School of Computation, Information and Technology - Informatics

Technische Universität München

Master's Thesis in Informatics

Scalable Sampling of Deep Neural Operators

Onur Eker

School of Computation, Information and Technology - Informatics

Technische Universität München

Master's Thesis in Informatics

Scalable Sampling of Deep Neural Operators

Skalierbares Abtasten von tiefen Operatornetzwerken

Author:	Onur Eker
Examiner:	Dr. Felix Dietrich
Advisor:	Iryna Burak, M.Sc.
Submission Date:	January 15th, 2024

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

January 15th, 2024 Onur Eker

Acknowledgments

I extend my gratitude to my supervisor, Dr. rer. nat. Felix Dietrich, and my advisor, Iryna Burak, for their invaluable guidance and support during the creation of this thesis.

Abstract

Maps between functions are important in many areas of science, serving as a fundamental tool for understanding complex relationships in data. Fourier Neural Operators (FNO) are a popular recent machine learning technique used to encode these maps, offering a powerful and efficient approach for learning and generalizing across diverse functional spaces in a way traditional neural networks cannot. This thesis delves into the scalability of two-dimensional Fourier Neural Operators (FNO2D) within the framework of Sampling Where It Matters (SWIM). It primarily focuses on enhancing the FNO2D model's efficiency and accuracy by comparing it with FNO1D. The research systematically explores various hyperparameters, including the number of modes, hidden channels, and layer width, to optimize the performance of FNO2D models, particularly in the context of complex partial differential equations. A series of experiments are conducted to assess the impact of these hyperparameters on model scalability and performance. The study demonstrates that while FNO2D models show great promise in handling multi-dimensional data, strategic hyperparameter tuning is crucial for balancing computational efficiency with predictive accuracy. This thesis contributes insights into the scalability and applicability of FNO2D models with its experiments results.

Contents

1. Introduction

Neural networks, inspired by the intricate workings of the human brain, have emerged as a powerful and versatile class of machine learning models. Over the years, these artificial neural networks have proven their prowess across a myriad of domains, revolutionizing fields such as computer vision, natural language processing, robotics, and many others. The foundation of neural networks lies in their ability to learn complex patterns and representations from vast amounts of data, enabling them to tackle challenging tasks that were once considered beyond the realm of computational capabilities. Through interconnected layers of neurons, these networks can capture hierarchical features, leading to remarkable advancements in areas like image recognition, speech synthesis, and even decision-making systems [17].

Classical optimization techniques such as Gradient Descent and Stochastic Gradient Descent face challenges in scalability when dealing with extensive datasets. Recent research indicates that addressing this concern could involve utilizing sampling methods for weights and biases. An illustrative instance is the Sampling Where It Matters (SWIM) [3] algorithm, which suggests sampling weights and biases using data points from the dataset. The upcoming chapters of this thesis will delve into the examination and exploration of this study and algorithm.

The application of neural networks to partial differential equations (PDEs) has emerged as a promising avenue that bridges the fields of physics, engineering, and machine learning. PDEs play a fundamental role in describing a wide range of phenomena, including heat conduction, fluid dynamics, electromagnetism, and quantum mechanics. Traditionally, solving PDEs involves analytical or numerical methods, which can be computationally intensive and challenging for complex systems. However, by harnessing the expressive power of neural networks, researchers have devised innovative techniques that approximate solutions to PDEs with remarkable accuracy and efficiency. PDEs are employed as a means of application in conducting experiments and evaluations within this thesis [1].

The Fourier Neural Operator (FNO) represents an advancement at the intersection of deep learning and scientific computing, offering a popular machine learning approach to solving partial differential equations [10]. FNO harnesses the expressive power of neural networks alongside the inherent efficiency of Fourier analysis to tackle complex PDEs with high accuracy and speed. By incorporating ideas from both fields, FNO leverages

the Fourier transform to learn an efficient spectral representation of the underlying PDEs, significantly reducing computational costs compared to traditional numerical methods. This methodology enables FNO to capture intricate spatial and temporal patterns in the data, leading to highly accurate and generalizable solutions for a wide range of physical systems. In this thesis, a comprehensive exploration is undertaken to analyze the utilization of sampling methods within FNO networks, as opposed to classical optimizers.

The subsequent structure of the thesis is outlined as follows: Section 2 presents an investigation into existing literature concerning deep neural operators, the application of PDEs to neural networks, and the utilization of weight sampling methods. In Section 3, a comprehensive analysis of the weight sampling technique and scalable sampling for FNOs is presented. Finally, Section 4 draws the thesis to a close by discussing the findings and their potential ramifications.

2. State of the Art

In this chapter, we will dive into the main ideas and methods that form the backbone of this thesis. The section start by looking closely at Partial Differential Equations (PDEs) and especially Burgers' Equation which is testbed for this thesis. Then, we will shift our focus to neural operators and what they involve. Finally, we will dig into how deep neural networks use sampling weights and biases. This exploration will give us the background we need to really grasp the results of the work presented in this thesis.

2.1. Partial Differential Equation

In the realm of mathematics and science, Partial Differential Equations (PDEs) stand as powerful tools for modeling and understanding a wide array of complex physical phenomena. PDEs are mathematical equations that involve multiple independent variables and their respective partial derivatives, making them indispensable in describing how physical quantities evolve and interact in continuous systems. From fluid dynamics and heat transfer to quantum mechanics and electromagnetism, PDEs serve as a unifying language that bridges the gap between theory and reality, enabling us to explore and analyze the intricate dynamics of the natural world. In this section, we will focus on the fundamental principles of Partial Differential Equations (PDEs), with a special emphasis on Burgers' Equation, a notable class of PDEs, examining its properties and applications.

A general form of a partial differential equation is given by [9]:

$$F(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_m}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots) = 0,$$

where u is the unknown function, $x_1, x_2, \ldots, x_n$ are the independent variables, and $\frac{\partial^k u}{\partial x_{i_1} \partial x_{i_2} \ldots \partial x_{i_k}}$ denotes the partial derivative of u with respect to the specified variables $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$.

The function F represents a mathematical expression involving u and its partial derivatives, which encapsulates the physical or mathematical behavior being described. The specific form of F depends on the context and the nature of the problem being addressed. PDEs are often classified based on their order, linearity, and whether they are elliptic, parabolic, or hyperbolic. The order of a PDE is determined by the highest order of the partial derivatives involved. For instance, the heat equation and wave equation are

examples of second-order PDEs.

Among the various classes of PDEs, Burgers' Equation stands out as a fundamental model in fluid dynamics, nonlinear acoustics, and several other fields. This equation is particularly notable for its use in this thesis, where the dataset employed is based on solutions to Burgers' Equation. Burgers' Equation is expressed in its simplest form as:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{2.1}$$

where u is the velocity field as a function of space x and time t, and ν represents the viscosity coefficient. This equation models the dynamics of a fluid, capturing the interplay between nonlinear advection (represented by $u\frac{\partial u}{\partial x}$) and diffusion or viscosity (represented by $\nu\frac{\partial^2 u}{\partial x^2}$). The nonlinearity of the advection term makes Burgers' Equation a quintessential example for studying shock waves and turbulence phenomena, while the diffusion term accounts for the dissipative effects in the fluid [4].

Burgers' Equation is particularly valuable in areas where understanding the balance between nonlinear advection and diffusion is crucial. For example, in meteorology and oceanography, it helps in modeling the movement of atmospheric fronts and ocean currents. In traffic flow analysis, a modified form of Burgers' Equation can describe the flow and density of vehicles, providing insights into traffic patterns and potential congestions [16].

In the context of this thesis, Burgers' Equation serves as a testbed for demonstrating the capabilities of Fourier Neural Operators (FNOs). The complex interplay of nonlinear and dissipative effects in the Burger's Equation provides a challenging yet insightful scenario for FNOs to learn and predict the evolution of fluid dynamics. By using a dataset derived from Burgers' Equation, the thesis aims to showcase the potential of FNOs in capturing and predicting the behavior of such complex systems governed by PDEs [10]. The ability of FNOs to efficiently learn from this data and generalize to new scenarios offers a promising avenue for advancing the application of neural networks in solving and understanding the intricacies of PDE-based phenomena.

2.2. Neural Operators

Traditional advancements in neural network research have predominantly concentrated on exploring relationships within bounded Euclidean spaces or finite sets. The seminal work by Kovachki et al. (2021) marked a paradigm shift with the introduction of "neural operators," a class of neural networks architected to learn mappings between infinite-dimensional function spaces. This innovative concept amalgamates the principles of linear integral operators with the transformative power of non-linear activation functions, forging a pathway for neural networks to comprehend and model complex functional mappings [8].

A cornerstone of this novel approach is the establishment of a universal approximation theorem for neural operators. This theorem rigorously demonstrates the neural operator's capacity to approximate any non-linear continuous operator, reflecting its potential to model a diverse range of phenomena with high fidelity. One of the profound implications of neural operators is their invariance to discretization. Unlike traditional neural network models that are intricately tied to the discretization of the input space, neural operators transcend this limitation. They maintain the ability to operate across varying discretizations of the function spaces, implying that a single set of model parameters is applicable across different resolutions of input data.

Building upon the foundational concept of neural operators, this section delves into three specific architectures that exemplify the application and versatility of neural operator networks:

1. Deep Operator Network (DeepONet) [12].
2. Fourier Neural Operator (FNO) [10].
3. Physics Informed Neural Network (PINN) & Physics Informed Neural Operator (PINO) [13], [14], [11].

The exploration of these neural operator networks in the subsequent subsections will provide an in-depth analysis of their structures, theoretical underpinnings, practical applications, and the challenges they address in the field of machine learning.

2.2.1. DeepONet

Deep Operator Networks (DeepONets) represent a significant stride in machine learning, especially in learning operators between infinite-dimensional function spaces. Introduced by [12], DeepONets are designed to surpass the limitations of traditional neural networks in approximating complex functional mappings. They offer a more generalizable and efficient approach to a range of problems.

The architecture of DeepONets is composed of two sub-networks: a branch net and a trunk net. The branch net learns representations of input functions, while the trunk net focuses on points where the output function is evaluated. This bifurcation allows DeepONets to effectively learn the operator by approximating the mapping from a set of function samples to corresponding output function values.

In practical applications, DeepONets have demonstrated remarkable capabilities, particularly in solving differential equations, learning dynamical systems, and modeling physical phenomena. They have been successfully applied to fluid dynamics problems, exemplifying their proficiency in handling complex, high-dimensional data [7].

DeepONets differ from traditional neural networks in their ability to handle irregular or unstructured data without requiring a grid-like data structure. This flexibility positions them as a more versatile solution compared to conventional methods, which often struggle with such data formats [12].

Future research in DeepONets aims at enhancing their efficiency, robustness, and capacity to generalize from limited data. Integrating DeepONets with other machine learning approaches for tackling more complex, real-world problems is an area of growing interest [12].

In conclusion, DeepONets mark a pivotal development in the field of neural operators. Their unique architecture and promising results across various applications underscore their significance in the advancement of machine learning techniques.

2.2.2. Fourier Neural Operator (FNO)

Fourier Neural Operators (FNOs) are at the forefront of this thesis due to their novel machine learning approach in learning continuous mappings in function spaces, especially for solving partial differential equations (PDEs). Introduced by [10], FNOs are distinct in their use of the Fourier transform to learn mappings in the spectral domain. This approach allows for the efficient learning of complex, high-dimensional PDEs, a core focus of this thesis.

The working principle of FNOs lies in their unique treatment of function space mappings. FNOs first apply a Fourier transform to input functions, converting them into the frequency domain. This transformation facilitates the learning of relationships between different frequencies, allowing the FNO to capture both local and global dependencies in the data. The network then learns a mapping in this spectral space, which is followed by an inverse Fourier transform to convert the output back to the spatial domain. This process enables FNOs to efficiently approximate complex functional mappings, a key advantage over traditional neural network approaches [10] [8].

