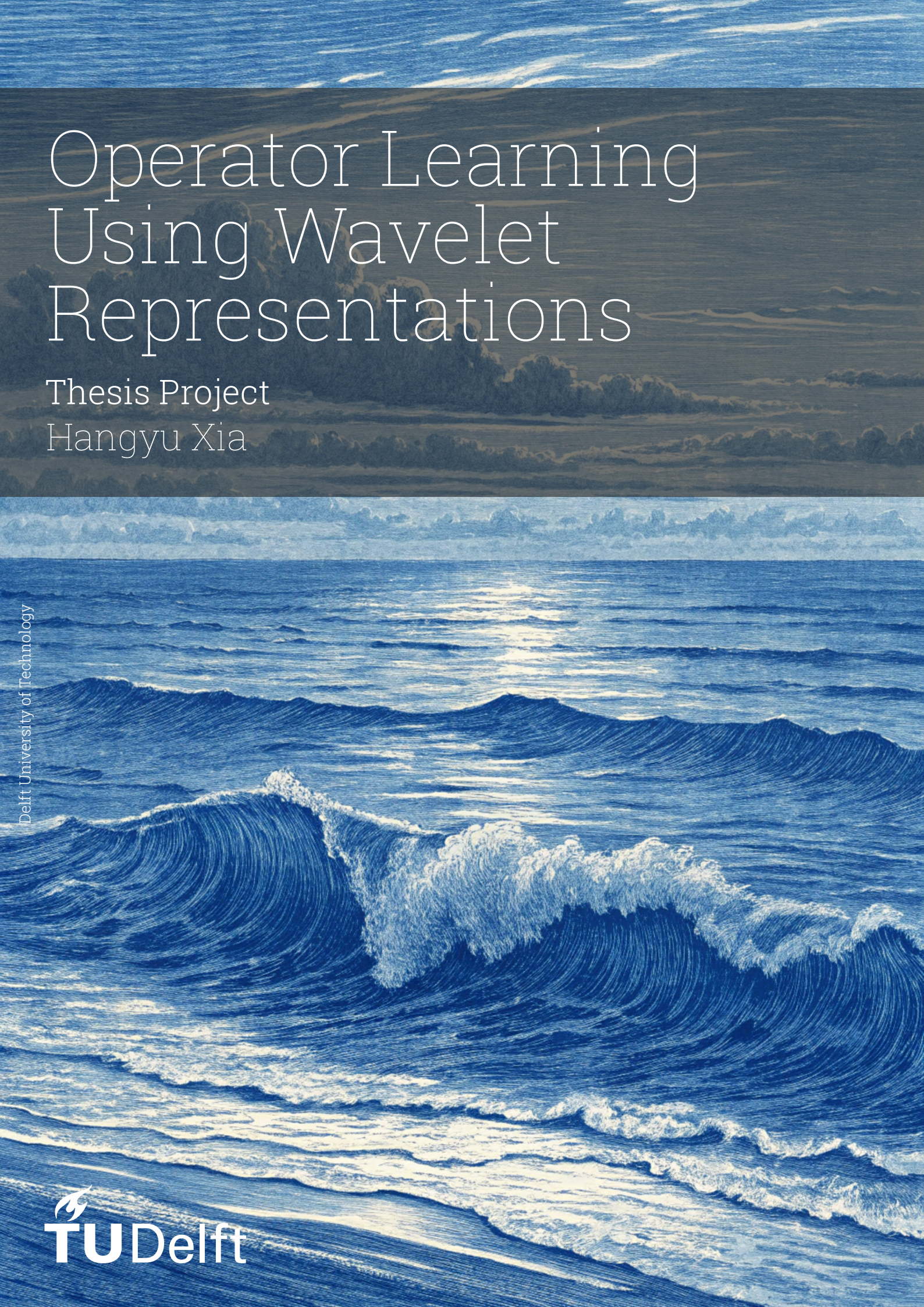
Operator Learning Using Wavelet Representations

Thesis Project

Hangyu Xia



Delft University of Technology

TU Delft

Operator Learning Using Wavelet Representations

by

Hangyu Xia

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday May 8, 2026 at 2:00 PM.

Student number:	5914779	
Project duration:	December 10, 2024 – April 29, 2026	
Thesis committee:	Dr. A. Heinlein,	TU Delft, supervisor
	Dr. W. van Horssen,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Acknowledgements

Through the completion of my studies and the writing of this thesis, I have systematically reviewed and strengthened my knowledge in mathematics and related quantitative fields. This process has significantly enhanced my analytical thinking, problem-solving abilities, and academic rigor. The journey has been both intellectually demanding and highly rewarding, providing me with a solid foundation for my future academic and professional development.

I would like to express my sincere gratitude to my supervisor, Dr. Alexander Heinlein, for his patient and dedicated guidance throughout the entire research process. His feedback was always precise, insightful, and highly targeted to the key issues of my work, which greatly improved both the depth and clarity of this thesis. I am especially thankful for the considerable degree of academic freedom and trust he granted me, allowing me to explore my ideas independently while still benefiting from his timely and thoughtful advice.

I am deeply grateful to my family for their unwavering support in both material and emotional aspects during my studies. Their constant care, deep trust in my abilities, and full support of my academic and career planning have given me the confidence and stability to move forward. I would also like to thank Mei for the companionship, encouragement, and positive presence throughout this important stage of my life.

Overall, this period of study has greatly broadened my horizons and enriched my academic perspective as well as personal growth. It has been an exceptionally meaningful and worthwhile experience that will continue to influence my future path.

Hangyu Xia
Delft, 28 April 2026

Declaration on the Use of Artificial Intelligence

This thesis is written with the aid of generative artificial intelligence (AI) tools, including ChatGPT and Recraft, to support the development of written content and code.

Specifically, AI tools were used for:

1. brainstorming and structuring of ideas,
2. language refinement and stylistic editing of the text,
3. generating and improving code snippets where applicable,
4. generating the visual design of the thesis cover and the illustrations.

All AI-generated outputs were critically reviewed, validated and modified by the author. All content in this thesis was produced with substantial human oversight and intellectual contribution. The author retains full responsibility for the accuracy, integrity, and originality of the content presented in this thesis.

The use of AI in this work complies with the applicable academic integrity guidelines of the institution.

Abstract

Operator learning has recently emerged as an effective approach for approximating mappings between function spaces. In this work, we study operator learning using wavelet representations (OL-Wavelet). It represents functions via discrete wavelet transforms (DWT), uses a neural network to learn mappings between the resulting wavelet coefficients, and reconstructs the output functions from the predicted coefficients to learn operators between functions. Several representative operators are considered, including 1D translation operators, operators mapping different source terms of 1D Poisson equations to the corresponding solutions, operators mapping different initial conditions of an 1D Diffusion euqation and an 1D Burgers' equation to their terminal solutions, and operators approximating functions with piecewise functions on dyadic partitions. We analyze the sources of error in the proposed framework by decomposing the total error into the neural network generalization error and the reconstruction error. This perspective provides insight into how different components of the model contribute to the final prediction accuracy. In a series of numerical experiments, we compare models trained with different numbers of DWT coefficients for function representation, motivated by the fact that functions can be well approximated using only a subset of DWT coefficients, which also reduces computational cost. The experimental results show that our model achieves higher prediction accuracy than comparable models based on Fourier transforms on certain tasks. Moreover, we observe a non-monotonic relationship between the model's prediction accuracy and the number of DWT coefficients used. In addition, experiments show that using a single neural network to learn the mappings among all wavelet coefficients is often more accurate than using multiple neural networks to separately learn the mappings of wavelet coefficients at different decomposition levels. Overall, the study demonstrates the effectiveness of the proposed model and analyzes the sources of its errors, thereby revealing its strengths and limitations.

Contents

Acknowledgements iii

Declaration on the Use of Artificial Intelligence v

Abstract vii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Literature Review . . . 2
  - 1.2.1 Typical Operator Learning Models . . . 2
  - 1.2.2 Categorization . . . 3
  - 1.2.3 Applications . . . 4
  - 1.2.4 Research Gaps . . . 5
- 1.3 Research Questions . . . 9
  - 1.3.1 Main Research Question . . . 9
  - 1.3.2 Sub-Questions . . . 9
- 1.4 Thesis Outline . . . 10

2 Fundamentals 11
- 2.1 Function Approximation . . . 11
  - 2.1.1 Principles of Function Approximation . . . 11
  - 2.1.2 Fourier Transform . . . 12
  - 2.1.3 Wavelet Transform . . . 12
  - 2.1.4 Gibbs Phenomenon for 1D Functions . . . 15
  - 2.1.5 Wavelet Approximation and Gibbs Phenomenon . . . 15
- 2.2 Neural Network . . . 16
  - 2.2.1 Mathematics of Neural Networks . . . 16
  - 2.2.2 Typical Structures of Neural Networks . . . 16
  - 2.2.3 Optimization Algorithms . . . 17
  - 2.2.4 Dataset . . . 17
- 2.3 Mathematics of Operator Learning . . . 18

3 Proposed Method 21
- 3.1 Design Rationale . . . 21
- 3.2 Overall Architecure . . . 21
- 3.3 Model Components . . . 24
- 3.4 Error Analysis . . . 26

4 Numerical Experiments 29
- 4.1 Overall Settings . . . 29
  - 4.1.1 Pipeline . . . 29
  - 4.1.2 Selection of Operators . . . 30
  - 4.1.3 Hyperparameters Settings . . . 31
- 4.2 Translation Operator . . . 32
  - 4.2.1 Problem Setup . . . 32
  - 4.2.2 Dataset . . . 32
  - 4.2.3 Experiment Result . . . 33
- 4.3 Translation Operator with Height Variation on Dyadic Partitions . . . 36
  - 4.3.1 Problem Setup . . . 37

4.3.2 Dataset . . . 37
4.3.3 Experiment Result . . . 37
4.4 Poisson Equation Solver . . . 40
4.4.1 Problem Setup . . . 40
4.4.2 Dataset . . . 41
4.4.3 Experiment Result . . . 41
4.5 Diffusion Equation Solver . . . 43
4.5.1 Problem Setup . . . 43
4.5.2 Dataset . . . 44
4.5.3 Experiment Result . . . 44
4.6 Burgers' Equation Solver . . . 46
4.6.1 Problem Setup . . . 46
4.6.2 Dataset . . . 47
4.6.3 Experiment Result . . . 47
4.7 Approximation by Piecewise Constant Functions on Dyadic Partitions . . . 49
4.7.1 Problem Setup . . . 49
4.7.2 Dataset . . . 50
4.7.3 Experiment Result . . . 50

5 Discussion **55**
5.1 Interpretation of Experimental Results . . . 55
5.1.1 Error Decomposition and the Role of the Cross Term . . . 55
5.1.2 Comparison Between DFT and DWT-Flat . . . 55
5.1.3 The Limitation of DWT-LevelWise . . . 56
5.2 Effect of the Representation Dimension . . . 56
5.2.1 The Dual Role of the Truncation Parameter K . . . 56
5.2.2 Behavior of the Reconstruction and Coefficient Errors . . . 56
5.2.3 Two Regimes for Choosing the Representation Dimension . . . 56
5.3 Applicability Analysis of the Level-wise Wavelet Mapping . . . 57

6 Conclusion **59**
6.1 Main Results . . . 59
6.2 Limitations and Challenges . . . 60
6.2.1 Limited Range of Benchmark Tasks . . . 60
6.2.2 Incomplete Understanding of the Prediction Error . . . 60
6.2.3 Restriction to One-dimensional Problems . . . 61
6.3 Future Work . . . 61
6.3.1 Expanding the Range of Operator Learning Tasks . . . 61
6.3.2 Developing a More Rigorous Theoretical Analysis . . . 61
6.3.3 Extending the Framework to Higher-dimensional Problems . . . 61
6.3.4 Exploring Adaptive Coefficient Selection Strategies . . . 61
6.3.5 Investigating Alternative Wavelet Bases . . . 61

References **63**

A A Low-Frequency Approximation Test Using DFT and DWT **69**
A.1 Main Idea . . . 69
A.2 Problem Setup . . . 71
A.3 Experiment Result . . . 72
A.4 Conclusion . . . 73

B An Empirical Study on the Effect of Network Width in Operator Learning **75**
B.1 Problem Setup . . . 75
B.2 Experiment Result . . . 76
B.3 Main Observation . . . 76

C Code Availability **79**

1

Introduction

This thesis explores a newly proposed wavelet-based neural operator model and the principles behind it. The field of neural operators encompasses a wide range of approaches, and the model in this work represents just one of them which exploits the fact that wavelet-based approximations can achieve higher accuracy for certain non-smooth functions. To clearly explain the origins, development, and rationale of this model, it is necessary to provide a comprehensive introduction to the relevant background knowledge. This chapter serves to provide background knowledge and motivation of the research, along with literature review, reserach questions and thesis outline.

1.1. Background

The operator learning [37] is an important branch of scientific machine learning [18] [70], a field that integrates machine learning with scientific computing to model and solve complex systems [53]. Different from traditional machine learning, which often serves as a black box, scientific machine learning emphasizes integrating physical laws, domain knowledge, and numerical solvers to build models which are not only accurate in prediction but also interpretable and physically consistent. Based on the existing review [18], scientific machine learning can be broadly categorized into several major paradigms in the Figure 1.1, including operator learning [37], physics-informed neural networks (PINNs) [11], hybrid modeling [62] and surrogate modeling [3] techniques, random feature methods [68] and probabilistic and Bayesian approaches [35]. These methods have found applications across diverse domains, including fluid dynamics [40], material design [52], climate modeling [31], molecular simulation [61], and biomedical engineering [2]. Therefore, scientific machine learning is becoming a powerful tool for advancing both fundamental science and engineering innovation to accelerate simulations, reduce computational costs, and enable data-efficient modeling.

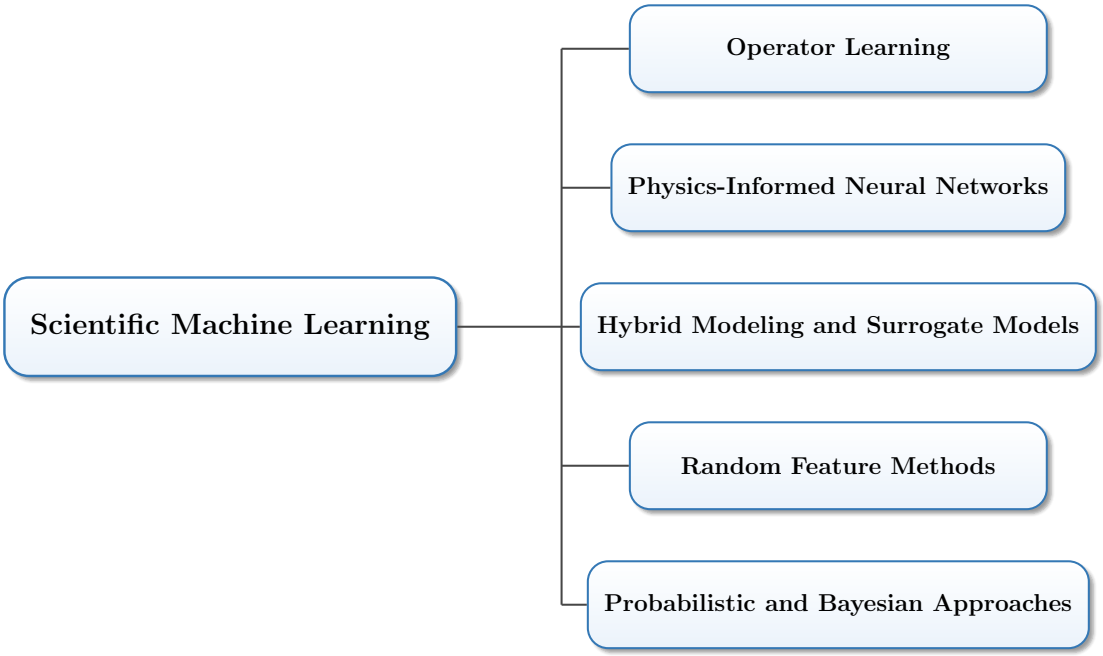


Figure 1.1: A Categorization of Scientific Machine Learning

Within this broad landscape, operator learning has emerged as a particularly important branch. While many scientific machine learning methods focus on approximating solutions of specific equations or models, operator learning aims to approximate entire mappings between infinite-dimensional function spaces. In other words, the goal of operator learning is to learn a function mapping. It offers a scalable and flexible framework for solving families of problems governed by partial differential equations (PDEs) and other complex operators. Therefore, operator learning can also be leveraged in a wide of range of applications to accelerate simulations, attracting growing attention as a research frontier.

1.2. Literature Review

Up to now, operator learning has been studied from multiple perspectives in terms of concepts, methods, and applications. Therefore, there are many references available. Some literature explores the definition and standards of operator learning, while others discuss specific models and their practical applications.

1.2.1. Typical Operator Learning Models

In recent years, many neural operator models have emerged. Early developments in this direction include the Deep Operator Network (DeepONet) [48], which introduces a branch–trunk architecture to approximate nonlinear operators. By decomposing the input function and spatial coordinates into separate representations, DeepONet provides a flexible and theoretically grounded framework for operator approximation. However, its formulation may face challenges in scalability and efficiency when applied to high-dimensional problems.

Building upon this foundation, a large body of work has focused on kernel-based neural operators, which approximate operators through integral transformations. A representative example is the Fourier Neural Operator (FNO) [43], which leverages spectral convolution in the Fourier domain to capture global dependencies efficiently. Similarly, the Graph Neural Operator (GNO) [42] extends this idea to irregular domains by incorporating graph-based structures, enabling greater flexibility in handling non-uniform discretizations. Subsequent research has explored various extensions to

improve expressiveness and scalability. Multi-scale representations have been introduced through approaches such as the Multiwavelet-based Operator Learning [27] and the Multi-grid Tensorized Fourier Neural Operator [36], which aim to capture both coarse and fine-grained features of the underlying solution space. Architectural refinements, such as the U-shaped Neural Operator [59], adopt encoder–decoder structures to enhance the integration of local and global information.

In addition to Fourier-based methods, alternative basis representations have been investigated. The Wavelet Neural Operator (WNO) [73], Spectral Neural Operator (SNO) [23], and Hermite Neural Operator [5] extend the operator learning framework by employing different functional bases, aiming to better capture localized or non-stationary behaviors. Furthermore, the Complex Neural Operator [71] enhances representational capacity by extending the formulation to complex-valued domains. Alongside these neural operator architectures, data-driven and reduced-order approaches have also been explored. For instance, PCA-Net [8] integrates dimensionality reduction techniques into the learning process, while DGN [25] focuses on efficient surrogate modeling through learned low-dimensional representations. These methods typically offer improved computational efficiency, though sometimes at the cost of reduced generalization capability.

Among recent architectures, Physics-Informed Neural Operators (PINO) [45], Fourier-Embedded Deep Operator Networks (FEDONet) [65], and General Neural Operator Transformer (GNOT) [28] represent important directions in operator learning. PINO enhances standard neural operators by incorporating physical constraints from governing partial differential equations into the training process, which improves physical consistency and data efficiency, especially in low-data regimes. FEDONet extends the classical DeepONet framework by introducing Fourier feature embeddings, enabling better representation of high-frequency and multi-scale patterns while preserving the flexibility of branch-trunk operator decomposition. GNOT replaces conventional kernel-based or spectral operators with attention mechanisms, providing stronger expressive power for handling multiple input functions and irregular meshes. Together, these architectures reflect the major trends in modern neural operator research: physics-guided learning, spectral enhancement, and attention-based global operator modeling.

Overall, existing operator learning methods differ in their representation strategies, computational efficiency, and generalization capabilities across discretizations and problem settings. Function decomposition approaches such as DeepONet [48] provide flexible operator representations but may face scalability challenges in high-dimensional scenarios. In contrast, kernel-based and spectral methods, including FNO [43] and its variants, achieve greater efficiency by exploiting structured transformations, particularly in the frequency domain. Extensions incorporating multi-scale representations and alternative basis functions further improve the ability to capture complex spatial patterns, albeit with increased model complexity. Meanwhile, reduced-order approaches such as PCA-Net [8] and DGN [25] offer improved efficiency through low-dimensional representations, but may be limited in generalization. Despite these advances, challenges remain in terms of data efficiency, robustness under distribution shifts, and theoretical understanding of generalization properties.

1.2.2. Categorization

From the perspective of representation learning, we propose to categorize neural operators into two classes: end-to-end operators and decoupled representation operators. This categorization is motivated by our focus on how functions are transformed throughout the learning process, namely whether the representation of functions is learned jointly with the operator mapping itself, or constructed separately before operator learning. Such a viewpoint highlights the role of latent functional representations in operator approximation and provides a clearer framework for understanding different architectural design principles.

End-to-end neural operators learn the function representation and the function mapping jointly, without relying on predefined coefficient representations. In this setting, the model directly takes discretized function observations as input and produces the target function values in a unified

framework, where the latent representation is implicitly optimized together with the operator mapping. This allows the model to adaptively discover task-dependent representations and directly approximate nonlinear mappings between infinite-dimensional function spaces.

Decoupled representation operators construct the representation of functions before learning the operator. The essence of this type of model is a reduced order model (ROM) [49] combined with regression learning in a vector space. In this paradigm, high-dimensional functional data are first projected into a compact latent space through predefined or learned basis representations, and the operator is then learned as a mapping between these low-dimensional coefficient vectors. As a result, the quality of the functional representation becomes a critical factor in the final approximation performance. Compared with end-to-end neural operators, they are often not regarded as entirely new architectures, since the operator-learning stage is essentially performed in a finite-dimensional vector space after representation construction. Nevertheless, they are equally effective for learning mappings between functions, according to experiments by PCA-Net [8] and SNO [23].

Table 1.1 summarizes the two representative types of neural operator models.

Table 1.1: Neural operator models classification based on representation

Representation Type	**Models**
End-to-end Neural Operators	DeepONet [48], FNO [43], DGN [25], GNO [42], WNO [73], Multigrid Tensorized Fourier Neural Operator [36] , Multiwavelet-based Operator Learning [27], U-shaped Neural Operator [59], Complex Neural Operator [71], Hermite Neural Operator [5], PINO [45], FEDONet [65], GNOT [28]
Decoupled Representation Operators	PCA-Net [8], SNO [23]

1.2.3. Applications

Neural operator models have many applications. In many industries, there exist problems that can be described by PDEs or some operators. Once trained, neural operators can rapidly approximate solutions to these problems, enabling efficient analysis and decision-making. As a result, they provide a powerful tool for addressing practical challenges in scientific computing and engineering applications. A few neural operators for applications are listed in the Table 1.2.

Table 1.2: Application Areas of Operator Learning

Application Category	Representative Models
Computational Fluid Dynamics	Geometry-informed Neural Operator [44]
Solid Mechanics	Geometry Physics Neural Operator [32]
Heat Transfer	Heat Transfer Neural Operators [19]
Energy Systems	Causal Fuzzy Neural Operator [50]
Battery Modeling	Physics-informed Adversarial Network [75]
Reservoir Simulation and Porous Media Flow	Feature Attention-based Deep Neural Operator [47]
Weather and Climate Modeling	Adaptive Fourier Neural Operator [38], Spherical Neural Operator Network [46]
Ocean and Hydrodynamic Modeling	Fourier Neural Operator with Convolutional Long Short-term Memory Units [55]
Electromagnetic Simulation	Extended Fourier Neural Operator [56]
Robotics and Dynamic Systems	Optical Flow-Integrated Markov Neural Operator [7]
Computational Biology	Physics-guided Neural Operator Learning Model [77], Variable Spiking Wavelet Neural Operator [24]
Finance	Pricing Operator Network [29]
Digital Twin Systems	OceanNet [13]

1.2.4. Research Gaps

Despite the rapid development of neural operators, several important gaps remain in the current literature.

Current research on neural operators has primarily focused on end-to-end architectures, while decoupled representation-based operator learning remains relatively underexplored. From this perspective, SNO [23] provides a representative decoupled framework by first representing functions through spectral coefficients and then learning the mapping between coefficient spaces via neural networks. This paradigm offers improved interpretability and reduced computational complexity, but its effectiveness fundamentally depends on the choice of basis functions, as shown in the experiments of SNO [23].

