EFFICIENT PDE SOLUTIONS USING HARTLEY NEURAL OPERATORS IN PHYSICS-INFORMED NETWORKS: POTENTIALS AND LIMITATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we introduce novel differentiable architectures for solving partial differential equations (PDEs) using well-known Discrete Hartley Transform. We incorporate Hartley Neural Operators (HNO) into Physics-Informed Neural Operator Networks (PINOs). Our analysis concentrates on two pivotal PDEs: 1. the one-dimensional diffusion equation, which holds significance not only in machine learning but also across a broad spectrum of physical sciences and engineering disciplines; and 2. the one-dimensional Thermodynamic Energy Equation that is commonly used in weather data analysis. Our implementation of HNO that employ real-valued linear transforms into the PINO architecture results in significant run time improvements. We show that reconstruction loss is lower than other recently introduced operators that may be used for the above PDEs. Importantly, we find that HNOs naturally satisfy the governing physical laws and equations specific to the PDEs under consideration. However, our empirical observations suggest that the benefits of HNO diminishes in certain scenarios where the underlying physical conditions at the boundary are less tractable and involve complex numbers. As an example of a potential failure mode we illustrate that in the case of the one-dimensional Burger's equation, traditional Fourier Neural Operators outperform their Hartley counterparts. Our results indicate that a combination of neural operators including Fourier and Hartley transforms may be better to effectively address the specific type, and/or context of physical problem at hand.

1 INTRODUCTION

Designing a real-valued Neural Operator for Deep Learning. Fourier Neural Operators (FNOs) Li et al. (2020), combines Fourier analysis with neural networks to study multi-scale phenomena. The conventional methodology employed in utilizing neural networks for solving differential equations often entails the process of discretizing the complete domain into a grid and subsequently predicting values at each individual grid point. Nevertheless, the scalability of this approach deteriorates significantly as the problem dimensionality grows. The approach of Fourier Neural Operators involves operating directly in the Fourier domain, leveraging its inherent capacity to handle differential operators. This methodology results in solutions that are both more salable and accurate.

While revolutionary, the FNO framework has certain limitations. For example, consider using Fourier Transforms to modify neural network representations to solve partial differential equations (PDEs) with real-valued variables, and solutions that are known a priori to be real valued also. In such a setting, it is possible that the resulting scheme provides suboptimal solutions unless an excessively precise resolution is used to eliminate imaginary components appropriately. The Hartley Transform Hartley (1942), like the Fourier Transform, can transform signals between the time (or spatial) and frequency domains with the key advantage is that it produces real-valued output for real-valued input. Therefore, it operates fully over the space of real numbers – so post processing conversions are not necessary. This is advantageous for real-valued PDEs because it eliminates the need to use complex numbers inherent to the Fourier Transform. This distinction has proven advantageous in the fields of image processing and the solution of certain partial differential equations in which the existence of real-valued solutions is guaranteed.

We introduce the concept of Hartley Neural Operators (HNO), which follows the same inspiration and framework as FNO while leveraging the advantages of Hartley Transform. While the concept of applyng the Hartley Transform to neural operators has, to our knowledge, not been done before, the concept of leveraging Hartley transforms for deep learning is nothing new Arenas & Serrano-Gotarredona (1996). Our study, however, incorporates HNO into neural operator learning frameworks. Specifically, we apply various combinations of HNO and FNO layers in a neural network to compare solutions of a small sample set of PDEs that are of interest to machine learning and the physical sciences alike. The first method combined the HNO and FNO layers to evaluate their potency and collaboration. The second procedure replaced all four layers of the structure with HNO layers. After rigorous evaluations, both approaches improved performance measures. These changes shed light on the potential of HNO within the PINO framework and lay the foundation for further investigation.

Our Contributions. Our key contribution is that we modify and adapt the FNO frameworks using Hartley Transforms as a drop-in replacement for the Fourier Transform within the neural operator layer architecture. These Hartley Neural Operators are explored for their potential use and efficacy in this organized framework. We also use a cyclic convolution method, instead of a standard element-wise multiplication for the HNO layers, which efficiently handles periodic signals and is computationally advantageous when using the Discrete Hartley Transform (DHT), as it maintains a consistent signal length without extending boundaries. Otherwise, the convolution theorem for the Hartley Transform is much more complex than a simple multiplication in the transformed basis.

2 Neural Hartley Transform

Preliminaries. The *continuous* Hartley transformation of a function $f(t)$ is mathematically defined by the following integral transform:

$$H_f(\omega) = \int_{-\infty}^{\infty} f(t) \cdot \text{cas}(\omega t)\, dt \tag{1}$$

where the cas function is defined as the sum of cosine and sine functions involving frequency ω, that is, $\text{cas}(\omega t) = \cos(\omega t) + \sin(\omega t)$. Similar to the Fourier transform, the Hartley transform has the unique property that it is involutionary, that is, the inverse is defined by:

$$f(t) = \int_{-\infty}^{\infty} H_f(\omega) \cdot \text{cas}(\omega t)\, d\omega \tag{2}$$

For transforming discrete signals such as vectors, we discretize the integral in equation 2 as follows. Given a sequence $f[n]$ of length N, the *Discrete* Hartley Transform (DHT) Bracewell (1983) is defined as:

$$H_k = \sum_{n=0}^{N-1} f[n] \cdot \left(\cos\left(\frac{2\pi kn}{N}\right) + \sin\left(\frac{2\pi kn}{N}\right)\right) \tag{3}$$

The Discrete Hartley Transform (DHT) is a linear operation that involves the multiplication of a vector $(x_0, \cdots, x_{N-1}) \in \mathbb{R}^N$ by an $N \times N$ matrix. In order to obtain the initial value of x_n from H_k, it suffices to perform the Discrete Hartley Transform (DHT) on H_k and afterwards normalize the output by a factor of $(1/N)$. Thus, the DHT can be considered a self-inverse structure, but incorporating a normalization/scaling factor.

Though the convolution of the DHT is not as straightforward and elegant as the discrete Fourier transform (DFT), it is still fairly computationally inexpensive. For the cyclic convolution of vectors, x and y producing z (all length N), post-DHT operations become simpler. If X, Y, and Z represent the DHTs of x, y, and z respectively, then Z can be obtained as,

$$\begin{aligned} Z_k &= \frac{X_k(Y_k + Y_{N-k}) + X_{N-k}(Y_k - Y_{N-k})}{2} \\ Z_{N-k} &= \frac{X_{N-k}(Y_k + Y_{N-k}) - X_k(Y_k - Y_{N-k})}{2}. \end{aligned} \tag{4}$$

Similarities to Wavelet transforms and Discrete Cosine Transforms. Integral transforms such as the wavelet transform (Grossmann & Morlet, Daubechies (1988)), the Hartley transform, and the discrete cosine transform (DCT) Ahmed et al. (1974), analyze and represent data in multiple domains. Both use basis functions to analyze and represent signals. For multi-resolution signal analysis, the wavelet transform derives basis functions from a ”mother wavelet” by dilation and translation Akansu & Haddad (1992). Whereas sinusoidal and cosinusoidal functions are used in the Hartley transform. Meanwhile, the DCT transforms a signal largely utilizing cosine functions, making it particularly effective at concentrating signal energy into a few coefficients. DCT is frequently used in image and signal processing, especially for JPEG compression Wallace (1991). However, DCT has limitations. It may introduce artifacts, specifically ”blocking artifacts”, when used in high-compression scenarios. Due to its inability to portray crisp transitions or edges, DCT reduces visual detail. Furthermore, its fixed basis functions are not adaptable, which can impede optimal signal representation in a variety of settings.