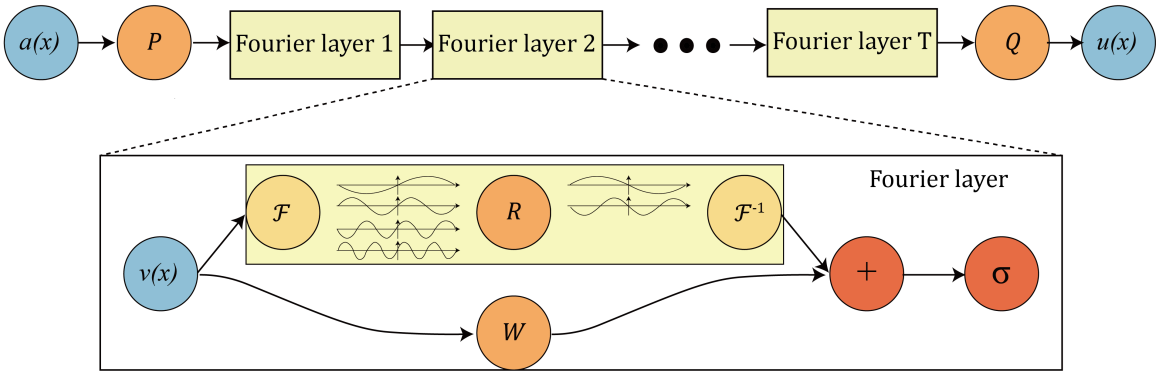

Figure 2.1.: Architecture of Fourier Neural Operator and one single Fourier layer. Taken from Li et al. [10].

A significant challenge in the application of Fourier Neural Operators (FNOs) is their reliance on data produced by numerical solvers. FNOs, designed to learn mappings in function spaces, particularly for solving partial differential equations (PDEs), depend heavily on the quality and quantity of data generated by these solvers. This reliance poses a constraint, as the accuracy and generalizability of FNOs are directly tied to the fidelity of the solver-generated data. Numerical solvers, while effective, may introduce errors or approximations that can propagate through the FNO, potentially impacting its learning efficacy and prediction accuracy. This dependency also raises concerns regarding the FNO's performance in scenarios where high-quality numerical solver data is scarce or computationally expensive to generate. Addressing this challenge is crucial for enhancing the robustness and applicability of FNOs, especially in complex real-world scenarios where solver data may be imperfect or limited [10].

In conclusion, FNOs, which is the primary model of this thesis, represent a significant advancement in neural operators. Their innovative use of the Fourier transform for function space mappings positions them as a powerful tool for solving complex problems in PDEs and various domains in science and engineering.

2.2.3. Physics-Informed Neural Network (PINN) & Physics Informed Neural Operator (PINO)

Physics-Informed Neural Networks (PINNs) represent a notable advancement in integrating deep learning with scientific computing, specifically for solving Partial Differential Equations (PDEs) in accordance with physical laws. Introduced by Raissi et al. [13] [14] [15], PINNs uniquely integrate physical principles into neural network training,

ensuring that solutions are both data-driven and physically accurate.

PINNs represent a paradigm shift in computational science by embedding physical laws into the structure of neural networks. This integration is achieved through a custom-designed loss function, which is the cornerstone of PINNs. The loss function in PINNs, denoted as $\mathcal{L}$, is a combination of two distinct components:

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{physics}, \tag{2.2}$$

where $\mathcal{L}_{data}$ targets data fidelity and $\mathcal{L}_{physics}$ enforces physical consistency.

The data fidelity term, $\mathcal{L}_{data}$, is defined as the mean squared error between the predictions of the neural network and the observed data:

$$\mathcal{L}_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \|u(\mathbf{x}_i, t_i) - u_i^{obs}\|^2, \tag{2.3}$$

where N_{data} represents the number of data points, $u(\mathbf{x}_i, t_i)$ is the prediction of the neural network, and u_i^{obs} is the observed value at the point $\mathbf{x}_i$ and time t_i.

The physics-informed term, $\mathcal{L}_{physics}$, is what distinguishes PINNs from traditional neural networks. It penalizes the network for deviations from the governing PDEs, ensuring that the predictions adhere to the physical laws:

$$\mathcal{L}_{physics} = \frac{1}{N_{col}} \sum_{i=1}^{N_{col}} \|\mathcal{F}(u(\mathbf{x}_i, t_i))\|^2, \tag{2.4}$$

where N_{col} is the number of collocation points used to enforce the physics, and $\mathcal{F}$ is the differential operator representing the PDE.

The architecture of PINNs usually involves a deep fully connected neural network, which is trained to learn the solution of a PDE over a given domain. This architecture allows PINNs to approximate complex functions and capture intricate patterns in the data. The training process involves optimizing the network parameters to minimize both the data fidelity loss and the physics-informed loss, thereby embedding the physics of the problem into the learning process [7].

PINNs have found applications in various scientific domains, including fluid dynamics, heat transfer, materials science, and even in biomedical engineering. Their ability to incorporate physical laws makes them particularly useful for problems where data is sparse, noisy, or expensive to obtain. Moreover, PINNs can be employed in solving inverse problems, where they can identify unknown parameters or governing laws from

observed data [15].

In addition to PINNs, the field of neural operators also encompasses Physics-Informed Neural Operators (PINO), which represent an advancement in the modeling and solution of complex differential equations. PINO combines the principles of physics-informed learning with the flexibility of neural operator frameworks to offer a more generalized and efficient approach to solving PDEs.

Similar to PINNs, PINO employs a specialized loss function that integrates physical laws into the learning process. However, PINO extends this concept by incorporating neural operator architectures, such as Fourier Neural Operators (FNOs), which enable the learning of mappings in function spaces. This integration allows PINO to handle a broader range of problems and achieve higher levels of accuracy and efficiency. [11]

The architecture of PINO typically involves a combination of a neural network, such as a deep fully connected network or a convolutional network, coupled with a neural operator. The neural network component focuses on learning spatial-temporal features, while the neural operator component generalizes the solution across different geometries and boundary conditions. This dual-component architecture enables PINO to learn complex dynamics more effectively than traditional neural networks or standalone physics-informed models.

PINO has shown potential in various applications where traditional methods fall short, particularly in problems with complex geometries, varying boundary conditions, and high-dimensional spaces. Its ability to generalize across different scenarios without the need for retraining makes it a powerful tool in computational physics and engineering.

The research in PINO is ongoing, with efforts directed towards enhancing its computational efficiency, improving its generalization capabilities, and exploring its application in a wider array of scientific problems. The combination of physics-informed principles and neural operator architectures in PINO marks a significant step forward in the field of scientific machine learning, pushing the boundaries of what is possible in the computational understanding of complex systems.

2.3. Sampled Networks

Sampled Networks represent an innovative direction in the field of neural network research, focusing on efficient learning mechanisms and strategic data utilization. This section will explore the core concept of Sampled Networks, emphasizing how they diverge from traditional neural network paradigms in their architecture and learning processes.

Central to the philosophy of Sampled Networks is the strategic sampling of data and network parameters to optimize learning efficiency. This approach is particularly beneficial in scenarios with limited data availability, computational resources, or where rapid model development and deployment are crucial. By employing strategic sampling methods, Sampled Networks aim to achieve a balance between model performance and computational efficiency, a key challenge in machine learning and artificial intelligence [2].

The adaptability and versatility of Sampled Networks enable their application across a variety of fields, including image processing, natural language processing, and predictive modeling. These networks are particularly adept at extracting valuable insights from sparse or unevenly distributed datasets, making them a valuable tool in data-driven disciplines.

Furthermore, the development of Sampled Networks aligns with the growing need for models that can function effectively under constraints like limited labeled data, real-time processing, and dynamic environments. This need makes Sampled Networks a significant asset in the rapidly evolving landscape of machine learning.

This subsequent subsection delves into two different weight sampling method: Extreme Learning Machine (ELM) [6] and Sampling Where It Matters (SWIM) [3].

2.3.1. Extreme Learning Machine (ELM)

Extreme Learning Machines (ELMs) offer an innovative approach in neural network research, characterized by their simplicity and efficiency in training. Developed initially by [6], ELMs are single-layer feedforward neural networks that stand out due to their unique training methodology, which significantly reduces computational time and resources.

In ELMs, the input weights and biases of the hidden layer are randomly assigned and remain fixed. The core of ELM training involves adjusting only the output weights, a process that is significantly faster than the iterative weight adjustments employed in traditional neural networks. This approach not only accelerates the training process but also avoids issues like local minima, which are common in backpropagation algorithms.

Mathematically, an ELM can be expressed through its output function:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}, \tag{2.5}$$

where $\mathbf{H}$ is the hidden layer output matrix for the input data $\mathbf{X}$, $\boldsymbol{\beta}$ represents the output weights, and $\mathbf{Y}$ denotes the target outputs. The hidden layer output matrix is computed as:

$$\mathbf{H} = g(\mathbf{W}\mathbf{X} + \mathbf{b}), \tag{2.6}$$

with g being the activation function, $\mathbf{W}$ the randomly assigned input weights, and $\mathbf{b}$ the biases. The training of ELMs involves finding the optimal output weights $\boldsymbol{\beta}$, typically through a least-squares method.

The random initialization of weights and biases in ELMs introduces an element of diversity in the feature mapping capability of the hidden layer. This diversity often leads to better generalization performance, despite the randomness involved. ELMs have been effectively utilized in various domains, including classification, regression, and clustering, particularly where fast training is crucial.

However, the randomness in ELMs can also lead to variability in model performance. Determining the optimal number of hidden nodes and managing the randomness in weight initialization are among the challenges in ELMs. Research efforts continue to focus on enhancing the stability and scalability of ELMs, making them more adaptable to a wide range of applications [5].

In conclusion, Extreme Learning Machines provide a fast, efficient, and often effective alternative to traditional neural network training methods. Their ability to handle large-scale and complex problems rapidly makes them a valuable tool in the field of machine learning.

2.3.2. Sampling Where It Matters (SWIM)

SWIM [3] introduces a new approach for constructing neural networks by sampling weights and biases based on input and output training data. This method contrasts with traditional iterative gradient-based optimization, offering a more efficient way to train both shallow and deep networks without the need for iterative optimization or gradient computations. The core concept of SWIM is the use of a data-driven probability distribution for determining the weights and biases, which are then informed by the training data.

Mathematically, SWIM defines each pair of weight and bias in all hidden layers based on two points from the input space. For a given layer l, pairs of points $(x_{0i}^{(1)}, x_{0i}^{(2)})$ are sampled,

and the weights w_{li} and biases b_{li} are derived as follows:

$$w_{li} = \frac{s_1(x^{(2)}_{l-1,i} - x^{(1)}_{l-1,i})}{\|x^{(2)}_{l-1,i} - x^{(1)}_{l-1,i}\|^2}, \quad b_{li} = \langle w_{li}, x^{(1)}_{l-1,i} \rangle + s_2, \tag{2.7}$$

where s_1 and s_2 are constants. The last layer of weights and biases W_{L+1}, b_{L+1} is optimized to minimize a suitable loss function, usually L2 loss. This approach effectively links the network parameters directly to the training data, facilitating an efficient network construction process.

The constants s_1 and s_2 are chosen depending on the activation function used. For ReLU activation, $s_1 = 1$ and $s_2 = 0$ are set, allowing linear interpolation between points. In the case of tanh activation, these constants are set to ensure that the points $x^{(1)}$ and $x^{(2)}$ map to $\pm\frac{1}{2}$, respectively, which is particularly useful for classification tasks.

The SWIM algorithm proceeds by randomly selecting pairs of input points and computing a probability distribution based on these points. This probability is proportional to the output difference and input space distance, focusing the network on the most informative aspects of the data. After determining the weights and biases for all hidden layers, the final step involves solving a linear optimization problem for the output layer's coefficients.

SWIM offers several advantages, including its efficiency in constructing networks and its invariance to rigid body transformations and scaling of the input data. This reduces the need for common pre-processing techniques. Moreover, SWIM has been shown to achieve comparable accuracy to iteratively trained networks but can be constructed much faster. However, a key challenge in SWIM is developing effective mechanisms for accurately identifying the most informative data points. Also, it is important to note that the SWIM methodology is not yet suitable for application in convolutional neural networks and transformers, as its sampling approach is primarily designed for fully-connected network architectures.

In conclusion, SWIM represents a significant advancement in neural network training methodologies. Its novel sampling algorithm, underpinned by a robust mathematical framework, aligns with the objectives of this thesis to develop efficient and effective solutions for neural network training.

3. Scalable Sampling of Deep Neural Networks

Fourier Neural Operators (FNO) represent a significant advancement in computational modeling, providing an efficient approach for understanding and learning mappings within function spaces. This method shows promise in addressing problems typically governed by partial differential equations (PDEs). The core advantage of FNOs lies in their utilization of the Fourier transform, which aids in converting functions into a spectral space. This conversion is useful for identifying broad correlations within data, an attribute that is particularly relevant for PDEs occurring in high-dimensional spaces.