First, existing SNO models predominantly rely on Fourier bases for function representation. While Fourier series provide efficient global approximations for smooth functions, they impose an implicit global smoothness assumption that limits their approximation capability for functions with discontinuities, sharp gradients, or localized structures. In particular, the Gibbs phenomenon [26] introduces persistent oscillatory errors near discontinuities, and these representation errors may propagate through the coefficient regression stage, ultimately degrading the quality of operator approximation.

Under this representation framework, another fundamental limitation arises from the lack of spatial localization and multiresolution adaptivity in Fourier-based representations. As a result, SNO may struggle to capture multi-scale patterns or spatially localized dynamics that frequently appear in practical operator learning problems. In contrast, wavelet bases naturally provide both spatial-frequency localization and multiscale representations, making them more suitable for approximating non-smooth and locally varying functions. Although wavelet-based neural operator methods such as WNO [73] have demonstrated the empirical potential of wavelet representations, their integration remains largely within end-to-end architectures, leaving wavelet-based decoupled operator learning insufficiently explored.

Building upon these observations, a more fundamental theoretical gap remains: the interaction

between function representation error and coefficient-space regression error is still not well understood, especially under non-Fourier basis representations. Existing studies mainly focus on the operator approximation itself, with limited analysis of how basis-induced approximation errors influence the subsequent regression process and the final reconstruction quality. This reveals a clear research gap: the absence of a wavelet-based decoupled neural operator framework that can leverage localized multiresolution representations while providing a principled understanding of representation-induced operator approximation errors.

Based on the research gaps identified above, we conducted two preliminary experiments, as presented in the appendices, to further investigate the key factors affecting the performance of decoupled neural operators. From the perspective of decoupled operator learning, the overall prediction error can be fundamentally decomposed into two sources: the function representation error induced by the chosen basis functions, and the latent-space mapping error introduced by the neural network regression model. Motivated by this observation, the first experiment examines whether wavelet-based representations can achieve lower approximation errors than Fourier-based representations for functions with discontinuities or localized features. The second experiment investigates the regression error of the neural network itself, focusing on whether the approximation performance of a fully connected network in the latent coefficient space is significantly affected by the network width.

The first experiment in Appendix A compares the discrete Fourier transform (DFT) [69] and the discrete wavelet transform (DWT) [15] for function approximation. We construct a multiscale function on the periodic domain $[0, 1]$ defined by

$$f(x) = \sum_{j=2}^{8} \epsilon_j f_j(x),$$

where $f_j \in V_j$ and V_j denotes the approximation space at level j of the Haar wavelet [67]. The weights designed to be decaying are given by

$$\epsilon_j = 2^{-\beta(j-2)}, \qquad \beta > 0.$$

The function is discretized on a uniform grid with $M = 1024$.

We compare DFT and Haar DWT approximations under an equal coefficient budget

$$K = 2^m, \qquad K \in \{4, 8, 16, 32, 64, 128, 256\}.$$

For each K, the DFT approximation retains the lowest K Fourier modes, while the DWT approximation retains the first K coefficients in the multiresolution ordering.

The resulting approximations are shown in Figure 1.2. From the figure, we observe that for this class of functions, the DWT consistently achieves higher approximation accuracy than the DFT.

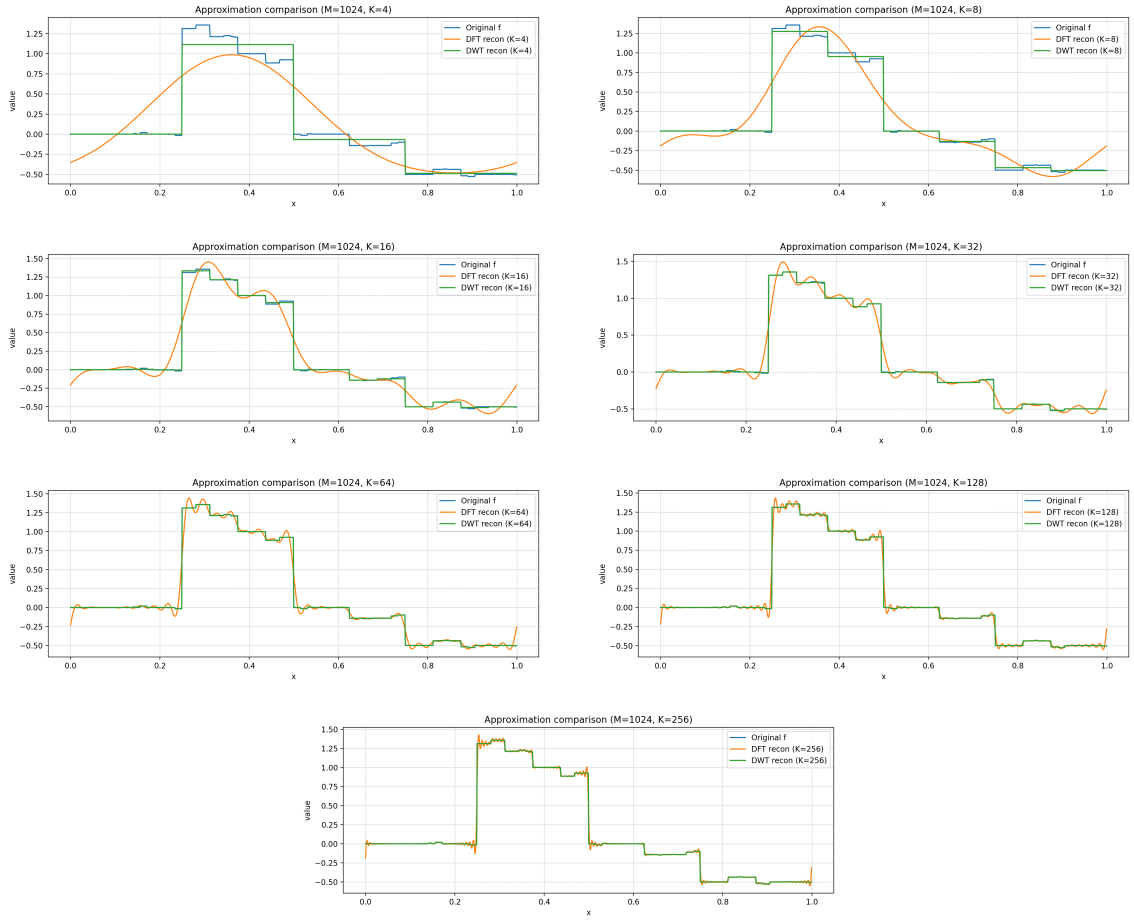


Figure 1.2: DFT and DWT Approximations under Equal Coefficient Budget K

The second experiment in Appendix B explores the dependence of neural operator approximation error on neural network width. We consider a supervised operator learning problem, where the input function $u \in L^2([-1,1])$ is parameterized as

$$u(x) = a\sin(5x) + b\sin(7x),$$

with coefficients $a, b \sim \mathcal{U}([-3,3])$. The target operator $\mathcal{G}$ is defined pointwise by

$$v(x) = \mathcal{G}(u)(x) = 1 - u(x)^2.$$

To obtain a finite-dimensional learning problem, both input and output functions are discretized on a uniform grid of size M,

$$x_j = -1 + \frac{2(j-1)}{M-1}, \qquad j = 1, \dots, M.$$

The discretized input and output are then represented as vectors

$$\mathbf{u} = (u(x_1), \dots, u(x_M)) \in \mathbb{R}^M, \qquad \mathbf{v} = (v(x_1), \dots, v(x_M)) \in \mathbb{R}^M,$$

so that the operator learning task is reduced to approximating a mapping

$$\mathcal{G}_M : \mathbb{R}^M \to \mathbb{R}^M.$$

A dataset $\mathcal{D} = \{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^{|\mathcal{D}|}$ is generated by independently sampling the coefficients a and b. We approximate the operator using fully connected neural networks $\mathcal{N}_\theta : \mathbb{R}^M \to \mathbb{R}^M$, where θ denotes the trainable parameters. Each network consists of three linear layers with hidden width W and pointwise activation functions between consecutive layers. The model is trained by minimizing the mean squared error

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{u}_i, \mathbf{v}_i) \in \mathcal{D}} \|\mathcal{N}_\theta(\mathbf{u}_i) - \mathbf{v}_i\|_2^2 .$$

The dataset is split into training and test sets with a ratio of $9{:}1$. For each choice of hidden width W, the training procedure is repeated with multiple random parameter initializations to reduce the influence of optimization randomness.

The training and test error curves, shown in Figure B.1, indicate that the approximation error varies with network width in a non-monotonic manner, rather than consistently decreasing as the network becomes wider. This behavior is observed under both ReLU [6] and SiLU [22] activation functions, suggesting that the phenomenon is not tied to a specific activation choice but reflects an intrinsic sensitivity of the regression model to network architecture. These results demonstrate that, even for this relatively simple operator learning task, the latent-space regression error remains strongly dependent on architectural design, and increasing network width alone does not necessarily guarantee improved approximation performance.

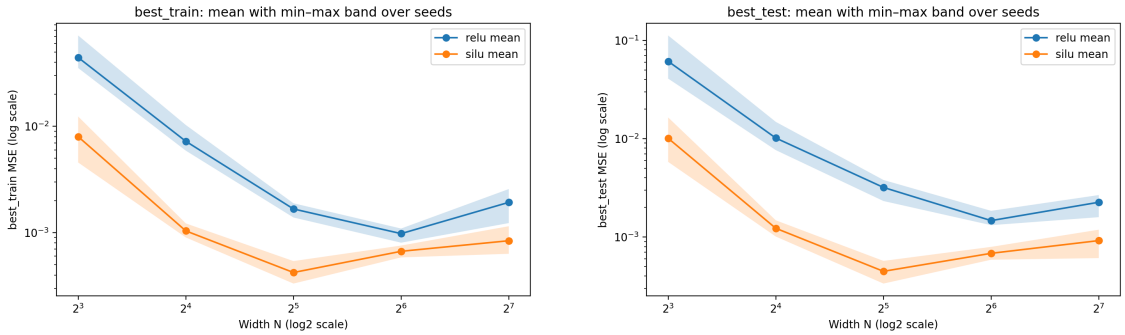


Figure 1.3: Change of Train and Test Loss with Network Width

Therefore, we obtained two key observations as follows:

1. As shown in the first experiment, for certain classes of functions, wavelet-based approximations can outperform Fourier-based ones when both methods reconstruct the function using the same number of selected coefficients. In the Fourier case, this means retaining only a subset of frequency components, while in the wavelet case it involves selecting coefficients across different resolution levels.
2. As shown in the second experiment, the width of a neural network influences the error of neural-network-based operator models, and this dependence is not strictly monotonic. For instance, the error may initially decrease and then increase as the width grows.

This naturally motivates the investigation of replacing the Fourier basis with wavelets in an SNO-based framework, with the goal of reducing the prediction error of neural operators by improving the accuracy of function representation. However, the prediction error of neural operators is not determined solely by the representation error; it is also influenced by the generalization error of the neural network itself. Furthermore, if the DWT is adopted, the number of retained coefficients must be considered. This number affects the width of the neural network, which in turn influences its generalization error. By constructing a wavelet-based spectral neural operator and conducting both theoretical analysis and experiments, we can address these questions.

1.3. Research Questions

In this work, we propose operator learning using wavelet representations (OL-Wavelet) for approximating nonlinear operators between function spaces. The core idea is to represent input and output functions using their DWT coefficients, and to learn a mapping between these coefficients via a multilayer perceptron (MLP) [57].

More specifically, given a function $f(x)$, we compute its wavelet coefficients $\mathcal{W}(f)$ using a chosen wavelet basis. The neural network is then trained to approximate an operator $\mathcal{G}$ in the coefficient space:

$$\mathcal{W}(f) \mapsto \mathcal{W}(\mathcal{G}(f)).$$

This approach can be interpreted as a modification of the SNO framework, where the DFT is replaced by the DWT. The motivation for this replacement is that wavelet bases provide both spatial and frequency localization, which may offer advantages when dealing with functions exhibiting discontinuities or localized structures.

1.3.1. Main Research Question

The main research question of this work is:

> Can a wavelet-based spectral neural operator accurately approximate specific classes of operators, including 1D translation operators, operators mapping different source terms of 1D Poisson equations to the corresponding solutions, operators mapping different initial conditions of an 1D Diffusion euqation and an 1D Burgers' equation to their terminal solutions, and operators approximating functions with piecewise functions on dyadic partitions?

1.3.2. Sub-Questions

In the process of investigating the main research question, we examine the following subproblems.

1. **Wavelet vs Fourier Representation**

   Does using a wavelet basis improve the prediction accuracy of PDE solution operators compared to a Fourier basis, particularly in scenarios involving discontinuous or spatially localized source terms?

2. **Error Decomposition and Analysis**

   When evaluating the model on a test set, the total prediction error arises from two primary sources:

   - the approximation error introduced by truncating the wavelet representation of functions,
   - the prediction error of the neural network in mapping between coefficient spaces.

   How can the total error be rigorously decomposed into these two components? Furthermore, which component is dominant under different experimental settings?

3. **Optimal Representation Dimension**

   In practice, only a finite number of wavelet coefficients are retained to represent functions, in order to reduce computational cost via information compression. The number of retained coefficients determines:

   - the input and output dimensions of the neural network,
   - the width of the network.

   Given that the generalization error of the neural network depends on its width, what is the optimal number of wavelet coefficients to retain such that the overall prediction accuracy is maximized?

4. **Architecture Design: Joint vs Level-wise Learning**

   Wavelet coefficients are naturally organized into multiple levels, corresponding to different resolutions. Two possible architectures can be considered:

   - concatenating all coefficients into a single vector and learning the mapping using one global MLP,
   - assigning separate MLPs to coefficients at each level, thereby reducing the total number of connections.

   Is it feasible to learn the mapping between wavelet coefficients independently at each level? Which of these architectural choices leads to better prediction accuracy?

1.4. Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 2, we provide the foundational mathematical background related to the proposed model. The architecture of our model and the error analysis are presented in Chapter 3. In Chapter 4 the proposed model is applied to a few 1D operator learning problems. Chapter 5 discuss the experiment results further and Chapter 6 concludes the paper with a summary.

2

Fundamentals

This chapter introduces the mathematical foundations of the thesis, including function approximation, neural networks and mathematics of operator learning.

2.1. Function Approximation

In operator learning, transform-based function representations [78] provide an effective way to represent and approximate functions by projecting them onto a set of basis functions. Instead of working directly with function values in the original domain, these transforms map functions into coefficient spaces, where the underlying structure of the function can often be represented more compactly and analyzed more efficiently. Among the most widely used transform methods, Fourier transforms [10] and wavelet transforms [15] are particularly important due to their strong approximation properties and practical applicability. This section introduces these two transform methods and discusses some of their key properties.

2.1.1. Principles of Function Approximation

Function transforms, such as the Fourier transform and the wavelet transform, are mathematical techniques used to convert functions from one domain to another. They allow complex functions to be approximated by a series of basis functions, such as in spectral methods. This property makes function transforms invaluable in solving differential equations and analyzing data. Furthermore, studying function transforms is crucial for machine learning tasks, as they enable efficient representation and manipulation of functions in feature spaces.

A generalized integral transform [16] is a broad class of mathematical operations that map a function $f(x)$ from its original domain, such as the physical, spatial, or time domain, into a transformed representation space, such as a frequency domain, scale domain, or coefficient space, typically through an integral involving a kernel function. These transforms are widely used in various fields such as signal processing, differential equations, numerical methods, and physics. The general form of these transforms can be written as:

$$F(a) = \int_{\Omega} K(a, x) f(x)\, dx$$

where:

- $f(x)$ is the original function,
- $F(a)$ is the transformed function,
- $K(a, x)$ is the kernel function that governs how $f(x)$ is mapped to $F(a)$,

- Ω is the domain of integration, often spanning over an interval or the entire real line.

The kernel function $K(a, x)$ is crucial in determining the nature of the transformation. Depending on the specific choice of $K(a, x)$, the integral transform can take different forms, such as Fourier transforms, Laplace transforms [66], or wavelet transforms.

2.1.2. Fourier Transform

Among various transform-based function representations, the Fourier transform is one of the most fundamental and widely used tools in harmonic analysis and signal processing. The following discussion follows standard treatments in Fourier analysis [10]. The Fourier transform maps a function from its original domain into the frequency domain, where the function is represented by its frequency components. For periodic functions, this representation corresponds to a Fourier series, while for non-periodic functions it corresponds to a continuous superposition of sinusoidal basis functions. Such spectral representations are particularly useful in function approximation and operator learning.

The fundamental principle of the Fourier transform is based on Euler's formula,

$$e^{ix} = \cos x + i \sin x,$$

and for a continuous function $f(t)$, its Fourier transform is defined as

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t}\, dt,$$

where $F(\omega)$ denotes the frequency-domain representation of $f(t)$, and ω is the angular frequency. This transform can be interpreted as projecting the function onto complex exponential basis functions of different frequencies, thereby extracting its spectral coefficients.

Conversely, the inverse Fourier transform reconstructs the original function from its frequency-domain representation,

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t}\, d\omega.$$

This shows that the original function can be expressed as the superposition of all frequency components weighted by their corresponding Fourier coefficients.

For discrete signals or discretized functions, the discrete Fourier transform (DFT) is used, which is defined by

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i \frac{2\pi}{N} kn},$$

where X_k is the k-th frequency coefficient, x_n is the n-th sample of the original signal, and N is the total number of samples. Direct computation of the DFT requires $O(N^2)$ operations. In practice, the Fast Fourier Transform (FFT) algorithm [20] is commonly used to reduce the computational complexity to $O(N \log N)$, making Fourier-based methods computationally efficient for large-scale problems.

2.1.3. Wavelet Transform

The wavelet transform [15] represents a function in terms of wavelets, which are localized basis functions. Unlike the Fourier transform, which uses sine and cosine functions as basis functions, the wavelet transform employs basis functions that are localized in both time and frequency domains. This localization allows wavelets to capture both global trends and local singularities of a signal.

A wavelet basis function $\psi_{a,b}(t)$ is derived from a single prototype function, called the mother wavelet $\psi(t)$, through dilation and translation:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right),$$

where:

- $\psi(t)$ is the mother wavelet,
- $a \in \mathbb{R} \setminus \{0\}$ is the scale parameter, controlling the dilation or compression of the wavelet,
- $b \in \mathbb{R}$ is the translation parameter, controlling the temporal position of the wavelet.

The scale parameter a determines the frequency resolution: small values of $|a|$ correspond to compressed wavelets and capture high-frequency details, while large values of $|a|$ correspond to stretched wavelets and capture low-frequency components. The translation parameter b shifts the wavelet along the time axis to analyze local signal behavior at different positions.

There are several commonly used mother wavelets, as listed in Table 2.1. The choice of wavelet depends on the regularity, symmetry, and oscillatory behavior of the target function.

Wavelet Name	Mathematical Expression
Haar Wavelet [67]	$\psi(t) = \begin{cases} 1, & 0 \le t < \frac{1}{2} \\ -1, & \frac{1}{2} \le t < 1 \\ 0, & \text{otherwise} \end{cases}$
Mexican Hat Wavelet [64]	$\psi(t) = (1-t^2)e^{-t^2/2}$
Morlet Wavelet [14]	$\psi(t) = e^{i\omega_0 t}e^{-t^2/2}$
Daubechies Wavelet [74]	Defined recursively by $\phi(t) = \sum_k h_k \phi(2t-k)$

Table 2.1: Common Mother Wavelets and Their Expressions

The continuous wavelet transform (CWT) [1] of a function $f(t)$ is defined as

$$W(a,b) = \int_{-\infty}^{\infty} f(t)\psi_{a,b}(t)\,dt.$$

This transformation decomposes the function into different scales and positions, producing a time-frequency representation of the signal.

To perform a continuous wavelet transform on a function $f(t)$, the following steps are required:

1. Choose a mother wavelet $\psi(t)$ according to the properties of the signal and the analysis objective.
2. Select an appropriate range of scale parameters a and translation parameters b. In practice, the scale parameter is chosen over an interval $a \in [a_{\min}, a_{\max}]$, where small scales correspond to high-frequency components and large scales correspond to low-frequency structures. For each fixed scale a, the translation parameter b is varied continuously (or densely sampled) over the domain of the signal to detect localized features at different positions.
3. Construct the scaled and translated wavelet basis function $\psi_{a,b}(t)$ for each pair (a,b).
4. Compute the wavelet coefficient

$$W(a,b) = \int_{-\infty}^{\infty} f(t)\psi_{a,b}(t)\,dt.$$

5. Repeat the computation for all selected scales and translations to obtain the complete time-scale representation. The resulting coefficient map measures the similarity between the signal and the wavelet at each scale and position.

The inverse continuous wavelet transform (ICWT) [76] reconstructs the original function from its wavelet coefficients:

$$f(x) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{a^2} W(a,b) \psi \left(\frac{x-b}{a} \right) db\, da.$$

The normalization constant C_ψ is defined by the admissibility condition:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty.$$

The CWT theoretically allows all possible continuous values of the scale and translation parameters, providing highly redundant and fine-grained time-frequency analysis. However, this redundancy leads to high computational cost and storage requirements, often producing more coefficients than the original signal itself. Therefore, the CWT is primarily used for signal analysis rather than efficient compression.

In contrast, the discrete wavelet transform (DWT) [15] restricts the analysis to dyadic scales and discrete translations. This significantly reduces computational complexity while preserving the essential multiscale characteristics of the signal.

The DWT is founded on the framework of multiresolution analysis (MRA) [30], which represents a signal through a hierarchy of nested approximation spaces. Let $\{V_j\}_{j\in\mathbb{Z}}$ be a sequence of closed subspaces of $L^2(\mathbb{R})$ satisfying

$$\cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots,$$

with

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\}, \qquad \overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}).$$

Here, V_j denotes the approximation subspace at resolution 2^{-j}, containing the coarse representation of the signal at scale j. A fundamental property of MRA is that each space V_{j-1} can be decomposed into the direct sum of a coarser approximation space V_j and a detail space W_j:

$$V_{j-1} = V_j \oplus W_j.$$

The subspace V_j is spanned by translations and dilations of a scaling function $\phi(t)$, which captures the low-frequency (smooth) components of the signal, while the detail subspace W_j is spanned by a wavelet function $\psi(t)$, which captures the high-frequency (detail) components. By recursively applying this decomposition, a multiscale representation is obtained:

$$V_0 = V_J \oplus W_J \oplus W_{J-1} \oplus \cdots \oplus W_1.$$

In practical discrete-time implementations, this decomposition is realized by filters. Given an input discrete signal $x[n]$, the initial approximation coefficients are defined as

$$cA_0[n] = x[n].$$

At each decomposition level j, the signal is processed by two analysis filters:

- a low-pass analysis filter $h[n]$, associated with the scaling function $\phi(t)$, which preserves the low-frequency components and produces the approximation coefficients;
- a high-pass analysis filter $g[n]$, associated with the wavelet function $\psi(t)$, which extracts the high-frequency components and produces the detail coefficients.

The approximation coefficients $cA_j[k]$ and detail coefficients $cD_j[k]$ at level j are computed as

$$cA_j[k] = \sum_n h[n-2k]cA_{j-1}[n],$$

$$cD_j[k] = \sum_n g[n-2k]cA_{j-1}[n],$$

where k denotes the downsampled discrete-time index. The factor $2k$ reflects the downsampling operation by a factor of two after filtering. The approximation coefficients cA_j are then recursively used as the input for the next decomposition level, producing a hierarchical multiscale representation of the signal.

The inverse discrete wavelet transform (IDWT) [15] reconstructs the original signal from the approximation and detail coefficients using synthesis filters. Specifically, the approximation coefficients $cA_j[k]$ and detail coefficients $cD_j[k]$ are first upsampled by inserting zeros between adjacent samples, and then filtered by the synthesis low-pass filter $\tilde{h}[n]$ and synthesis high-pass filter $\tilde{g}[n]$, respectively. The reconstruction at level $j-1$ is given by

$$cA_{j-1}[n] = \sum_k \tilde{h}[n-2k]cA_j[k] + \sum_k \tilde{g}[n-2k]cD_j[k].$$

By recursively applying this reconstruction from level J back to level 0, the original signal $x[n]$ can be exactly recovered, provided that the analysis and synthesis filters satisfy the perfect reconstruction condition.