Both wavelet and Hartley transformation formulas utilize integrals of the original signal multiplied by a kernel (the wavelet function Kaiser (1994) for wavelet transforms and the cas function for Hartley transforms, respectively). The transforms enable practitioners to decompose the frequency or scale components of a signal, thereby revealing its underlying properties. Signal processing requires switching between time (or spatial) and frequency (or scale) domains, making these techniques necessary. Although versatile in storing signals across scales, wavelet transforms have limitations. Choosing the optimal wavelet function necessitates domain-specific knowledge, making the decision difficult. A suboptimal selection can reduce the precision of signal representation. Shift variance and edge effects can also present difficulties. While wavelets excel at capturing transient events, they may fall short when it comes to recording sinusoidal signals or phase information. Due to its real-valued structure and coupled sinusoidal and cosinusoidal functions, the representations provided by Hartley transform are often significantly more robust. Thus, many applications that prioritize computational performance and clarity may benefit more from the Hartley transform than DCT or wavelet transforms.

Combining HNO with Physics Informed Neural Networks. Physics-Informed Neural Networks (PINNs) for nonlinear PDEs are recently explored for various reasons Raissi et al. (2019). Compared to traditional neural networks, PINNs offer better adherence to the laws of physics, efficiency, and versatility. These naturally led to the development of Physics-Informed Neural Operator Networks (PINOs) Li et al. (2023), which amalgamate physics principles into neural operator structures, proving invaluable in areas like fluid dynamics and environmental modeling. The utilization of the Hartley transform in PINO systems has the potential to enhance computational efficiency and strengthen learning mechanisms. A recent study Rosofsky et al. (2023) introduces a framework for integrating partial differential equations (PDEs), essential for understanding and simulating many physical occurrences. The framework includes initial data generation, boundary conditions, and physics-informed neural operators. Their solutions demonstrate accuracy comparable to the literature. A primarily focus of theirs is the one-dimensional Burgers' equation Burgers (1948) - a fundamental partial differential equation that encompasses the interplay between non-linear advection and diffusion, which can be written as:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \tag{5}$$

Here, we apply the Hartley Neural Operator (HNO) to the one-dimensional Burgers' equation for comparison to their work. We also apply the HNO framework to two other PDEs, which are traditionally real-valued.

Operating on the Diffusion Equation, a key equation in Machine Learning. First, the diffusion equation Fick (1855), often also known as the heat equation, describes how heat, concentration, and population density spread through time and space. In Machine Learning applications, PDEs are used in various ways. For example, Laplacian Eigenmaps smooth or interpolate high-dimensional data sets using diffusion equation-like constructions **?**. Label propagation in graph-based semi-supervised learning uses the diffusion equation **?**. Modeling the 'flow' of data points in a feature space helps uncover anomalies and is used in time-series analysis **?**. Understanding stochastic gradient descent and other optimization techniques can be done using diffusion equation concepts **?**. The diffusion equation is also extremely versatile in physics due to its being key to understanding heat transmission between states of matter. In one dimension, this PDE can be written as,

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \tag{6}$$

Applications to PDEs Useful In Weather Forecasting. Second, the thermodynamic energy equation Gill (1982) with advection terms and assuming hydrostatic balance - a fundamental concept that links gravity with vertical pressure gradient force - that simplifies the equation by focusing on vertical forces, which is a good approximation for most meteorological uses. For our choice of diabatic heating term, we chose a linear function in T to simplify matters.

$$\frac{\partial T}{\partial t} + w\frac{\partial T}{\partial z} = Q_0 T + Q_1 \tag{7}$$

Defining Losses. We primarily the methodology used in setting up FNO architectures to our losses:

1. $\mathcal{L}_{L2}$: This loss aligns the model predictions with the training data. It's calculated as the relative mean squared error (MSE) between the training data and the model outputs, normalized by the norm of the actual values.
2. $\mathcal{L}_{reconstruction}$: This loss ensures the model's adherence to known physical laws, like PDEs or conservation laws. Defined as the MSE between the breach of the physical principle and its ideal value, which here for practicality is the MSE between the residuals and zero, indicating the physics fit exactly.

Finally, define the Total Loss function as the weighted sum of the loss functions described above:

$$w_{L2} \times \mathcal{L}_{L2} + w_{reconstruction} \times \mathcal{L}_{reconstruction} \tag{8}$$

Unlike the previous study, we ignore initial condition losses (since their GitHub implementation has this commented out in their code anyway). We use unity weights for these runs. Both their studies and ours use experimental weight selection for the loss function rather than hyperparameter optimization, though our choice of unity is more arbitrary than anything.

Using GRF For Initial Conditions. In the construction of initial conditions, similar to the aforementioned study, Gaussian Random Fields (GRFs) are utilized Gudder (1978). GRFs are a type of spatially correlated random processes that have played a crucial role in generating realistic initial conditions for diverse simulations, particularly in the fields of geostatistics and cosmology. The Matérn function Matérn (1986) is widely used as a covariance function in Gaussian random fields (GRFs) because of its flexibility to represent various degrees of smoothness in the data. Through the utilization of the Matérn function within a Gaussian Random Field (GRF) Tilmann Gneiting & Schlather (2010), it is possible to create spatially consistent formations that possess distinct attributes dictated by the parameters of the function. The Matérn function is fundamentally responsible for introducing a predetermined degree of smoothness and correlation length to the resulting field, hence facilitating a deliberate synthesis of spatial data. When employed for the purpose of establishing initial conditions, this methodology guarantees that the resulting data exhibits statistical qualities that closely correspond to real-world observations.

Defining Hartley Transforms using PyTorch Fast Fourier Transforms. Our investigation required Fourier transforms to depict the Hartley transform since PyTorch lacks native Hartley Transform capabilities. The Hartley transform (H(u)) is closely related to the Fourier transform (F(u)): $H(u) = \mathrm{Re}[F(u)] - \mathrm{Im}[F(u)]$, where Re and Im represent the real and imaginary components of the Fourier transform. The equation was crucial to our analytical and computational work. We used PyTorch tools to adapt a Numpy version Panizza (2021) and enhance its multidimensional functionality.

3 EXPERIMENTS

Setting Up Our Neural Network Architecture to Parallel Recent Work for Comparison. Our research followed the same structure as Rosofsky et al. (2023). The model is characterized by its widths, modes, and total layers, with a strong emphasis on Gaussian error linear units (GELUs) as the activation function Hendrycks & Gimpel (2023). GELUs are preferred in physics-informed deep learning because they have a non-zero second derivative, making ReLUs less suited. In our 1-D configurations, the FNO structure includes four layers with widths [16, 24, 24, 32] and modes

	$\mathcal{L}_{L2}$	$\mathcal{L}_{reconstruction}$	$\mathcal{L}_{TOTAL}$
HNO	0.6099	0.3726	0.9825
2 HNO, 2 FNO	0.0292	0.0255	0.0547
1 HNO, 3 FNO	0.0217	0.0169	0.0386
FNO	0.0240	0.0255	0.0495

Table 1: Comparison of L2 and reconstruction losses between architectures using combinations of the standard Fourier Neural Operator method (FNO) and our Hartley Neural Operator (HNO) substitution.

[15, 12, 9, 9]. The design produces a 128-width, completely linked layer. We optimized our model on PyTorch using the Adam method with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, across 500 epochs, with an initial learning rate of 0.001.

Is it beneficial to use Swish as an Activation Function in PINO? Strategic use of GELUs combines ReLU with sigmoid functions to provide non-linearity and handle the vanishing gradient issue in PDE situations. However, another important activation function in this area is Swish, created by Google Brain Ramachandran et al. (2017), which combines input and sigmoid. Swish overcomes constraints via ReLU and LeakyReLU. Both GELUs and Swish produce smoother function outputs, although Swish's simple architecture may be more computationally efficient. Their performance depends on the application. GELUs are recommended for physics-informed jobs in Fourier Neural Operator frameworks, and Swish is versatile. Problem specifics, computational limits, and empirical outcomes determine the final choice.