When FNOs apply the Fourier transform to input functions, they effectively shift the representation of these functions into the frequency domain. Within this domain, the FNOs are designed to carry out linear operations, harnessing the spectral space's inherent properties to perform computations that might be challenging or less efficient in the original spatial representation. Subsequent to these operations, FNOs utilize inverse Fourier transforms, a process that reconverts the frequency-domain data back into the spatial domain, thus producing the final output functions.

The proficiency of FNOs in efficiently handling complex PDEs is fundamentally attributed to their intrinsic architecture, which is adept at parameterizing the functional mappings. This neural network structure is tailored to operate proficiently across various frequencies, effectively learning the intricate patterns and behaviors dictated by the PDEs. It is this capacity to work across different frequencies, leveraging the nuances of the spectral space, that empowers FNOs to offer a potent solution for modeling and solving high-dimensional PDEs with remarkable efficiency.

Sampling Where It Matters (SWIM), on the other hand, introduces a methodology for sampling weights and biases in neural networks based on input-output training data, avoiding the need for iterative, gradient-based optimization. SWIM's strategy is to select weights and biases that are most informative given the training data, thereby constructing neural networks that are finely tuned to the specifics of the data and the task at hand. SWIM's approach can be particularly beneficial for networks where the input space is high-dimensional and complex, making traditional training methods computationally expensive or infeasible.

When combining FNO and SWIM, the goal is to leverage the strengths of both methods to create a sampling-based approach that is both scalable and effective for learning operators over function spaces. In this integration, SWIM can be utilized to initialize the FNO architecture by selectively sampling weights and biases that are crucial for capturing the underlying dynamics of the problem space, as indicated by the training data. This can lead to a more data-efficient learning process and can potentially improve the generalization of the FNO model.

In practice, this means using SWIM to inform the initial parameters of the FNO's layers (see Figure 3.1). The spectral coefficients that the FNO learns might be initialized or constrained based on the significance of the corresponding frequencies in the training data, as determined by SWIM's sampling algorithm. This hybrid approach aims to reduce the computational burden of training FNOs while enhancing their capacity to learn more accurate representations of the operators governing the data.

The integration of FNO and SWIM thus proposes a novel direction for scalable sampling in deep neural networks. It offers the promise of constructing networks that are not only computationally efficient but also capable of learning complex functional mappings with a high degree of accuracy. This method could be particularly impactful in fields such as computational fluid dynamics, climate modeling, and other areas where PDEs play a critical role, and where data efficiency and scalability are of utmost importance.

The FNO architecture from [3] combines linear operations in the Fourier space with skip connections in the signal space. The process involves lifting an input signal to a higher dimensional channel space, applying a discrete Fourier transform to keep only the lowest frequencies, and then using a spectral convolution followed by a 1x1 convolution with bias. Several Fourier blocks are stacked, and the output signal is projected to the target dimension, with both the projection and lifting operators typically parameterized by neural networks.

[3] discusses the adaptation of the FNO concept within the framework of sampled deep neural operators, where traditional convolution kernels are not used, and instead, a fully-connected network is trained for each channel in Fourier space. This approach involves appending grid coordinates to the input signal, normalizing the input data, applying Fourier transform to both input and target data, and employing skip connections by subtracting the input data from the lifted target function during training, then adding it back before moving to the output of the block.

The results of the experiments show that sampled models, while not directly following the original FNO architecture, are comparable to those trained with gradient-based methods like Adam. Although the sampled models only involve fully-connected layers, they demonstrate the advantage of sampled networks in terms of training speed, with signif-

icant speed-ups observed even when sampling is run on a CPU compared to gradient-based training on a GPU.

3.1. Model

This thesis extends the methodologies developed in the SWIM paper to two-dimensional Fourier Neural Operators (FNO2D). The FNO2D model is specifically tailored to address the complexities associated with multi-dimensional data spaces, which pose unique challenges in terms of spatial data representation and computational efficiency.

The FNO architecture commences with a sophisticated lifting pipeline. This stage is pivotal in transforming the low-dimensional input data into a higher-dimensional channel space. The lifting pipeline is essential in equipping the model with the capability to encode complex patterns and interactions inherent in the input data. This step is particularly crucial when dealing with multi-dimensional spatial data, as it ensures the preservation of intricate spatial structures and relationships, foundational for accurately modeling and predicting complex phenomena.

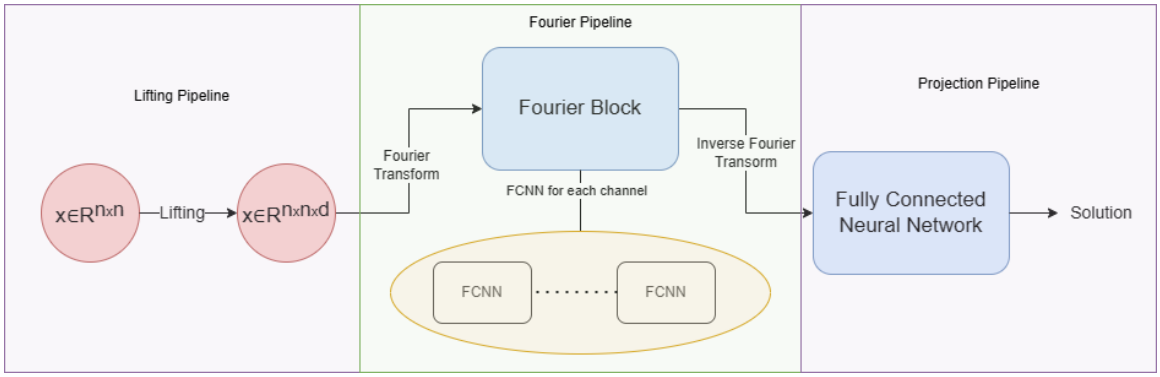

Figure 3.1.: Architecture of FNO2D: Weights and biases of Fully Connected Neural Networks (FCNNs) are sampled using the SWIM algorithm.

The heart of the FNO's functionality is the Fourier pipeline, which applies Fourier transforms to both the input and the target data. This transformation into the spectral domain allows the model to efficiently capture and learn the intrinsic patterns of the data. In this spectral space, the model employs a fully connected network for each channel, ensuring a comprehensive learning process across the spectrum. The inclusion of skip connections refines this process further. By selectively removing and reintegrating the input data from the lifted target function during training, the model enhances its ability to preserve essential information throughout the learning process. The application of inverse

Fourier transforms at the end of this pipeline is a critical step, converting the processed spectral data back into a meaningful spatial representation.

The culmination of the FNO's processing occurs in the projection pipeline. This stage functions as an advanced fully connected network, adept at mapping the transformed signals to a solution space that aligns with the model's intended output. This pipeline synthesizes the model's learned spectral and spatial representations, generating predictive outputs that provide solutions to the underlying complex PDEs.

A key pat in the FNO2D model that this thesis proposes is the integration of the SWIM algorithm for sampling all weights and biases. This integration ensures that the model's parameters are optimally initialized based on insights drawn from the training data. The application of SWIM facilitates a more informed and efficient optimization process, potentially reducing extensive training requirements and enhancing the model's generalization capabilities from limited data sets.

A methodological enhancement in this thesis is the implementation of a meshgrid-based lifting procedure within the context of FNO2D. This approach represents a strategic departure from traditional linear interpolation methods, which often fall short in multi-dimensional domains. The meshgrid function creates a two-dimensional array of coordinates, ensuring the fidelity of spatial relationships in the input data. This nuanced representation is critical for the Fourier and projection pipelines to operate effectively, enabling the FNO2D model to perform spectral operations with increased precision and deliver more accurate predictions for PDEs modeled in two-dimensional spaces.

The enhancements introduced in the FNO2D model – particularly the meshgrid-based lifting procedure and the integration of SWIM for parameter sampling – collectively elevate the model's capability to address the challenges of multi-dimensional data spaces. These improvements not only ensure the integrity of spatial information in the model's input but also enrich the data representation, crucial for the effective functioning of the Fourier and projection pipelines. The implications of these advancements are far-reaching, offering a blueprint for future developments in neural operator models, especially in solving complex scientific problems that involve high-dimensional data.

3.2. Dataset

The empirical evaluation of Fourier Neural Operators in one and two dimensions (FNO1D and FNO2D) is facilitated by datasets generated from the Burgers' equation, a fundamental partial differential equation used to model various physical phenomena, as mentioned in the 2.1. For a thorough comparative analysis, distinct datasets for 1D and 2D scenarios have been curated.

1D Burgers' Dataset: The 1D Burgers' dataset comprises 1000 simulation instances with a spatial resolution of 64 points across the domain $[0, 2\pi]$. These simulations are generated by solving the 1D Burgers' equation with a viscosity (ν) of 0.1, ensuring an accurate representation of diffusion phenomena. The temporal resolution is within the interval $[0, 1]$.

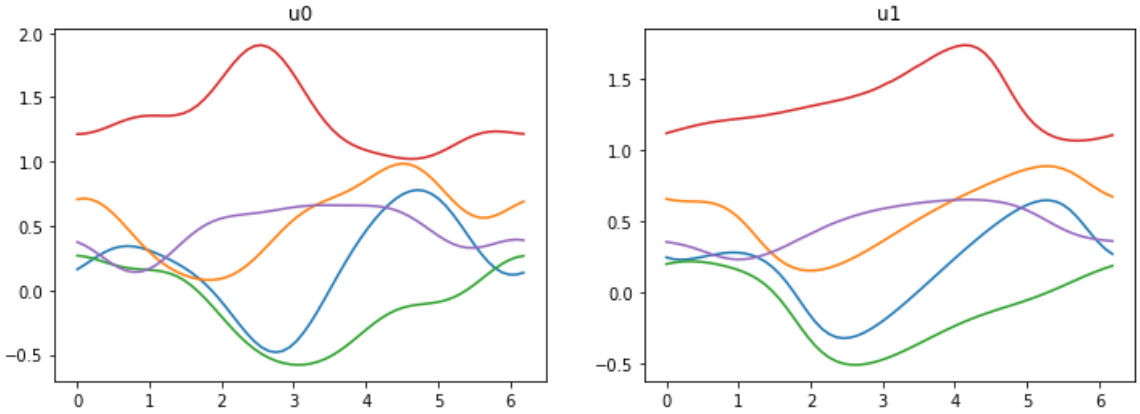

Figure 3.2.: 5 different samples from 1D dataset. Each color represents a distinct sample. u0 represents the sample at t=0 while u1 represents the sample at t=1

2D Burgers' Dataset: In parallel, the 2D Burgers' dataset includes 1000 simulation instances as well, each resolved on a 64×64 grid, covering the spatial domain $[0, 1] \times [0, 1]$. These simulations address the additional complexity introduced in two dimensions and are generated with a reduced viscosity (ν) of 0.01, maintaining consistency with the physical realism of higher-dimensional flows. The temporal resolution mirrors that of the 1D case, with 1000 time steps within the interval $[0, 1]$.

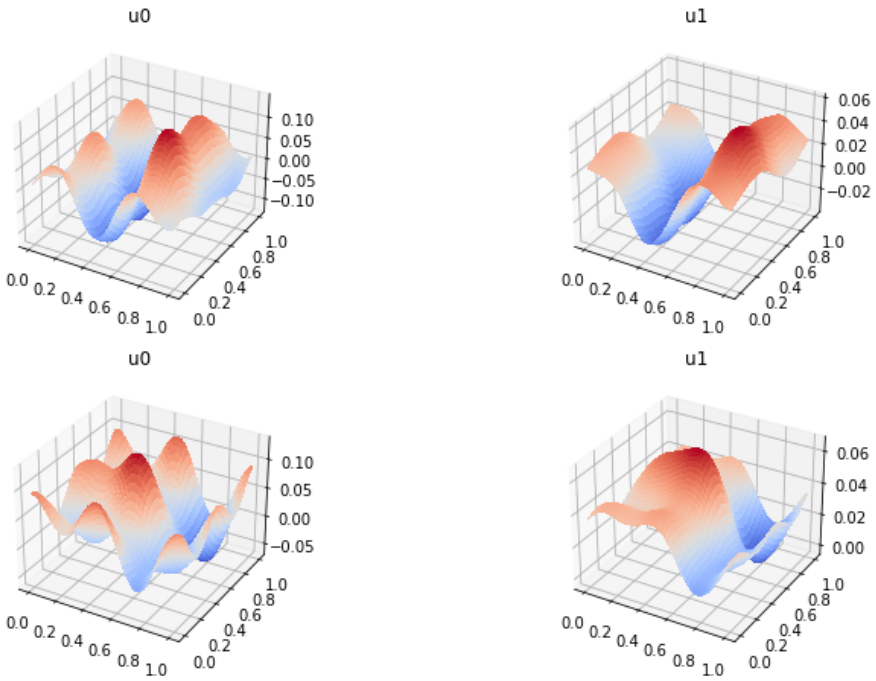

Figure 3.3.: 2 different sample from 2D dataset. u0 represents the sample at t=0 while u1 represents the sample at t=1

Of these 1000 data points, a significant portion, comprising 900 data points, is allocated for the training phase. The remaining 100 data points are reserved exclusively for testing purposes.