2.1.4. Gibbs Phenomenon for 1D Functions

The Gibbs phenomenon [26] is an effect that cannot be ignored when using Fourier series to approximate discontinuous functions.

The Gibbs phenomenon occurs when a Fourier series is used to approximate a function that has a discontinuity. Near the point of discontinuity, the partial sum of the Fourier series produces oscillations and an overshoot above and below the true function value. Even as more Fourier terms are added, these oscillations become narrower but do not disappear completely, and the overshoot approaches a fixed proportion of the jump.

This phenomenon affects function approximation because it introduces persistent local errors near discontinuities. As a result, Fourier-based approximations may struggle to accurately represent functions with sharp jumps or localized changes, which can reduce the overall approximation quality in those regions.

In the Fourier series approximation of a function with a jump discontinuity, the Gibbs phenomenon produces an overshoot near the discontinuity. As the number of Fourier terms increases, the oscillations become narrower, but the maximum overshoot does not vanish. Instead, it approaches a fixed value proportional to the size of the jump.

2.1.5. Wavelet Approximation and Gibbs Phenomenon

Wavelet-based approximations generally do not exhibit the typical Gibbs phenomenon observed in Fourier series. Due to the localization property of wavelet basis functions, approximation errors near discontinuities tend to remain spatially localized, rather than producing global oscillations around the discontinuity. As a result, the overshoot behavior characteristic of Fourier approximations is usually significantly suppressed in wavelet expansions and may only appear under specific conditions depending on the choice of wavelet basis [33].

In particular, existing literature shows that Haar wavelets do not exhibit the Gibbs phenomenon when approximating discontinuous functions [58].

2.2. Neural Network

In neural operators, neural networks are used as learnable parameterizations for approximating mappings between function spaces. They provide strong nonlinear approximation capability, but their performance depends on optimization, data quality, and generalization ability.

2.2.1. Mathematics of Neural Networks

Mathematically, a neural network is a parameterized function

$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m,$$

where n is the input dimension, m is the output dimension, and θ denotes all trainable parameters (weights and biases). The goal is to approximate an unknown mapping between input $x \in \mathbb{R}^n$ and output $y \in \mathbb{R}^m$.

Given a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N,$$

where N is the number of samples, training is formulated as minimizing a loss function

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i),$$

where $\ell(\cdot, \cdot)$ measures the difference between prediction and target.

From a statistical viewpoint, this optimization is often interpreted as maximum likelihood estimation (MLE) [54]. If the loss function is the negative log-likelihood of a probabilistic model $p_\theta(y|x)$, minimizing $L(\theta)$ is equivalent to maximizing the probability of the observed data.

Neural networks are also universal function approximators: under mild assumptions, they can approximate arbitrary continuous functions. Thus, neural networks combine statistical inference and nonlinear function approximation into a unified framework.

2.2.2. Typical Structures of Neural Networks

The simplest and most common neural network architecture is the multilayer perceptron (MLP), also called a fully connected neural network.

An L-layer neural network is defined recursively by

$$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right), \quad l = 1, \ldots, L,$$

where:

- $h^{(0)} = x \in \mathbb{R}^n$ is the input vector,
- $h^{(l)}$ is the hidden representation at layer l,
- $W^{(l)}$ is the weight matrix of layer l,
- $b^{(l)}$ is the bias vector of layer l,
- $\sigma(\cdot)$ is an activation function such as ReLU.

The final output is

$$f_\theta(x) = h^{(L)}.$$

Each layer performs an affine transformation followed by a nonlinear activation. The composition of multiple layers enables the network to learn hierarchical nonlinear features.

The trainable parameter set is

$$\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^L.$$

The depth L determines how many transformations are applied, while the width (the dimension of $h^{(l)}$) controls the representation capacity of each layer.

MLPs are universal function approximators and form the basic building block of many neural operator architectures. Although simple, they provide the fundamental mechanism for learning nonlinear mappings from data.

2.2.3. Optimization Algorithms

Training a neural network means solving the optimization problem

$$\min_{\theta} L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i),$$

where $L(\theta)$ is the empirical loss over the dataset.

The most basic optimization method is gradient descent:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t),$$

where θ_t is the parameter at iteration t, $\nabla_\theta L(\theta_t)$ is the gradient of the loss, and $\eta > 0$ is the learning rate.

Computing the exact gradient over all N samples is expensive for large datasets. Therefore, stochastic gradient descent (SGD) [4] uses a small mini-batch of data to estimate the gradient.

Several improved variants have been developed, including AdaGrad [17], RMSProp [63], and Adam [34]. These methods adapt the step size using historical gradient information, often improving convergence speed and stability.

Optimization algorithms are the computational foundation of neural network training and directly affect efficiency and final performance.

2.2.4. Dataset

A dataset for supervised learning is written as

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N},$$

where x_i is the input and y_i is the target output.

In neural operators, the dataset has a different structure because the input and output are functions rather than finite-dimensional vectors. The target mapping is an operator

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$

where $\mathcal{X}$ and $\mathcal{Y}$ are function spaces.

Each sample is a pair of functions

$$(u_i, v_i), \quad v_i = \mathcal{G}(u_i).$$

In practice, functions are discretized on a domain $\Omega \subset \mathbb{R}^d$:

$$u \approx (u(x_1), u(x_2), \ldots, u(x_M)),$$

where $\{x_j\}_{j=1}^{M}$ are discretization points and M is the number of grid points.

Thus, the practical dataset becomes

$$\mathcal{D} = \{(u_i(x_j), v_i(x_j))\}_{i=1,j=1}^{N,M}.$$

Such datasets are often generated by numerical solvers. For example, for a parametric partial differential equation

$$\mathcal{L}_a u = f,$$

where a is a parameter field and f is a forcing term, solving the equation for different a produces training pairs (a, u).

The dataset structure is essential because it determines the operator that the neural network learns.

2.3. Mathematics of Operator Learning

Operator learning aims to approximate mappings between functions rather than mappings between finite-dimensional vectors [9]. Mathematically, let $\mathcal{U}$ and $\mathcal{V}$ be two Banach spaces [37][9], where a Banach space is a complete normed vector space. Here, $\mathcal{U}$ denotes the input function space and $\mathcal{V}$ denotes the output function space. Both spaces are defined on a domain $\Omega \subset \mathbb{R}^d$, where Ω is a d-dimensional Euclidean domain and d is the spatial dimension.

The goal is to learn an operator

$$\mathcal{A} : \mathcal{U} \to \mathcal{V},$$

where $\mathcal{A}$ maps an input function $u \in \mathcal{U}$ to an output function $v \in \mathcal{V}$. That is,

$$\mathcal{A}(u) = v.$$

Given a dataset of paired functions

$$\{(u_i, v_i)\}_{i=1}^N,$$

where N is the number of training samples, the objective is to construct an approximation $\hat{\mathcal{A}}$ of $\mathcal{A}$.

In practice, $\hat{\mathcal{A}}$ is parameterized by a neural network with trainable parameters $\theta \in \mathbb{R}^P$, where P is the total number of parameters. Training is formulated as an optimization problem:

$$\min_{\theta \in \mathbb{R}^P} \sum_{i=1}^{N} \mathcal{L}(\hat{\mathcal{A}}(u_i; \theta), v_i),$$

where $\mathcal{L}$ is a loss function measuring the difference between the predicted output $\hat{\mathcal{A}}(u_i; \theta)$ and the target output v_i.

A common neural operator layer is defined through an integral transformation:

$$u_{l+1}(x) = \sigma\left(\int_\Omega K^{(l)}(x, y) u_l(y)\, dy + b_l(x)\right).$$

Here, $u_l(y)$ denotes the input feature function at layer l evaluated at point $y \in \Omega$, and $u_{l+1}(x)$ denotes the output feature function at layer $l+1$ evaluated at point $x \in \Omega$. The function $K^{(l)}(x, y)$ is a learnable kernel that defines the interaction between positions x and y, $b_l(x)$ is a bias function, and $\sigma(\cdot)$ is a nonlinear activation function.

This kernel-based structure directly mimics the action of continuous operators. Different neural operator architectures mainly differ in how the kernel $K^{(l)}(x, y)$ is parameterized.

DeepONet [48] represents an operator using a finite-dimensional basis decomposition:

$$G(u)(x) = \sum_{k=1}^{p} b_k(u) t_k(x).$$

Here, G denotes the learned operator, u is the input function, and $x \in \Omega$ is the spatial coordinate. The coefficient $b_k(u)$ is generated by the branch network and depends on the input function u,

while $t_k(x)$ is generated by the trunk network and depends on the spatial location x. The integer p denotes the number of basis functions.

Fourier Neural Operator (FNO) [43] parameterizes the operator in Fourier space:

$$u_{l+1}(x) = \sigma\left(\mathcal{F}^{-1}\big(R_l(k)\mathcal{F}(u_l)(k)\big) + W_l u_l(x) + b_l(x)\right).$$

Here, $\mathcal{F}$ denotes the Fourier transform and $\mathcal{F}^{-1}$ its inverse. The function $u_l(x)$ is the input feature at layer l, and $u_{l+1}(x)$ is the output feature at layer $l+1$. The variable k denotes the frequency index. The term $R_l(k)$ is a learnable linear transformation in frequency space, W_l is a local linear transformation in physical space, and $b_l(x)$ is a bias function.

The Deep Green Network (DGN) [25] uses a Green's function representation:

$$(Gu)(x) = \int_\Omega G_\theta(x,y)u(y)\,dy.$$

Here, G denotes the operator, $u(y)$ is the input function evaluated at $y \in \Omega$, and $G_\theta(x,y)$ is a neural network approximation of the Green's function parameterized by θ.

The Graph Neural Operator (GNO) [42] defines the operator on graph-structured data:

$$u_{l+1}(x) = \sigma\left(\sum_{y\in\mathcal{N}(x)} K_\theta(x,y)u_l(y)\right).$$

Here, x denotes a graph node, $\mathcal{N}(x)$ denotes the neighborhood of node x, and $K_\theta(x,y)$ is a learnable kernel between connected nodes x and y.

The Wavelet Neural Operator (WNO) [73] applies operator learning in wavelet space:

$$u_{l+1}(x) = \sigma\left(\mathcal{W}^{-1}(R_l\mathcal{W}(u_l)) + b_l(x)\right).$$

Here, $\mathcal{W}$ denotes the wavelet transform and $\mathcal{W}^{-1}$ denotes the inverse wavelet transform. The matrix R_l is a learnable transformation in wavelet space.

PCA-net [8][39] first projects functions into a finite-dimensional basis using principal component analysis (PCA), and then learns a mapping between coefficient vectors using a fully connected neural network.

Spectral Neural Operator (SNO) [23] directly learns mappings between spectral coefficients:

$$f_{\text{in}}(x) = \sum_{i=1}^{M} g_i(x)d_i, \qquad f_{\text{out}}(x) = \sum_{i=1}^{M} g_i(x)b_i.$$

Here, $f_{\text{in}}(x)$ is the input function and $f_{\text{out}}(x)$ is the output function. The functions $g_i(x)$ are basis functions, d_i are the input spectral coefficients, and b_i are the output spectral coefficients. A neural network learns the mapping from the input coefficients $\{d_i\}_{i=1}^M$ to the output coefficients $\{b_i\}_{i=1}^M$, where M is the number of spectral modes.

In summary, operator learning extends neural networks from vector-to-vector mappings to function-to-function mappings. Its mathematical foundation lies in Banach spaces, integral operators, and data-driven approximation of nonlinear mappings between function spaces.

3

Proposed Method

This chapter provides a detailed description of the proposed model and analyzes the possible sources of errors that occur during prediction tasks. Error analysis in return helps to determine the optimal size of neural networks.

3.1. Design Rationale

Prior works PCA-net [8] and SNO [23] use only fully connected neural networks. Fully connected neural networks have sufficient capacity to approximate linear or nonlinear mappings between vectors, especially in 1D cases. Therefore, our model is designed and implemented based on fully connected neural networks instead of CNN [41] and U-net [60], to maintain consistency with prior works.

In our work and experiments, DWT and DFT are used as feature engineering representations for functions. Therefore, we have a stable way to encode and decode functions, without learning parameters as FNO [43] and WNO [73]. Due to computational constraints, these function approximation algorithms apply truncation to the number of modes used in the function representation. Therefore, in this neural network design we can expect a trade-off between computational cost and function representation error. Higher input dimensionality of neural networks with more number of modes generally leads to more accurate function representations, but it also increases the computational cost of neural network training and inference. On the other hand, under fixed computational resources high dimensionality may make neural networks more difficult to train and increase the prediction error of the neural network. As a result of the combined effects of these two types of errors, the overall network error may change non-monotonically with respect to the input dimension. Therefore, determining the optimal input dimensionality and network size to minimize the total error is an important task.

3.2. Overall Architecure

Based on the design ideas from the previous subsection, an operator learning model using wavelet representations (OL-Wavelet) is proposed. This modeling approach first transforms both the input function $f(x)$ and the output function $g(x)$ into the wavelet domain via multilevel discrete wavelet decomposition. The neural network is trained to learn the mapping between the wavelet coefficients of f and those of g. This framework decouples the learning process from the physical space and instead operates in the multiscale wavelet representation of the functions.

The diagram in Fig. 3.1 illustrates the sequence of mappings in the proposed model. Suppose there exists an operator $\mathcal{A}$ such that $\mathcal{A}f = g$, with $f \in X$ and $g \in Y$, where X and Y are Banach spaces. The operator H maps a function f to its truncated wavelet coefficient vector c, while H^{-1}

reconstructs a function from a given coefficient vector d. The neural network $\mathcal{N}$ is trained to learn a mapping from c to d.

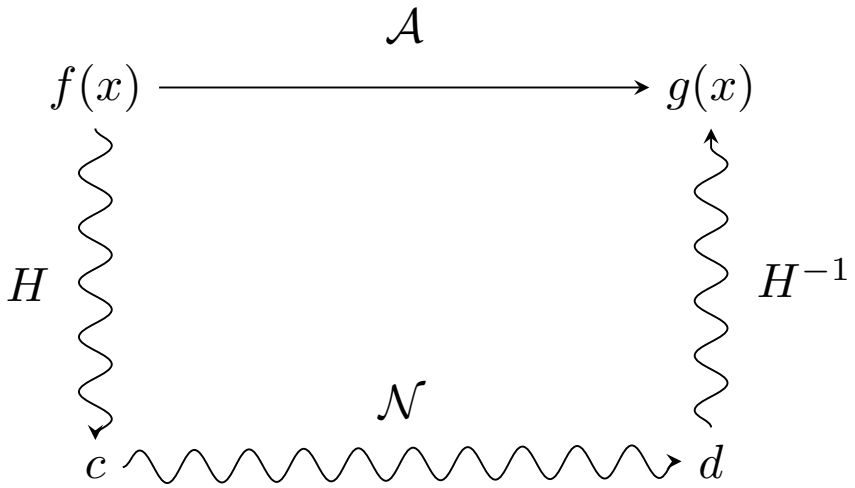


Figure 3.1: Overall Architecture of the Wavelet Spectral Neural Operator

It is important to emphasize that the diagram is not strictly commutative. The top horizontal arrow represents the exact operator $\mathcal{A}$, whereas the wavy arrows indicate approximate mappings. In particular, the composition

$$H^{-1} \circ \mathcal{N} \circ H$$

only approximates $\mathcal{A}$, rather than reproducing it exactly. Therefore, the diagram should be interpreted as approximately commutative, in the sense that

$$H^{-1} \circ \mathcal{N} \circ H \approx \mathcal{A}.$$

It is worth noting that the coefficients produced by the discrete wavelet transform (DWT) are not naturally organized as a single one-dimensional vector, but rather as a hierarchical, tree-structured collection of subband coefficients. Consider a discrete-time signal, which can be viewed as samples of a one-dimensional function. The DWT decomposes this signal in a multilevel manner, as illustrated in Fig. 3.2. At each level of decomposition, the signal is passed through two complementary filters: a low-pass (LP) filter and a high-pass (HP) filter. The LP branch captures the low-frequency (coarse or approximation) content of the signal, producing the approximation coefficients cA_j, while the HP branch captures the high-frequency (detail) content, producing the detail coefficients cD_j. After filtering, a downsampling operation by a factor of two is typically applied, reducing the resolution at each level.

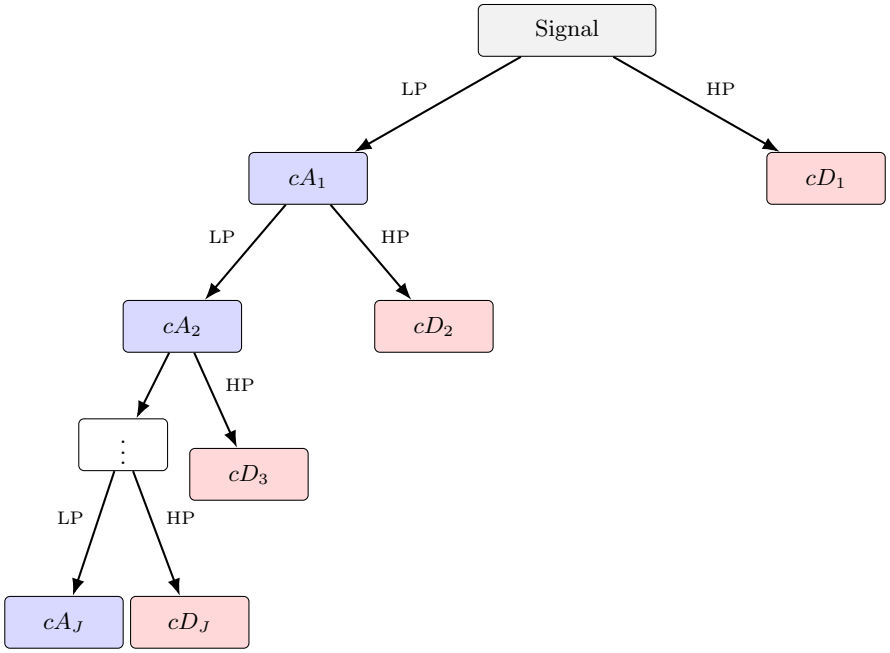


Figure 3.2: 1D Discrete Wavelet Transform Tree

This process is applied recursively only to the approximation coefficients. Starting from the original signal, the first level decomposition yields cA_1 and cD_1. The approximation part cA_1 is then further decomposed into cA_2 and cD_2, and so on. After J levels, the full set of DWT coefficients can be expressed as

$$cA_J, cD_J, \cdots, cD_2, cD_1.$$

Here, cA_J and cD_J correspond to the coarsest scale (lowest frequency resolution), whereas the coefficients cD_1 capture the finest-scale (highest frequency) details. In other words, as the level index decreases from J to 1, the corresponding detail coefficients represent progressively finer structures of the original signal.

Although these coefficients are naturally organized across multiple scales, it is often convenient in practical applications to convert them into a single representation vector. This can be achieved by concatenating the coefficient arrays in a prescribed order, for example,

$$c = [cA_J, cD_J, \cdots, cD_2, cD_1]\,.$$

Depending on the application, one may either include all coefficients or truncate the representation by retaining only a subset such as

$$c = [cA_J, cD_J, \cdots, cD_L]\,, \quad L \leq J.$$

Such a truncated representation preserves the coarse-scale structure while discarding finer details, which is useful for compression, denoising, or feature extraction. Importantly, the original signal can be reconstructed either exactly (if all coefficients are retained) or approximately (if only a subset is used) through the inverse DWT, highlighting that the vector c serves as a compact yet informative representation of the underlying function signal.

The overall architecture in the figure 3.1 also applies to SNO if we replace the DWT with DFT. Therefore, OL-Wavelet and SNO are highly similar and can be compared under equally truncated sequences. Based on the structure of DWT, the number of modes selected is generally 4, 8, 16, and so on, in powers of 2.

3.3. Model Components

As illustrated in Fig. 3.3, we consider a forward mapping defined on discrete signals via a wavelet-based representation.

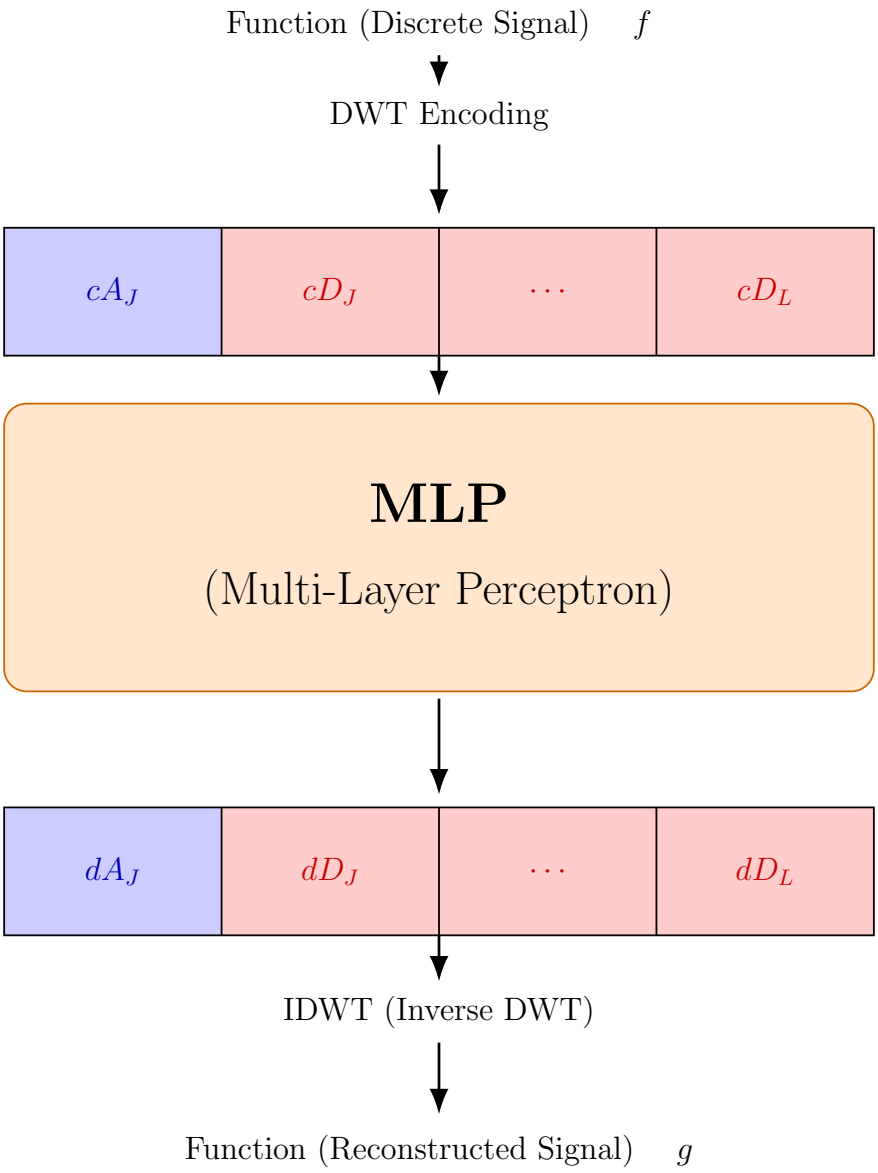


Figure 3.3: OL-Wavelet Schematic

Let $f \in \mathbb{R}^M$ denote the input discrete signal. We first apply a full discrete wavelet transform (DWT), using a fixed wavelet basis (e.g., Haar) and a maximal decomposition level $J = \log_2 M$. This yields a collection of wavelet coefficients organized as

$$\mathcal{W}(f) = \big[cA_J,\ cD_J,\ cD_{J-1},\ \ldots,\ cD_L\big],$$

where cA_J denotes the coarse-scale approximation coefficients and cD_ℓ denotes the detail coefficients at level ℓ. For notational simplicity, we view this collection as a single vector

$$\mathbf{c} \in \mathbb{R}^K,$$

obtained by concatenating the coefficients in a fixed coarse-to-fine ordering, as depicted in Fig. 3.3.