Using RK4 with Finite Difference to Obtain Exact Solutions. The Runge-Kutta order 4 (RK4) method, a popular numerical integration method, was used to solve partial differential equations in our work (just as Rosofsky et al. (2023) did in their study) in order to have ”ground truth” for what the exact solutions to the PDEs were to compare the predictions of the neural operator networks we employed. The complexity of PDEs, which incorporate several independent variables like time and space, required a two-pronged approach: We first used finite difference methods to discretize the spatial components of the PDE, turning it into a time-dependent ODE system. The RK4 approach then was used to integrate the system across time after spatial discretization. Given a 1-dimensional spatial domain, we set the domain boundaries as $xmin = 0$ and $xmax = 1$. The domain is discretized into $Nx = 100$ spatial grid points, which provides the spatial resolution of the simulation. The time step size for advancing the solution is selected as $dt = 1 \times 10^{-3}$, which determines the temporal resolution and influences the stability of the algorithm. The simulation is run until an end time $tend = 1.0$, where we choose these parameters to ensure both stability and efficiency of the numerical method.

4 Results

A Hybrid HNO/FNO Approach Gives the Best Results for Burgers'. The first experiment, comparisons of the loss values for different architectures of PINO run to solve the one dimensional Burgers' equation is shown in Table 1. The outcomes of the losses related to several variations of PINO, which incorporate varying combinations of Fourier and Hartley layers in their 4-layer architecture, are presented in Table 1. The table provides a systematic comparison of the effectiveness of these configurations, bringing valuable insights into the influence of various layer combinations on the performance of the network. Each row in the table represents a distinct variation of the PINO algorithm, indicating the specific number of Fourier and Hartley layers that have been included. The losses associated with each model, which are potentially presented in the following columns, provide a quantifiable assessment of their correctness or error rate. The Hartley design with four layers has the most unfavorable loss values among the four variations, whereas the architecture including a single Hartley layer followed by three Fourier layers demonstrates the most favorable loss outcomes. Furthermore, as depicted in Figure 1, the comparison between the exact solution (obtained using a fourth-order Runge-Kutta solver) and the solution predicted by the PINOs with partial Hartley and partial Fourier layers exhibits a remarkable level of agreement. Notably, the absolute error remains consistently low as the solution evolves over time.

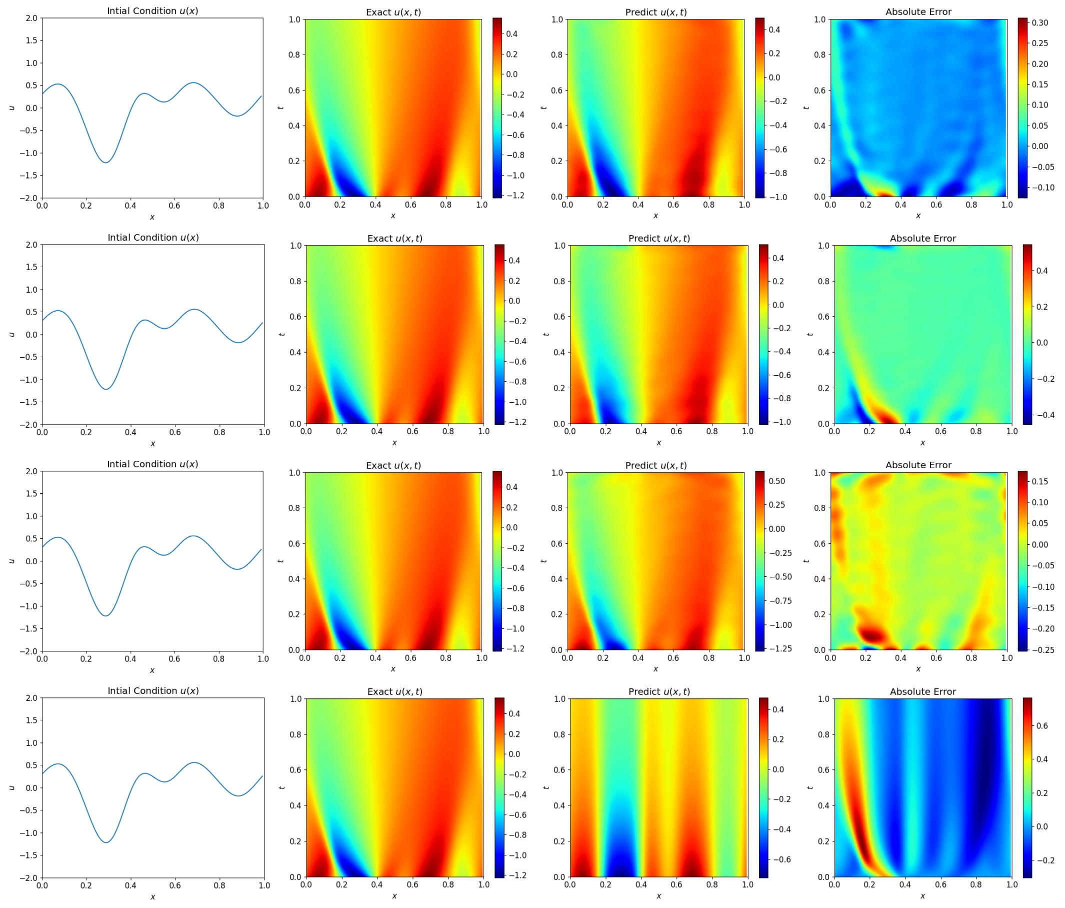

Figure 1: Comparison of Exact vs. Predicted (with absolute error shown) of the 1-D Burger's equation with the given initial condition and periodic boundary conditions using 1.) Top Panel: FNO with 4 layerss; 2.) Middle Top Panel: 1 HNO layer and 3 FNO layers; 2.) Middle Bottom Panel: 2 HNO layers and 2 FNO layers; 4.) Bottom Panel: HNO with 4 layers.

	activation	FNO $\mathcal{L}_{L2}$	FNO $\mathcal{L}_{reconstruction}$	HNO $\mathcal{L}_{L2}$	HNO $\mathcal{L}_{reconstruction}$
Heat/Diffusion Eq	GELU	0.00126	0.00216	0.00385	0.00214
Heat/Diffusion Eq	Swish	0.00116	0.00216	0.00468	0.00214
TDE, Linear Q(T)	GELU	0.00457	0.01085	0.04044	0.01069
TDE, Linear Q(T)	Swish	0.00472	0.01086	0.04047	0.01069

Table 2: Comparison of L2 and Function error between the standard Fourier Neural Operator method (FNO) and our Hartley Neural Operator (HNO) substitution.

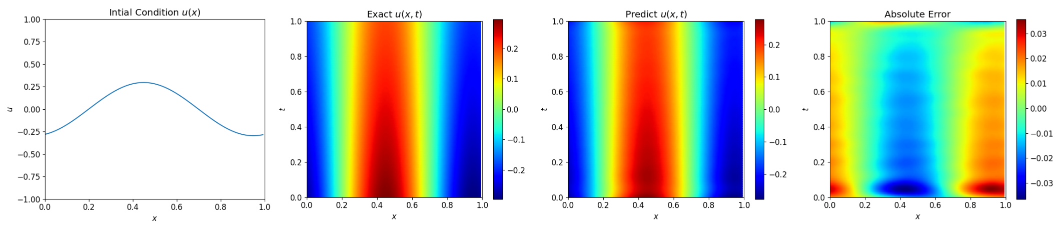

Figure 2: 1-D diffusion equation results with GELU activation function using a 4-layer FNO architecture.