3.3. Experiments

3.3.1. Experiment Setup

In the experimental evaluation, the performance of Fourier Neural Operators is fine-tuned through a series of hyperparameter optimizations. Six critical hyperparameters are considered: the number of modes (`n_modes`), which determines the resolution in the spectral domain of the FNO, the number of hidden channels `n_hidden_channels`), the width of each layer (`layer_width`), the number of blocks in the FNO architecture (`n_blocks`), the regularization scale (`regularization_scale`), which is vital for controlling model complexity and avoiding overfitting, and the type of parameter sampler combined with the activation function used (`Parameter Sampler & Activation`). These hyperparameters are selected based on their potential impact on the model's capacity to capture the dynamics of the Burgers' equation in both one and two-dimensional domains.

The following table shows the hyperparameters to be tuned and its set of values. For FNO2D, n_modes is used for both dimension.

Hyperparameter	Description	Values
n_modes	Number of Fourier modes	1, 2, 4, 8, 16, 32
n_hidden_channels	Number of hidden channels	1, 2, 4, 8, 16, 32, 64
layer_width	Width of each layer	2, 4, 8, 16, 32, 64, 128, 256
n_blocks	Number of blocks in FNO	1, 2, 3, 4, 5, 6, 7, 8
regularization_scale	Regularization Scale	$10^{-8}, 10^{-7}, ..., 10^{-2}, 10^{-1}$
activation	Activation function	tanh, ReLU

Table 3.1.: Hyperparameter tuning for FNO models, with a distinction between FNO1D and FNO2D in terms of the number of modes.

Continuing with the experimental setup, it is crucial to address the methodological decisions regarding the model's architecture. One such decision is the alignment of the parameter sampler function with the activation function within the neural network. This alignment is implemented to ensure consistency in the network's processing mechanism. Utilizing the same function for both parameter sampling and activation guarantees that the model's approach to introducing non-linear transformations remains uniform throughout its architecture. This uniformity is particularly important as it helps in maintaining the integrity of the network's learning dynamics, ensuring that the sampling of parameters harmonizes with their subsequent activation during the model's operation. Such a consistent approach is essential for preserving the coherence and predictability of the neural operator's behavior.

Furthermore, the experimental configuration places a constraint on the number of hidden channels, largest value at 64 (n_hidden_channels). This limitation is imposed due to the memory capacity of the computational device used for conducting the experiments. Expanding the number of hidden channels beyond this threshold leads to an exponential increase in the computational memory requirement, surpassing the capabilities of the available hardware.

3.3.2. Method

The experimental methodology in this study is designed to evaluate the performance of Fourier Neural Operator (FNO) models under various hyperparameter settings. The default parameters for these experiments are established as: number of blocks (n_blocks) at 1, number of hidden channels (n_hidden_channels) at 16, number of modes (n_modes at 2, regularization scale at 10^{-8} and layer width (layer_width) also at 16. These serve as a baseline against which the impact of hyperparameter variations is measured.

Hyperparameter Tuning: Each hyperparameter configuration undergoes testing using both tanh and relu activation functions to evaluate the impact of these distinct non-linear transformations on model performance. In this experimentation, the emphasis is on altering one of three key aspects — the number of hidden channels, the width of each layer, or the number of blocks within the FNO architecture — while maintaining the other parameters at their default settings. This focused approach aims to isolate and understand the influence of each individual hyperparameter on the evaluation metrics described below.

Evaluation Metrics: Performance evaluation of each model configuration is based on three key metrics:

1. **Fit Time (Train):** Time required for the model to train on the dataset, indicative of the computational efficiency under different hyperparameter setups.
2. **Transform Time (Predict):** Time taken for the model to transform the training data, reflecting operational efficiency during the predicting phase.
3. **Relative L2 Error:** A measure of model accuracy, calculated as the L2 norm of the difference between predicted and actual values, normalized by the L2 norm of the actual values. It is given by:

$$\text{Relative L2 Error} = \frac{\|y_{\text{pred}} - y_{\text{true}}\|_2}{\|y_{\text{true}}\|_2}, \tag{3.1}$$

   where y_{pred} and y_{true} denote the predicted and true values, respectively.

Experimental Procedure: The models, trained on datasets derived from the Burgers' equation in both 1D and 2D forms, undergo a process of fitting and subsequent performance evaluation on a validation set using the defined metrics. Each hyperparameter configuration is iteratively tested to ensure the reliability and reproducibility of the results.

This methodological framework is designed to offer comprehensive insights into the effects of various hyperparameters on FNO model performance, facilitating an in-depth understanding of optimal configurations for neural operators in complex PDE solutions.

3.3.3. Results

The results section presents the outcomes of the experiments conducted to evaluate the performance of FNO models under different configurations and activation functions. The following table delineates the performance metrics — fit time, transform time, and relative L2 error — for default configuration with both tanh and ReLU activation functions.

Dimension	Activation	Fit Time (s)	Transform Time (s)	Relative L2 Error
1D	tanh	0.40	0.02	0.240
1D	relu	0.47	0.02	0.262
2D	tanh	23.68	1.68	0.418
2D	relu	23.08	1.11	0.391

Table 3.2.: Baseline performance metrics for FNO models with default parameters

Number of Blocks (`n_blocks`):
The experimental analysis conducted on FNO1D and FNO2D models offers insights into the nuanced behavior of neural operators as the number of blocks (`n_blocks`) is varied. Notably, an increase in `n_blocks` does not uniformly lead to a decrease in the relative L2 error 3.4. This outcome suggests that adding complexity to the model, in terms of depth, does not inherently enhance its predictive accuracy. In terms of activation functions, tanh generally exhibits marginally superior performance over ReLU when evaluating the relative L2 error across different configurations. However, an exception is observed in the case of FNO2D with a single block, where ReLU marginally outperforms tanh. This indicates that the effectiveness of activation functions may be context-dependent, varying with the dimensionality and architecture of the model.

Regarding the computational time, a linear increase in fit time is observed as `n_blocks` is augmented, which aligns with the expected growth in computational workload with deeper models 3.5. Conversely, transform times do not exhibit significant changes with increasing `n_blocks`, suggesting that the computational expense during the transform phase remains relatively stable. When comparing the time efficiency of tanh and ReLU, the results show no substantial difference, implying that the choice of activation function does not critically impact the temporal aspect of model training and transformation.

A stark contrast in time consumption of fitting is noted when comparing FNO2D to FNO1D; the former requires approximately 75 times more time than the latter. This considerable disparity underscores the complexity introduced by higher-dimensional data and the associated increase in computational resources required for processing.

In summation, the findings from the experiments elucidate that while model depth, as controlled by `n_blocks`, is a pivotal factor in the architecture of FNOs, its optimization for enhanced performance is not straightforward and warrants a balanced consideration of accuracy and computational efficiency. The comparative slight edge of tanh in accuracy and the substantial time demands of FNO2D models are salient points that must be factored into the design and application of neural operators.

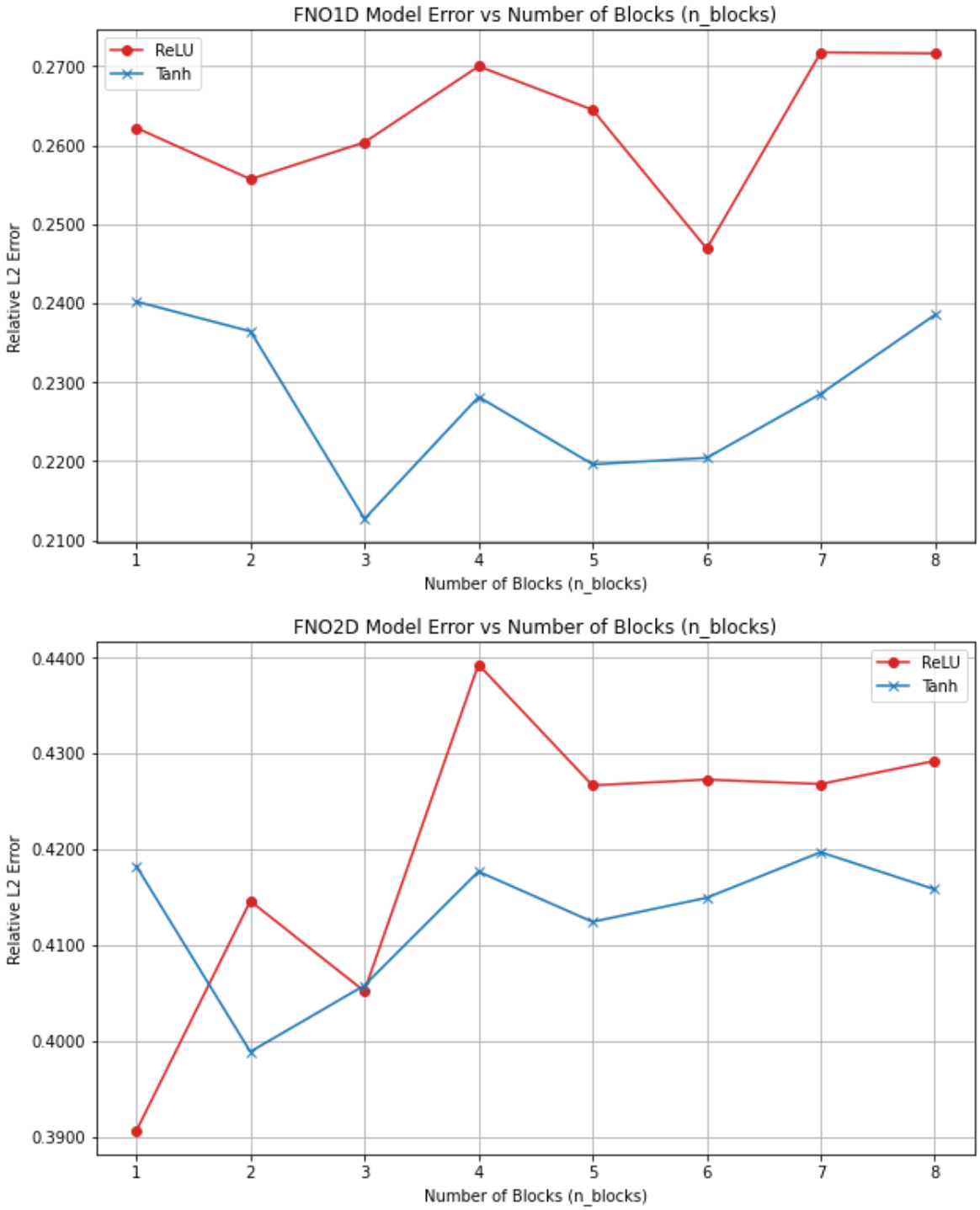

Figure 3.4.: For both FNO2D and FNO1D, figure shows the graphs of n_blocks vs relative L2 error