Next, the coefficient vector $\mathbf{c}$ is passed through a multi-layer perceptron (MLP), which defines a nonlinear mapping

$$\Phi_\theta : \mathbb{R}^K \to \mathbb{R}^K,$$

parameterized by θ. The network produces an output vector

$$\mathbf{d} = \Phi_\theta(\mathbf{c}) \in \mathbb{R}^K,$$

which is interpreted as the predicted wavelet coefficients

$$\mathbf{d} = \big[dA_J,\ dD_J,\ dD_{J-1},\ \ldots,\ dD_L\big].$$

Finally, the predicted coefficients are mapped back to the signal domain via the inverse discrete wavelet transform (IDWT),

$$\tilde{g} = \mathcal{W}^{-1}(\mathbf{d}) \in \mathbb{R}^M,$$

yielding the reconstructed output signal $\tilde{g}$, as shown at the bottom of Fig. 3.3.

In summary, the overall forward operator can be written compactly as

$$\tilde{g} = \mathcal{W}^{-1} \circ \Phi_\theta \circ \mathcal{W}(f),$$

which corresponds to learning a nonlinear transformation in the wavelet coefficient domain.

We consider two neural network architectures for learning mappings in the wavelet coefficient domain, which differ in how they exploit the multilevel structure of the discrete wavelet transform.

In the flat variant, all wavelet coefficients across different scales are concatenated into a single vector

$$\mathbf{c} = \big[cA_J,\ cD_J,\ cD_{J-1},\ \ldots,\ cD_L\big] \in \mathbb{R}^K.$$

A fully connected neural network (MLP) is then used to learn a global mapping

$$\Phi_\theta : \mathbb{R}^K \to \mathbb{R}^K,$$

which directly maps the input coefficients to the output coefficients.

To train the network, we define a loss function based on the mean squared error between the predicted output coefficients and the target output coefficients:

$$\mathcal{L}(\theta) = \frac{1}{K} \sum_{k=1}^{K} \left(\Phi_\theta(c^{\text{in}})_k - c_k^{\text{out}}\right)^2.$$

Here, $c^{\text{in}} \in \mathbb{R}^K$ denotes the input coefficient vector, $c^{\text{out}} \in \mathbb{R}^K$ denotes the target output coefficient vector, and $\Phi_\theta(c^{\text{in}})_k$ denotes the k-th predicted coefficient produced by the neural network. Minimizing this loss enables the network to learn the mapping between input and output coefficient spaces.

This formulation is conceptually simple and aligns naturally with standard neural network architectures. In particular, all coefficients are treated uniformly, and the fully connected structure allows unrestricted interactions across different scales. As a result, the model is capable of capturing cross-scale dependencies and complex global relationships in the coefficient space.

However, this flexibility comes at the cost of a large number of parameters and dense connectivity. Every coefficient can interact with every other coefficient, regardless of their scale, which may lead to increased computational complexity and reduced interpretability.

In contrast, the level-wise variant preserves the multilevel structure of the wavelet decomposition. The coefficient vector is partitioned into blocks corresponding to different scales,

$$\mathbf{c} = \left[\mathbf{c}^{(J)},\ \mathbf{c}^{(J-1)},\ \ldots,\ \mathbf{c}^{(L)}\right],$$

where each $\mathbf{c}^{(\ell)}$ represents the coefficients at level ℓ.

For each level ℓ, a separate neural network is introduced to learn a mapping

$$\Phi_\theta^{(\ell)} : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_\ell},$$

and the overall mapping is defined by applying these networks independently across levels. This design explicitly enforces a block-diagonal structure in the coefficient mapping.

Such a construction significantly reduces the number of parameters and neural connections, since interactions are restricted within each level. It also aligns with the multiscale nature of wavelet representations, allowing the model to capture scale-specific behaviors in a structured and interpretable manner.

On the other hand, the independence across levels implies that cross-scale interactions cannot be learned directly. Each level is processed in isolation, which may limit the expressive power of the model when the target mapping involves strong dependencies between different scales. Consequently, this architectural constraint can lead to a potential loss in approximation accuracy compared to the fully connected flat variant.

3.4. Error Analysis

The error in the model during the prediction phase comes from multiple sources. These errors are essentially determined by the network width. Conversely, error analysis can help us find the optimal network width. Within the proposed framework, we are thus able to partially analyze the factors that affect the predictive accuracy of neural networks. The errors mainly come from the reconstruction error of the function and the error of the neural network itself. Although these two types of errors have different sources, they are not necessarily independent of each other. We can derive both an error expression.

Define the notions:

- f: An input function.
- g: An output function.
- H: DWT with truncation of K modes.
- H^{-1}: IDWT.
- N: a neural network.
- $c^* = H(f) \in \mathbb{C}^M$: full spectral coefficients of $f(x)$, with M modes.
- $c = c^* + \delta_c$: actual coefficients with representation error δ_c, so $\|\delta_c\| \leq \varepsilon_H$. If $K = M$, $c = c^*$.
- $d^* = H(g)$: full spectral coefficients of the target function $g(x)$, with M modes.
- $d = \mathcal{N}(c)$: neural network output based on perturbed input c, containing also generalization error of neural networks.
- $\bar{d}$ is the ground-truth mapping target of c, truncated by K modes. Due to the generalization error of neural networks, d is often not equal to $\bar{d}$.
- $\tilde{g}(x) = H^{-1}(d)$: reconstructed function from network output d.

For one sample of mapping, the total error can be decomposed by splitting terms

$$\begin{aligned}\|\tilde{g}(x) - g(x)\|_2 &= \|H^{-1}(d) - H^{-1}(d^*)\|_2 \\ &= \|H^{-1}(d) - H^{-1}(\bar{d}) + H^{-1}(\bar{d}) - H^{-1}(d^*)\|_2\end{aligned}$$

We define that

$$e_{\text{coef}} := \tilde{g} - H^{-1}(\bar{d}), \qquad e_{\text{rec}} := H^{-1}(\bar{d}) - g.$$

Then we have the exact identity

$$\hat{g} - g = e_{\text{coef}} + e_{\text{rec}}.$$

Taking squared norms, the overall error under K modes is decomposed as:

$$\|\tilde{g} - g\|_2^2 = \|e_{\text{coef}}\|_2^2 + \|e_{\text{rec}}\|_2^2 + 2\langle e_{\text{coef}}, e_{\text{rec}}\rangle. \tag{3.1}$$

We see in Equation 3.1 an cross term $\langle e_{\text{coef}}, e_{\text{rec}}\rangle$ which can be positive, negative or zero.

For both DFT and DWT, let $T \in \mathbb{K}^{M\times M}$ be the analysis transform matrix ($T = F$ for DFT, $T = W$ for DWT), where $T^{-1} = T^*$ since T is unitary/orthogonal. Let $\Pi_K \in \mathbb{K}^{M\times M}$ be the diagonal projector that keeps the prescribed K coefficients and sets the other $M - K$ coefficients to zero. Then the truncated reconstruction operator is explicitly defined by

$$P_K := T^{-1}\Pi_K T.$$

Hence P_K is a rank-K orthogonal projector onto the retained coefficient subspace. It projects vector into a K-rank space S_K.

$$e_{\text{coef}} = \tilde{g} - \bar{g}$$

By the neural network, we have

$$\tilde{g},\ \bar{g} \in \text{Ran}(P_K).$$

Equivalently, this can be written as

$$P_K\tilde{g} = \tilde{g}, \qquad P_K\bar{g} = \bar{g}.$$

Resulting in

$$P_K e_{\text{coef}} = e_{\text{coef}}$$

Therefore,

$$\begin{aligned}\text{cross} &= 2\langle e_{\text{coef}}, e_{\text{rec}}\rangle \\ &= 2\langle e_{\text{coef}}, (P_K - I)g\rangle \\ &= 2\langle P_K e_{\text{coef}}, (P_K - I)g\rangle \\ &= 2\langle e_{\text{coef}}, P_K(P_K - I)g\rangle \\ &= 2\langle e_{\text{coef}}, (P_K^2 - P_K)g\rangle \\ &= 0.\end{aligned}$$

The equation indicates the orthogonality of e_{coef} and e_{rec}. Therefore, Equation 3.1 is changed to Equation 3.2 which represents the error decomposition of our model.

$$\|\tilde{g} - g\|_2^2 = \|e_{\text{coef}}\|_2^2 + \|e_{\text{rec}}\|_2^2 \tag{3.2}$$

Let

$$S = \{(f_i, g_i)\}_{i=1}^N$$

be a dataset of function pairs, where each function is discretized on a fixed physical grid of size M. For a given truncation dimension K, denote by $\tilde{g}_i^{(K)}$ the reconstructed output produced by the neural operator using K spectral coefficients.

We define the mean squared error as

$$E(S, K) := \frac{1}{|S|} \sum_{(f,g)\in S} \left\|\tilde{g}^{(K)} - g\right\|_2^2.$$

Given a set of truncation dimensions

$$\mathcal{K} \subset \{1, 2, \ldots, M\},$$

we define the optimal truncation dimension as the solution of the empirical risk minimization problem

$$K^* = \arg\min_{K\in\mathcal{K}} E(S, K). \tag{3.3}$$

By using Equation 3.3 we can find the optimal representation dimension of functions.

4

Numerical Experiments

This chapter presents numerical experiments using several operators that map one-dimensional functions to one-dimensional functions. The experimental results in this chapter are mainly presented in the form of line plots, together with some preliminary analysis. More in-depth discussion is provided in Chapter 5.

4.1. Overall Settings

Before presenting the specific numerical experiments, we first introduce the common pipeline and experimental settings shared across all experiments, as well as the selected 1D operators used for training and evaluating our model.

4.1.1. Pipeline

We consider a general operator learning problem of the form

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$

where $\mathcal{X}$ and $\mathcal{Y}$ are function spaces, and the goal is to learn the mapping from an input function $f \in \mathcal{X}$ to its image $\mathcal{G}(f) \in \mathcal{Y}$ from finitely many sample pairs. Rather than restricting attention to a specific operator, we adopt a general experimental framework that can be applied to a broad class of linear or nonlinear operators.

First, we specify a parametric family of input functions $\{f_\theta\}_{\theta \in \Theta}$, where the parameter θ controls the variability of the dataset. For each sampled parameter θ, we construct an input-output pair

$$(f_\theta,\, g_\theta), \qquad g_\theta = \mathcal{G}(f_\theta).$$

These pairs form the supervised dataset used for operator learning. The training set and test set are sampled independently from prescribed parameter ranges, and duplicate samples are excluded so that the evaluation is carried out on previously unseen functions.

Since learning algorithms operate on finite-dimensional vectors, all functions are discretized on a fixed uniform grid of size M over the spatial domain. Thus each function f_θ and g_θ is represented by a vector in $\mathbb{R}^M$. This discretization step converts the operator learning problem into the task of learning a map between finite-dimensional representations of input and output functions.

To study how the choice of representation affects learnability, we further transform the discretized functions into compressed coefficient representations. In particular, we consider both Fourier-based and wavelet-based representations, and retain only the first K effective coefficients, where K is varied across a prescribed range. This yields a family of reduced models indexed by the

resolution parameter K. The purpose of this step is to compare how different bases encode the information relevant to the target operator.

Given the reduced representations of the input and output functions, we train neural networks to approximate the induced finite-dimensional mapping between coefficients. Different architectures may be used depending on the representation; for example, one may use a single multilayer perceptron on a flattened coefficient vector, or separate subnetworks acting on different wavelet levels. In our experiments, we refer to these two architectures as DWT-Flat and DWT-LevelWise, respectively. Overall, three models—DWT-Flat, DWT-LevelWise, and DFT (i.e., the SNO model based on Fourier representations)—are compared in our experiments. In all cases, the networks are trained in a supervised manner by minimizing the mean squared error between predicted and target coefficients on the training set.

After training, the predicted output coefficients are mapped back to the physical domain to reconstruct an approximation $\widetilde{g}_\theta$ of the true output $g_\theta = \mathcal{G}(f_\theta)$. The reconstruction is then evaluated on the test set using function-level error metrics with the L^2 error. In addition, when appropriate, the total error may be decomposed into a coefficient learning error and a representation truncation error, in order to distinguish approximation errors caused by the reduced basis from errors caused by the learned model itself.

Finally, to examine the effect of representation size, the entire procedure is repeated for multiple values of K. This produces error curves as functions of K, allowing us to compare different representations and architectures in terms of both approximation quality and learning efficiency. In this way, the experimental pipeline provides a unified framework for evaluating operator learning methods for arbitrary target operators.

4.1.2. Selection of Operators

We selected the learned 1D operators according to the continuity of the functions, including several combinations such as from continuous functions to discontinuous functions, from discontinuous functions to continuous functions, and so on. Therefore, these cases cover different kinds of continuity for operators. The operators to be learned are listed in Table 4.1.

Operator Name	Continuity of Mapped Functions
Translation Operator	Discontinuous function to discontinuous function
Translation Operator with Height Variation	Discontinuous function to discontinuous function
Poisson Equation Solver Operator	Discontinuous function to continuous function
Diffusion Equation Solver Operator	Continuous function to continuous function
Burgers Equation Solver Operator	Continuous function to continuous function
Piecewise Constant Approximation	Continuous function to continuous function

Table 4.1: Operators and the continuity of mapped functions

The six operators considered in this study are deliberately chosen to span a diverse set of continuity mappings and structural properties, in order to systematically evaluate the behavior of operator learning models under different functional regimes.

The translation operator serves as a fundamental baseline. Both the input and output functions are discontinuous (e.g., square waves), making this case particularly relevant for studying representations of non-smooth signals. Such functions are, in principle, well-suited for approximation by Haar wavelets due to their localized and piecewise-constant nature. However, empirical results later demonstrate that, even in this setting, the DFT can still achieve superior approximation performance compared to the DWT, highlighting nontrivial differences between global and local basis representations.

For the translation operator with height variation, this operator is constructed following the observation that the standard translation operator is not well approximated by DWT. By introducing height

variations aligned with dyadic partitions, the resulting functions exhibit structures that are more compatible with wavelet bases. In particular, the approximation error of DWT for such functions can be made arbitrarily small, approaching zero. This operator therefore provides a favorable scenario for wavelet-based methods and serves as a contrast to the first case.

For the Poisson equation solver operator, it corresponds to a one-dimensional boundary value problem, where the input is the source term (right-hand side of the PDE) and the output is the global solution. Notably, discontinuous inputs are mapped to continuous outputs due to the smoothing property of elliptic operators. This example is representative of a broad class of PDE-based operators and is essential for evaluating how well models capture nonlocal smoothing effects.

For the diffusion equation solver operator, it represents a one-dimensional initial value problem with inherent smoothing dynamics. Both the initial condition and solution at later times are continuous. This operator provides a canonical example of continuous-to-continuous mappings with dissipative behavior, allowing us to assess how models handle gradual regularization over time.

For the Burgers' equation solver operator, it is another one-dimensional initial value problem but exhibits nonlinear dynamics. Although the initial condition is smooth, the solution may develop steep gradients or even shock-like structures over time. This operator is therefore particularly challenging, as it tests the ability of models to capture the transition from smooth to nearly discontinuous behavior within a continuous function space.

For the piecewise constant approximation operator,it is a constructed operator designed to favor wavelet-based representations. The input function is sampled over dyadic partitions, and each interval is mapped to its maximum value, producing a piecewise constant function that acts as an upper envelope. Such outputs are highly compatible with Haar wavelets, and we expect the DWT to achieve superior approximation performance in this setting. Consequently, this operator is intended to highlight potential advantages of OL-Wavelet methods in operator learning tasks.

Overall, the selected operators cover a wide range of scenarios, including discontinuous-to-discontinuous, discontinuous-to-continuous, and continuous-to-continuous mappings, as well as both linear and nonlinear dynamics. This diversity ensures a comprehensive evaluation of different approximation mechanisms (e.g., DFT vs. DWT) within the operator learning framework.

4.1.3. Hyperparameters Settings

In all experiments, we adopt a unified set of hyperparameters that are independent of the specific model architecture.

First, the ambient discretization size is fixed to

$$M = 1024,$$

which provides a sufficiently fine grid for representing functions while maintaining computational tractability.

The number of retained modes of DWT or DFT is chosen from the set

$$K \in \{4, 8, 16, 32, 64, 128, 256\}.$$

This allows us to study how well functions can be approximated using a reduced number of coefficients. A smaller value of K leads to a more compact representation and lower computational cost. On the other hand, choosing K close to M improves approximation accuracy but introduces redundant computation and storage overhead. Therefore, this range provides a meaningful trade-off between efficiency and accuracy.

Once K is fixed, the input and output dimensions of the neural network are determined accordingly. Since complex coefficients are represented via their real and imaginary parts, the network operates in $\mathbb{R}^{2K}$. We use a multilayer perceptron with depth 3 (i.e., three hidden layers) and ReLU activation

functions. The architecture is given by

$$N_K : \mathbb{R}^{2K} \to \mathbb{R}^{2K}, \qquad 2K \to 2K \to 2K \to 2K,$$

with nonlinear activations in the first three layers and a linear output layer.

The prediction pipeline is:

$$f \xrightarrow{H_K} c^* \xrightarrow{\text{pack}} \mathbb{R}^{2K} \xrightarrow{N_K} \mathbb{R}^{2K} \xrightarrow{\text{unpack}} \hat{d} \xrightarrow{H_K^{-1}} \hat{g}.$$

For the dataset, we use 1000 training samples and 100 test samples. All samples are generated independently, and there is no overlap between the training and test sets.

To systematically evaluate generalization performance, we construct three different types of test sets:

- **Interpolation:** the test distribution is identical to the training distribution.
- **Distribution shift:** the test distribution is partially overlapping with the training range, but shifted relative to it.
- **Extrapolation:** the test distribution is completely disjoint from the training distribution.

The purpose of this design is to assess different aspects of the learned operator. The interpolation setting evaluates the in-distribution performance, measuring how well the model fits the training distribution and captures the underlying mapping when no distributional change is present. The distribution shift setting probes the robustness of the model under moderate deviations from the training data. This reflects more realistic scenarios where inputs may not follow the exact same distribution as the training set but still share partial overlap. The extrapolation setting is the most challenging regime, where the model must generalize beyond the support of the training data. This tests whether the learned operator captures structural properties of the mapping rather than merely memorizing patterns within the training range. Together, these three regimes provide a comprehensive evaluation of approximation accuracy, robustness to distributional changes, and true out-of-distribution generalization.

4.2. Translation Operator

Translation operators are very common in various scientific branches, and they are also used to solve linear transport equations. The translation operator maps a square-wave function to another square-wave function by shifting it along the domain while preserving its basic shape.

4.2.1. Problem Setup

We learn a 1D translation operator on $[0, 1]$:

$$g(x) = (\mathcal{T}f)(x) := f(x - \Delta), \qquad \Delta = 0.2.$$

Inputs are pulse functions of width $w = 0.2$:

$$f_a(x) = \mathbf{1}_{[a,a+w]}(x), \qquad g_a(x) = \mathbf{1}_{[a+\Delta,a+\Delta+w]}(x).$$

4.2.2. Dataset

The training parameters are sampled from a uniform distribution as

$$a_{\text{train}} \sim \mathcal{U}(0.1, 0.25).$$

The parameter ranges are chosen within the interval $(0, 1)$, which provides a representative and bounded parameter regime for the considered operator. This choice allows us to study the model behavior under moderate variations of the input while avoiding extreme values.

We construct three different test sets to evaluate generalization:

- Interpolation: $a_{\text{test}} \sim \mathcal{U}(0.12, 0.24)$.
- Distribution shift: $a_{\text{test}} \sim \mathcal{U}(0.13, 0.28)$.
- Extrapolation: $a_{\text{test}} \sim \mathcal{U}(0.26, 0.30)$.

The interpolation setting evaluates performance within the training distribution, the distribution-shift setting tests robustness under partially shifted distributions, and the extrapolation setting examines the model's ability to generalize beyond the training range.

4.2.3. Experiment Result

Interpolation

As shown in the experimental results in the Figure 4.1 below, the prediction accuracy of all three models increases with K, consistently following the pattern that DFT outperforms DWT-Flat, which in turn outperforms DWT-LevelWise. For any value of K, the prediction accuracy of DWT-LevelWise is significantly lower than that of the other two models. The optimal value of K is 256.

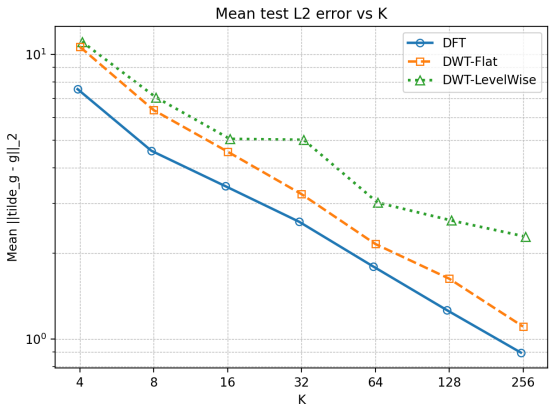
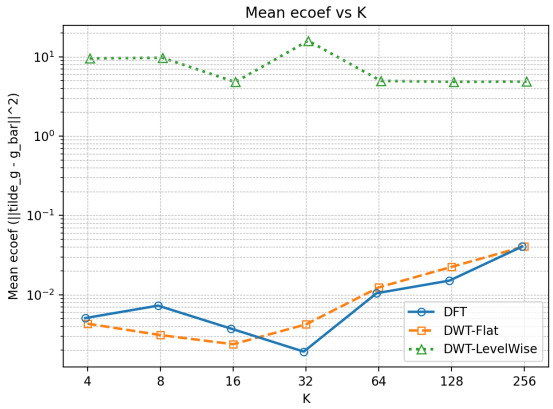
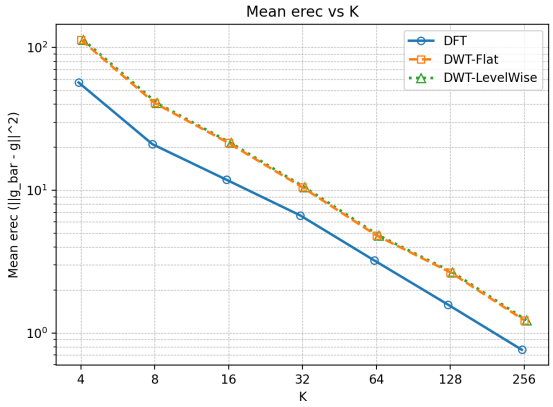


Figure 4.1: Translate Operator Mean $L2$ Error Changing with K

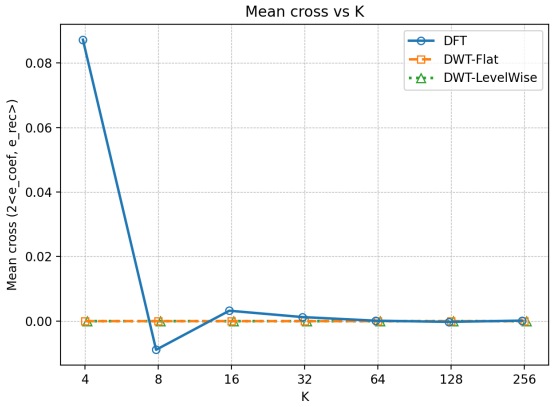
The decomposition of error in this experiment can be found in the Figure 4.2. The e_{coef} of DFT and DWT-Flat are relatively small. The reconstruction error e_{rec} dominates in all three models. The cross terms are all close to zero, which is consistent with the theoretical analysis.



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.2: Translation Operator Learning Error Decomposition

From the example solution shown in the Figure 4.3, it can be observed that in interpolation-type test tasks, under the setting of $K = 256$, both the DFT and DWT-Flat models produce highly accurate predictions. This is because, in this case, both their neural network approximation errors and reconstruction errors are very small. In contrast, the DWT-LevelWise model exhibits noticeable deviations in its predictions due to insufficient training of the neural network.