Four-Layer HNO Improves Reconstruction Loss for Solving the Diffusion Equation and TDE. Table 2 displays the comparative reconstruction and L2 losses for the FNO and HNO structures, using GELU and Swish activation functions. One noteworthy finding derived from the table is the comparatively reduced reconstruction loss exhibited by the HNO model. This outcome implies a more advanced level of alignment with the fundamental principles of the Diffusion and TDE equations. Nevertheless, the aforementioned observation is accompanied with an elevation in L2 losses for the HNO, suggesting a somewhat reduced level of agreement with the empirical data.

FNO Causes Periodic Error, HNO Low Random Error in Diffusion. Figures 2 and 3 offer significant insights into the performance of the FNO and HNO designs when employed for the resolution of the Diffusion equation. The evaluation of these structures is conducted under varying initial conditions, utilizing randomized Gaussian Random Field (GRF) fields. In the context of the 4-layer FNO, it is worth mentioning that the discrepancy between the precise and projected observations displays a recurring pattern, a periodicity that diminishes over time, whereas in the case of the 4-layer HNO, the error is comparatively reduced and demonstrates a greater degree of randomness as the solution progresses through time.

For TDE with Linear Q(T) FNO Again Causes Periodic Error, while HNO's Error is Non-periodic, Both Increase as the Solution Evolves in Time. In addressing the TDE equation, as illustrated in Figure 4 and Figure 5, we offer a thorough perspective on the outcomes obtained through the inclusion of a linear diabatic heating component and the utilization of identical randomized Gaussian random field (GRF) beginning conditions. It is noteworthy that the absolute error between the exact and predicted observations exhibits a similar periodicity when the 4-layer FNO

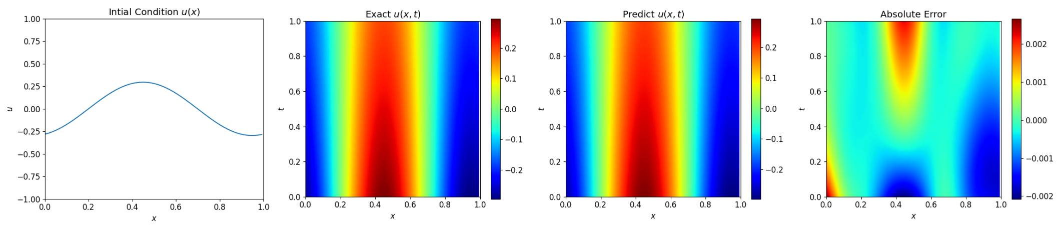

Figure 3: 1-D diffusion equation results with GELU activation function using a 4-layer HNO architecture.

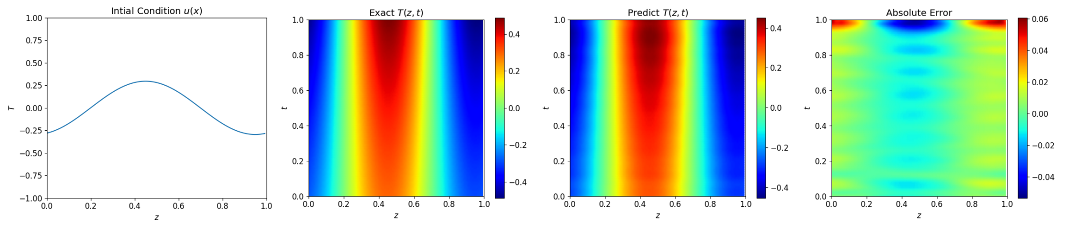

Figure 4: 1-D TDE equation, with linear Q results with GELU activation function using a 4-layer FNO architecture.

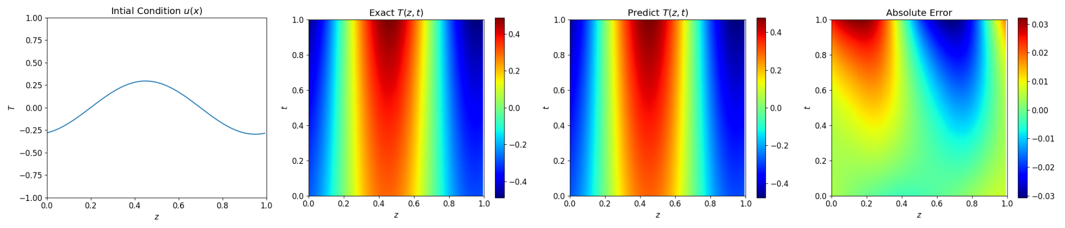

Figure 5: 1-D TDE equation, with linear Q results with GELU activation function using a 4-layer HNO architecture.

architecture is employed, akin to the case of the diffusion equation. However, this periodicity is not as pronounced as the solution progresses in time. Conversely, when the 4-layer HNO is utilized, the absolute error displays less periodicity than the FNO case, is an order of magnitude less than the FNO case, and displays less randomness than the HNO case in the 1-D diffusion equation. In fact, the error initially remains relatively low, but as the solution evolves over time, the error begins to increase.

No Appreciable Difference in Activation Function Choice. As shown in table 2, there was no appreciable difference in the overall neural network performance when using Swish over GELU, despite being self-gating and having been shown to work better for certain neural network architectures designed to solve PDEs.

5 Discussion

We undertook a comprehensive adaptation of the PINO framework, originally proposed by Rosofsky et al. (2022). Our study's chief aim was to explore the integration of Hartley Neural Operators (HNO) within this structure. Upon examining the Burgers' equation, we found that using HNO resulted in a heightened overall loss compared to its FNO equivalent. Notably, a combined approach employing both HNO and FNO layers displayed superior performance and a much better fit to the exact solution, then Rosofsky et al. (2023). Further, our exploration of the Diffusion equation and the TDE equation, using a linear depiction of the diabatic heating component, revealed subtle differences between the two neural operators. The HNO showed a lesser reconstructive loss compared to the FNO, but there was a minor rise in L2 loss for both.

When analyzing the time-dependent evolution of absolute errors in the diffusion equation, the FNO four-layer architecture revealed a distinct periodic error trend, whereas the HNO four-layer architecture revealed a more random error trajectory, albeit with an order of magnitude lower discrepancy. The observed periodicity in FNO error may suggest that the FNO architecture fails to account for specific data characteristics, such as boundary conditions or periodic patterns. The presence of oscillatory error patterns in the Partial Differential Equation (PDE) suggests that the FNO architecture may be responsive to specific solution frequencies. These results may be ascribed to resonance phenomena, in which specific frequencies of the partial differential equation's solutions are either amplified or attenuated.

An analogous discrepancy was noted in the context of the TDE PDE with regard to the FNO. The HNO four-layer architecture initially matched the correct solution, but with time, prediction and ground truth diverged. The model's inability to effectively describe the underlying dynamics of the partial differential equation (PDE), especially over longer time periods, may explain these disparities. The observed divergence and larger L2 loss values may be due to overfitting, numerical instability, temporal discretization issues, boundary condition errors, or TDE complications with the selection of a linear diabatic heating term. In our scenario, where a randomized Gaussian random field (GRF) with a Matérn covariancThat said, however, the HNO still outperformed the FNO in the TDE solution prediction.

Finally, the choice of activation function between GELU and Swish did not lead to any substantial differences between either the HNO or FNO four-layer architectures and was not applied to the hybrid architecture. That said since we were working strictly in one-dimension and using simplified versions of the PDEs under investigation, solving the fully-fledged PDEs in three-dimensions may still benefit from such a choice in activation function.