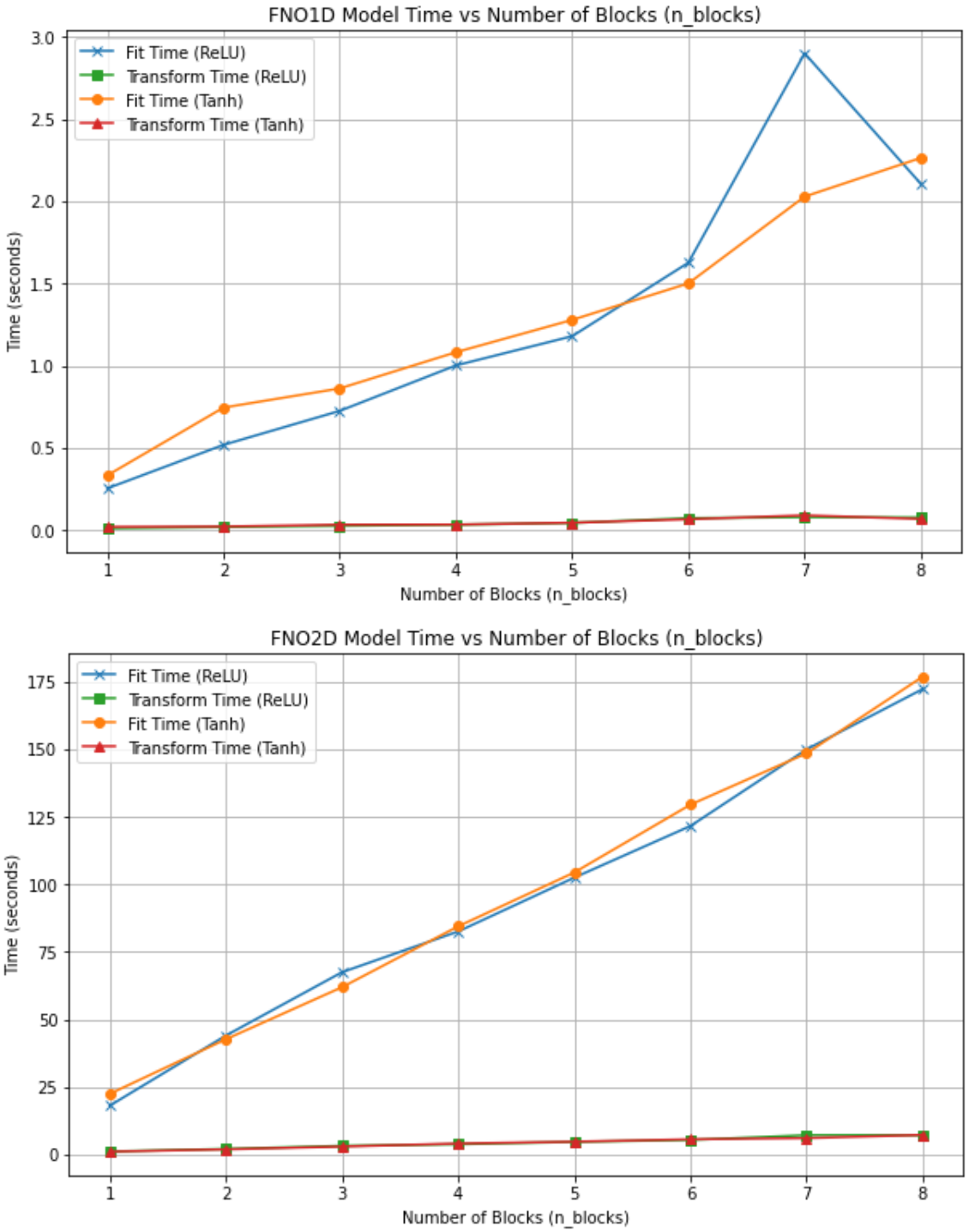

Figure 3.5.: For both FNO2D and FNO1D, figure shows the graphs of n_blocks vs time

Number of Hidden Channels (`n_hidden_channels`):
Building upon the previous observations, the experimental focus shifted to tuning the number of hidden channels (`n_hidden_channels`) within the FNO1D and FNO2D models. The impact of this hyperparameter on the relative L2 error and computational times offers further insights into the neural operators' performance characteristics.

For FNO1D models, the tanh activation function consistently outperforms ReLU in terms of error minimization across various settings of `n_hidden_channels`. In contrast, for FNO2D models, the trend is reversed, with ReLU achieving slightly better error metrics compared to tanh. However, incrementally increasing the number of hidden channels does not translate into substantial gains in performance for either model, in terms of error reduction. This plateau in performance improvement, despite increasing the number of channels, may be attributed to limitations such as a relatively small dataset size and a primitive default setup of other hyperparameters, which could be restricting the models' ability to fully leverage the increased complexity.

In terms of computational time, there is a clear linear relationship between the number of hidden channels and the fit time for both FNO1D and FNO2D models, irrespective of the activation function employed. This linear trend highlights the direct impact of model capacity on the resources required during the training phase. Notably, FNO2D models exhibit a drastically large fit time compared to their 1D counterparts, underscoring the heightened computational demands when dealing with two-dimensional data. Despite the variations in fit time, transform times remain largely unaffected by the number of hidden channels, indicating a consistent performance during the data transformation phase of the models

The absence of significant variations in transform times, akin to the observations in the tuning of `n_blocks`, points to a stable transformation phase that is less sensitive to changes in model depth and capacity. This stability in transform times, despite the linear rise in fit times, reinforces the importance of a judicious choice of hyperparameters to balance accuracy against computational efficiency, particularly when working with the more resource-intensive FNO2D models.

The exploration into the scaling of the number of hidden channels (`n_hidden_channels`) within FNO models has brought to light the primary challenge associated with this hyperparameter: memory constraints. For FNO2D models, increasing `n_hidden_channels` leads to the creation of a huge multi-dimensional array during training, the dimensions of which are directly proportional to the number of data points, the number of hidden channels, and the spatial resolution in both the x and y directions.

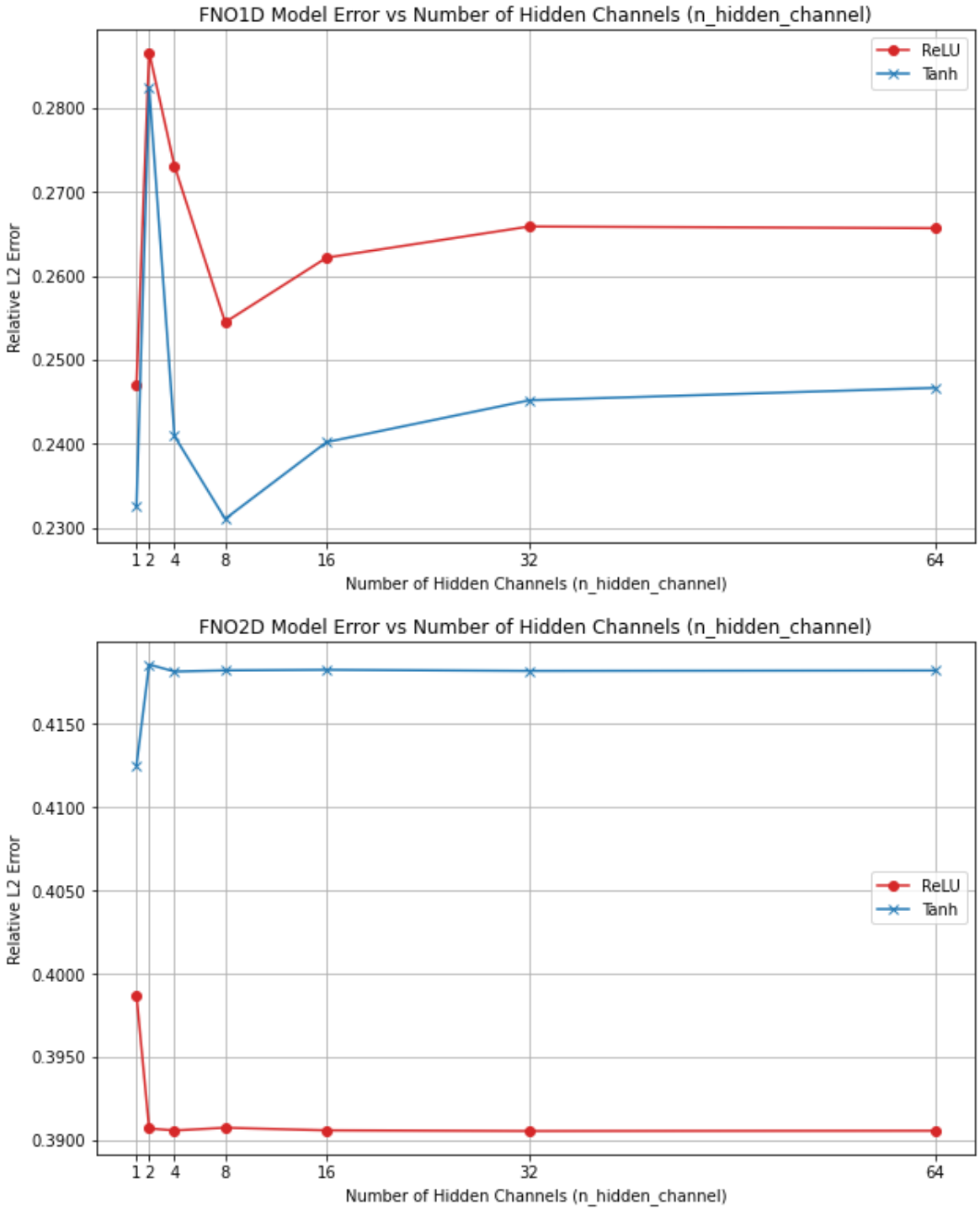

Figure 3.6.: For both FNO2D and FNO1D, figure shows the graphs of n_hidden_channels vs relative L2 error

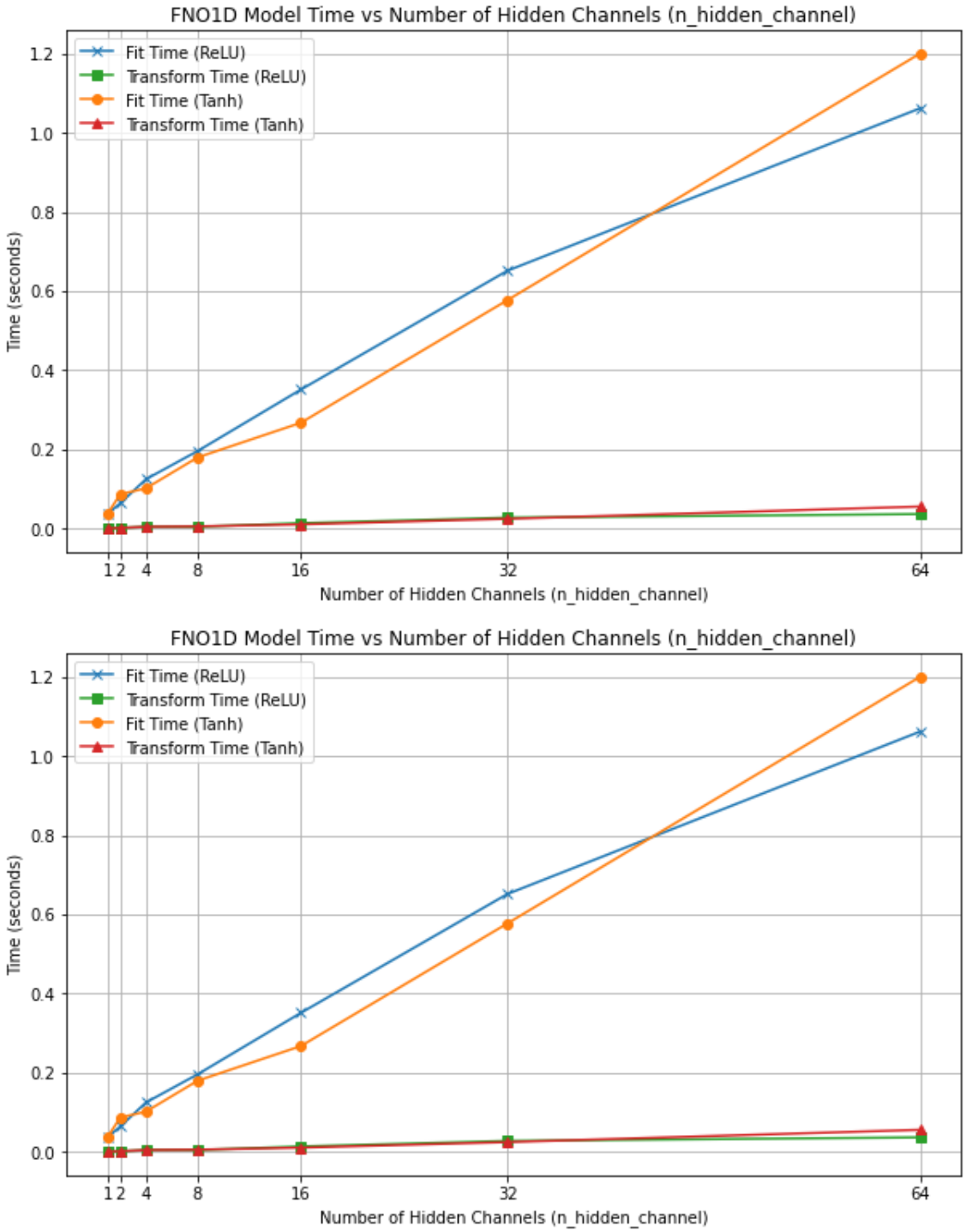

Figure 3.7.: For both FNO2D and FNO1D, figure shows the graphs of n_hidden_channels vs time

Concretely, for FNO2D, the training process necessitates the instantiation of an array with shape (Number of Data Points,Number of Hidden Channels,x resolution,y resolution). This requirement for high-dimensional arrays significantly escalates the memory usage, especially as the resolution and the number of hidden channels grow. The problem is exacerbated when dealing with 2D data, where the resolution along two spatial dimensions compounds the memory demands.