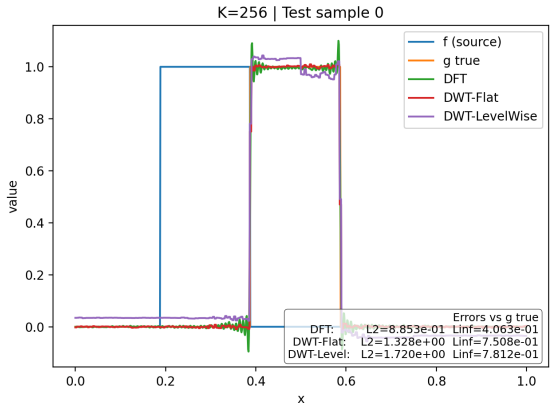


Figure 4.3: Example Solution Translation Operator on Interpolation Test Set

Distribution Shift and Extrapolation

Since the three models show similar performance on the distribution shift and extrapolation test sets, we only present and analyze the results on the extrapolation setting here.

In the experiments as shown in the Figure 4.4, the prediction accuracy of the three models exhibits a heavier non-monotonic trend compared with the interpolation and distribution shift cases. The optimal values of K are 16, 32, and 64, respectively.

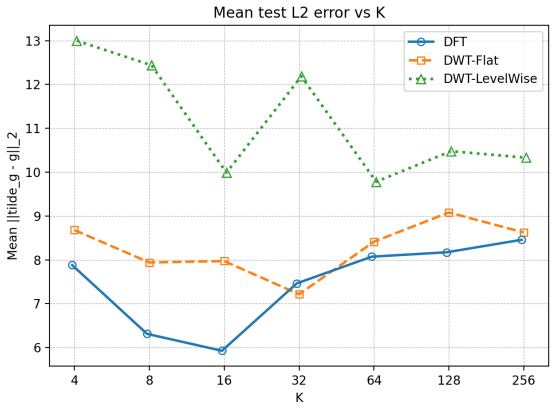
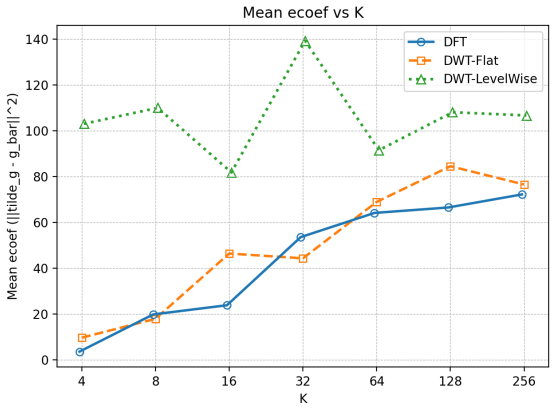
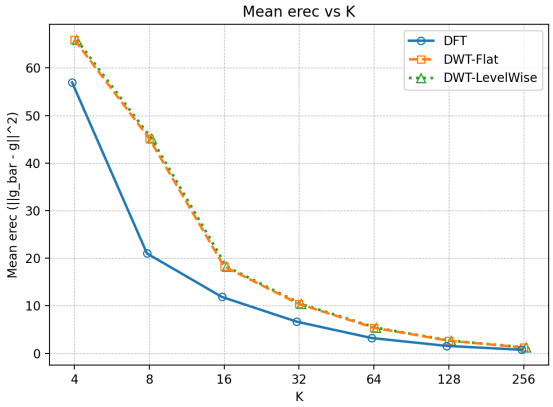


Figure 4.4: Translate Operator Mean $L2$ Error Changing with K

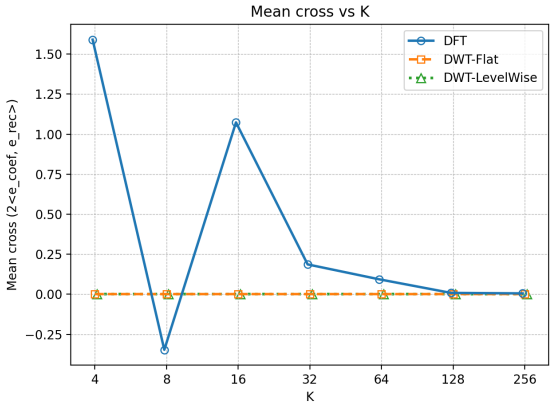
From the error decomposition in the Figure 4.5, it can be observed that the coefficient error e_{coef} for the DFT and DWT-Flat models is even monotonically increasing, eventually reaching the same order of magnitude as the reconstruction error e_{rec}. As a result, the total error exhibits a trend of first decreasing and then increasing.



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.5: Translate Operator Learning Error Decomposition

4.3. Translation Operator with Height Variation on Dyadic Partitions

A variant of the translation operator performs not only a horizontal translation but also on dyadic partitions of the axis. In the previous translation experiment, we observed that for learning a general translation operator, the DWT-based model does not demonstrate a clear advantage over the DFT model. However, when the translated function is aligned with a dyadic grid, the multiresolu-

tion structure of DWT may become more favorable. To investigate this hypothesis, we design the following experiment.

4.3.1. Problem Setup

The operator to be learned maps a source signal supported on a fixed interval to a target signal supported on another fixed interval on dyadic partitions, while preserving its amplitude. More precisely, for a scalar height parameter $h > 0$, the input function is

$$f_h(x) = \begin{cases} h, & x \in [0.125,\, 0.25], \\ 0, & \text{otherwise}, \end{cases}$$

and the corresponding output is

$$g_h(x) = \begin{cases} h, & x \in [0.5,\, 0.625], \\ 0, & \text{otherwise}. \end{cases}$$

Hence, the target operator $\mathcal{T}$ is defined by

$$\mathcal{T}(f_h) = g_h,$$

that is, $\mathcal{T}$ translates a rectangular pulse from the source interval $[0.125, 0.25]$ to the target interval $[0.5, 0.625]$ without changing its height. In this sense, the learning task is an operator learning problem rather than a standard finite-dimensional regression problem: the objective is to learn the map between input and output functions.

4.3.2. Dataset

The dataset is synthetically generated from the family of piecewise constant functions defined above. For every sample, only the height parameter h is varied, while the source interval and the target interval remain fixed. Therefore, the variability of the dataset comes entirely from the amplitude of the pulse.

For training, the height is sampled independently from the uniform distribution

$$h \sim \mathcal{U}(0.1,\, 0.25).$$

For testing the height is sampled from

$h \sim \mathcal{U}(0.12,\, 0.24)$, $h \sim \mathcal{U}(0.15,\, 0.28)$ and $h \sim \mathcal{U}(0.26,\, 0.30)$, corresponding to interpolation, distribution shift and extrapolation respectively.

4.3.3. Experiment Result

Interpolation

From the total error in the Figure 4.6, we observe that for the DWT-based models, the DWT-Flat model achieves nearly zero error once K exceeds a certain threshold, while the DWT-LevelWise model performs slightly worse. In contrast, the DFT model only gradually improves as K increases, eventually reaching a comparable level of accuracy to the DWT-Flat model at $K = 256$. All three models achieve their optimal prediction accuracy at the upper bound of $K = 256$ set in our experiments.

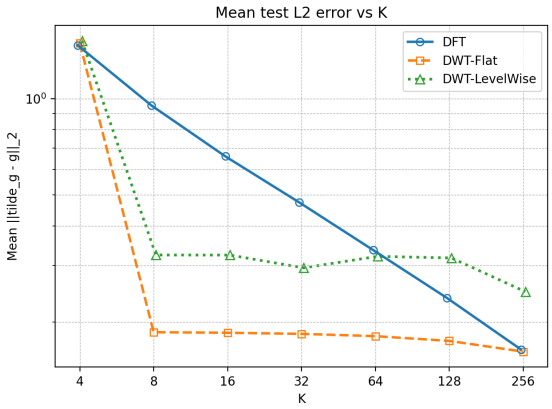
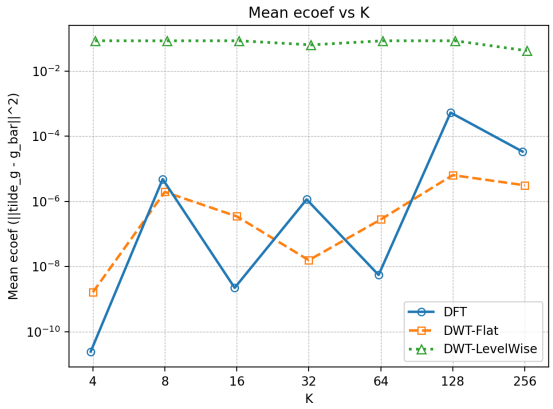
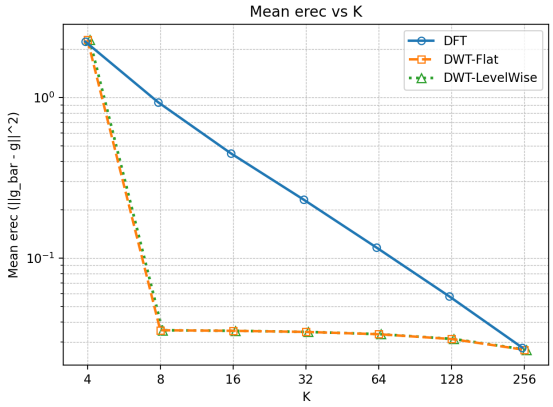


Figure 4.6: Translation Operator on Dyadic Partitions Mean $L2$ Error Changing with K

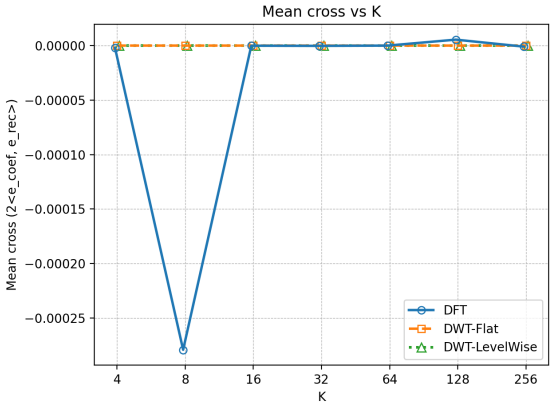
From the Figure 4.7, we can observe that the reconstruction error e_{rec} plays a decisive role. Since DWT can better approximate piecewise constant functions on dyadic grids, both the DWT-Flat and DWT-LevelWise models exhibit relatively smaller reconstruction errors.



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.7: Translation Operator on Dyadic Partitions Learning Error Decomposition

From the example solution in the Figure 4.8, we can observe that at $K = 256$, both the DFT and DWT-Flat models can approximate the target function quite well. Due to the oscillatory nature of Fourier basis functions, which introduces additional approximation error, the DWT-Flat model retains a slight advantage. In contrast, although the DWT-LevelWise model has learned a constant function, its predicted value deviates significantly from the true one.

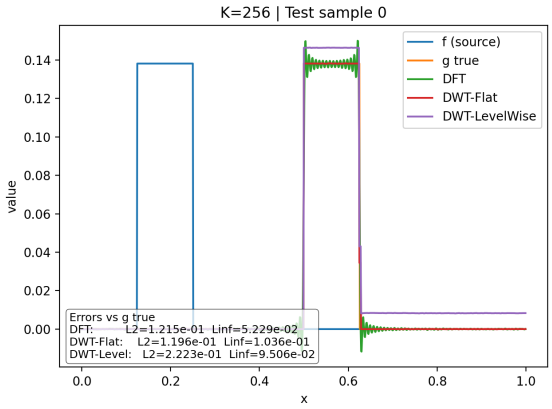


Figure 4.8: Example Solution Translation Operator on Dyadic Partitions on Interpolation Test Set

By comparing this example with the previous translation operator, we observe that although both are translation operators, when the target function is defined on a dyadic grid, the wavelet basis can approximate the function very accurately, resulting in a much smaller overall prediction error.

Distribution Shift and Extrapolation

The results observed in these two sets of experiments do not differ significantly from those in the interpolation setting. This indicates that the operator learning approach in this setup possesses strong generalization capability.

4.4. Poisson Equation Solver

The Poisson equation solver operator corresponds to solving the boundary value problem of the Poisson equation. It defines a mapping from the source term in the equation to the corresponding solution. In this setting, the source term can be chosen as a discontinuous function, while the resulting solution remains continuous.

4.4.1. Problem Setup

Consider the one-dimensional problem

$$\frac{d^2u}{dx^2} = f(x), \qquad x \in (0,1),$$

with fixed homogeneous Dirichlet boundary conditions

$$u(0) = 0, \qquad u(1) = 0.$$

The goal is to learn the mapping from the source term $f(x)$ to the solution $u(x)$.

Since

$$u''(x) = f(x),$$

integrating once gives

$$u'(x) = C_1 + \int_0^x f(s)\, ds.$$

Integrating again gives

$$u(x) = C_2 + C_1 x + \int_0^x \int_0^\xi f(s)\, ds\, d\xi.$$

Using $u(0) = 0$, we obtain

$$C_2 = 0.$$

Using $u(1) = 0$, we obtain

$$C_1 = -\int_0^1 \int_0^\xi f(s)\, ds\, d\xi.$$

Therefore,

$$u(x) = \int_0^x \int_0^\xi f(s)\, ds\, d\xi - x \int_0^1 \int_0^\xi f(s)\, ds\, d\xi.$$

When $f(x)$ is not continuous, the solution can be understood in the weak sense.

The forcing term $f(x)$ is defined as a square pulse function with an amplitude parameter p:

$$f(x; p) = \begin{cases} p, & 0.25 \leq x \leq 0.75, \\ 0, & \text{otherwise.} \end{cases}$$

4.4.2. Dataset

We use uniform sampling to construct the dataset. The training set is sampled with amplitudes $p \in [1.0, 1.9]$, and the test sets are sampled with amplitudes $p \in [1.0, 1.9]$, $p \in [1.2, 2.3]$ and $p \in [2.0, 2.5]$, according with interpolation, distribution shift and extrapolation respectively.

4.4.3. Experiment Result

Interpolation

From the Figure 4.9, it can be observed that the DWT-LevelWise model consistently exhibits noticeable variation as K changes, whereas the DFT and DWT-Flat models are able to steadily reduce their prediction errors as K increases.

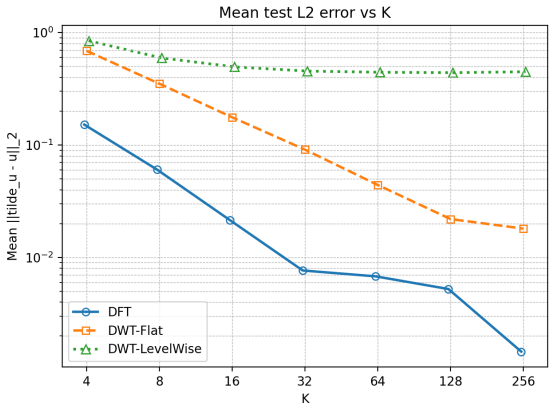
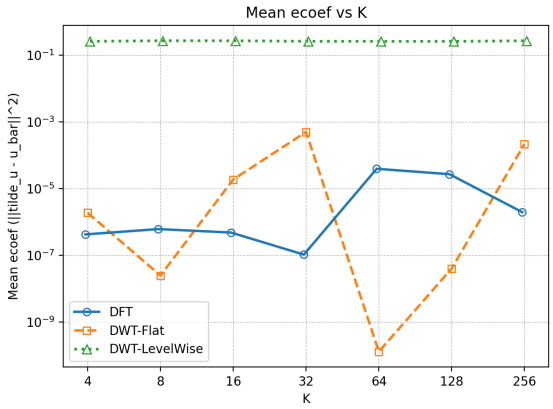
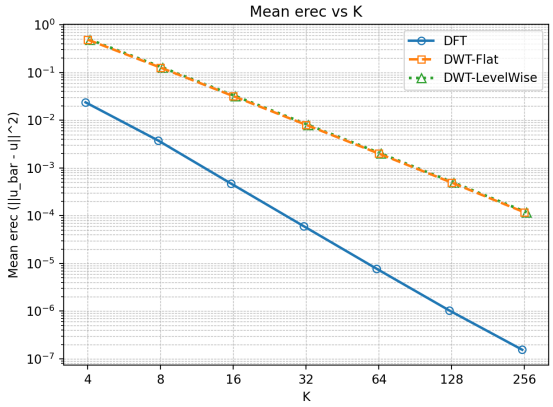


Figure 4.9: Poisson Solver Operator Mean $L2$ Error Changing with K

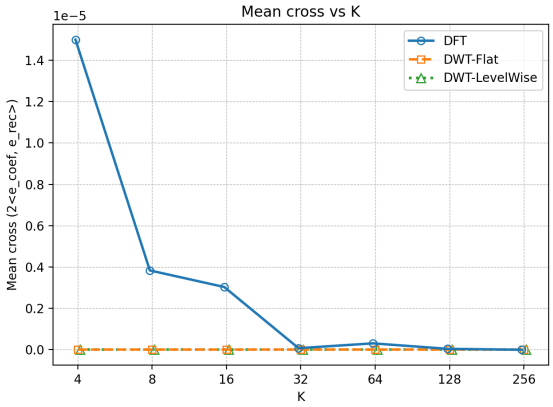
The error decomposition in the Figure 4.10 further confirms this observation. For the smooth target functions considered in the experiment, the DFT provides a more accurate approximation than the DWT, although both can achieve nearly zero reconstruction error at $K = 256$. However, for the DWT-LevelWise model, the neural network error remains consistently high across all values of K. As a result, even though its reconstruction error can be reduced to nearly zero, the total error remains large.



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.10: Poisson Solver Operator Learning Error Decomposition

From an example solution at $K = 256$ in the Figure 4.11, it can be observed that the predictions produced by the DFT and DWT-Flat models closely align with the true solution, making the difference between the two models hardly noticeable. In contrast, although the DWT-LevelWise model captures the overall shape, it exhibits a significant deviation in the vertical direction.

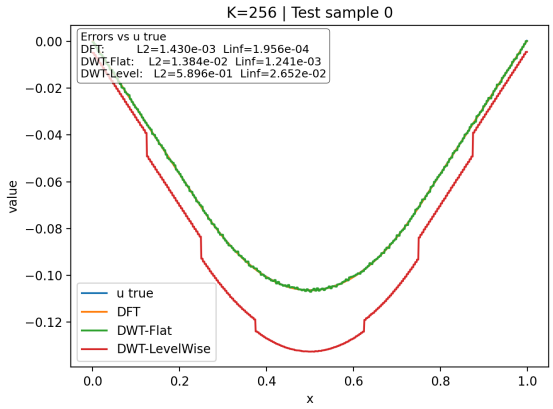


Figure 4.11: Example Solution Poisson Solver Operator on Interpolation Test Set

Distribution Shift and Extrapolation

The results observed in these two sets of experiments do not differ significantly from those in the interpolation setting. This indicates that the operator learning approach in this setup possesses strong generalization capability on this Possion solver task.

4.5. Diffusion Equation Solver

The diffusion equation solver operator corresponds to the initial value problem of the diffusion equation. It maps a continuous initial condition to the corresponding solution at a later time, which remains continuous due to the smoothing effect of diffusion.

4.5.1. Problem Setup

We consider the one-dimensional diffusion equation

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \qquad x \in [0,1], \quad t > 0,$$

with homogeneous Dirichlet boundary conditions

$$u(0,t) = u(1,t) = 0,$$

and initial condition

$$u(x,0) = u_0(x).$$

In this work, we set

$$D = 0.1,$$

and study the solution operator evaluated at the final time $t = 1$, namely

$$\mathcal{T} : u_0(x) \mapsto u(x,1).$$

To ensure compatibility with the homogeneous Dirichlet boundary conditions and to obtain a classical benchmark problem, we restrict the initial conditions to a finite-dimensional sine basis. Specifically, we consider

$$u_0(x) = a_1 \sin(\pi x) + a_2 \sin(2\pi x) + a_3 \sin(3\pi x),$$

where a_1, a_2, a_3 are real-valued coefficients.

Since the sine functions are eigenfunctions of the Laplacian under homogeneous Dirichlet boundary conditions, the corresponding exact solution at time $t = 1$ is given by

$$u(x,1) = a_1 e^{-D\pi^2} \sin(\pi x) + a_2 e^{-4D\pi^2} \sin(2\pi x) + a_3 e^{-9D\pi^2} \sin(3\pi x).$$

Therefore, the learning task is to approximate the operator that maps the coefficients and shape of the initial condition u_0 to the diffused solution $u(\cdot, 1)$.

4.5.2. Dataset

The training dataset is constructed from the family of initial conditions

$$u_0(x) = a_1 \sin(\pi x) + a_2 \sin(2\pi x) + a_3 \sin(3\pi x),$$

where the coefficients are sampled independently from the uniform distribution

$$a_1, a_2, a_3 \sim \mathcal{U}(-1, 1).$$

For test set, the coefficients are sampled from the uniform distribution

$a_1, a_2, a_3 \sim \mathcal{U}(-1, 1)$, $a_1, a_2, a_3 \sim \mathcal{U}(-0.8, 1.2)$ and $a_1, a_2, a_3 \sim \mathcal{U}(1.1, 1.5)$, corresponding to interpolation, distribution shift and extrapolation respectively.

For each sampled initial condition, the corresponding target output is defined as the exact solution of the diffusion equation at time $t = 1$, namely

$$u(x, 1) = a_1 e^{-D\pi^2} \sin(\pi x) + a_2 e^{-4D\pi^2} \sin(2\pi x) + a_3 e^{-9D\pi^2} \sin(3\pi x),$$

with $D = 0.1$.

4.5.3. Experiment Result

Interpolation

From the Figure 4.12 and the Figure 4.13, we observe that the model predictions are similar to those of the Poisson solver. Since the target functions are smooth, the DFT model provides the most accurate predictions. However, at $K = 256$, its advantage over the DWT-Flat model is not significant. In contrast, the DWT-LevelWise model performs worse than the other two models due to its relatively large neural network error.

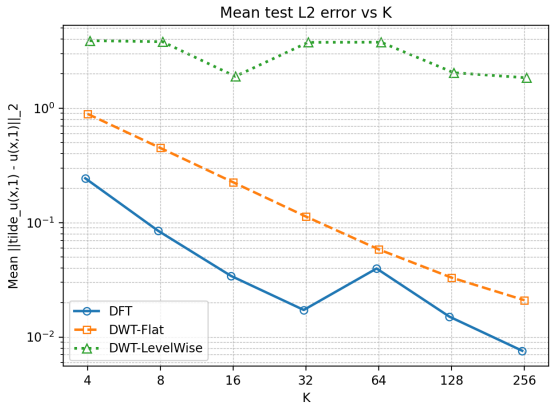
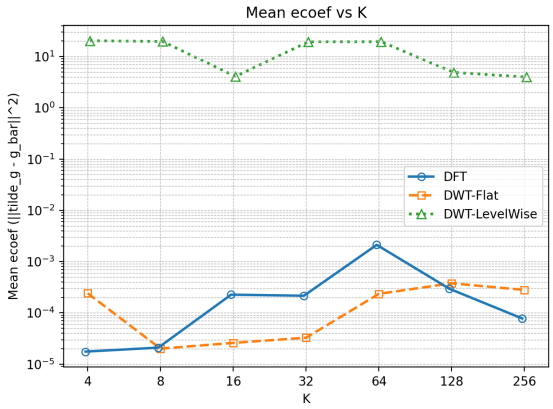
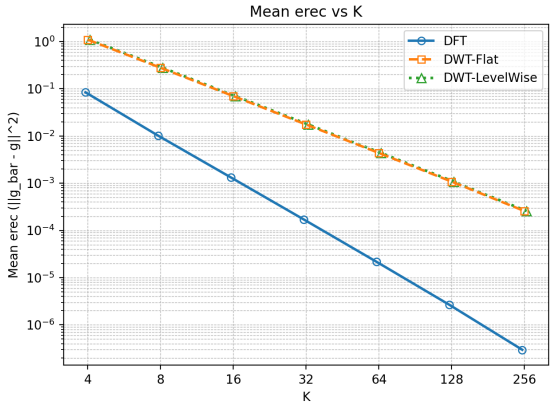


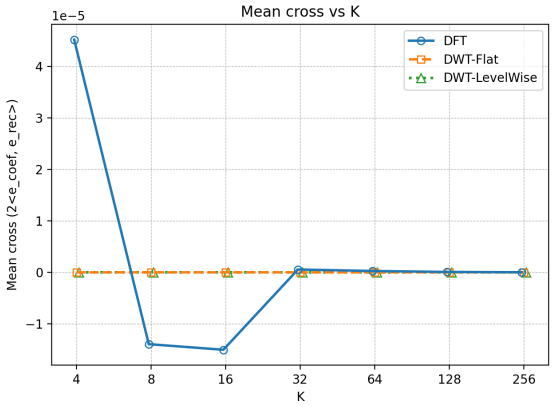
Figure 4.12: Diffusion Solver Operator Mean $L2$ Error Changing with K



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.13: Diffusion Solver Operator Learning Error Decomposition

From the example solution shown in the Figure 4.14, we observe that both the DFT and DWT-Flat models almost perfectly match the target function. In contrast, the DWT-LevelWise model makes errors in predicting the wavelet coefficients; as a result, although the overall shape of the predicted curve is similar, the detailed features are inaccurate.