6 Conclusion & Future Work

In conclusion, our research compellingly underscores the profound implications of melding real-valued transforms, with particular emphasis on the Hartley Transform, into the architecture of neural operators when grappling with streamlined primitive equations. By embarking on this innovative path, we have illuminated a pioneering methodology adept at decoding pivotal meteorological PDEs, such as the Thermodynamic Heat Equation. The merger of the Hartley Transform with these equations not only allows for a more nuanced understanding of atmospheric dynamics but also bestows upon our models an enhanced level of fidelity, ensuring predictions that resonate closely with real-world weather phenomena. The juxtaposition of our findings from the 1-D Diffusion and Thermodynamic Energy Equations with the Burgers' equation offers a holistic perspective. The comparative results accentuate that domain-specific PDEs, especially those inherently real-valued in their solutions, align more harmoniously with real-valued transforms like the Hartley. In stark contrast, equations like the Burgers', which might exhibit complex components or behaviors in their solutions, reveal the nuances and potential limitations of a purely real-valued approach.

It is also clear that additional ablation experiments that isolate and analyze specific neural operator elements or arrangements within the model architecture can help us understand their effects. An in-depth investigation of the training process, particularly the periodic error patterns, may be beneficial. In a similar manner, boundary conditions, temporal discretizations, and brain architecture may explain model efficacy differences. The quality and representativeness of the training data must also be closely examined, taking into account long-term discrepancies. Different activation functions, dropout rates, and batch normalization could also be examined in the neural operator's structure. The main goal of this ablation research is to identify specific sources of absolute error and high L2 loss while improving the model's design and training methods to reduce reconstruction loss and better fit the exact, numerical solutions.

Future research should aim to further explore the PINO framework, particularly in three dimensions. Further expanding the utilization to a wider range of intricate partial differential equations (PDEs), with particular emphasis on the extensive collection of fundamental equations that are vital to numerical weather prediction is also of interest. It would be beneficial, therefore, to employ a Hartley Neural Operator network to process weather model data with varying resolutions. For example, in recent studies, it has been demonstrated that Deep Learning models, such as FourCastNet Pathak et al. (2022), exhibit efficacy as viable options for NWP with the utilization of the Adaptive Fourier Neural Operator (AFNO) attention mechanism, have the capability to make accurate forecasts with low-resolution data with up to a maximum of seven days in advance. Future research, which is already ongoing, will strive to answer the question of whether or not can it do better if we apply the HNO framework modifications and create an AHNO attention mechanism instead.

As the research landscape continues to evolve, we anticipate that additional applications of the Hartley transform within PINOs specifically, and Neural Operator learning in general will materialize, contributing innovative resolutions to a diverse set of scientific and engineering challenges.

REFERENCES

N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

Ali N Akansu and Richard A Haddad. *Multiresolution signal decomposition: transforms, subbands, and wavelets*. Academic Press Professional, Inc., 1992.

A. B. Arenas and T. Serrano-Gotarredona. Neural networks based on the hartley transform. *Electronics Letters*, 32(21):1962–1963, 1996.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

Kaushik Bhattacharya et al. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2021.

R. N. Bracewell. Discrete hartley transform. *J. Opt. Soc. Am.*, 73(12):1832–1835, Dec 1983. doi: 10.1364/JOSA.73.001832. URL `https://opg.optica.org/abstract.cfm?URI=josa-73-12-1832`.

R. N. Bracewell. *The Hartley transform*. Oxford University Press, USA, 1984.

J.M. Burgers. A mathematical model illustrating the theory of turbulence. volume 1 of *Advances in Applied Mechanics*, pp. 171–199. Elsevier, 1948. doi: https://doi.org/10.1016/S0065-2156(08)70100-5. URL `https://www.sciencedirect.com/science/article/pii/S0065215608701005`.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

L. Debnath. *Wavelet transforms and their applications*. Birkhäuser Boston, 2007.

Adolf Fick. Ueber diffusion. *Annalen der Physik und Chemie*, 170(1):59–86, 1855. doi: 10.1002/andp.18551700105. URL `https://doi.org/10.1002/andp.18551700105`.

A.E. Gill. *Atmosphere-Ocean Dynamics*. Number v. 30 in Atmosphere-ocean Dynamics. Elsevier Science, 1982. ISBN 9780122835223. URL `https://books.google.com/books?id=1WLNX_lfRp8C`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT Press, 2016.

Alex Grossmann and Jean Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM journal on mathematical analysis*, 15(4):723–736.

S. P. Gudder. Gaussian random fields. *Foundations of Physics*, 8(3-4):295–302, April 1978. doi: 10.1007/bf00715214. URL `https://doi.org/10.1007/bf00715214`.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

R.V.L. Hartley. A more symmetrical fourier analysis applied to transmission problems. *Proceedings of the IRE*, 30(3):144–150, 1942. doi: 10.1109/JRPROC.1942.234333.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

Gerald Kaiser. *A friendly guide to wavelets*. Birkhäuser, 1994.

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, May 2021. doi: 10.1038/s42254-021-00314-5. URL `https://doi.org/10.1038/s42254-021-00314-5`.

G. Kutyniok, P. Petersen, and F. Raslan. A theoretical analysis of deep neural networks and parametric pdes. *arXiv preprint arXiv:1903.00358*, 2019.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

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.

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, 2023.

Z. Long, Y. Lu, X. Ma, and B. Dong. Pde-net: Learning pdes from data. *arXiv preprint arXiv:1710.09668*, 2020.

L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2019.

Bertil Matérn. *Spatial Variation*. Springer New York, 1986. doi: 10.1007/978-1-4615-7892-5. URL `https://doi.org/10.1007/978-1-4615-7892-5`.

Michael A Osborne, Stephen J Roberts, Alistair McLean, Peter M Brown, and Mark Ebden. Machine learning of linear differential equations using gaussian processes. *Journal of Chemical Physics*, 146(12):124106, 2017.

Andrea Panizza. GitHub - AndreaPi/DHT: A naive Python implementation of the Discrete Hartley Transform — github.com. `https://github.com/AndreaPi/DHT`, 2021. [Accessed 11-08-2023].

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022.

J. Radecki. Hartley image compression. In *Proceedings., International Conference on Image Processing*, volume 1, pp. 457–460. IEEE, 1996.

M. Raissi and G. E. Karniadakis. Hidden fluid mechanics: A navier–stokes informed deep learning framework for assimilating flow visualization data. *Journal of Fluid Mechanics*, 845:202–233, 2018a.

M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018b.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

Maziar Raissi, Paris Perdikaris, and George Em 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.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

K. R. Rao and P. Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 1990.

Lewis Fry Richardson and Peter Lynch. *Weather Prediction by Numerical Process*. Cambridge Mathematical Library. Cambridge University Press, 2 edition, 2007. doi: 10.1017/CBO9780511618291.

Shawn G Rosofsky, Hani Al Majed, and E A Huerta. Applications of physics informed neural operators. *Machine Learning: Science and Technology*, 4(2):025022, may 2023. doi: 10.1088/2632-2153/acd168. URL `https://dx.doi.org/10.1088/2632-2153/acd168`.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven solutions of nonlinear partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

T. Sun, J. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 2(5):356–366, 2020.

T. Sun, C. Bao, and G. E. Karniadakis. Fourier neural operator for high-dimensional pdes. *Proceedings of the National Academy of Sciences*, 118(6), 2021.

William Kleiber Tilmann Gneiting and Martin Schlather. Matérn cross-covariance functions for multivariate random fields. *Journal of the American Statistical Association*, 105(491):1167–1177, 2010. doi: 10.1198/jasa.2010.tm09420. URL `https://doi.org/10.1198/jasa.2010.tm09420`.

Gregory K Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34 (4):30–44, 1991.

Ahmed I Zayed (ed.). *Advances in the theory and applications of the Hartley transform*. World Scientific, 1993.

D. Zhang, Z. Li, C. Bao, and G. E. Karniadakis. Multiscale fourier neural operators. *arXiv preprint arXiv:2107.12283*, 2021.

