earl J. Kirkland. "Bilinear Interpolation". In: *Advanced Computing in Electron Microscopy*. Ed. by Earl J. Kirkland. Boston, MA: Springer US, 2010, pp. 261–263. ISBN: 978-1-4419-6533-2. DOI: 10.1007/978-1-4419-6533-2_12. URL: https://doi.org/10.1007/978-1-4419-6533-2_12 (visited on 05/10/2024).

[22] Nikola Kovachki et al. "Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs". In: (Aug. 18, 2021). URL: https://arxiv.org/abs/2108.08481.

[23] Enes Krijestorac, Samer Hanna, and Danijela Cabric. "Spatial Signal Strength Prediction using 3D Maps and Deep Learning". In: *ICC 2021 - IEEE International Conference on Communications*. ICC 2021 - IEEE International Conference on

Communications. ISSN: 1938-1883. June 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500970. URL: https://ieeexplore.ieee.org/document/9500970 (visited on 04/10/2024).

[24] Ranjan Kumar and Ranber Singh. *Thermoelectricity and Advanced Thermoelectric Materials*. Google-Books-ID: 2eL7DwAAQBAJ. Woodhead Publishing, June 3, 2021. 358 pp. ISBN: 978-0-12-820439-9.

[25] Ron Levie et al. "RadioUNet: Fast Radio Map Estimation With Convolutional Neural Networks". In: *IEEE Transactions on Wireless Communications* 20.6 (June 2021). Conference Name: IEEE Transactions on Wireless Communications, pp. 4001–4015. ISSN: 1558-2248. DOI: 10.1109/TWC.2021.3054977. URL: https://ieeexplore.ieee.org/document/9354041 (visited on 04/10/2024).

[26] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. May 16, 2021. DOI: 10.48550/arXiv.2010.08895. arXiv: 2010.08895[cs,math]. URL: http://arxiv.org/abs/2010.08895 (visited on 02/06/2024).

[27] Zongyi Li et al. *Neural Operator: Graph Kernel Network for Partial Differential Equations*. Mar. 6, 2020. DOI: 10.48550/arXiv.2003.03485. arXiv: 2003.03485[cs,math,stat]. URL: http://arxiv.org/abs/2003.03485 (visited on 02/07/2024).

[28] *P.2040 : Effects of building materials and structures on radiowave propagation above about 100 MHz*. URL: https://www.itu.int/rec/R-REC-P.2040/en (visited on 04/12/2024).

[29] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. 2nd edition. Amsterdam Boston Heidelberg London New York Oxford Paris San Diego San Francisco Singapore Sydney Tokyo: Morgan Kaufmann, Nov. 25, 2016. 1266 pp. ISBN: 0-12-378580-4.

[30] Petar Popovski. *Wireless connectivity: an intuitive and fundamental guide*. Hoboken, NJ Chichester: Wiley, 2020. 384 pp. ISBN: 978-0-470-68399-6.

[31] *Primer — Sionna 0.16.2 documentation*. URL: https://nvlabs.github.io/sionna/em_primer.html (visited on 04/12/2024).

[32] *Ray Tracing — Sionna 0.16.2 documentation*. URL: https://nvlabs.github.io/sionna/api/rt.html?highlight=fibonacci# (visited on 04/12/2024).

[33] Daniel Romero and Seung-Jun Kim. "Radio Map Estimation: A data-driven approach to spectrum cartography". In: *IEEE Signal Processing Magazine* 39.6 (Nov. 2022), pp. 53–72. ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2022.3200175. URL: https://ieeexplore.ieee.org/document/9931518/ (visited on 02/06/2024).

[34] Daniel Romero et al. *Theoretical Analysis of the Radio Map Estimation Problem*. Nov. 7, 2023. DOI: 10.48550/arXiv.2310.15106. arXiv: 2310.15106[cs,eess,math]. URL: http://arxiv.org/abs/2310.15106 (visited on 03/07/2024).

[35] Raju Shrestha et al. *Radio Map Estimation: Empirical Validation and Analysis*. Jan. 22, 2024. DOI: 10.48550/arXiv.2310.11036. arXiv: 2310.11036[physics]. URL: http://arxiv.org/abs/2310.11036 (visited on 03/07/2024).