This memory bottleneck presents a substantial challenge, as it not only limits the potential depth and complexity of the models that can be trained but also restricts the scalability of the approach to larger datasets or higher resolutions. As such, careful consideration must be given to the balance between model complexity and available computational resources, with memory efficiency becoming a critical factor in the design and optimization of neural operator models, particularly in the context of two-dimensional domains.

Layer Width (`layer_width` Another hyperparameter examined is the layer width (`layer_width`), which has demonstrated a considerable influence on the performance of both FNO1D and FNO2D models. Diverging from the patterns observed with other hyperparameters, increasing the layer width has resulted in exponential improvements in error reduction. This is a significant finding, suggesting that layer width plays a crucial role in the model's ability to capture and represent the complexities of the underlying functions.

For FNO2D models, in particular, the increase in layer width to 256, paired with the tanh activation function, has led to a notable decrease in error, achieving values as low as 0.2. This improvement is compelling, considering that other hyperparameters were assigned basic, default values, indicating the potent impact of layer width on model accuracy.

Another salient observation is the superior performance of the tanh activation function over ReLU, especially at higher layer widths. This trend is consistent across both FNO1D and FNO2D models and becomes more pronounced as the layer width expands. The tanh function's ability to outperform ReLU under these conditions may be attributed to its smoother gradient behavior, which could facilitate better learning in networks with a larger capacity.

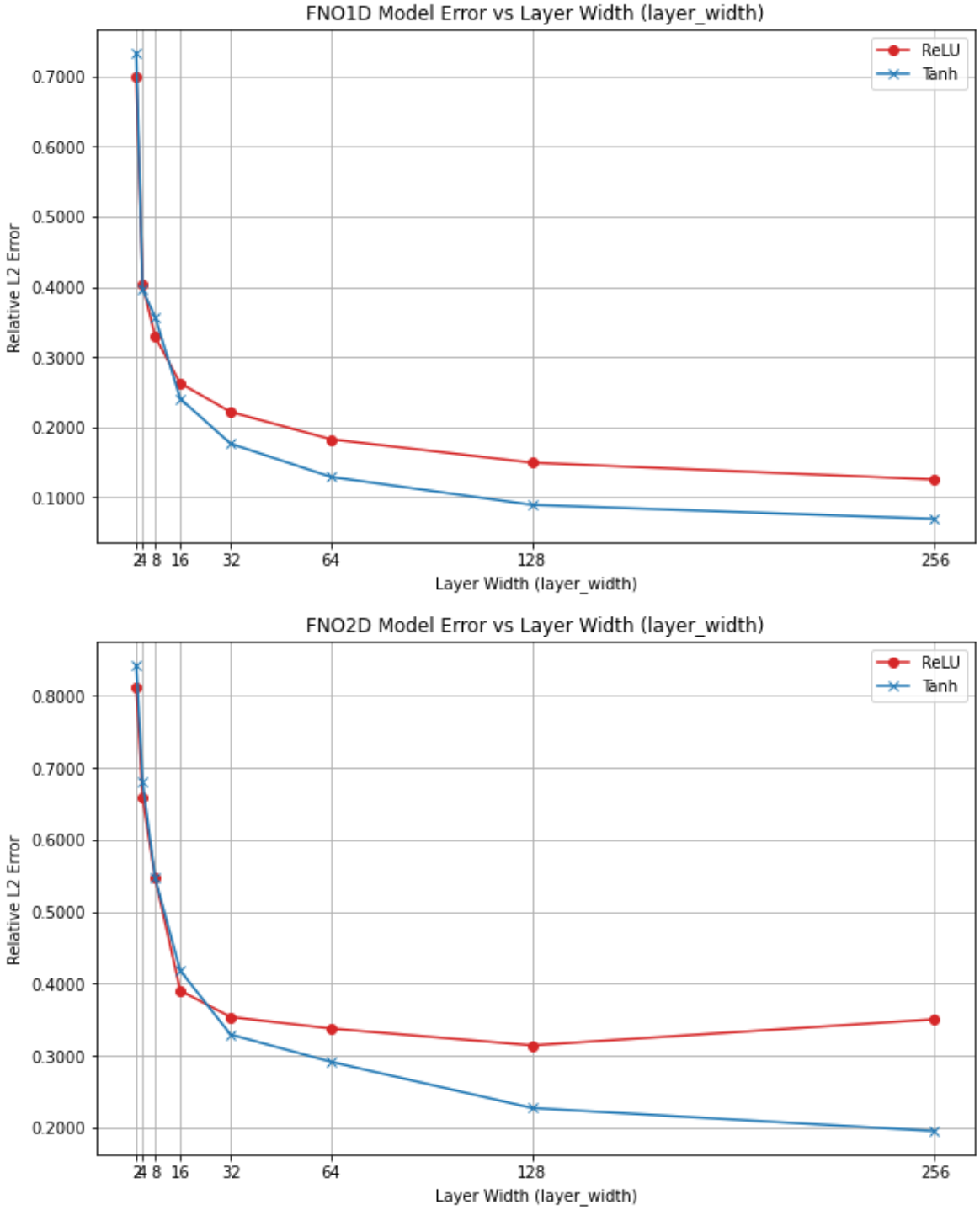

Figure 3.8.: For both FNO2D and FNO1D, figure shows the graphs of layer width vs relative L2 error

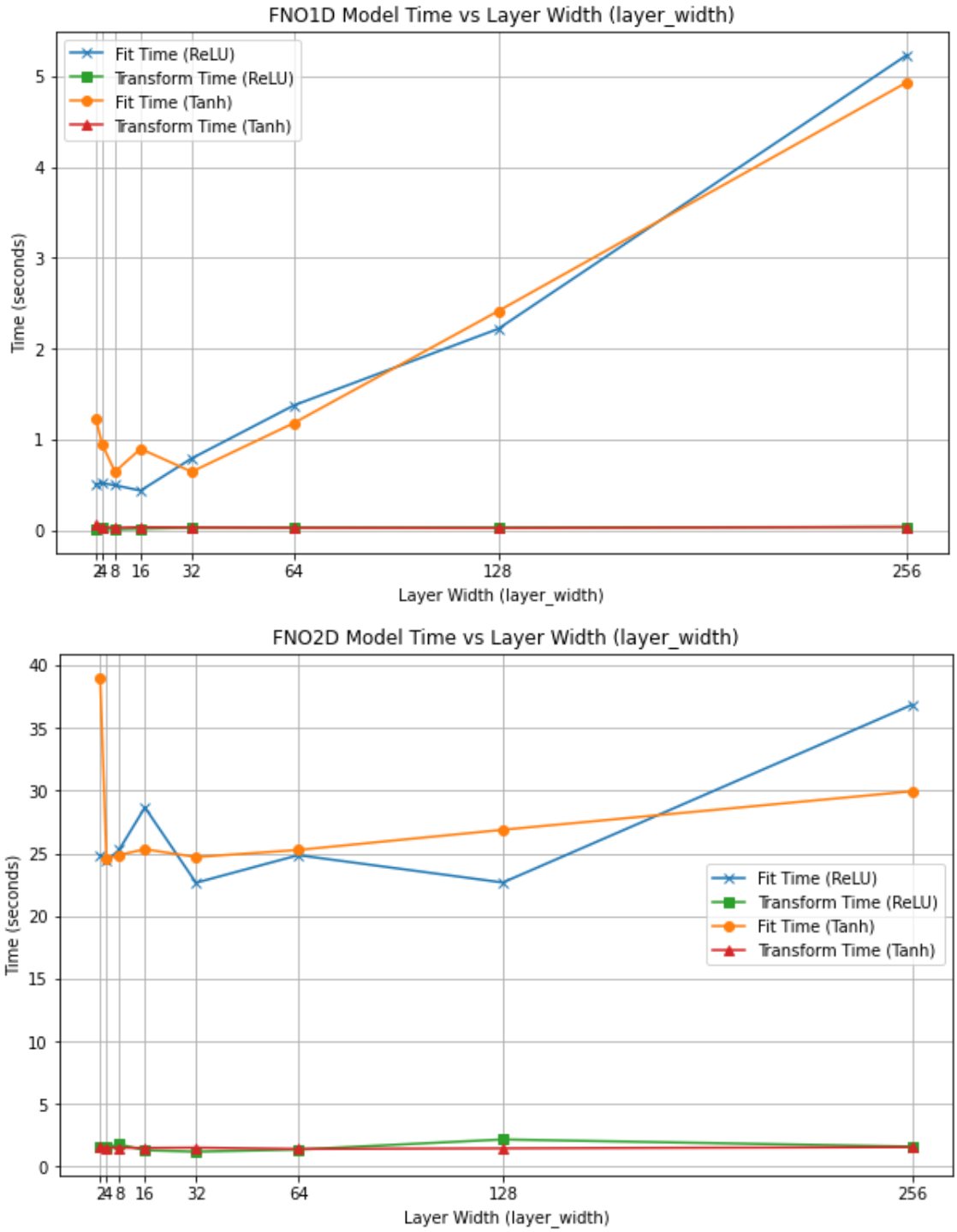

Figure 3.9.: For both FNO2D and FNO1D, figure shows the graphs of layer width vs time

Contrary to the linear relationships observed with the number of blocks and hidden channels, the increase in layer width does not correspond to a uniform rise in time consumption. This departure indicates that while larger layer widths significantly boost model accuracy, they do not necessarily lead to proportionally higher computational costs. This non-linear scaling of time with respect to layer width presents a more complex picture, where the additional computational overhead is not as predictable as with other hyperparameters. It underscores the potential for achieving higher accuracy without a straightforward increase in computational burden, emphasizing the importance of layer width as a hyperparameter in the design of efficient and effective neural operator models.

Number of Modes (`n_modes`)

The comprehensive analysis of the n_modes hyperparameter within the context of Fourier Neural Operators (FNO) has yielded insightful observations. For both FNO1D and FNO2D models, the increment in the number of Fourier modes, which essentially determines the resolution in the spectral domain, has not consistently translated into improved accuracy, as measured by the relative L2 error.

In the FNO2D models, the alteration of n_modes across a range of values revealed no discernible enhancement in error reduction. This was consistent across the spectrum, suggesting that for FNO2D, the complexity added by increasing n_modes does not necessarily equate to better model performance in terms of accuracy. Similarly, the FNO1D models largely mirrored this trend, with one notable exception. When n_modes was set to 2, the FNO1D models demonstrated a marginal improvement in error metrics. This specific configuration appears to be an optimal setting for the one-dimensional case, suggesting a potential sweet spot where the model is able to leverage the spectral resolution for a more accurate representation of the underlying functions.

Regarding computational time, although a linear relationship between increased n_modes and time consumption was not observed, the general trend indicates that higher n_modes generally lead to greater time requirements. This is likely due to the additional computational effort needed to process the larger spectral representation of the data. Even though this increase is not strictly linear, the overall impact on time consumption is clear, with higher n_modes contributing to longer training and transformation times for both FNO1D and FNO2D models.

These results suggest that while increasing n_modes may intuitively seem like a straightforward approach to enhancing model fidelity by capturing more detailed frequency information, the benefits in terms of error reduction are not as clear-cut. This nuanced finding underscores the complexity of neural network optimization, where not all increases in model parameters lead to direct improvements in performance, and highlights the importance of identifying parameter settings that are both time-efficient and effective in error minimization.