J. Zhang, Y. Yang, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 2(5):356–366, 2020.

A Appendix

A.1 HNO Skeleton

The Hartley Neural Operator applies the operations in algorithm 1 and then follows the same skeleton as Rosofsky et al. (2023), the code for which can be found at https://github.com/shawnrosofsky/PINO_Applications, except that anywhere there is a Fast Fourier Transform operation taken, we drop in our DHT from Algorithm 1.

A.2 Additional Figures from results

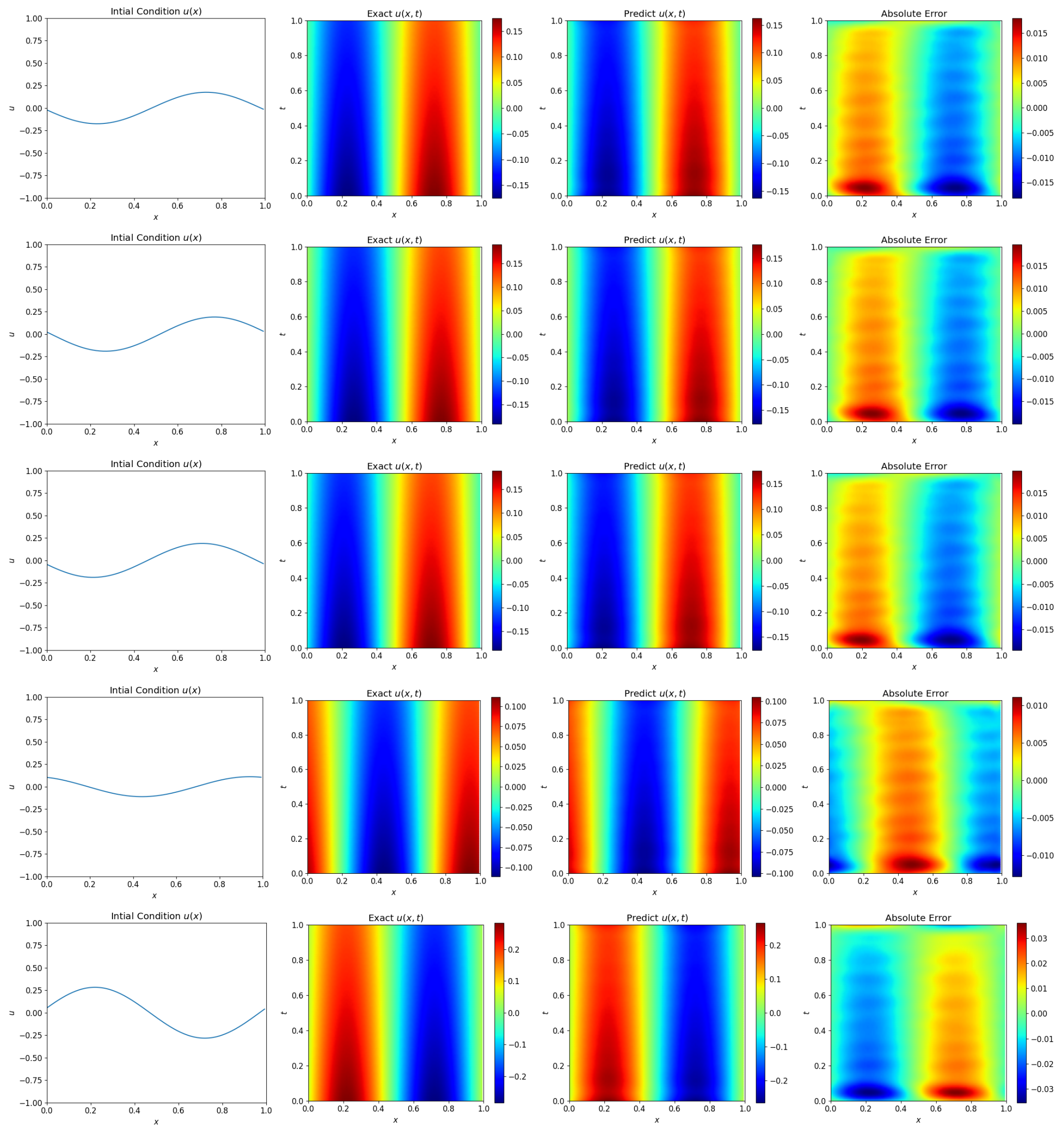

Figure 6: Additional 1-D diffusion equation results with GELU activation function using a 4-layer FNO architecture.

Algorithm 1 Discrete Hartley Transform and Complex Multiplication

```
1:  function DHT($x$)
2:      $X \leftarrow$ Compute FFT of $x$
3:      return Real part of $X$ minus Imaginary part of $X$
4:  end function
5:  function IDHT($X$)
6:      $dims \leftarrow$ Dimensions of $X$
7:      $n \leftarrow$ Product of all $dims$
8:      $X \leftarrow$ DHT($X$)
9:      return $X$ divided by $n$ element-wise
10: end function
11: function COMPL_MUL1D($x, y$)
12:     $X \leftarrow$ DHT($x$)
13:     $Y \leftarrow$ DHT($y$)
14:     $X_{\text{flip}} \leftarrow$ Flip and roll $x$
15:     $Y_{\text{flip}} \leftarrow$ Flip and roll $y$
16:     $Y_{\text{plus}} \leftarrow Y + Y_{\text{flip}}$
17:     $Y_{\text{minus}} \leftarrow Y - Y_{\text{flip}}$
18:     $Z \leftarrow 0.5 \times$ (Element-wise multiply $X$ with $Y_{\text{plus}}$ and $X_{\text{flip}}$ with $Y_{\text{minus}}$)
19:     return IDHT(Z)
20: end function
```

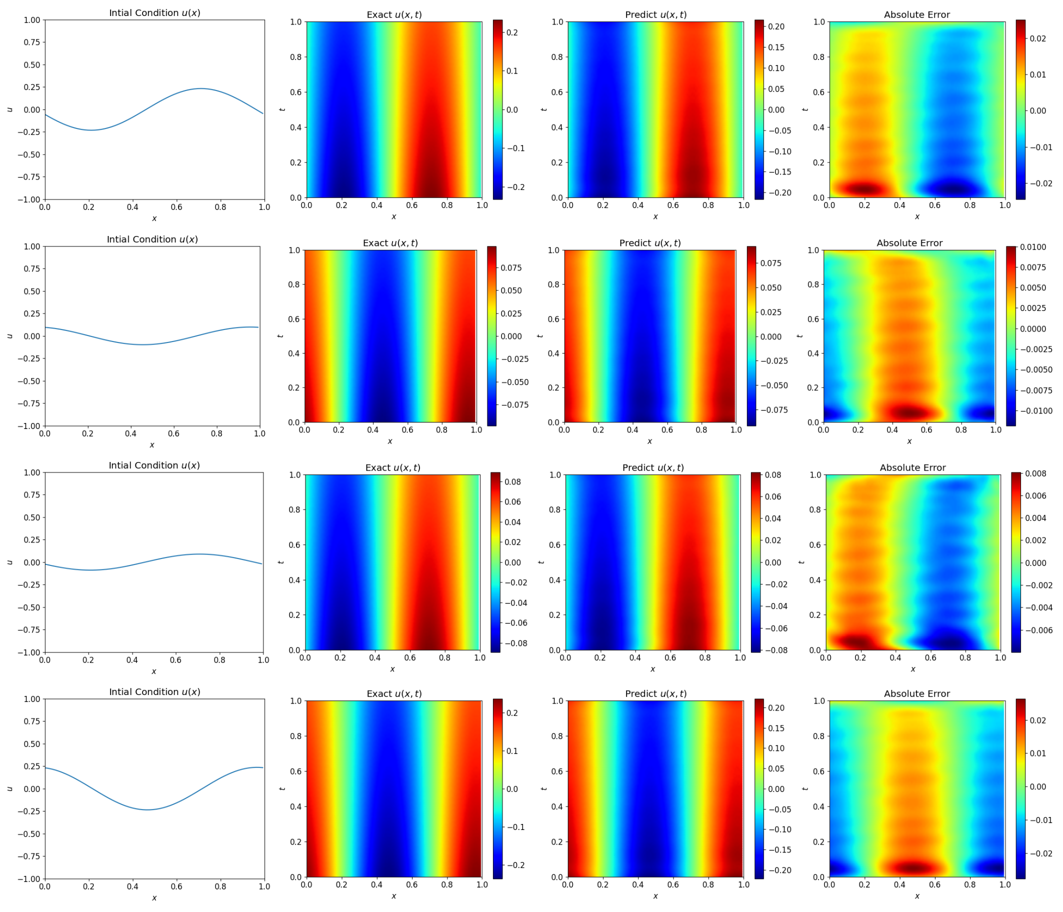

Figure 7: Additional 1-D diffusion equation results with GELU activation function using a 4-layer FNO architecture.

