A Numerical Study of Chaotic Dynamics of K-S Equation with FNOs

Surbhi Khetrapal*
School of Physics, University of Hyderabad
Hyderabad, Telangana, India.
surbhikhetrapal@uohyd.ac.in

Jaswin Kasi
Mercedez-Benz Research and Development
Bengaluru, Karnataka, India.
kasi.jaswin@mercedes-benz.com

Abstract

Solving non-linear partial differential equations which exhibit chaotic dynamics is an important problem with a wide-range of applications such as predicting weather extremes and financial market risk. Fourier neural operators (FNOs) have been shown to be efficient in solving partial differential equations (PDEs). In this work we demonstrate simulation of dynamics in the chaotic regime of the two-dimensional ($2d$) Kuramoto-Sivashinsky equation using FNOs. Particularly, we analyze the effect of Fourier mode cutoff on the results obtained by using FNOs vs those obtained using traditional PDE solvers. We compare the outputs using metrics such as the $2d$ power spectrum and the radial power spectrum. In addition we propose the normalised error power spectrum which measures the percentage error in the FNO model outputs. We conclude that FNOs capture the dynamics in the chaotic regime of the $2d$ K-S equation, provided the Fourier mode cutoff is kept sufficiently high.

1 Introduction

Solving Partial Differential Equations (PDEs) is one of the most important and challenging problems in scientific computing. PDEs are used to model a wide range of physical phenomena occurring in scientific and industrial problems. Traditionally, PDEs arising from complex industrial problems are solved on High-Performing Computers (HPC) based on numerical methods Press William et al. [1992] such as Finite Difference Methods (FDM), Finite Element Method (FEM), Spectral Methods etc. In spite of having high accuracy, these methods require significantly high computational costs and a long time to converge to an accurate solution, prompting researchers to explore novel alternatives that are more efficient and faster.

Interestingly, Deep Learning (DL), and particularly Deep Neural Networks (DNNs) are emerging as a radically new approach to solve PDEs. DNNs, being a type of universal function approximator, are well known to efficiently handle high-dimensional complexity, making them a promising tool for PDEs Sirignano and Spiliopoulos [2018], Huang et al. [2022]. By default DNNs admit parallelization of computations, making them easily trainable and infer-able on GPUs. Therefore solving PDEs with them, can straight-away deployed on GPUs, without requiring much mathematical trickery as compared to traditional solvers. Broadly, Deep Learning techniques for PDEs can be divided into two approaches - physics based and data-driven.

1.1 Physics-Informed Neural Networks

Physics-based (commonly referred to as, Physics-Informed Neural Networks (PINNs)) approach, involves integrating underlying physical laws described by PDEs directly into a DNNś training process Raissi et al. [2019], Lu et al. [2021], Cuomo et al. [2022]. The key idea, is to approximate

*Corresponding author

1st Conference on Applied AI and Scientific Machine Learning (CASML 2024).

the solution of PDE $\vec{u}(\vec{x}, t)$ as a DNN, and train the network with a total loss given by,

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{PDE} + \mathcal{L}_{BC} + \mathcal{L}_{IC} \tag{1}$$

where, $\mathcal{L}_{data}$ are few sample data collected from traditional solvers, and the remaining components are self-supervising loss terms. $\mathcal{L}_{PDE}$ ensures that the DNN solution satisfies the governing equation by using Automatic differentiation. $\mathcal{L}_{IC}$, $\mathcal{L}_{BC}$ ensures that the network respects initial and boundary conditions respectively. The pros of PINNs are that they can approximate high-dimensional PDEs, are mesh-free and more efficient to train with limited or noisy training data. However, there are various limitations of PINNs - they require an explicit way of incorporating the PDE and modeling of complex initial and boundary conditions. Also, for a different initial/boundary condition, PINNs needs to be retrained all over again!