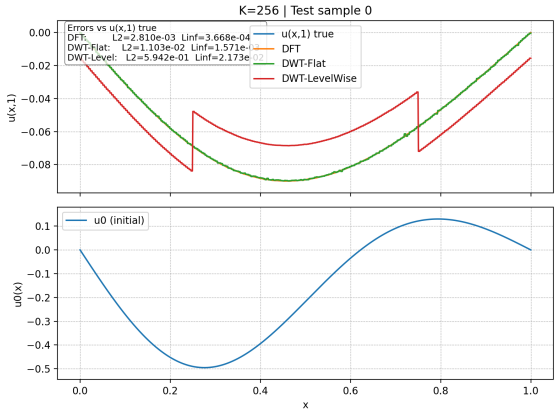


Figure 4.14: Example Solution Diffusion Solver Operator on Interpolation Test Set

Distribution Shift and Extrapolation

The results observed in these two sets of experiments do not differ significantly from those in the interpolation setting. This indicates that the operator learning approach in this setup possesses strong generalization capability on this diffusion solver task.

4.6. Burgers' Equation Solver

The solution operator for Burgers' equation maps an initial condition to the corresponding solution at a later time. For the inviscid case, smooth initial data can develop discontinuities (shocks) in finite time.

4.6.1. Problem Setup

We study the solution operator of the one-dimensional inviscid Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0,$$

posed on the spatial domain

$$x \in [-5, 5].$$

The initial condition is parameterized by a scalar k_f:

$$u(x, 0) = u_0(x; k_f) = 1 - k_f \tanh(x).$$

For each initial condition, we evolve the solution up to the final time

$$T = 1,$$

and aim to learn the nonlinear solution operator

$$\mathcal{S}_T : u_0 \mapsto u(T, \cdot).$$

In the numerical implementation, the Burgers' equation is solved using a conservative Rusanov [72] scheme with outflow boundary conditions. The time step is chosen adaptively according to a CFL condition [51] with CFL number 0.4.

4.6.2. Dataset

The training parameter is sampled from

$$k_f \sim \mathcal{U}[0.5, 2.0].$$

The test set consists of three regimes:

$$k_f \sim \mathcal{U}[0.5, 2.0],$$

for interpolation,

$$k_f \sim \mathcal{U}[0.8, 2.2],$$

for distribution shift, and

$$k_f \sim \mathcal{U}[2.1, 2.3],$$

for extrapolation.

For each sampled k_f, we construct the initial profile on a uniform spatial grid and compute the corresponding solution at $T = 1$. Each dataset sample is therefore an input-output pair

$$\big(u_0(\cdot; k_f), u(T, \cdot; k_f)\big).$$

4.6.3. Experiment Result

Interpolation

From the Figure 4.15, it can be observed that the prediction errors of all three models decrease monotonically as K increases, implying that the optimal choice of K is the upper bound $K = 256$ set in our experiments.

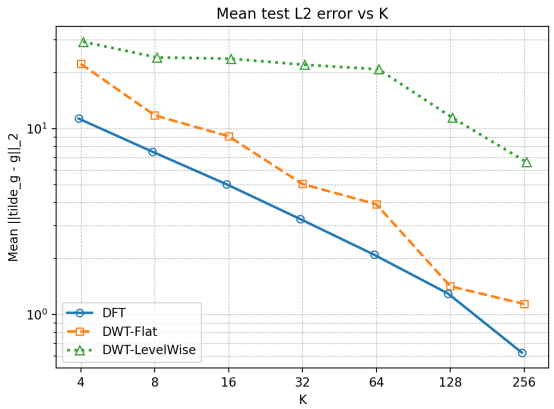
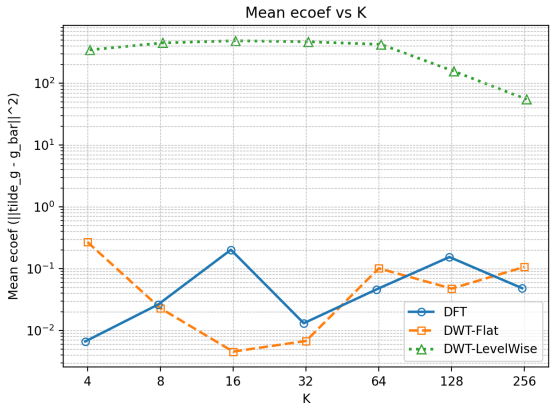
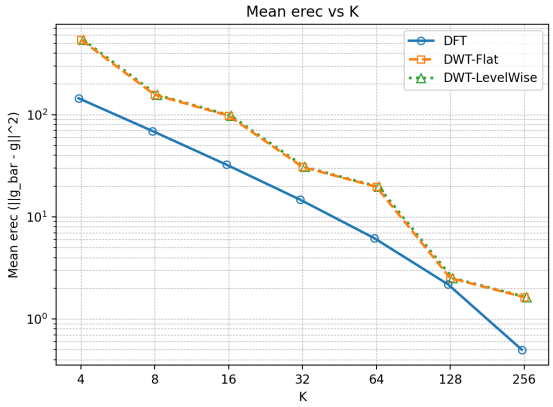


Figure 4.15: Burgers' Solver Operator Mean $L2$ Error Changing with K

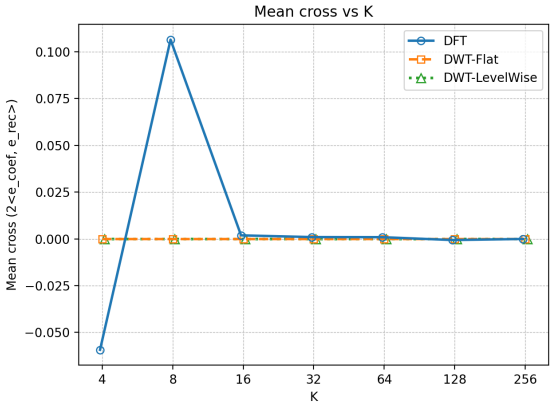
From the error decomposition in the Figure 4.16, it can be observed that although the neural network error of the DWT-LevelWise model remains relatively large, it also exhibits a noticeable variation with K, first increasing and then decreasing. In this case, the coefficient error e_{coef} is the dominant source of error.



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.16: Burgers' Solver Operator Learning Error Decomposition

From the example solution shown in the Figure 4.17, we observe that the initial condition of the Burgers equation evolves over time into a final state with a very steep gradient, which is nearly discontinuous. Despite this, at $K = 256$, both the DFT and DWT-Flat models still provide highly accurate predictions. In contrast, the DWT-LevelWise model exhibits significant deviations, as its neural network fails to capture the correct fine-scale features of the solution.

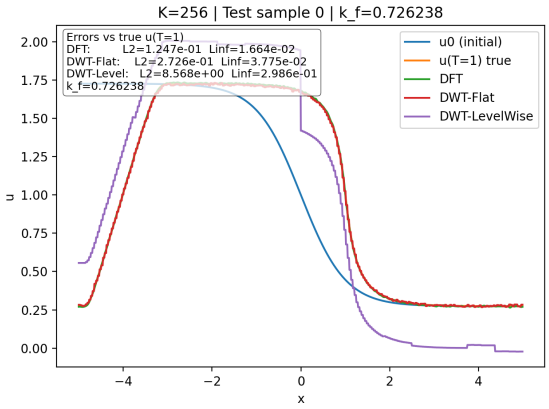


Figure 4.17: Example Solution Burgers' Solver Operator on Interpolation Test Set

Distribution Shift and Extrapolation

The results observed in these two sets of experiments do not differ significantly from those in the interpolation setting. This indicates that the operator learning approach in this setup possesses strong generalization capability on this Burgers solver task.

4.7. Approximation by Piecewise Constant Functions on Dyadic Partitions

We introduce an operator that maps a continuous function to its supremum-based piecewise constant approximation on a dyadic grid. This operator enables us to study the learnability of OL-Wavelet of such nonlinear transformations. It is closely related to the upper envelope [21].

4.7.1. Problem Setup

We consider the problem of learning an operator that maps a family of continuous functions to their dyadic piecewise constant approximations.

Let $\phi \in [0, 2\pi)$ be a phase parameter. We define a parametric family of functions $\{f_\phi\}_{\phi\in[0,2\pi)}$ on the unit interval $[0, 1]$ by

$$f_\phi(x) = \sin(4\pi x + \phi), \qquad x \in [0, 1].$$

Each function f_ϕ is smooth and oscillatory, with fixed frequency and varying phase.

Let $J \in \mathbb{N}$ denote the resolution level. We define the dyadic partition Δ_J of $[0, 1]$ as

$$\Delta_J = \{I_k^{(J)}\}_{k=0}^{2^J-1},$$

where

$$I_k^{(J)} = \begin{cases} \left[\dfrac{k}{2^J}, \dfrac{k+1}{2^J}\right), & k = 0, 1, \ldots, 2^J - 2, \\ \left[\dfrac{2^J-1}{2^J}, 1\right], & k = 2^J - 1. \end{cases}$$

This partition provides a dyadic decomposition of the interval $[0, 1]$ into 2^J subintervals of equal length.

We define the approximation operator Q_J that maps a function f to a piecewise constant function on Δ_J via a maximum-based rule:

$$(Q_J f)(x) = \sum_{k=0}^{2^J-1} \left(\max_{y\in I_k^{(J)}} f(y) \right) \mathbf{1}_{I_k^{(J)}}(x).$$

Since each f_ϕ is continuous and each interval $I_k^{(J)}$ is compact, the maximum is attained on every dyadic cell.

For the family $\{f_\phi\}$, the target output corresponding to f_ϕ is therefore

$$Q_J f_\phi.$$

If one works with purely discrete data, it is natural to define a discrete analogue of the operator by taking the maximum only over the sampled grid points contained in each dyadic cell. In that case one may use

$$(Q_J^h f)(x_j) = \max_{x_m \in I_k^{(J)} \cap \mathcal{G}_h} f(x_m), \qquad x_j \in I_k^{(J)},$$

where $\mathcal{G}_h$ denotes the sampling grid.

This discrete version is consistent with a fully sampled-data viewpoint, where the function is available only through its values on the grid. Although it is not identical to the continuous maximum-based operator in general, it provides a natural alternative when the problem is formulated entirely in discrete form.

Our goal is to learn the mapping

$$\mathcal{G}_J : f_\phi \mapsto Q_J f_\phi,$$

using a neural operator model.

4.7.2. Dataset

To construct the dataset, we sample phase parameters $\phi_i \in [0, 2\pi)$ for $i = 1, \ldots, N$, and form paired data

$$\mathcal{D} = \left\{ \left(f_{\phi_i},\, Q_J f_{\phi_i} \right) \right\}_{i=1}^{N}.$$

The variability in the dataset is introduced through the phase parameter ϕ. We generate a total of 1000 training samples and 100 testing samples, ensuring that all samples are mutually distinct.

Although the parameter space is given by $\phi \in [0, 2\pi)$, for training and evaluation we restrict sampling to smaller subintervals in order to study interpolation, distribution shift, and extrapolation.

For the training set, the phase parameter is sampled uniformly from the interval

$$\phi \sim \mathcal{U}([0, 0.25]).$$

We construct three different test sets to evaluate generalization:

- Interpolation:
$$\phi \sim \mathcal{U}([0, 0.25]).$$
- Distribution shift:
$$\phi \sim \mathcal{U}([0.05, 0.30]).$$
- Extrapolation:
$$\phi \sim \mathcal{U}([0.26, 0.50]).$$

4.7.3. Experiment Result

Interpolation

From the Figure 4.18 and the Figure 4.19, we observe that in this task, the DWT-LevelWise model achieves relatively high prediction accuracy for the first time. This is because, when K is sufficiently large, both its neural network error and reconstruction error become small. Compared to the DFT model, the DWT-Flat and DWT-LevelWise models do not exhibit smaller reconstruction errors when K takes values in $\{8, 16, 32\}$. However, when $K \in \{4, 64, 128, 256\}$, the DWT-Flat

and DWT-LevelWise models achieve better prediction accuracy than the DFT model due to their smaller reconstruction errors.

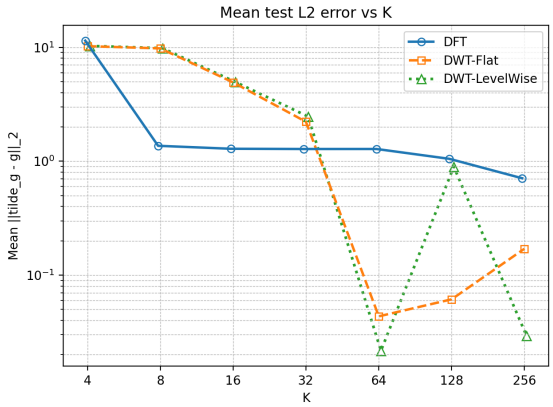
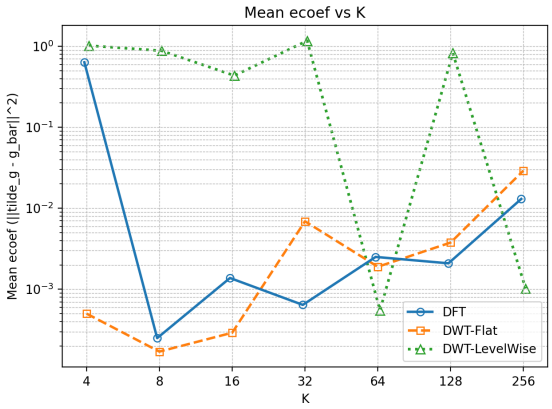
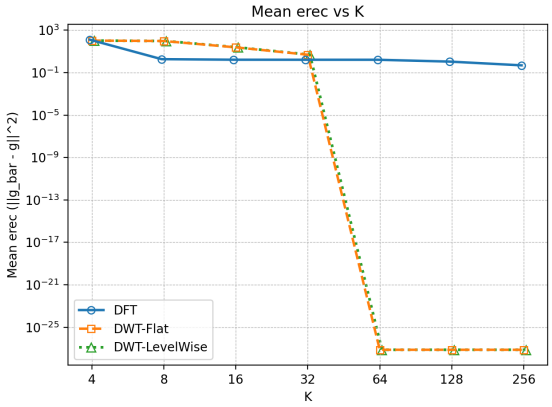


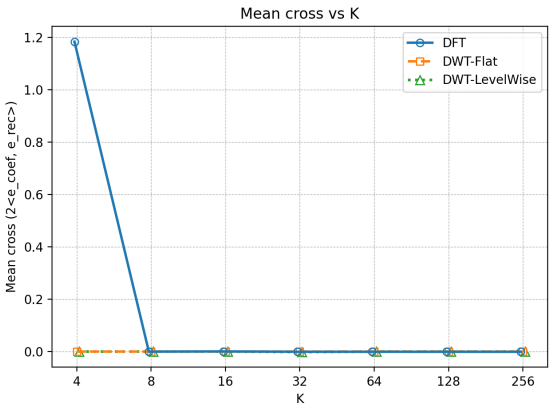
Figure 4.18: Approximation by Piecewise Constant Operator Mean $L2$ Error Changing with K



(a) e_{coef}



(b) e_{rec}



(c) cross term

Figure 4.19: Approximation by Piecewise Constant Operator Learning Error Decomposition

The distribution shift and extrapolation cases are similar to the interpolation case.

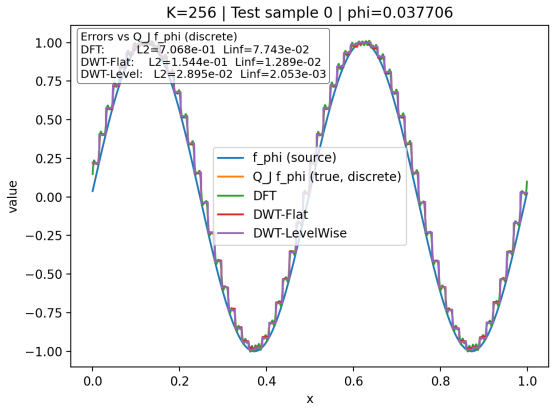


Figure 4.20: Example Solution Approximation by Piecewise Constant Operator on Interpolation Test Set

From the example solution in the Figure 4.20, we observe that when $K = 256$, all three models successfully capture the overall trend of the target equation, and their predictions closely match the true solution.

5 Discussion

Through theoretical analysis of the model and experiments on multiple operators, we obtained a wealth of practical data results. From these results, we can summarize and further analyze many conclusions.

5.1. Interpretation of Experimental Results

The experiments presented in Chapter 4 record the variations of several quantities, which collectively reflect the validity of the model.

5.1.1. Error Decomposition and the Role of the Cross Term

A first important observation from the experiments is that the cross terms of DWT models are consistently 0. This verifies that the coefficient-learning error and the reconstruction error are orthogonal as we have shown in Equation 3.2, so the total prediction error is governed entirely by the balance between these two terms. As a result, the error decomposition provides a transparent way to interpret the experimental outcomes: the overall performance can be understood by separately examining the neural network approximation quality and the representation quality.

The DFT model is slightly different. In the present implementation, the truncated Fourier modes are selected by retaining a finite set of low-frequency coefficients that is not always perfectly conjugate-symmetric when K is small. As a result, the reconstruction is not the exact orthogonal projection onto a Fourier subspace for real-valued functions. Consequently, the coefficient error and the reconstruction error are not strictly orthogonal, and the cross term may be nonzero. As K increases, this asymmetry becomes less significant relative to the total retained frequency range, so the cross term becomes negligible.

5.1.2. Comparison Between DFT and DWT-Flat

This decomposition clarifies the comparison between DFT and DWT-Flat. For many operator-learning tasks, the DFT-based model achieves slightly better overall prediction accuracy. The main reason is that the outputs of many such operators are relatively smooth, and smooth target functions are often more efficiently approximated by low-frequency Fourier modes. Consequently, the reconstruction error of the truncated Fourier representation is typically smaller than that of the corresponding wavelet representation.

In the experiments, the coefficient-learning errors of DFT and DWT-Flat are usually of similar magnitude and already quite small. Once e_{coef} is sufficiently reduced, the reconstruction term becomes the decisive factor. In this regime, the advantage of Fourier truncation in representing smooth outputs directly translates into better overall prediction accuracy.

However, this advantage is not universal. For some operators, the structure of the target functions is better aligned with wavelet representations. Representative examples are the translation operator on a dyadic grid and the constant approximation on dyadic partitions, where the outputs naturally match the multiscale structure of the Haar wavelet basis. In such cases, DWT-based truncation can preserve the essential features of the output more effectively, leading to a smaller reconstruction error and therefore better overall prediction accuracy when the coefficient-learning errors remain comparable.

These observations show that the comparison between DFT and DWT-Flat should not be interpreted as a universal superiority of one basis over the other. Rather, performance depends on how well the chosen representation matches the intrinsic structure of the operator outputs: smooth global features favor Fourier modes, while localized multiscale structures favor wavelets.

5.1.3. The Limitation of DWT-LevelWise

The behavior of DWT-LevelWise is qualitatively different. Its prediction accuracy is substantially worse than that of both DFT and DWT-Flat, and the error decomposition shows that this is mainly caused by a much larger coefficient-learning error rather than by the reconstruction term.

Although the levelwise architecture is motivated by the multiscale organization of wavelet coefficients, splitting the learning task across levels makes the coefficient mapping harder to learn effectively in the present setup. As a result, the neural network approximation error remains large and directly dominates the total prediction error. This suggests that, in practice, the success of a low-dimensional operator-learning method depends not only on the quality of the representation but also on the learnability of the induced coefficient mapping.

5.2. Effect of the Representation Dimension

5.2.1. The Dual Role of the Truncation Parameter K

An important factor in the experiments is the representation dimension, which is controlled by the truncation parameter K. In both the DFT and DWT representations, only the lowest K coefficients are retained, so K directly determines the dimension of the truncated representation.

In the present experimental setup, the role of K is more significant because the neural networks are deliberately kept simple. For both DFT and DWT-Flat, the truncated coefficient vector has dimension $2K$, and the width of the fully connected neural network is chosen proportionally to this dimension. Therefore, increasing K simultaneously increases both the expressiveness of the representation and the capacity of the neural network.

5.2.2. Behavior of the Reconstruction and Coefficient Errors

The reconstruction error e_{rec} behaves predictably as K increases. Since retaining more basis coefficients yields a more accurate approximation of the target function, the truncation error decreases monotonically with K. In the experiments, increasing K consistently reduces e_{rec}.

The behavior of the coefficient error e_{coef} is more complex. Unlike the reconstruction error, it depends on how well the neural network can learn the coefficient-to-coefficient mapping. Increasing K makes this mapping higher-dimensional and potentially more difficult to learn, while at the same time increasing the network width and its expressive power. As a result, the dependence of e_{coef} on K is generally non-monotonic.

5.2.3. Two Regimes for Choosing the Representation Dimension

The experiments reveal two distinct regimes. In the first regime, the coefficient error is already very small compared with the reconstruction error. In this case, the total prediction error is dominated by e_{rec}, so increasing K consistently improves performance by reducing the dominant source of error. In our experiments, this means that the largest tested value, $K = 256$, gives the best results.

In the second regime, the coefficient error is comparable to the reconstruction error. In this case,

increasing K does not necessarily improve performance. Although a larger K reduces the reconstruction error, it may also increase the difficulty of learning the coefficient map and thus increase e_{coef}. The optimal choice of K is therefore determined by a trade-off between representation accuracy and learnability, and it often occurs at an intermediate value.

5.3. Applicability Analysis of the Level-wise Wavelet Mapping

In the experiments, the prediction accuracy of the DWT-LevelWise model is consistently much lower than that of the other models. This is mainly because the architecture only learns coefficient mappings within each individual wavelet layer, without allowing interactions between different wavelet levels, which significantly restricts its expressive power. For many operators, however, the output wavelet coefficients depend on input coefficients across multiple scales. When such cross-level dependencies are strong, the levelwise restriction makes the learning task considerably more difficult, leading to a large neural network approximation error and poor prediction performance. Therefore, the effectiveness of the DWT-LevelWise model depends on the degree of correlation between wavelet coefficients across different levels. If these cross-level correlations are strong, the model is unlikely to perform well. A simple numerical analysis of the correlations between wavelet coefficients at different scales can therefore be used to assess whether a given operator is suitable for learning with the DWT-LevelWise architecture. Inspired by wavelet cross-correlation [12] methods for analyzing multiscale dependencies, we introduce the following method to quantify cross-scale coupling induced by the operator.