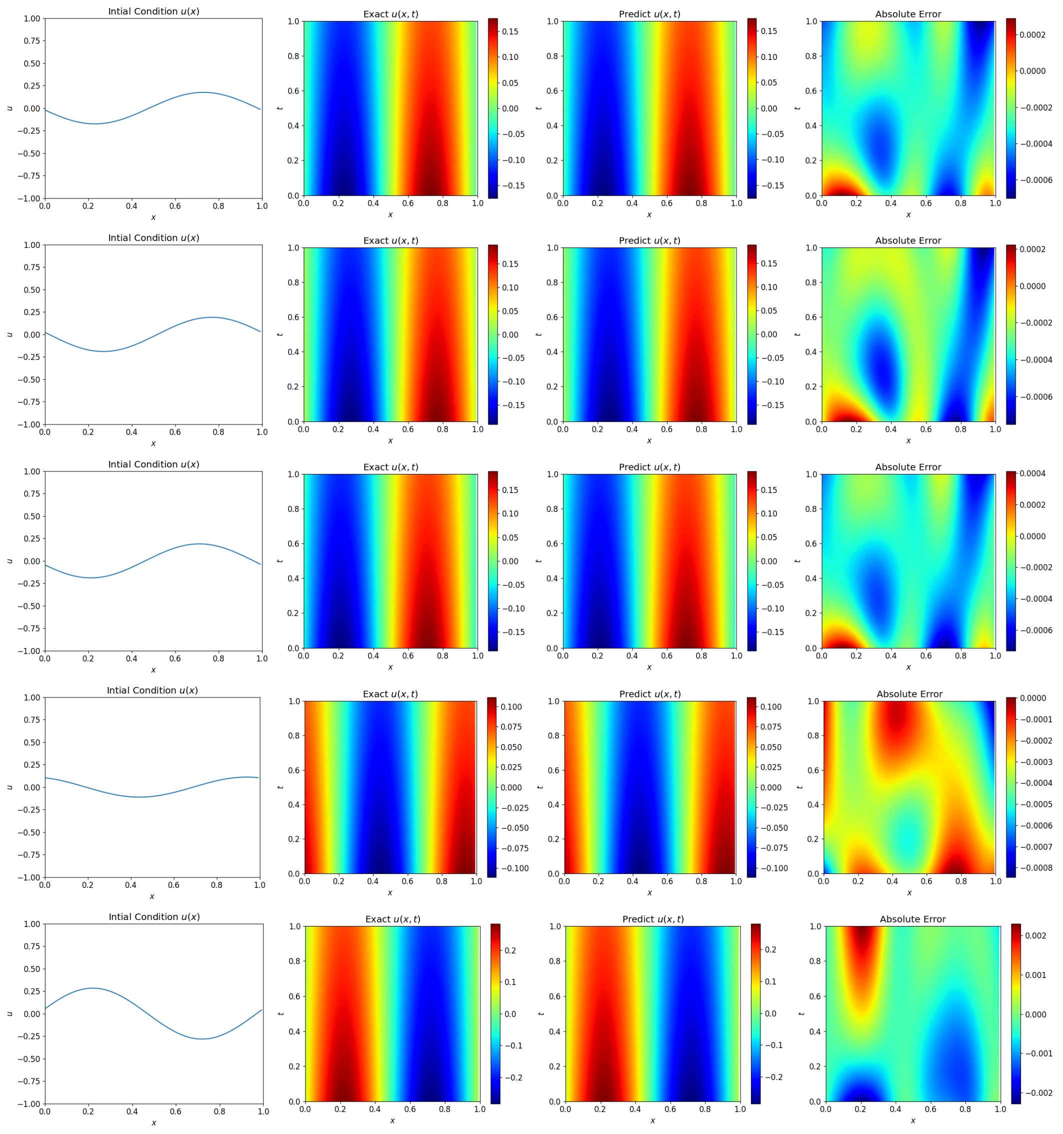

Figure 8: Additional 1-D diffusion equation results with GELU activation function using a 4-layer HNO architecture.

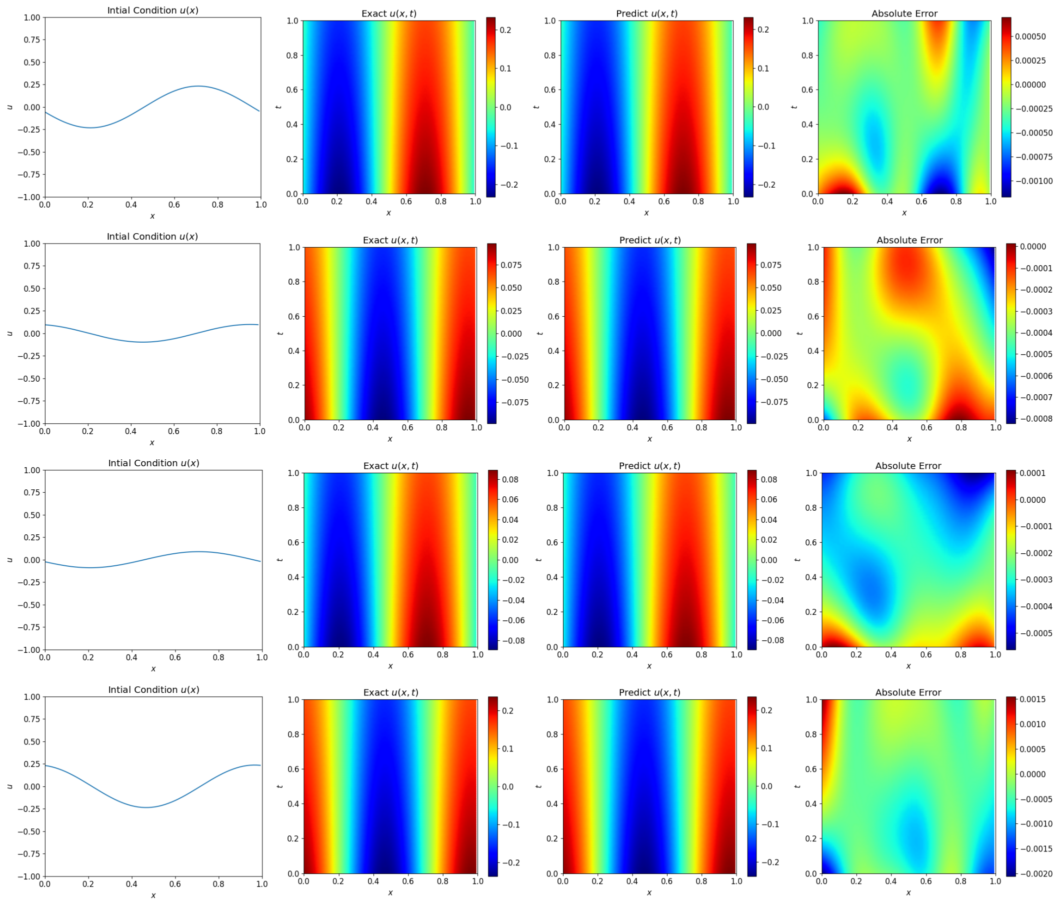

Figure 9: Additional 1-D diffusion equation results with GELU activation function using a 4-layer HNO architecture.