1.2 Data-Driven Approaches

To avoid explicitly modeling the physics, recently, multiple works have emerged which solve PDEs by a purely data-driven approach. These solvers utilize dynamical data obtained from conventional solvers for training. They then train neural networks to model the dynamics as a transformation between infinite-dimensional function spaces, eliminating the need for explicit physical laws. The class of such approaches are called Neural Operators, and few examples are Deep Operator Networks (DeepONet) Lu et al. [2019] and Fourier Neural Operators (FNO) Li et al. [2021].

DeepONet is a DNN designed to learn non-linear operators. It consists of two components : branch net and trunk net. The branch net processes the input function at discrete spatial and temporal values and transforms it into feature vectors. Concurrently, the trunk net takes the input spatio-temporal coordinates , and transforms them into feature vectors of same dimensions as branch net's feature vectors. These two feature vectors are merged by dot-product and further processed in DNN layers to predict the value at the inputted spatio-temporal coordinates. DeepONets are highly flexible and versatile and perform well for different initial and boundary conditions. They are particularly effective in reducing the generalization errors compared to PINNs. Despite their strengths, DeepONets cannot guarantee overall physics knowledge, involve complex architecture making it difficult to train and struggle at scaling to higher dimensional input functions.

1.3 Fourier Neural Operators (FNOs)

To overcome limitations of previously mentioned PDE solvers, Li et al. [2021] came up with a novel approach. FNO is a class of neural operators, which learns the mapping between infinite-dimensional function spaces as a sequence of integral operators acting directly on the Fourier space. FNOs map the input function $a(x)$ to a solution $u(x)$ using a parameterized neural network G_ϕ. The form of G_ϕ is given by,

$$G_\phi = \mathcal{P} \circ \sigma(W_n + \hat{K}_n) \circ \ldots \circ \sigma(W_1 + \hat{K}_1) \circ \mathcal{Q}a(x) \tag{2}$$

here, $\mathcal{P}$, $\mathcal{Q}$ are lifting operators to higher dimensions and W_i is local linear transformation operator at the i^{th} layer. $\hat{K}_i$, is the non-linear operator given by,

$$\hat{K} = \mathcal{F}^{-1} \circ R \circ \mathcal{F} \tag{3}$$

where $\mathcal{F}$ represents a non-local Fourier transformation and R is a linear transformation. σ is a non-linear activation function which is typically used in PINNs. FNOs have proven to be exceptionally efficient in capturing turbulent flows, physics on complex geometries Li et al. [2023a], Li et al. [2024], weather modeling Pathak et al. [2022] and many more applications. Motivated by the success of FNOs, in this work we undertake a numerical study of FNOs for a dynamical system which is well known to exhibit chaos.

1.4 Chaotic dynamics and Kuramoto-Sivashinsky equation

Chaos in PDEs arises when the equations governing the system exhibit non-linearities that lead to complex, unpredictable behavior. These non-linearities can arise from various physical phenomena, such as fluid turbulence, nonlinear optics, and chemical reactions. When these non-linearities are sufficiently strong, the system can exhibit chaotic behavior, characterized by extreme sensitivity to initial conditions, irregular patterns, and the inability to predict future states with certainty. This

chaotic behavior can make it challenging to analyze and understand the dynamics of complex systems, as small perturbations can lead to drastically different outcomes. Despite the challenges, studying such systems has great societal impact, particularly to predict weather extremes Blanchard et al. [2022].

The Kuramoto-Sivashinsky (K-S) equation is a nonlinear PDE which exhibits chaos. It has applications in various fields including fluid dynamics, combustion theory, and materials science. It is a canonical model for studying pattern formation and spatio-temporal chaos. In fluid dynamics, it describes the evolution of thin films and interfaces, while in combustion theory, it models the propagation of flames. In materials science, it is used to study the formation of surface patterns and the dynamics of thin films. Additionally, the K-S equation has found applications in finance, where it can be used to model the dynamics of asset prices and the emergence of market bubbles. Due to its versatility and ability to capture complex nonlinear phenomena, the K-S equation remains an active area of research in various scientific disciplines.