[36] J. Tang and F. P. Resurreccion. "1 - Electromagnetic basis of microwave heating". In: *Development of Packaging and Products for Use in Microwave Ovens*. Ed. by Matthew W. Lorence and Peter S. Pesheck. Woodhead Publishing, Jan. 1, 2009, 3–38e. ISBN: 978-1-84569-420-3. DOI: 10.1533/9781845696573.1.3. URL: https://www.sciencedirect.com/science/article/pii/B9781845694203500019 (visited on 03/26/2024).

[37] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. 1st ed. Cambridge University Press, May 26, 2005. ISBN: 978-0-521-84527-4. DOI: 10.1017/CBO9780511807213. URL: https://www.cambridge.org/core/product/identifier/9780511807213/type/book (visited on 02/27/2024).

[38] Hongyu Wei et al. "Techniques to enhance magnetic permeability in microwave absorbing materials". In: *Applied Materials Today* 19 (June 1, 2020), p. 100596. ISSN: 2352-9407. DOI: 10.1016/j.apmt.2020.100596. URL: https://www.sciencedirect.com/science/article/pii/S2352940720300433 (visited on 03/26/2024).

[39] Xin Zhang et al. *Cellular Network Radio Propagation Modeling with Deep Convolutional Neural Networks*. Oct. 5, 2021. arXiv: 2110.01848[cs,math]. URL: http://arxiv.org/abs/2110.01848 (visited on 02/27/2024).

A Appendix

A.1 Fourier Transform

In this section definitions and theorems related to the Fourier transform of vector-valued functions are presented. This will be used to develop the Fourier neural operator.

The Fourier transform is first defined on a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ and then extended to a vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$.

Definition A.1 (Fourier Transform)
Let $f : \mathbb{R}^n \to \mathbb{R}$. Then its Fourier transform $\hat{f} = \mathcal{F}(f)$ is defined as

$$\hat{f}(\boldsymbol{\omega}) = \mathcal{F}(f)(\boldsymbol{\omega}) = \int_{\mathbb{R}^n} \mathrm{e}^{-i\boldsymbol{\omega}^\intercal \mathbf{x}} f(\mathbf{x}) \mathrm{d}\mathbf{x} \quad \text{for } \boldsymbol{\omega} \in \mathbb{R}^n.$$

If it is assumed that $\hat{f} \in L^1$ then the inverse Fourier transform $f = \mathcal{F}^{-1}(\hat{f})$ is defined as

$$f(\mathbf{x}) = \mathcal{F}^{-1}(f)(\mathbf{x}) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} \mathrm{e}^{i\boldsymbol{\omega}^\intercal \mathbf{x}} \hat{f}(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \quad \text{for } \mathbf{x} \in \mathbb{R}^n.$$

[9, pp. 243–244]

Definition A.2 (Fourier Transform of Vector-Valued Functions)
Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a vector-valued function. Then its Fourier transform $\hat{\mathbf{f}} = \mathcal{F}(\mathbf{f})$ is defined as the Fourier transform element-wise

$$\hat{f}_j(\boldsymbol{\omega}) = (\mathcal{F}(\mathbf{f}))_j(\boldsymbol{\omega}) = \int_{\mathbb{R}^n} \mathrm{e}^{-i\boldsymbol{\omega}^\intercal \mathbf{x}} f_j(\mathbf{x}) \mathrm{d}\mathbf{x} \quad \text{for } \boldsymbol{\omega} \in \mathbb{R}^n, \quad j = 1, \ldots, m.$$

If it is assumed that $\hat{f}_i \in L^1$ then the inverse Fourier transform $\mathbf{f} = \mathcal{F}^{-1}(\hat{\mathbf{f}})$ is defined as

$$f_j(\boldsymbol{x}) = (\mathcal{F}^{-1}(\mathbf{f}))_j(\boldsymbol{x}) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} \mathrm{e}^{i\boldsymbol{\omega}^\intercal \mathbf{x}} \hat{f}_j(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \quad \text{for } \boldsymbol{x} \in \mathbb{R}^n, \quad j = 1, \ldots, m.$$

The Fourier transform of a matrix-valued function $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ is likewise defined as the Fourier transform element-wise. [9, pp. 243–244][26, p. 5]

A.2 Convolution

In this section, the convolution theorem between a vector-valued function and a matrix-valued function is presented. Initially, convolution for scalar-valued functions is defined followed by stating the convolution theorem.