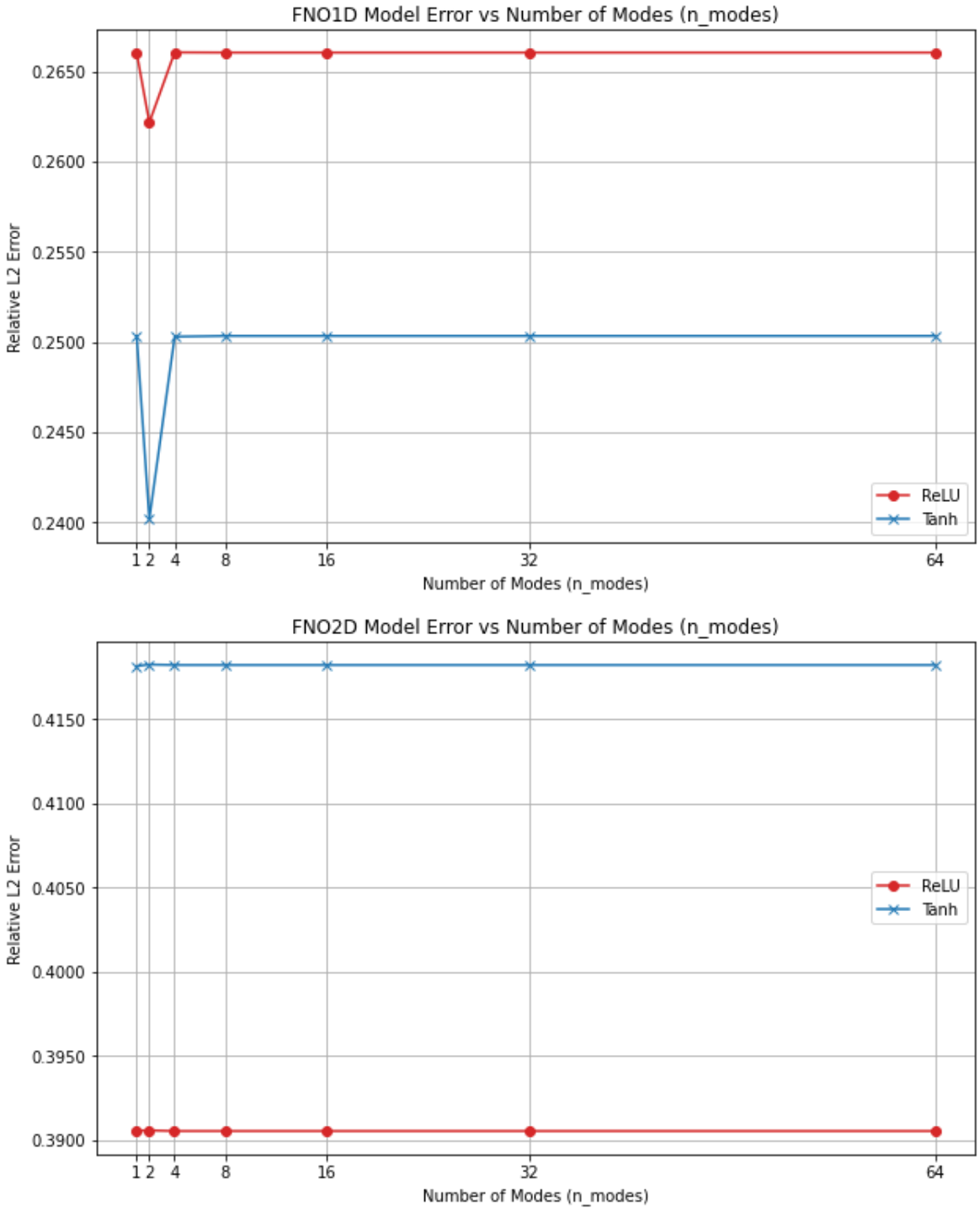

Figure 3.10.: For both FNO2D and FNO1D, figure shows the graphs of n_modes vs relative L2 error

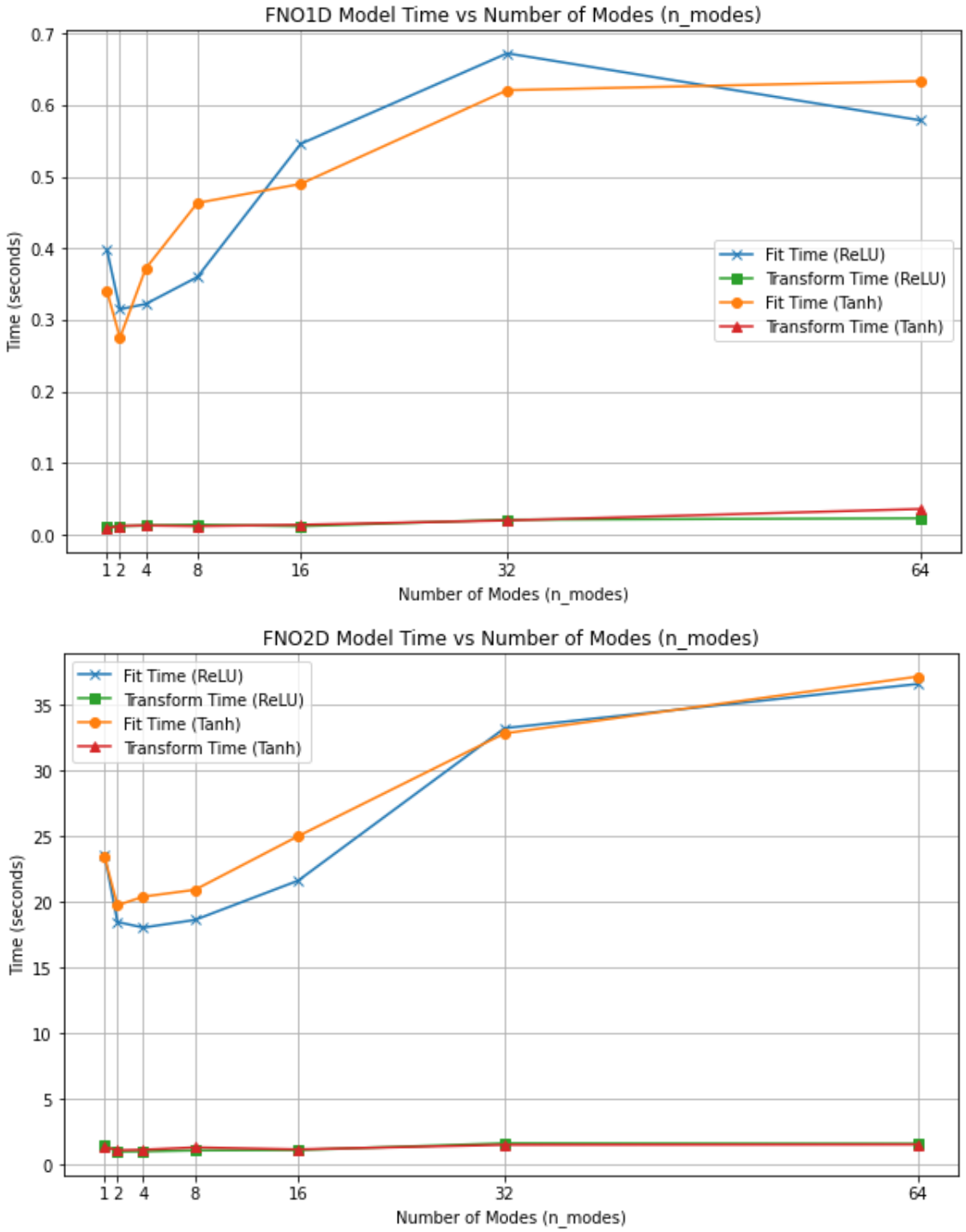

Figure 3.11.: For both FNO2D and FNO1D, figure shows the graphs of n_mode vs time

Regularization Scale

The comprehensive study focused on exploring the influence of varying regularization scales on the performance metrics of Fourier Neural Operators, specifically the FNO1D and FNO2D models. These models are integral components in the field of neural network architectures and are particularly noted for their efficiency in handling complex spatial-temporal data. The research meticulously analyzed the impact of different regularization scales, which are critical parameters in machine learning that contribute to preventing overfitting and ensuring that the models generalize well to new, unseen data.

Throughout the span of the study, a broad spectrum of regularization scales was scrutinized. The findings revealed a consistent and interesting pattern across both FNO1D and FNO2D models. It was discovered that when the regularization scale was set to a lower value, there was a significant enhancement in the performance of the models. This improvement was quantitatively measured in terms of error metrics, a crucial criterion in assessing the accuracy and reliability of neural network predictions. Lower regularization scales effectively reduced the error rates, indicating a more precise and dependable model performance.

In stark contrast, an increase in the regularization scale led to a notable escalation in error rates. This trend was uniformly observed in both models and across various scale settings. Such a rise in error rates is indicative of the models' decreasing ability to accurately predict and represent the data. This phenomenon underscores the delicate balance required in setting the appropriate regularization scale to optimize model performance.

Furthermore, this trend was consistent regardless of the choice of activation function in the neural networks. Both the tanh and ReLU activation functions were considered in this study. These functions play a pivotal role in introducing non-linearity to the models, enabling them to learn and represent more complex data patterns. The observation that the choice of activation function did not significantly alter the influence of regularization on model error is a notable insight. It suggests that the impact of regularization scale is a more dominant factor in influencing model performance than the specific type of activation function used.

Another aspect of the study was examining the impact of regularization scale on the time efficiency of the FNO1D and FNO2D models. Time efficiency, in this context, refers to the computational time required for the models to fit to the training data and subsequently transform new data. Interestingly, it was found that the computational time remained largely unaffected by changes in the regularization scale. This observation is significant as it indicates that adjusting the regularization scale mainly affects the model's generalization capabilities and complexity control, rather than its computational efficiency. This insight is crucial for practitioners in the field, as it allows them to fine-tune the regularization scale to optimize model performance without concerns about increased computational demands.

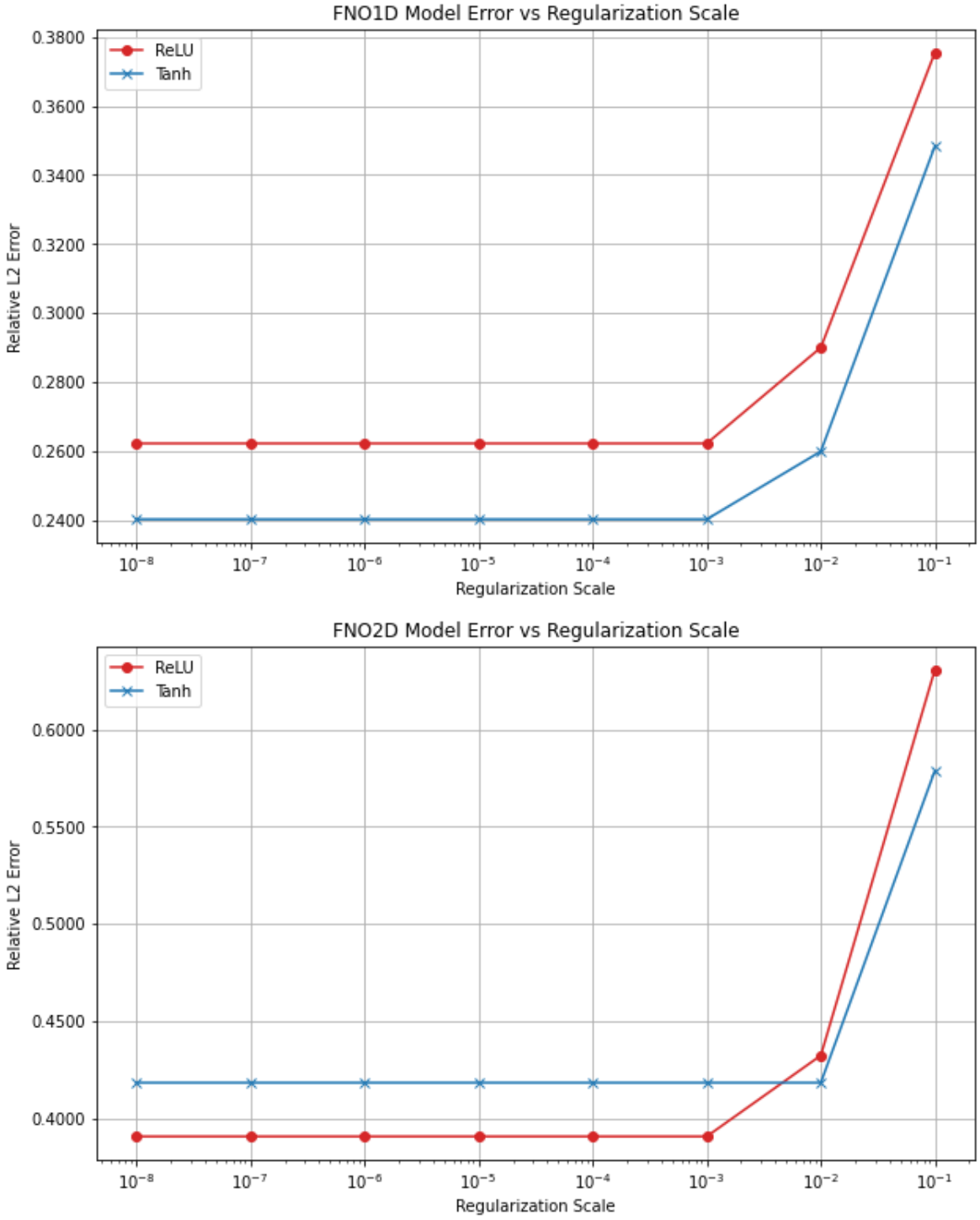

Figure 3.12.: For both FNO2D and FNO1D, figure shows the graphs of regularization scale vs relative L2 error