FNOs have been used to study the chaotic regime in the two-dimensional ($2d$) Kolmogrov equation Li et al. [2023b] and one-dimensional K-S equation Lippe et al. [2023]. In this paper, we study the $2d$ K-S equation in a square box with Dirichlet boundary conditions,

$$\partial_t u = -\frac{1}{2}|\nabla u|^2 - \nabla^2 u - \nabla^4 u \tag{4}$$

where $u(x, y, t)$ is a scalar field. Here the dimenionality $1d$ vs $2d$ refers to spatial dimensions.

2 Methodology

In this section, we describe the details regarding generation of the data which is used for training and testing the FNOs, as well as the model architecture.

2.1 Data generation

The training data is generated by solving the K-S equation given in equation (4) using finite difference method for a unit spatial grid of size 128×128 ($x, y \in (0, 128)$) with Dirichlet boundary conditions. The PDE is evolved up-to time $t = 10$ with time step $\delta t = 0.01$. A dataset of 128 samples is generated where the initial state of the scalar field $u_0(x, y)$ is populated with random values drawn from a uniform distribution with values between 0 and 1. Data obtained thus is referred to as *ground truth* in this paper.

Furthermore, on performing spectral analysis on the data generated, we observe a continuous distribution of frequencies which suggests a lack of periodicity or a complex, non-periodic structure, characteristic of chaotic systems (figure 4).

2.2 Fourier modes truncation in FNOs

In this sub-section we mention details regarding the architecture of the neural network used. In addition to the initial and final fully connected layers, the neural network is made up of 4 FNO layers between these fully connected layers. Since we analyse the effect of the Fourier mode cutoff on the resulting model output, we compare the following two FNO models:

- *FNO modes-12* with frequency modes: 12 and hidden channels: 64,
- *FNO modes-24* with frequency modes: 24 and hidden channels: 64.

The difference in frequency modes in the above two models led to a difference in number of parameters to be trained namely 4,743,937 weights for FNO modes-12 vs 18,899,713 for FNO modes-24. The dataset of 128 samples was divided into train, validation and test datasets of 80:20:20 samples. Further details regarding the training of the FNO models are mentioned in appendix A.2. The output of the two FNO models is compared to the ground truth for a particular sample in the test dataset in figure 1.

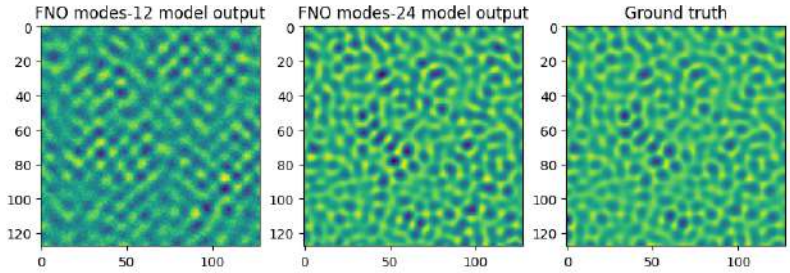

Figure 1: A comparison of output from FNO models vs ground truth.

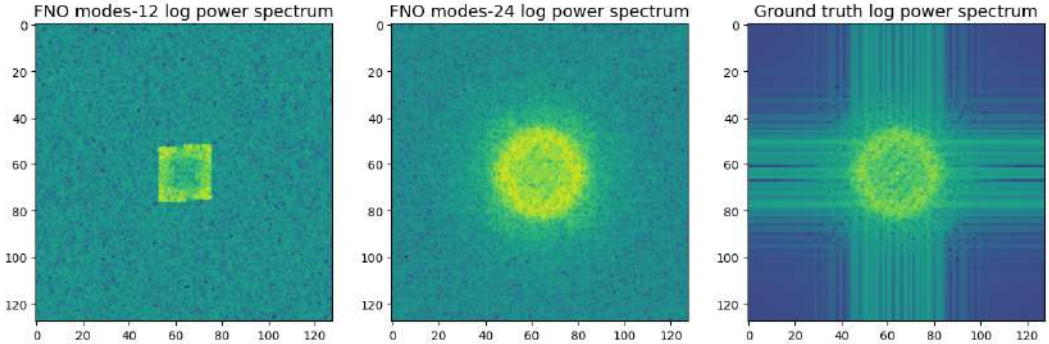

Figure 2: The two-dimensional log power spectrum of FNO model outputs vs ground truth.