Definition A.3 (Convolution for Scalar-Valued Functions)
Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. Then the convolution between the two functions is defined as

$$f * g(\mathbf{x}) = \int_{\mathbb{R}^n} f(\mathbf{y})g(\mathbf{x} - \mathbf{y})\mathrm{d}\mathbf{y} \quad \text{for } \mathbf{x} \in \mathbb{R}^n,$$

provided that the integral exists. [9, p. 242]

Theorem A.4 (Convolution Theorem for Scalar-Valued Functions)
Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. If $g \in L^1$ and $f * g \in L^1$ then

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g).$$

[9, Theorem 7.8(d)]

The proof is omitted but can be found in [9, p. 244].

The convolution theorem will now be extended to the convolution between a vector-valued function and a matrix-valued function. This theorem is used in the derivation of the Fourier neural operator.

Theorem A.5 (Convolution Theorem for Matrix- and Vector-Valued Functions)
Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$. It is assumed that all the entries of $\mathbf{f}$ and $\mathbf{g}$ are in L^1. Then convolution between the two functions is given by

$$\mathbf{f} * \mathbf{g}(\mathbf{x}) = \int_{\mathbb{R}^n} \mathbf{f}(\mathbf{y})\mathbf{g}(\mathbf{x} - \mathbf{y})\mathrm{d}\mathbf{y} \quad \text{for } \mathbf{x} \in \mathbb{R}^n.$$

The Fourier transform of the convolution is the matrix-vector product of $\hat{\mathbf{f}}$ and $\hat{\mathbf{g}}$

$$\mathcal{F}(\mathbf{f} * \mathbf{g})(\boldsymbol{\omega}) = \hat{\mathbf{f}}(\boldsymbol{\omega})\hat{\mathbf{g}}(\boldsymbol{\omega}) \quad \text{for } \boldsymbol{\omega} \in \mathbb{R}^n.$$

Proof
The convolution can be rewritten as

$$\begin{aligned}
\mathbf{f} * \mathbf{g}(\mathbf{x}) &= \int_{\mathbb{R}^n} \mathbf{f}(\mathbf{y})\mathbf{g}(\mathbf{x}-\mathbf{y})\mathrm{d}\mathbf{y} \\
&= \int_{\mathbb{R}^n} \begin{bmatrix} \sum_{j=1}^m f_{j,1}(\mathbf{y})g_j(\mathbf{x}-\mathbf{y}) \\ \sum_{j=1}^m f_{j,2}(\mathbf{y})g_j(\mathbf{x}-\mathbf{y}) \\ \vdots \\ \sum_{j=1}^m f_{j,m}(\mathbf{y})g_j(\mathbf{x}-\mathbf{y}) \end{bmatrix} \mathrm{d}\mathbf{y} \\
&= \sum_{j=1}^m \begin{bmatrix} (f_{j,1} * g_j)(\mathbf{x}) \\ (f_{j,2} * g_j)(\mathbf{x}) \\ \vdots \\ (f_{j,m} * g_j)(\mathbf{x}) \end{bmatrix}.
\end{aligned}$$

The Fourier transform is then applied

$$\begin{aligned}
\mathcal{F}(\mathbf{f} * \mathbf{g})(\boldsymbol{\omega}) &= \mathcal{F}\left(\sum_{j=1}^m \begin{bmatrix} (f_{j,1} * g_j)(\mathbf{x}) \\ (f_{j,2} * g_j)(\mathbf{x}) \\ \vdots \\ (f_{j,m} * g_j)(\mathbf{x}) \end{bmatrix} \right) \\
&= \sum_{j=1}^m \begin{bmatrix} \mathcal{F}(f_{j,1} * g_j)(\boldsymbol{\omega}) \\ \mathcal{F}(f_{j,2} * g_j)(\boldsymbol{\omega}) \\ \vdots \\ \mathcal{F}(f_{j,m} * g_j)(\boldsymbol{\omega}) \end{bmatrix} \\
&= \begin{bmatrix} \sum_{j=1}^m \hat{f}_{j,1}(\boldsymbol{\omega})\hat{g}_j(\boldsymbol{\omega}) \\ \sum_{j=1}^m \hat{f}_{j,2}(\boldsymbol{\omega})\hat{g}_j(\boldsymbol{\omega}) \\ \vdots \\ \sum_{j=1}^m \hat{f}_{j,m}(\boldsymbol{\omega})\hat{g}_j(\boldsymbol{\omega}) \end{bmatrix} \\
&= \hat{\mathbf{f}}(\boldsymbol{\omega})\hat{\mathbf{g}}(\boldsymbol{\omega}).
\end{aligned}$$

■