We diagnose cross-scale coupling of an operator $T : \mathbb{R}^M \to \mathbb{R}^M$ by estimating a matrix $E \in \mathbb{R}^{L\times L}$ in a Haar wavelet domain. Assume $M = 2^J$ and let W denote the orthonormal Haar DWT up to level J (with periodic boundary handling), so that Wf is the standard coefficient list

$$Wf = \big(cA_J,\ cD_J,\ cD_{J-1}, \ldots, cD_1\big), \qquad L = J + 1.$$

For each level index $m \in \{0, 1, \ldots, J\}$, define the projection operator P_m acting on the coefficient list by keeping only the m-th block at one scale and setting all other blocks to zero. Using this projection, we form a single-scale input by inverse transforming only that level:

$$f_{(m)} := W^{-1}\big(P_m(Wf)\big).$$

We then apply the operator and re-analyze the output in the same wavelet basis:

$$g_{(m)} := T\big(f_{(m)}\big), \qquad Wg_{(m)} = WT\big(f_{(m)}\big).$$

To quantify how much energy injected at input level m appears at output level l, we measure the squared ℓ_2 norm of the corresponding coefficient blocks. Writing $\|\cdot\|_2$ for the Euclidean norm of the coefficients within a level, the (normalized) energy transfer from m to l for a given sample f is

$$\frac{\big\|P_l\, WT\big(f_{(m)}\big)\big\|_2^2}{\big\|P_m\, Wf\big\|_2^2 + \varepsilon},$$

and the matrix entry is defined as its expectation over random inputs:

$$E_{l,m} \approx \mathbb{E}_s\left[\frac{\big\|P_l\, WT\big(f_{(m)}\big)\big\|_2^2}{\big\|P_m\, Wf\big\|_2^2 + \varepsilon}\right].$$

In practice, the expectation $\mathbb{E}_s[\cdot]$ is approximated by Monte Carlo averaging over S independent samples $\{f^{(s)}\}_{s=1}^S$:

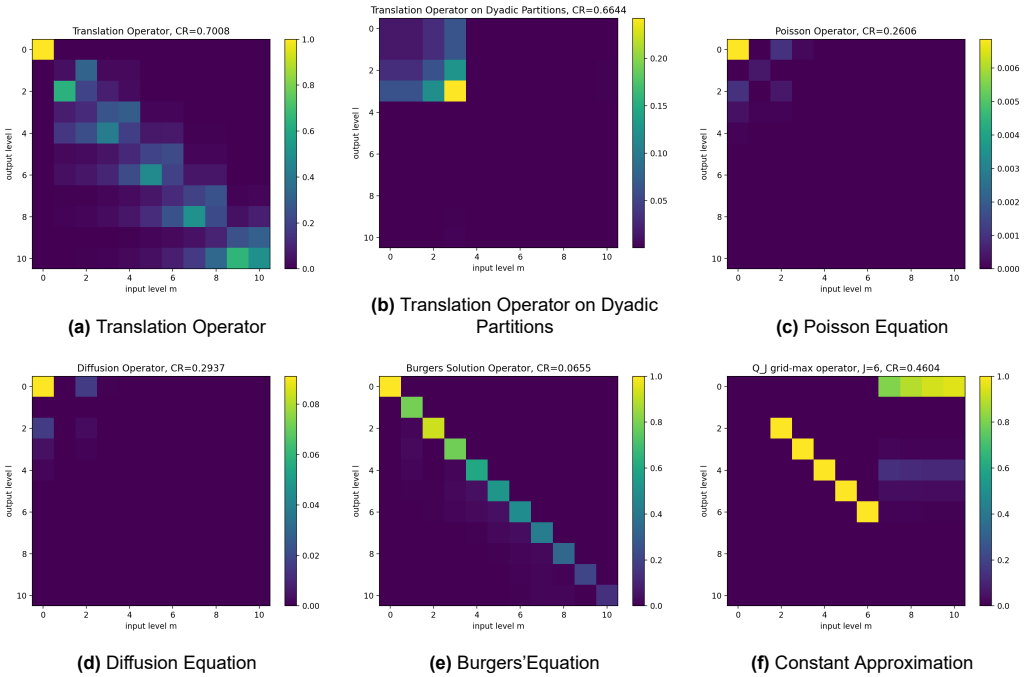
$$\widehat{E}_{l,m} = \frac{1}{S}\sum_{s=1}^{S}\frac{\big\|P_l\, WT\Big(f_{(m)}^{(s)}\Big)\big\|_2^2}{\big\|P_m\, Wf^{(s)}\big\|_2^2 + \varepsilon}, \qquad f_{(m)}^{(s)} := W^{-1}\big(P_m(Wf^{(s)})\big).$$

The normalization by $\|P_m W f^{(s)}\|_2^2$ makes each column m comparable across scales, interpreting $E_{l,m}$ as "output energy at level l produced per unit input energy at level m." The small constant $\varepsilon > 0$ is a numerical regularizer (noise floor) that prevents instability when the sampled input happens to have extremely small energy at level m, which would otherwise lead to division by near-zero and dominate the Monte Carlo average. If T is scale-decoupled in this wavelet representation, then $\widehat{E}$ concentrates near the diagonal; significant off-diagonal mass indicates cross-scale coupling induced by T.

To summarize the amount of cross-scale coupling by a single scalar, we further define the coupling ratio

$$\mathrm{CR} = \frac{\sum_{l \neq m} \widehat{E}_{l,m}}{\sum_{l,m} \widehat{E}_{l,m}}.$$

A larger value of CR indicates stronger off-diagonal energy transfer and hence stronger cross-scale coupling induced by the operator.



(a) Translation Operator

(b) Translation Operator on Dyadic Partitions

(c) Poisson Equation

(d) Diffusion Equation

(e) Burgers'Equation

(f) Constant Approximation

Figure 5.1: Heatmaps of Wavelet Level Coupling Analysis

The analysis in Figure 5.1 indicates that the operators studied in this paper exhibit strong cross-level correlations among wavelet coefficients, and therefore are not suitable for learning with the DWT-LevelWise architecture.

6

Conclusion

This chapter presents the conclusions of the work.

6.1. Main Results

This work proposes OL-Wavelet, a wavelet-based neural operator designed to approximate operator mappings in a compressed spectral representation. The method represents input and output functions by truncated Haar wavelet coefficients and learns the induced mapping between coefficient spaces using neural networks. Through theoretical error analysis and numerical experiments, we investigate the effectiveness of OL-Wavelet on several classes of operators, including translation operators, PDE solution operators, and operators involving dyadic piecewise structures.

The main findings can be summarized as follows.

1. **Feasibility of wavelet-based operator learning.** For the main research question, the experiments show that OL-Wavelet is capable of approximating the considered operator classes with good accuracy. In particular, for tasks involving piecewise constant functions on dyadic partitions, such as translation operators and piecewise approximation functions on dyadic partitions, the wavelet representation provides a natural and efficient description of the underlying functions. This supports the feasibility of using wavelet coefficients as the representation space for spectral neural operator learning.
2. **Comparison between wavelet and Fourier representations.** Compared with the Fourier-based spectral neural operator, OL-Wavelet achieves higher prediction accuracy in some specific settings, especially when the target functions are piecewise constant functions on dyadic partitions. However, the experiments also show that OL-Wavelet is not uniformly superior to the Fourier-based method. For certain PDE solution operators, the reconstruction error caused by truncating the wavelet expansion can be larger than the corresponding Fourier reconstruction error, which limits the overall prediction accuracy of OL-Wavelet. Therefore, the advantage of the wavelet representation is problem-dependent rather than universal.
3. **Error decomposition.** The total prediction error can be decomposed into two terms: the neural network prediction error in the coefficient space, the reconstruction error caused by truncating the spectral representation.

   This decomposition provides a useful diagnostic tool for identifying whether the dominant source of error comes from the neural network approximation or from the loss of information due to coefficient truncation.
4. **Choice of representation dimension.** The number of retained coefficients K plays a crucial

role in the performance of OL-Wavelet. A small value of K leads to strong compression but may produce large reconstruction error, whereas a larger value of K preserves more information but increases the dimension and width of the neural network. The experiments indicate that the optimal choice of K is determined by this trade-off and should be selected empirically for each operator-learning task.

5. **Joint learning versus level-wise learning.** The numerical results suggest that learning the mapping between wavelet coefficients independently at each resolution level is generally not sufficient. Although the level-wise architecture reduces the number of connections and respects the multiresolution structure of wavelets, it ignores interactions between different wavelet levels. The joint architecture, which concatenates the retained coefficients and learns the mapping using a global MLP, generally provides better prediction accuracy. This indicates that cross-level dependencies are important for approximating the considered operators.

Overall, the results show that OL-Wavelet is a feasible and effective wavelet-based neural operator, particularly for problems with localized or discontinuous structures. At the same time, its performance depends strongly on the reconstruction accuracy of the truncated wavelet representation and on the choice of representation dimension. These findings suggest that wavelet-based operator learning is most advantageous when the target operator is compatible with the multiscale and spatially localized structure of wavelet bases.

6.2. Limitations and Challenges

Despite the promising performance of the proposed model in several experiments, there remain several limitations and challenges that should be addressed in future work.

6.2.1. Limited Range of Benchmark Tasks

At present, OL-Wavelet has been evaluated only on a limited number of constructed operator learning tasks. These tasks are mainly designed for controlled experimental analysis, such as translation operators, selected one-dimensional PDE solution operators, and dyadic piecewise function approximation. Although these experiments provide useful evidence for the feasibility of the proposed method, they are not sufficient to demonstrate that OL-Wavelet can accurately approximate a broad class of commonly studied operators. In particular, the current results do not yet fully establish the robustness of OL-Wavelet across more diverse operator learning problems, such as operators with different boundary conditions, irregular coefficients, multiscale forcing terms, or more complex nonlinear dynamics. Therefore, additional benchmark problems from scientific computing and engineering applications are needed to further validate the general applicability of the model.

6.2.2. Incomplete Understanding of the Prediction Error

The present error analysis decomposes the total prediction error into the neural network prediction error, the reconstruction error, and their cross term. This decomposition is useful for identifying different sources of error. However, the behavior of the neural network prediction error is not yet fully understood. In the experiments, this term does not exhibit a simple or consistent trend relative to the reconstruction error. For example, increasing the number of retained wavelet coefficients may reduce the reconstruction error, but it also increases the dimension of the coefficient space and the width of the neural network, which can make the prediction error harder to control.

A more rigorous theoretical analysis is therefore needed to explain how the neural network prediction error depends on the representation dimension, the network architecture, the training sample size, and the regularity of the target operator. Possible directions include deriving generalization bounds for the coefficient-space neural network, studying approximation rates of MLPs for mappings between wavelet coefficient spaces, and analyzing the trade-off between spectral truncation error and statistical learning error. Such results would provide a stronger theoretical foundation

for selecting the representation dimension and designing more stable OL-Wavelet architectures.

6.2.3. Restriction to One-dimensional Problems

The current formulation and experiments are restricted to one-dimensional domains. While this setting provides a useful proof of concept, many practical problems in scientific computing involve two-dimensional or three-dimensional domains. Extending OL-Wavelet to higher-dimensional operator learning problems is therefore an important but nontrivial direction. In higher dimensions, the number of wavelet coefficients grows rapidly, and the multilevel structure becomes more complex. This may increase both the computational cost and the difficulty of designing efficient neural architectures. Future work should investigate multidimensional wavelet bases, tensor-product wavelets, adaptive sparse representations, and architectures that can exploit multiscale structure without suffering from excessive dimensional growth.

6.3. Future Work

Several directions can be explored in future work to further improve and extend the current OL-Wavelet framework.

6.3.1. Expanding the Range of Operator Learning Tasks

Future studies should evaluate OL-Wavelet on a wider variety of operator learning problems, particularly those arising from practical scientific and engineering applications. Incorporating more complex PDE systems, irregular domains, variable coefficients, and real-world datasets would provide stronger evidence of the model's effectiveness and robustness, and help assess its applicability beyond the constructed benchmark tasks considered in this work.

6.3.2. Developing a More Rigorous Theoretical Analysis

A more comprehensive theoretical analysis of the model error is an important direction for future research. In particular, the interaction between the neural network prediction error and the reconstruction error deserves deeper investigation. A key challenge is to understand how the prediction error depends on the representation dimension, the architecture of the neural network, and the size of the training data. Establishing sharper generalization bounds, approximation rates for coefficient-space mappings, and theoretical criteria for selecting the optimal representation dimension would significantly strengthen the mathematical foundation of the method.

6.3.3. Extending the Framework to Higher-dimensional Problems

Another important direction is to extend the current approach from one-dimensional settings to two-dimensional and three-dimensional problems. Many practical operator learning tasks arise in higher-dimensional domains, such as fluid dynamics, heat transfer, and elasticity. Developing efficient wavelet representations and scalable training strategies for higher-dimensional operators will be crucial for applying OL-Wavelet to more realistic scenarios.

6.3.4. Exploring Adaptive Coefficient Selection Strategies

The current framework uses a fixed truncation strategy by retaining a prescribed number of wavelet coefficients. A natural extension is to develop adaptive coefficient selection strategies that allocate representation resources according to the structure of the input and output functions. For example, thresholding strategies based on coefficient magnitude, energy distribution, or learned sparsity patterns may improve the balance between compression efficiency and approximation accuracy. Such adaptive representations may be particularly useful for operators with localized singularities or strongly nonuniform behavior.

6.3.5. Investigating Alternative Wavelet Bases

The current implementation mainly uses the Haar wavelet basis due to its simplicity and suitability for piecewise structures. However, different operator classes may be better represented by differ-

ent wavelet families. Future work could investigate smoother wavelet bases, such as Daubechies wavelets [74] or Morlet wavelets [14], to improve approximation quality for smoother PDE solutions. A systematic comparison between different wavelet bases may also provide practical guidelines for choosing the most suitable representation for a given operator learning problem.

References

[1] L. Aguiar-Conraria and M. J. Soares, "The continuous wavelet transform: Moving beyond uni- and bivariate analysis," *Journal of Economic Surveys*, vol. 28, no. 2, pp. 344–375, 2014. DOI: 10.1111/joes.12012.

[2] N. Ahmadi, Q. Cao, J. D. Humphrey, and G. E. M. Karniadakis. "Physics-informed machine learning in biomedical science and engineering." arXiv: 2510.05433. (2025), [Online]. Available: https://arxiv.org/abs/2510.05433.

[3] R. Alizadeh, J. K. Allen, and F. Mistree, "Managing computational complexity using surrogate models: A critical review," *Research in Engineering Design*, vol. 31, no. 3, pp. 275–298, 2020. DOI: 10.1007/s00163-020-00336-7.

[4] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4–5, pp. 185–196, 1993. DOI: 10.1016/0925-2312(93)90006-O.

[5] R. Bai, Z. Liu, X. Wu, Y. Wu, and X. Qian, "Hermite neural operator for solving partial differential equations on unbounded domains," *Machine Learning: Science and Technology*, vol. 7, no. 1, p. 015 004, 2026.

[6] Y. Bai, "ReLU-function and derived function review," in *SHS Web of Conferences*, vol. 144, 2022, p. 02 006. DOI: 10.1051/shsconf/202214402006.

[7] R. Bhaskara, H. Viswanath, and A. Bera, "Trajectory prediction for robot navigation using flow-guided markov neural operator," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 15 209–15 216. DOI: 10.1109/ICRA57147.2024.10611154.

[8] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart, "Model reduction and neural networks for parametric pdes," *SMAI Journal of Computational Mathematics*, vol. 7, pp. 121–157, 2021. DOI: 10.5802/smai-jcm.74.

[9] N. Boullé and A. Townsend, "A mathematical guide to operator learning," in *Handbook of Numerical Analysis*, vol. 25, Elsevier, 2024, pp. 83–125. DOI: 10.1016/bs.hna.2024.05.003.

[10] R. N. Bracewell, "The Fourier transform," *Scientific American*, vol. 260, no. 6, pp. 86–95, 1989. DOI: 10.1038/scientificamerican0689-86.

[11] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. M. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021. DOI: 10.1007/s10409-021-01148-1.

[12] E. Casagrande, B. Mueller, D. G. Miralles, D. Entekhabi, and A. Molini, "Wavelet correlations to reveal multiscale coupling in geophysical systems," *Journal of Geophysical Research: Atmospheres*, vol. 120, no. 15, pp. 7555–7572, 2015. DOI: 10.1002/2015JD023265.

[13] A. Chattopadhyay, M. Gray, T. Wu, A. B. Lowe, and R. He, "OceanNet: A principled neural operator-based digital twin for regional oceans," *Scientific Reports*, vol. 14, no. 1, p. 21 181, 2024. DOI: 10.1038/s41598-024-72145-0.

[14] M. X. Cohen, "A better way to define and describe morlet wavelets for time-frequency analysis," *NeuroImage*, vol. 199, pp. 81–86, 2019. DOI: 10.1016/j.neuroimage.2019.05.048.

[15] I. Daubechies, *Ten Lectures on Wavelets* (CBMS-NSF Regional Conference Series in Applied Mathematics). SIAM, 1992, vol. 61. DOI: 10.1137/1.9781611970104.

[16] L. Debnath and D. Bhatta, *Integral Transforms and Their Applications*, 3rd ed. Chapman and Hall/CRC, 2015. DOI: 10.1201/b17670.

[17] A. Défossez, L. Bottou, F. Bach, and N. Usunier. “A simple convergence proof of adam and adagrad.” arXiv: 2003.02395. (2020), [Online]. Available: https://arxiv.org/abs/2003.02395.

[18] F. Dietrich and W. Schilders, “Scientific machine learning,” *Mathematische Semesterberichte*, vol. 72, no. 2, pp. 89–115, 2025. DOI: 10.1007/s00591-025-00399-4.

[19] P. Du, K. H. Tan, and S. H. Cheung, “Transfer learning enhanced heat transfer neural operators (HTNO) for efficient thermal analysis in concrete structures,” *Engineering Structures*, vol. 329, p. 119 782, 2025. DOI: 10.1016/j.engstruct.2025.119782.

[20] P. Duhamel and M. Vetterli, “Fast fourier transforms: A tutorial review and a state of the art,” *Signal Processing*, vol. 19, no. 4, pp. 259–299, 1990. DOI: 10.1016/0165-1684(90)90158-U.

[21] H. Edelsbrunner, L. J. Guibas, and M. Sharir, “The upper envelope of piecewise linear functions: Algorithms and applications,” *Discrete & Computational Geometry*, vol. 4, no. 4, pp. 311–336, 1989. DOI: 10.1007/BF02187733.

[22] S. Elfwing, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” *Neural Networks*, vol. 107, pp. 3–11, 2018. DOI: 10.1016/j.neunet.2017.12.012.

[23] V. S. Fanaskov and I. V. Oseledets, “Spectral neural operators,” *Doklady Mathematics*, vol. 108, no. Suppl. 2, S226–S232, 2023. DOI: 10.1134/S1064562423701107.

[24] S. Garg and S. Chakraborty, “Neuroscience-inspired neural operator for partial differential equations,” *Journal of Computational Physics*, vol. 515, p. 113 266, 2024. DOI: 10.1016/j.jcp.2024.113266.

[25] C. R. Gin, D. E. Shea, S. L. Brunton, and J. N. Kutz, “DeepGreen: Deep learning of green’s functions for nonlinear boundary value problems,” *Scientific Reports*, vol. 11, no. 1, p. 21 614, 2021. DOI: 10.1038/s41598-021-00773-x.

[26] D. Gottlieb and C.-W. Shu, “On the gibbs phenomenon and its resolution,” *SIAM Review*, vol. 39, no. 4, pp. 644–668, 1997. DOI: 10.1137/S0036144596301390.

[27] G. Gupta, X. Xiao, and P. Bogdan, “Multiwavelet-based operator learning for differential equations,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 24 048–24 062. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/c9e5c2b59d98488fe1070e744041ea0e-Abstract.html.

[28] Z. Hao, Z. Wang, H. Su, *et al.*, “GNOT: A general neural operator transformer for operator learning,” in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023, pp. 12 556–12 569. [Online]. Available: https://proceedings.mlr.press/v202/hao23c.html.

[29] E. N. Hellström, “Pricing operators for financial instruments with machine learning,” Master’s thesis, KTH Royal Institute of Technology, 2025. [Online]. Available: https://kth.diva-portal.org/smash/record.jsf?pid=diva2:2023390.

[30] B. Jawerth and W. Sweldens, “An overview of wavelet-based multiresolution analyses,” *SIAM Review*, vol. 36, no. 3, pp. 377–412, 1994. DOI: 10.1137/1036095.

[31] P. Jiang, Z. Yang, J. Wang, *et al.*, “Efficient super-resolution of near-surface climate modeling using the fourier neural operator,” *Journal of Advances in Modeling Earth Systems*, vol. 15, no. 7, e2023MS003800, 2023. DOI: 10.1029/2023MS003800.

[32] C. Kaewnuratchadasorn, J. Wang, C.-W. Kim, and X. Deng, “Geometry physics neural operator solver for solid mechanics,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 40, no. 10, pp. 1388–1404, 2025. DOI: 10.1111/mice.13405.

[33] S. E. Kelly, “Gibbs phenomenon for wavelets,” *Applied and Computational Harmonic Analysis*, vol. 3, no. 1, pp. 72–81, 1996. DOI: 10.1006/acha.1996.0006.

[34] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” arXiv: 1412.6980. (2014), [Online]. Available: https://arxiv.org/abs/1412.6980.

[35] I. Kononenko, “Bayesian neural networks,” *Biological Cybernetics*, vol. 61, no. 5, pp. 361–370, 1989. DOI: 10.1007/BF00200801.

[36] J. Kossaifi, N. Kovachki, K. Azizzadenesheli, and A. Anandkumar. “Multi-grid tensorized fourier neural operator for high-resolution pdes.” arXiv: 2310.00120. (2023), [Online]. Available: https://arxiv.org/abs/2310.00120.

[37] N. B. Kovachki, S. Lanthaler, and A. M. Stuart, “Operator learning: Algorithms and analysis,” in *Handbook of Numerical Analysis*, vol. 25, Elsevier, 2024, pp. 419–467. DOI: 10.1016/bs.hna.2024.05.009.

[38] T. Kurth, S. Subramanian, P. Harrington, *et al.*, “FourCastNet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators,” in *Proceedings of the Platform for Advanced Scientific Computing Conference*, 2023, pp. 1–11. DOI: 10.1145/3592979.3593412. [Online]. Available: https://dl.acm.org/doi/10.1145/3592979.3593412.

[39] S. Lanthaler, “Operator learning with PCA-Net: Upper and lower complexity bounds,” *Journal of Machine Learning Research*, vol. 24, no. 318, pp. 1–67, 2023. [Online]. Available: https://www.jmlr.org/papers/v24/23-0478.html.

[40] K. R. Lennon, G. H. McKinley, and J. W. Swan, “Scientific machine learning for modeling and simulating complex fluids,” *Proceedings of the National Academy of Sciences*, vol. 120, no. 27, e2304669120, 2023. DOI: 10.1073/pnas.2304669120.

[41] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: Analysis, applications, and prospects,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2021. DOI: 10.1109/TNNLS.2021.3084827.

[42] Z. Li, N. Kovachki, K. Azizzadenesheli, *et al.*, *Neural operator: Graph kernel network for partial differential equations*, 2020. DOI: 10.48550/arXiv.2003.03485. arXiv: 2003.03485 [cs.LG].

[43] Z. Li, N. Kovachki, K. Azizzadenesheli, *et al.*, “Fourier neural operator for parametric partial differential equations,” in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=c8P9NQVtmnO.

[44] Z. Li, N. Kovachki, C. Choy, *et al.*, “Geometry-informed neural operator for large-scale 3D PDEs,” in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 35 836–35 854. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/70518ea42831f02afc3a2828993935ad-Abstract-Conference.html.

[45] Z. Li, H. Zheng, N. Kovachki, *et al.*, “Physics-informed neural operator for learning partial differential equations,” *ACM/IMS Journal of Data Science*, vol. 1, no. 3, pp. 1–27, 2024. DOI: 10.1145/3648506. [Online]. Available: https://doi.org/10.1145/3648506.

[46] K. Lin, X. Li, Y. Ye, *et al.*, “Spherical neural operator network for global weather prediction,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 6, pp. 4899–4913, 2024. DOI: 10.1109/TCSVT.2023.3337857.

[47] R. Lin, T. Song, and J. Li, “Feature attention-based deep neural operator for solving seepage flow equations in porous media reservoir simulation,” *Physics of Fluids*, vol. 37, no. 6, p. 067 152, 2025. DOI: 10.1063/5.0272866.

[48] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators,” *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, 2021. DOI: 10.1038/s42256-021-00302-5.

[49] D. J. Lucia, P. S. Beran, and W. A. Silva, “Reduced-order modeling: New approaches for computational physics,” *Progress in Aerospace Sciences*, vol. 40, no. 1–2, pp. 51–117, 2004. DOI: 10.1016/j.paerosci.2003.12.001.