3 Experimental Results

In order to compare the output of the FNO models with the ground truth, in this section we perform spectral analysis. Particularly, we compare the logarithm of the $2d$ power spectrum and the radial power spectrum of the prediction error between FNO outputs and ground truth.

3.1 $2d$ Power Spectrum

The $2d$ power spectrum is a powerful tool for analyzing the frequency content of $2d$ data. It provides a visual representation of the energy distribution in the frequency domain and helps identify dominant patterns. In order to obtain this we first perform the $2d$ Fourier transform:

$$F(k_x, k_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i2\pi(k_x x + k_y y)} dx dy. \tag{5}$$

The power spectrum is obtained by taking the square of the magnitude of the Fourier transform:

$$P(k_x, k_y) = |F(k_x, k_y)|^2 = \mathrm{Re}(k_x, k_y)^2 + \mathrm{Im}(k_x, k_y)^2. \tag{6}$$

We compare the logarithm of this power spectrum $\log P(k_x, k_y)$ for FNO modes-12 and modes-24 with the ground truth in figure 2. The component of the Fourier transform with zero frequency (DC component) is at the centre of the matrix. We see that the FNO model with Fourier modes cutoff at 24 captures a lot more spectral features of the ground truth compared to the FNO model with Fourier modes cutoff at 24.

3.2 Radial power spectrum for prediction error

The radial power spectrum provides a clear visualization of the energy distribution in the frequency domain, focusing on the radial variation rather than the $2d$ grid. Thus, it is useful in analyzing patterns and features in the frequency content of the output. The steps for computing the same are given below:

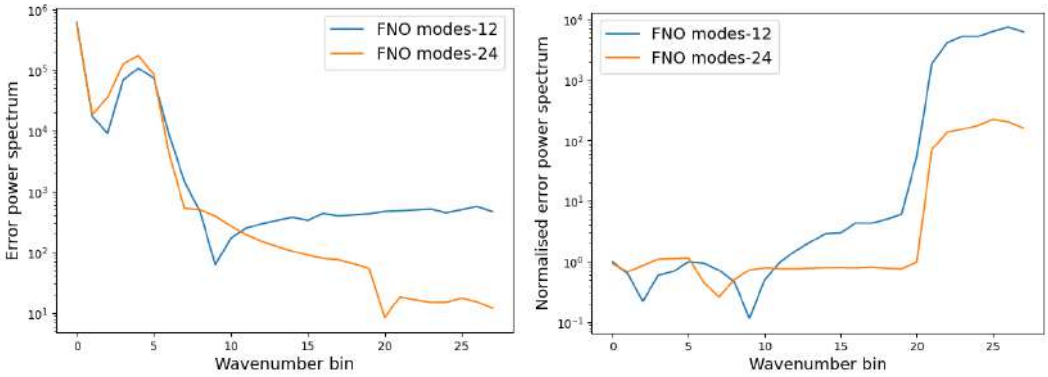

Figure 3: A comparison of error power spectrum (left) and normalised error power spectrum (right) vs wavenumber for FNO modes-12 and FNO modes-24.

1. Obtain the frequency-domain representation of the output of the FNO model by performing the $2d$ fast Fourier transform (FFT) given in equation (5).
2. Determine the radial wave number for each entry in the Fourier-transformed matrix, which is the distance from the center of the matrix $P(k_x, k_y)$.
3. Average the power spectrum along the radial wavenumbers. This is done by binning the spectrum elements based on their radial wavenumber and calculating the average power within each bin. As an example, for the 128×128 output, we use 28 bins.

We call the results obtained by following these steps: *radial power FNO (modes-12/modes-24) output* and *radial power ground truth*. Next, the absolute difference between these two is computed, shown in figure 3 (left). We find that at higher wavenumber, the energy in prediction error in case of FNO modes-24 model is lower compared to FNO modes-12 model.