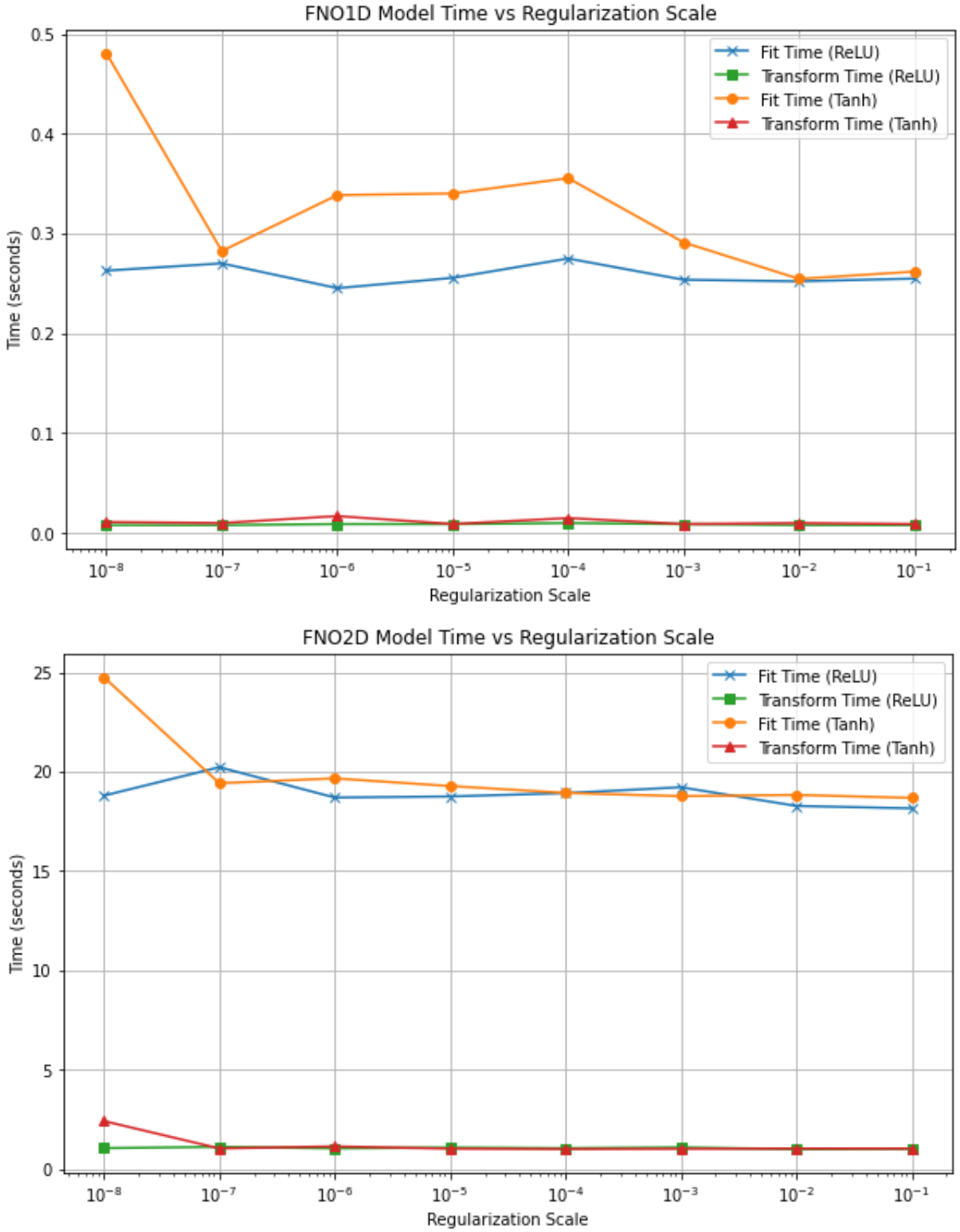

Figure 3.13.: For both FNO2D and FNO1D, figure shows the graphs of regularization scale vs time

Best Hyperparameter Setups and Their Results The numerical results obtained from the experimental evaluation of the FNO models are pivotal in demonstrating the efficacy of the chosen hyperparameter configurations. Two distinct setups are highlighted to illustrate the performance differences between the FNO1D and FNO2D models under optimized hyperparameters.

Model	n_blocks	nhc*	layer_width	Activation	Fit Time (s)	Relative L2 Error
FNO2D	2	4	256	tanh	19.19	1.980×10^{-1}
FNO1D	3	8	256	tanh	5.46	6.029×10^{-2}

Table 3.3.: Results for best hyperparameter setup in the range. The hyperparameters that does not appear in the table is same as its default (*nhc: n_hidden_channels)

These findings reinforce the hypothesis that layer width is a highly influential hyperparameter, especially when coupled with the tanh activation function. The numerical results underscore the delicate interplay between model architecture, dimensionality, and activation function, shaping the performance and computational efficiency of FNO models. These insights contribute significantly to the understanding of neural operator scalability and optimization.

3.4. Discussion of Results

The extensive experimental analysis conducted in this thesis offers a deep dive into the structural intricacies and operational capabilities of Fourier Neural Operator (FNO) models. This section aims to investigate and reflect upon the key outcomes derived from meticulous evaluations of the models, the distinct characteristics of the datasets used, and the comprehensive process of systematic experimentation that was undertaken.

FNO models, in their one-dimensional and two-dimensional variants, have showcased a commendable proficiency in learning and mapping complex functional relationships. The process of hyperparameter tuning, a pivotal aspect of this research, has shed light on the delicate equilibrium that exists between the complexity of the model, its computational demands, and its ability to accurately predict outcomes. The layer width has emerged as a particularly influential hyperparameter, with its adjustments showing a substantial impact on the model's error reduction capabilities. This effect is even more pronounced in the context of two-dimensional models, where the multifaceted nature of interactions within the data sets a higher bar for accuracy and complexity.

In the realm of experimental design, a strategic approach was adopted to evaluate how different hyperparameters influence the performance of FNO models. The findings from these experiments paint a comprehensive picture. On one hand, FNO1D models

are characterized by their rapid training and transformation capabilities. On the other hand, FNO2D models, while more computationally demanding, exhibit a strong potential in achieving lower error rates. This potential is particularly evident when the models are configured with an adequate layer width and an optimal count of hidden channels, highlighting the significance of these parameters in enhancing model accuracy.

Furthermore, the research indicates that the effectiveness of FNO2D models could be significantly amplified with the availability of larger datasets and more precise hyperparameter tuning. Such advancements, however, hinge on the provision of enhanced computational resources. With access to more powerful computing infrastructure, it becomes feasible to process larger volumes of data and to delve into more complex and deeper network architectures. This progression holds immense promise, especially for applications that necessitate a precise and detailed representation of two-dimensional spaces. The implications of such improvements are far-reaching, potentially leading to groundbreaking advancements in fields that rely heavily on accurate two-dimensional data modeling.

4. Conclusion

4.1. Summary

This thesis presented an in-depth exploration of Fourier Neural Operators (FNO), a novel approach in the field of computational science for learning mappings of function spaces, particularly those governed by partial differential equations (PDEs). The study extended the principles of the Sampling Where It Matters (SWIM) methodology to both one-dimensional and two-dimensional FNO models (FNO1D and FNO2D), thereby addressing the challenges posed by multi-dimensional data spaces.

Key to the advancement of FNO models was the systematic tuning of several critical hyperparameters, including the number of modes (`n_modes`), the number of hidden channels (`n_hidden_channels`), the layer width (`layer_width`), the number of blocks in the FNO architecture (`n_blocks`), and the regularization scale. Each of these parameters was carefully evaluated for its impact on the model's performance, with particular emphasis on their ability to capture the dynamics of the Burgers' equation.

The study demonstrated that while FNO1D models were efficient in terms of training and transformation times, FNO2D models showed a promising potential in achieving lower error rates with the right configuration of layer width and hidden channels. A significant observation was the limited impact of increasing the number of modes on the models' accuracy, suggesting that this parameter does not linearly correlate with improved performance.

An innovative aspect of this research was the integration of the meshgrid-based lifting procedure in the FNO2D models. This methodology enhanced the model's capacity to process and learn from two-dimensional spatial data with higher fidelity. It was observed that the integration of SWIM for parameter sampling and the adjustment of layer width were crucial in optimizing the FNO models' accuracy and efficiency.

Throughout the research, the balance between model complexity, computational efficiency, and predictive accuracy was a recurring theme. The results indicated that while larger layer widths and optimal hyperparameter settings could significantly improve model performance, they also increased the computational burden, especially for FNO2D models.

4.2. Discussion

The findings from this thesis contribute to a deeper understanding of FNO models and their potential applications. One of the key observations is the interplay between model complexity and computational efficiency. While FNO1D models exhibited swifter training and transformation capabilities, FNO2D models, with their enhanced complexity, demonstrated a stronger proficiency in error reduction, albeit at the cost of increased computational resources.

The integration of the meshgrid-based lifting procedure and the SWIM algorithm for parameter sampling in FNO2D models marked a significant methodological advancement. This integration underscored the importance of spatial integrity in data representation, especially for complex multi-dimensional problems.

However, the study also highlighted the limitations of increasing certain hyperparameters, such as the number of modes, which did not linearly correlate with performance improvement. This observation is critical in guiding future model optimization strategies, where the focus might shift from simply increasing model complexity to optimizing hyperparameter configurations.

Nevertheless, the general performance of the models, with a relative error around 20%, indicates room for improvement. This suboptimal result could be due to constraints such as limited dataset size and inadequate computational resources during the research. To enhance the models' performance, future research should consider using larger datasets, which would provide a richer training ground for the models to learn from more diverse and complex patterns. Additionally, overcoming the hyperparameter limitation due to memory and computational time constraints is essential. Utilizing more powerful computational hardware could allow for more extensive hyperparameter tuning and experimentation, potentially leading to significant improvements in model accuracy and efficiency. These steps could greatly advance the utility of FNO models in various complex computational tasks, driving them closer to achieving lower error rates and higher reliability in practical applications.

4.3. Future Work

Looking ahead, there are several promising directions for future research:

Enhanced Computational Resources: With the advancement of computational technologies, future work could explore the performance of FNO models on larger datasets and more complex PDEs. Enhanced computational resources could allow for deeper and more intricate network architectures, potentially leading to breakthroughs in accuracy

and efficiency.

Advanced Hyperparameter Optimization: Further research could focus on more sophisticated hyperparameter optimization techniques, possibly employing automated machine learning (AutoML) approaches to identify the most effective configurations.

Expansion to Other PDEs: While this study focused on the Burgers' equation, applying the findings to a broader range of PDEs could offer insights into the versatility and adaptability of FNO models across various scientific and engineering disciplines.

The research conducted lays a foundation for future investigations that could significantly advance our understanding and application of machine learning techniques in solving higher dimensional PDEs.

Bibliography

[1] Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *arXiv preprint arXiv:2012.12348*, 2020.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Erik Lien Bolager, Iryna Burak, Chinmay Datar, Qing Sun, and Felix Dietrich. Sampling weights of deep neural networks. *arXiv preprint arXiv:2306.16830*, 2023.

[4] Johannes Martinus Burgers. A mathematical model illustrating the theory of turbulence. *Advances in applied mechanics*, 1:171–199, 1948.

[5] Shifei Ding, Xinzheng Xu, and Ru Nie. Extreme learning machine and its applications. *Neural Computing and Applications*, 25:549–556, 2014.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[7] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[8] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

[9] Harold Levine. *Partial differential equations*, volume 6. American Mathematical Soc., 1997.

[10] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[11] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[12] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[13] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[14] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[15] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[16] Gerald Beresford Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011.

[17] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

Appendix

A. Technical Specifications

The research utilizes a device with 16GB RAM and an Intel(R) Core(TM) i7-7700HQ CPU (2.80GHz, 4 cores, 8 logical processors) and Python 3.10.11. This setup is employed for data generation and experimentation in the thesis.

List of Figures

List of Tables