[50] A. Mehrotra, S. Pavithra, and I. Singh, “Causal-fuzzy neural operators for smart energy forecasting: A multi context learning framework for grid aware load prediction,” *Energy Reports*, vol. 15, p. 108 946, 2026. DOI: 10.1016/j.egyr.2025.108946.

[51] C. A. de Moura and C. S. Kubrusly, Eds., *The Courant–Friedrichs–Lewy (CFL) Condition: 80 Years After Its Discovery*. Birkhäuser, 2013. DOI: 10.1007/978-0-8176-8394-8.

[52] C. Nyshadham, M. Rupp, B. Bekker, *et al.*, "Machine-learned multi-system surrogate models for materials prediction," *npj Computational Materials*, vol. 5, no. 1, p. 51, 2019. DOI: 10.1038/s41524-019-0189-9.

[53] Oden Institute for Computational Engineering and Sciences, *An overview: Scientific machine learning*, https://oden.utexas.edu/research/crosscutting-research-areas/scientific-machine-learning/, Accessed: 2026-04-27.

[54] J.-X. Pan and K.-T. Fang, "Maximum likelihood estimation," in *Growth Curve Models and Statistical Diagnostics*, Springer, 2002, pp. 77–158. DOI: 10.1007/978-0-387-21812-0_3.

[55] M. Pang, "Efficient river hydrodynamics modelling in realistic river systems using a fourier neural operator-based network," *Journal of Hydrology*, vol. 637, p. 131 345, 2024. DOI: 10.1016/j.jhydrol.2024.131345.

[56] Z. Peng, B. Yang, Y. Xu, F. Wang, L. Liu, and Y. Zhang, "Rapid surrogate modeling of electromagnetic data in frequency domain using neural operator," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–12, 2022. DOI: 10.1109/TGRS.2022.3222507.

[57] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009. [Online]. Available: https://www.wseas.us/e-library/transactions/circuits/2009/29-485.pdf.

[58] K. Raeen, "A study of the gibbs phenomenon in fourier series and wavelets," Master's thesis, The University of New Mexico, 2008. [Online]. Available: https://math.unm.edu/~crisp/students/kouroshMSthesis.pdf.

[59] M. A. Rahman, Z. E. Ross, and K. Azizzadenesheli, *U-NO: U-shaped neural operators*, 2022. DOI: 10.48550/arXiv.2204.11127. arXiv: 2204.11127 [cs.LG].

[60] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[61] M. Schreiner, O. Winther, and S. Olsson, "Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 36 449–36 462. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/7274ed909a312d4d869cc328ad1c5f04-Abstract-Conference.html.

[62] A. M. Schweidtmann, D. Zhang, and M. von Stosch, "A review and perspective on hybrid modeling methodologies," *Digital Chemical Engineering*, vol. 10, p. 100 136, 2024. DOI: 10.1016/j.dche.2023.100136.

[63] N. Shi, D. Li, M. Hong, and R. Sun, "RMSProp converges with proper hyper-parameter," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=3UDSdyIcBDA.

[64] A. Singh, A. Rawat, and N. Raghuthaman, "Mexican hat wavelet transform and its applications," in *Methods of Mathematical Modelling and Computation for Complex Systems*, Springer, 2021, pp. 299–317. DOI: 10.1007/978-3-030-77169-0_12.

[65] A. Sojitra, M. Dhingra, and O. San, "FEDONet: Fourier-embedded DeepONet for spectrally accurate operator learning," *Journal of Computational Physics*, vol. 559, p. 114 931, 2026. DOI: 10.1016/j.jcp.2026.114931.

[66] M. R. Spiegel, *Laplace Transforms*. New York: McGraw-Hill, 1965.

[67] R. S. Stanković and B. J. Falkowski, "The haar wavelet transform: Its status and achievements," *Computers & Electrical Engineering*, vol. 29, no. 1, pp. 25–44, 2003. DOI: 10.1016/S0045-7906(01)00011-8.

[68] J. Sun, S. Dong, and F. Wang, “Local randomized neural networks with discontinuous galerkin methods for partial differential equations,” *Journal of Computational and Applied Mathematics*, vol. 445, p. 115 830, 2024. DOI: 10.1016/j.cam.2024.115830.

[69] D. Sundararajan, *The Discrete Fourier Transform: Theory, Algorithms and Applications*. Singapore: World Scientific, 2001, ISBN: 9789810249734.

[70] J. Thiyagalingam, M. Shankar, G. Fox, and T. Hey, “Scientific machine learning benchmarks,” *Nature Reviews Physics*, vol. 4, no. 6, pp. 413–420, 2022. DOI: 10.1038/s42254-022-00441-7.

[71] K. Tiwari, N. M. A. Krishnan, and P. A. P., “CoNO: Complex neural operator for continous dynamical physical systems,” *APL Machine Learning*, vol. 3, no. 2, p. 026 101, 2025. DOI: 10.1063/5.0254013.

[72] E. F. Toro, “Conservative methods,” in *Computational Bodily Fluid Dynamics: Models, Algorithms and Applications*, Cham: Springer, 2025, pp. 449–505. DOI: 10.1007/978-3-031-92598-6_13.

[73] T. Tripura and S. Chakraborty, “Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 404, p. 115 783, 2023. DOI: 10.1016/j.cma.2022.115783.

[74] C. Vonesch, T. Blu, and M. Unser, “Generalized daubechies wavelet families,” *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4415–4429, 2007. DOI: 10.1109/TSP.2007.896255.

[75] Y. Wang, C. Xiong, C. Ju, G. Yang, Y.-W. Chen, and X. Yu, “A deep transfer operator learning method for temperature field reconstruction in a lithium-ion battery pack,” *IEEE Transactions on Industrial Informatics*, vol. 20, no. 6, pp. 8089–8101, 2024. DOI: 10.1109/TII.2024.3369708.

[76] F. Weisz, “The inverse of the continuous wavelet transform,” in *International Conference on Computer Aided Systems Theory*, ser. Lecture Notes in Computer Science, vol. 10672, Springer, 2017, pp. 246–253. DOI: 10.1007/978-3-319-74727-9_29.

[77] H. You, Q. Zhang, C. J. Ross, C.-H. Lee, M.-C. Hsu, and Y. Yu, “A physics-guided neural operator learning approach to model biological tissues from digital image correlation measurements,” *Journal of Biomechanical Engineering*, vol. 144, no. 12, p. 121 012, 2022. DOI: 10.1115/1.4055918.

[78] A. I. Zayed, *Handbook of Function and Generalized Function Transformations*. Boca Raton, FL: CRC Press, 1996, ISBN: 9780849386643.

A

A Low-Frequency Approximation Test Using DFT and DWT

In this chapter, we study a case to show that for certain specific functions, the approximation accuracy of DWT is significantly higher than that of DFT. This phenomenon provides a basis for using DWT in spectral operator learning. Therefore, we construct a specific set of functions for approximation experiments with DWT and DFT.

A.1. Main Idea

We consider a fixed low-frequency K-term truncation rule for both Fourier and wavelet representations instead of the best K-term approximation. We restrict

$$K = 2^m, \qquad m \in \mathbb{N},$$

and define:

- DFT approximation: keep the K lowest Fourier frequencies (including the zero frequency and symmetric high-end indices).
- DWT approximation (Haar): keep the first K wavelet coefficients in multiresolution order, i.e.,

$$[cA_J,\ cD_J,\ cD_{J-1}, \dots].$$

We recall the Haar multiresolution analysis on $[0, 1]$ with periodic boundary conditions.

For each $m \geq 0$, define the dyadic partition

$$I_{\ell,m} = \Big[\frac{\ell}{2^m},\ \frac{\ell+1}{2^m}\Big), \qquad \ell = 0, \dots, 2^m - 1.$$

The Haar approximation space is

$$V_m := \left\{ f :\ f(x) = \sum_{\ell=0}^{2^m-1} c_\ell\, \mathbf{1}_{I_{\ell,m}}(x) \right\}.$$

Thus V_m consists of functions that are piecewise constant on the 2^m dyadic intervals. These spaces are nested:

$$V_0 \subset V_1 \subset \cdots \subset V_m \subset \cdots$$

The Haar detail space is defined by the orthogonal difference

$$W_m := V_{m+1} \ominus V_m.$$

The multiresolution structure satisfies

$$V_{m+1} = V_m \oplus W_m, \qquad \dim(V_m) = 2^m, \qquad \dim(W_m) = 2^m.$$

In the Haar setting, W_m captures the local oscillatory corrections between resolutions m and $m+1$.

A key structural fact is the following.

Let $f \in V_{m^*}$. Then the Haar DWT truncated at

$$K = 2^{m^*}$$

reconstructs f exactly (up to machine precision).

The space V_{m^*} is spanned by the first 2^{m^*} Haar scaling coefficients in the multiresolution ordering. Therefore the truncated inverse DWT coincides with the orthogonal projection onto V_{m^*}, which leaves f unchanged.

This observation motivates constructing test signals aligned with the dyadic multiresolution structure.

We define the dyadic signal class $\mathcal{D}$ for approximation.

Fix integers

$$m_{\min} \le m_{\max}, \qquad \beta > 0.$$

We define the dyadic signal class

$$\mathcal{D} = \left\{ f : \ f(x) = \sum_{j=m_{\min}}^{m_{\max}} \epsilon_j f_j(x), \quad f_j \in V_j \right\},$$

where the scale weights satisfy

$$\epsilon_j = 2^{-\beta(j-m_{\min})}.$$

This class represents signals with multiscale dyadic structure and coarse-scale energy dominance.

Let

$$K = 2^m, \qquad m_{\min} \le m \le m_{\max}.$$

Denote by T_K^{W} the Haar low-frequency truncation operator. By linearity and the multiresolution nesting,

$$f - T_K^{\mathrm{W}} f = \sum_{j>m} \epsilon_j \big(f_j - P_{V_m} f_j\big),$$

where P_{V_m} is the orthogonal projector onto V_m.

Since $f_j \in V_j$ with $j > m$ contains only finer-scale variations, its projection onto V_m removes most of its energy. Consequently, the dominant contribution to the reconstruction error comes from the discarded fine-scale terms, leading to the heuristic bound

$$\|f - T_K^{\mathrm{W}} f\| \ \lesssim \ \sum_{j>m} \epsilon_j \|f_j\|.$$

Because ϵ_j decays exponentially in j, the Haar truncation error decreases rapidly as K increases.

This mechanism underlies the systematic advantage of the Haar DWT on the dyadic signal class $\mathcal{D}$.

A.2. Problem Setup

We construct a fixed test function from the dyadic signal class introduced in the previous section. Throughout the experiment we work on the periodic domain $[0, 1]$ discretized by a uniform grid of size

$$M = 1024 = 2^{10}.$$

We consider dyadic approximation spaces V_m and build the signal as a finite multiscale superposition

$$f(x) = \sum_{j=2}^{8} \epsilon_j f_j(x), \qquad f_j \in V_j,$$

where the scale weights are given by

$$\epsilon_j = 2^{-\beta(j-2)}, \qquad \beta > 0.$$

Each component f_j is specified by prescribing its piecewise-constant values on the 2^j dyadic subintervals. Concretely, we fix the coefficient arrays $\{c_\ell^{(j)}\}_{\ell=0}^{2^j-1}$ as follows:

- $j = 2$ (length 4):
$$(0.0,\ 1.0,\ 0.0,\ -0.5).$$
- $j = 3$ (length 8):
$$(0.0,\ 0.0,\ 0.6,\ 0.0,\ 0.0,\ -0.4,\ 0.0,\ 0.0).$$
- $j = 4$ (length 16): sparse nonzero entries at selected dyadic blocks with amplitudes 0.8, -0.6, and 0.5.
- $j = 5$ (length 32): sparse nonzero entries with amplitudes 1.0, -0.9, 0.7, and -0.6.
- $j = 6$ (length 64): sparse multiscale perturbations with amplitudes up to 1.2.
- $j = 7$ (length 128): further localized oscillatory components.
- $j = 8$ (length 256): fine-scale sparse perturbations.

The resulting function belongs to V_8 and therefore admits an exact representation in the Haar multiresolution framework when sufficient coefficients are retained. A visualization of the function can be shown in Figure A.1.

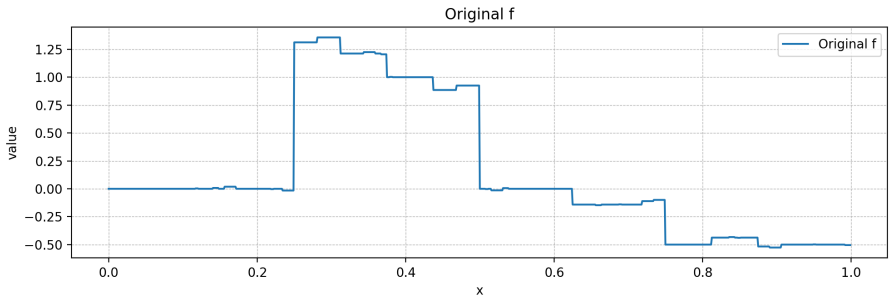


Figure A.1: The constructed function on the dyadic grid for low frequency approximation

We compare two fixed-coefficient truncation strategies under an equal budget K, where

$$K \in \{4, 8, 16, 32, 64, 128, 256\}, \qquad K = 2^m.$$

Let $\tilde{f}_K^{\mathrm{F}}$ and $\tilde{f}_K^{\mathrm{W}}$ denote the reconstructions obtained by the DFT and DWT truncations, respectively. We evaluate the reconstruction quality using

- the discrete ℓ^2 error

$$\|f - \tilde{f}\|_2,$$

- the discrete ℓ^∞ error

$$\|f - \tilde{f}\|_\infty.$$

These metrics are computed on the $M = 1024$ grid for each value of K.

This setup isolates the effect of the representation structure under a fixed low-frequency coefficient budget and highlights the behavior of Fourier and Haar approximations on multiscale dyadic signals.

A.3. Experiment Result

The approximation results for DFT and DWT with different K are shown in Figure A.2 and in Table A.1.

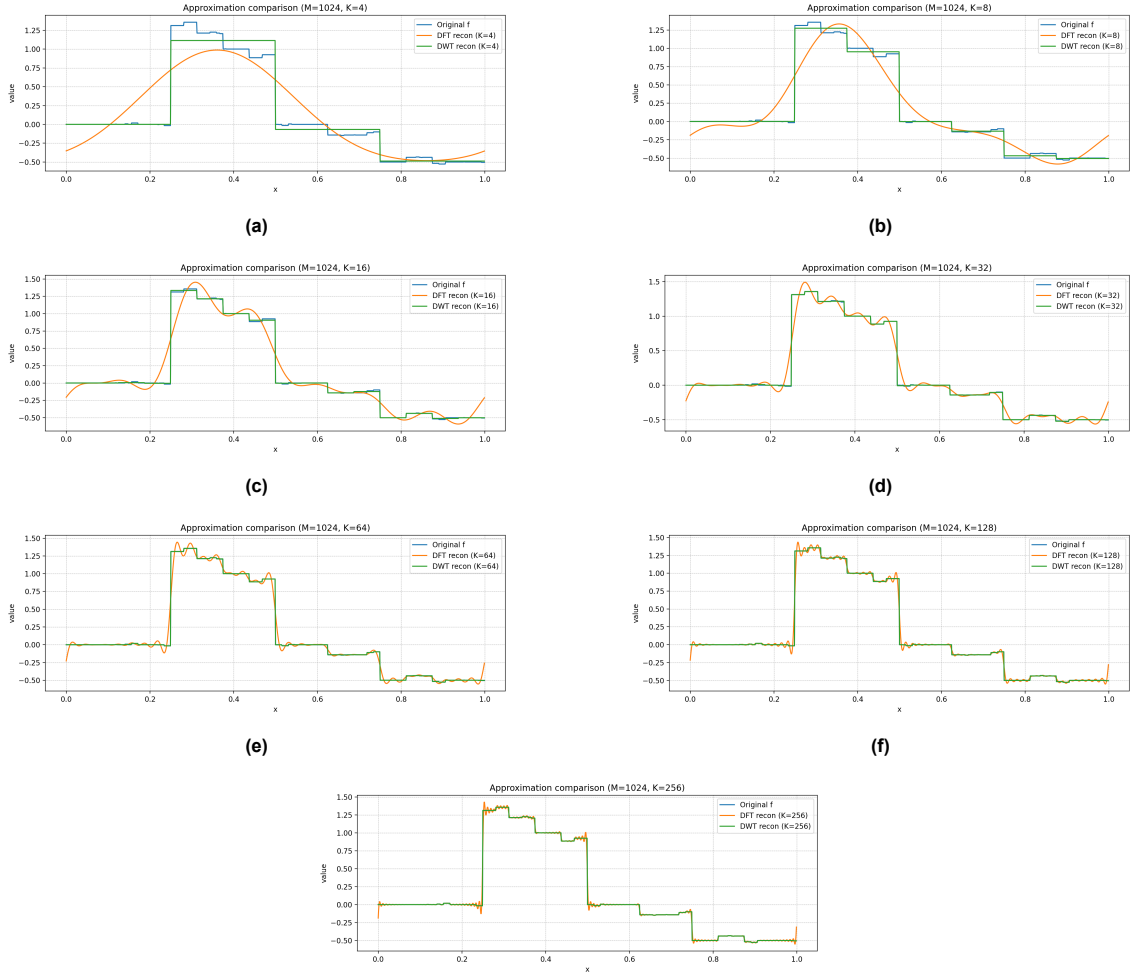


Figure A.2: Low frequency approximation with different K

Table A.1: Comparison of DFT and Haar DWT under fixed K-coefficient truncation.

	DFT		Haar DWT	
K	L_2	L_∞	L_2	L_∞
4	8.526145e+00	7.358144e-01	2.967376e+00	2.428613e-01
8	6.287922e+00	6.819151e-01	9.970870e-01	8.203198e-02
16	4.524185e+00	6.703653e-01	3.160176e-01	2.354902e-02
32	3.190831e+00	6.746697e-01	1.055379e-01	1.163616e-02
64	2.254425e+00	6.355666e-01	3.179997e-02	3.866990e-03
128	1.586208e+00	5.858667e-01	8.903337e-03	1.562500e-03
256	1.099638e+00	5.002827e-01	2.796015e-14	2.220446e-15

A.4. Conclusion

We investigated fixed low-frequency truncation of the DFT and the Haar DWT on a class of multiscale dyadic signals constructed from the spaces V_m. Under an identical coefficient budget $K = 2^m$, the Haar wavelet approximation consistently achieves significantly smaller reconstruction errors than the Fourier low-frequency truncation in both ℓ^2 and ℓ^∞ norms.

This performance gap is not incidental. The constructed signals belong to a multiresolution hierarchy aligned with the Haar scaling spaces, so the DWT truncation effectively performs a near-optimal projection onto the underlying model class. In contrast, the Fourier low-frequency subspace is globally smooth and poorly matched to the localized, piecewise-constant structure of the test signals.

More broadly, the experiment demonstrates that under a fixed low-frequency coefficient budget, approximation quality is strongly governed by the compatibility between the signal model and the representation basis. For multiscale dyadic signals with localized discontinuities, Haar wavelets provide a markedly more efficient representation than global Fourier modes.

The observed advantage arises without best-K selection, but purely from the intrinsic ordering of the Haar multiresolution coefficients.

B

An Empirical Study on the Effect of Network Width in Operator Learning

The purpose of this chapter is not to establish a general theoretical claim, but rather to provide a practical reminder. Even in simple, analytically known function-to-function mappings, the performance of neural networks can be sensitive to the choice of width. Therefore, careful empirical validation is necessary when selecting network architectures for operator learning tasks.

B.1. Problem Setup

We consider a supervised operator learning problem, where the goal is to learn a nonlinear mapping from functions to functions using finite-dimensional neural networks.

Let $x \in [-1, 1]$. The input function $u(x)$ is defined as

$$u(x) = a\sin(5x) + b\sin(7x),$$

where the coefficients a, b are independently sampled from the uniform distribution $\mathcal{U}([-3, 3])$.

The target operator $\mathcal{F}$ maps the input function u to the output function v via a pointwise nonlinear transformation:

$$\mathcal{F}(u)(x) = 1 - u(x)^2.$$

This operator is analytically known, deterministic, and noise-free.

Both the input and output functions are discretized on a uniform grid of N points:

$$x_j = -1 + \frac{2(j-1)}{N-1}, \quad j = 1, \ldots, N.$$

Accordingly, each function is represented as a vector in $\mathbb{R}^N$.

We use fully connected feedforward neural networks to approximate the discretized operator mapping $\mathbb{R}^N \to \mathbb{R}^N$. For each width N, the network consists of three linear layers:

$$\mathbb{R}^N \xrightarrow{\text{Linear}+\sigma} \mathbb{R}^N \xrightarrow{\text{Linear}+\sigma} \mathbb{R}^N \xrightarrow{\text{Linear}} \mathbb{R}^N,$$

where σ denotes a pointwise activation function. Unless otherwise stated, ReLU is used as the activation function, and alternative smooth activations (e.g., SiLU) are employed as controls.

The network parameters are optimized using stochastic gradient-based methods to minimize the mean squared error (MSE) loss:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(u_i, v_i) \in \mathcal{D}} \|\mathcal{N}_\theta(u_i) - v_i\|_2^2 .$$

The dataset is randomly split into training and test sets with a ratio of $9{:}1$. For each network width N, the training process is repeated with multiple random initializations in order to assess the variability induced by nonconvex optimization.

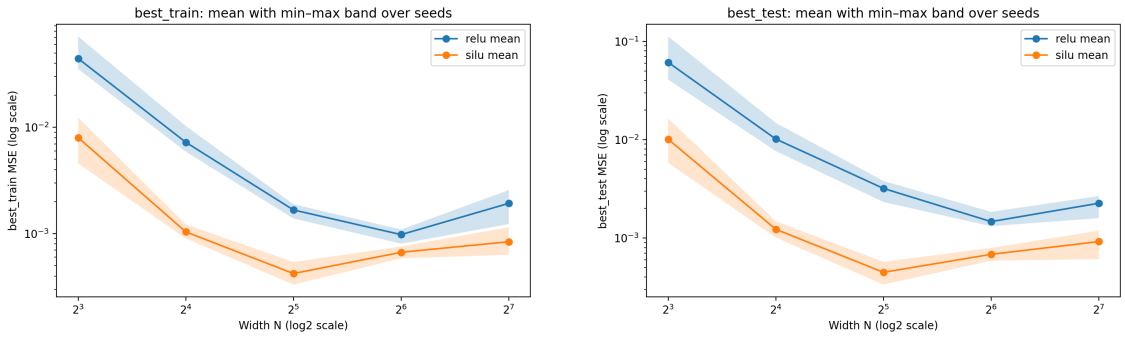
For each run, we record:

- the training error,
- the test error,
- and the best (minimum) training and test errors achieved during optimization.

For each N, we report the mean, minimum, and maximum errors across different random initializations.

B.2. Experiment Result

The experiment is repeated multiple times, and the loss variation curves during the training and testing phases of the neural operator are recorded in the form of min-max bands. The shape of the curves is shown in Figure B.1.



(a) Train loss curve

(b) Test loss curve

Figure B.1: Train and test loss curves

B.3. Main Observation

Despite the simplicity and analytic availability of the target operator, our numerical experiments reveal a clear non-monotonic dependence of both training and test errors on the network width N.

In particular, as the width increases from small to moderate values, the approximation error decreases, indicating improved expressive capacity. However, beyond a certain regime, further increasing the network width does not lead to continued error reduction. Instead, both the training and test errors may increase, and the variability across different random initializations becomes more pronounced.

These results provide an empirical counterexample to the commonly held intuition that larger neural networks necessarily achieve smaller training and test errors. In the context of function-to-function mappings, our experiments demonstrate that increasing network width alone does not guarantee improved approximation quality under fixed training strategies and finite computational budgets.

This phenomenon highlights the interplay between expressivity and optimization difficulty in operator learning, and underscores the fact that optimization-related errors can dominate the performance of finite-width neural networks, even when the target operator is analytically known.

C

Code Availability

The Python source code used in this thesis, including all Jupyter notebooks for data preprocessing, model training, and analysis, is available at:

`https://gitlab.ewi.tudelft.nl/scala/student-projects/master-thesis-projects/hangyu-xia`