While the radial power spectrum for prediction error is a useful tool used in literature Qin et al. [2024], Lippe et al. [2023], it fails to provide a clear visualization of percentage error in the radial power spectrum of the FNO model outputs vs ground truth. In order to take this into account, we propose the following metric to analyze the error:

$$\text{Normalised error power spectrum} = \left| \frac{\text{radial power FNO output} - \text{radial power ground truth}}{\text{radial power ground truth}} \right| \quad (7)$$

The plot of the normalised error power spectrum vs radial wavenumber is shown in figure 3 (right). We find that the percentage error in the radial power spectrum remains similar at small wavenumber for both the cases, i. e. Fourier modes cutoff 12 and 24. However, there is an exponential increase in power for higher wavenumbers for FNO modes-12 at wavenumber bin 10 compared to at wavenumber bin 20 for FNO modes-24. Also, the energy in the latter remains lower at the highest wavenumbers.

4 Conclusion

In this work we demonstrate that FNOs can capture the dynamics in the chaotic regime of the $2d$ K-S equation, given that Fourier mode cutoff is kept sufficiently high. We compare the performance of two FNO models where Fourier mode cutoff is kept at 12 and 24 respectively. On comparing the $2d$ power spectrum, radial power spectrum of prediction error and normalised error power spectrum, we find that the latter model, i. e. FNO modes-24 captures the chaotic dynamics of the ground truth better upto to higher wavenumbers. This is also due to the higher number of parameters trained in FNO modes-24 compared to FNO modes-12. However, even for FNO modes-24, the training loss and validation loss achieved are 0.13677 and 0.29458, respectively. This suggests that the model can get better convergence with more training data.

4.1 Future Work

While the study has shown that FNOs can simulate the temporal dynamics of the chaotic KS equation, it would be interesting to see if it can also capture the higher order statistical properties of chaos in the system. More specifically, measuring the Lyapunov exponent in this system and bench-marking against traditional solvers will provide us a better understanding of how well FNOs perform in this setup Edson et al. [2019].

In a recent work, it has been demonstrated that FNOs can learn dynamics in quantum spin systems Shah et al. [2024]. While the systems studied in this work are integrable, it would be interesting to see how efficient are FNOs in learning dynamics of quantum chaotic systems Alba and Calabrese [2019], Khetrapal and Pedersen [2024].

Acknowledgements

SK's research is supported by Department of Science and Technology's INSPIRE grant DST/INSPIRE/04/2020/001063.

References

Vincenzo Alba and Pasquale Calabrese. Quantum information scrambling after a quantum quench. *Phys. Rev. B*, 100(11):115150, 2019. doi: 10.1103/PhysRevB.100.115150.

Antoine Blanchard, Nishant Parashar, Boyko Dodov, Christian Lessig, and Themistoklis Sapsis. A multi-scale deep learning framework for projecting weather extremes, 2022. URL `https://arxiv.org/abs/2210.12137`.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Russell A Edson, Judith E Bunder, Trent W Mattner, and Anthony J Roberts. Lyapunov exponents of the kuramoto–sivashinsky pde. *The ANZIAM Journal*, 61(3):270–285, 2019.

Shudong Huang, Wentao Feng, Chenwei Tang, and Jiancheng Lv. Partial differential equations meet deep neural networks: A survey. *arXiv preprint arXiv:2211.05567*, 2022.

Surbhi Khetrapal and Emil Tore Mærsk Pedersen. Mutual information scrambling in ising spin chain, 2024. URL `https://arxiv.org/abs/2402.13558`.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. URL `https://arxiv.org/abs/2010.08895`.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

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, 2023b. URL `https://arxiv.org/abs/2111.03794`.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E. Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers, 2023. URL `https://arxiv.org/abs/2308.05732`.

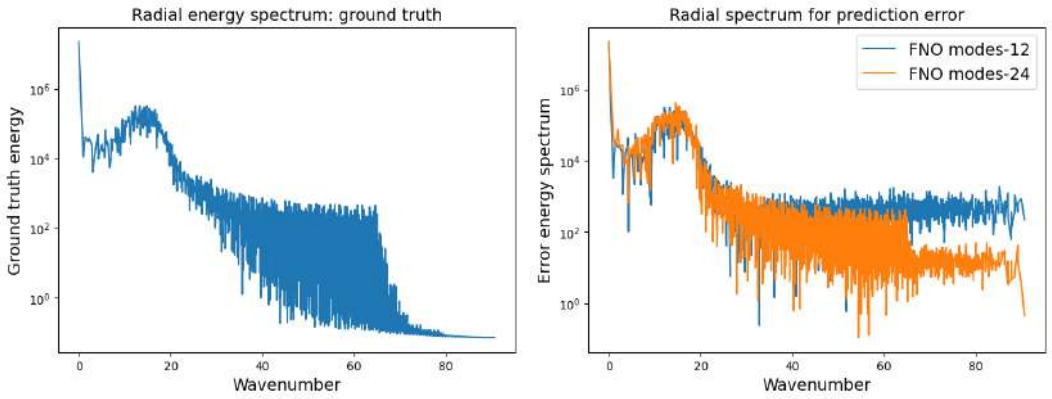

Figure 4: Radial power spectrum of ground truth vs radial wavenumber (left) and power spectrum of error between FNO model outputs and ground truth vs wavenumber (right).

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

H Press William, A Teukolsky Saul, T Vetterling William, and P Flannery Brian. Numerical recipes: the art of scientific computing, 1992.

Shaoxiang Qin, Fuyuan Lyu, Wenhui Peng, Dingyang Geng, Ju Wang, Naiping Gao, Xue Liu, and Liangzhu Leon Wang. Toward a better understanding of fourier neural operators: Analysis and improvement from a spectral perspective, 2024. URL `https://arxiv.org/abs/2404.07200`.

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.

Freya Shah, Taylor L. Patti, Julius Berner, Bahareh Tolooshams, Jean Kossaifi, and Anima Anandkumar. Fourier neural operators for learning dynamics in quantum spin systems, 2024. URL `https://arxiv.org/abs/2409.03302`.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

A Appendix

A.1 Spectral analysis of training data

The radial power spectrum of ground truth result shows a continuous distribution of frequencies, i. e. a broadband spectrum as seen in figure 4 (left) which implies chaotic behavior. This is because chaotic systems tend to exhibit complex, non-periodic patterns that lack a clear, repeating structure. Conversely, a discrete spectrum, featuring distinct, sharp peaks, suggests more regular behavior. Such peaks correspond to periodic or quasi-periodic components, implying a predictable or cyclical pattern.

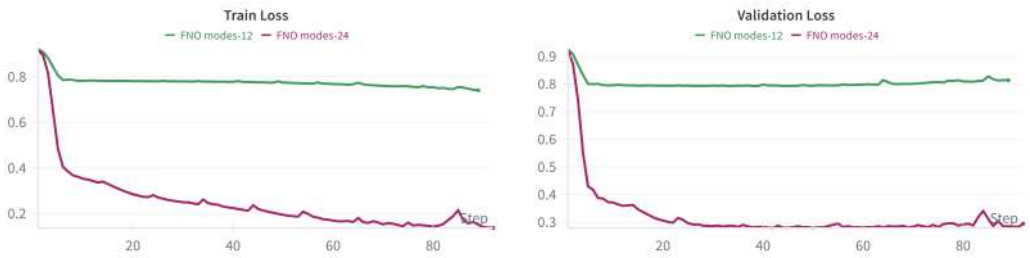

Figure 5: A comparison of training and validation loss of the two FNO models

A.2 Details of FNO model training

This appendix provides details regarding the training of the two FNO models mentioned in section 2.2. The hyper-parameters used in experimental study of FNOs are the following:

- Loss function: relative L_2 loss
- Optimizer: Adam
- Scheduler: step learning rate
- Test loss metric: MSE
- Weight decay: 0.0001

The FNO modes-12 model was trained for 89 epochs and FNO modes-24 model was trained for 92 epochs. Due to difference in trainable parameters viz. 4,743,937 for the former vs 18,899,713 for the latter, this led to a vast difference in training and validation loss achieved in the training of these two model as shown in figure 5.