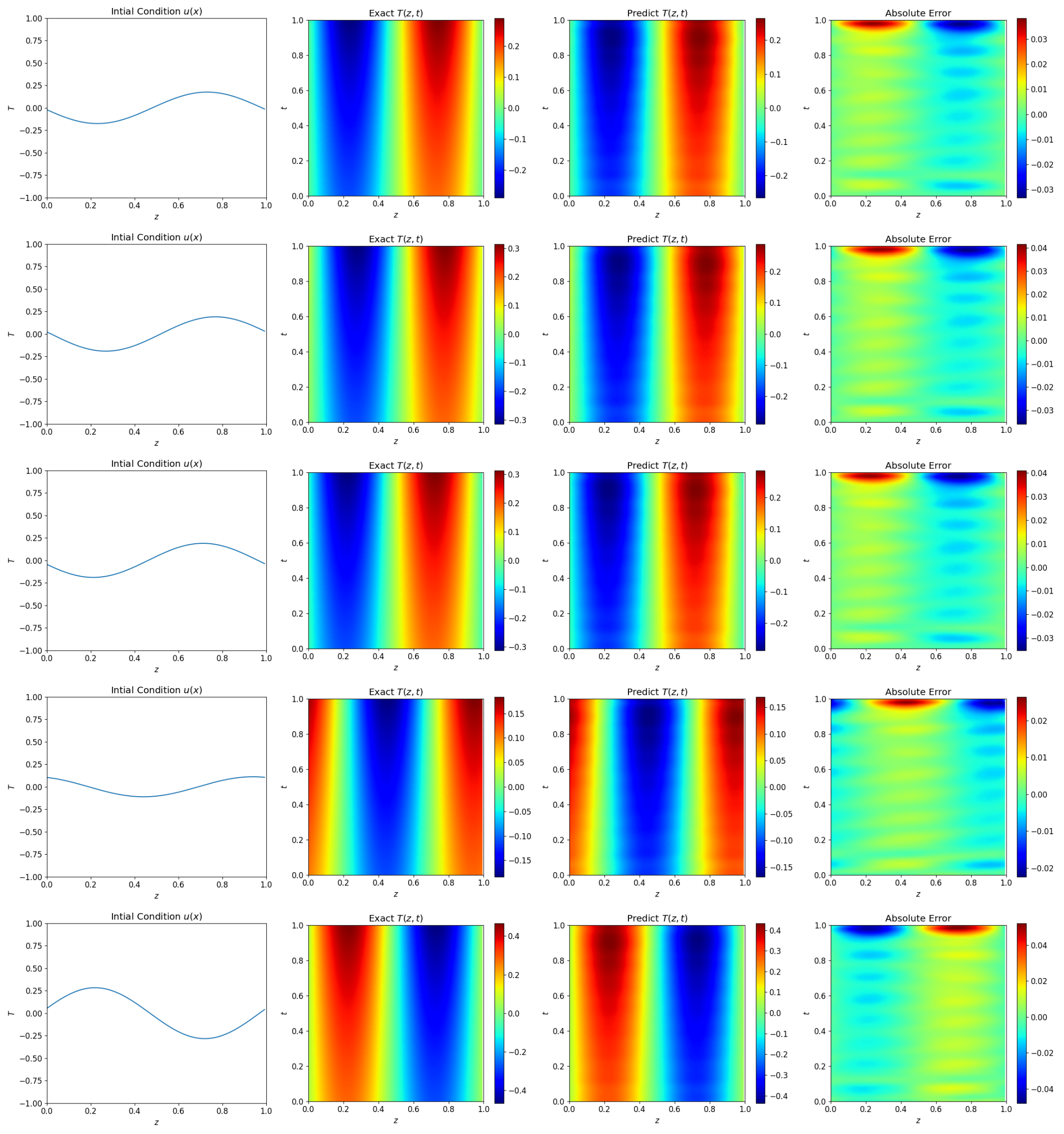

Figure 10: Additional 1-D TDE equation, with linear Q results with GELU activation function using a 4-layer FNO architecture.

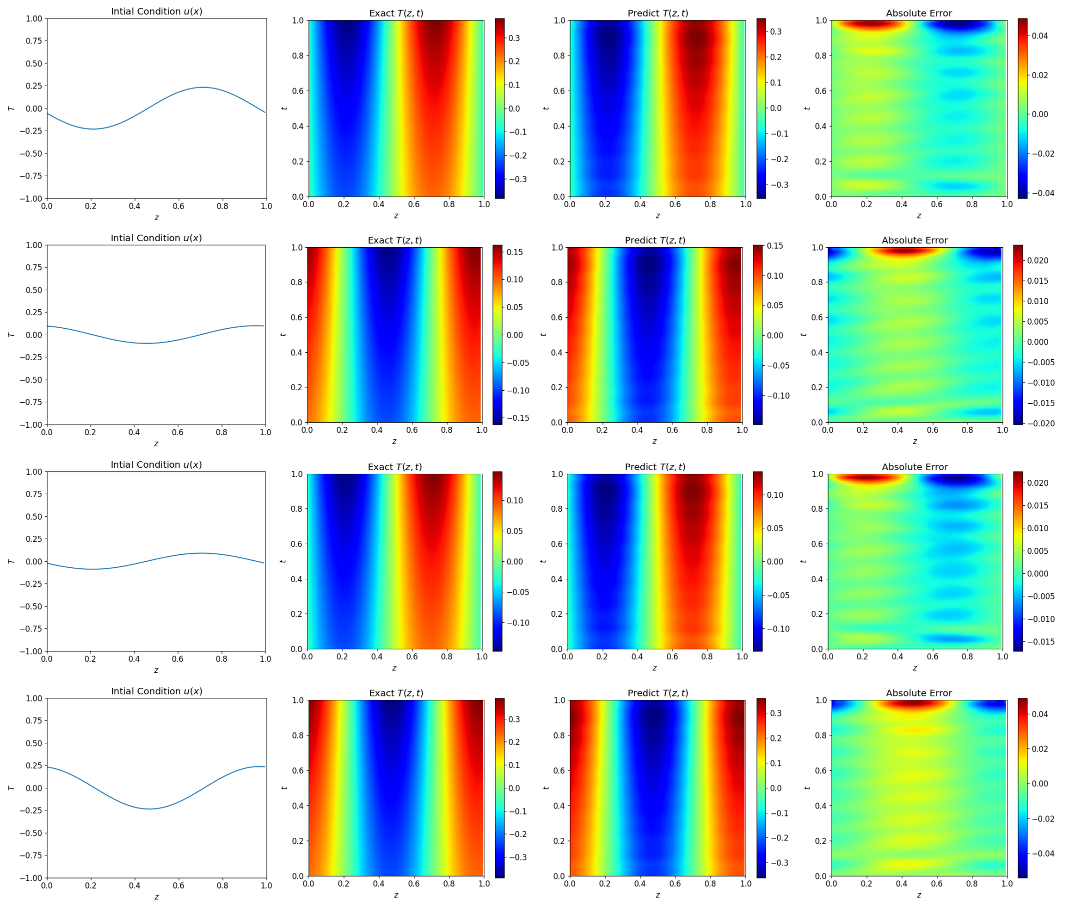

Figure 11: Additional 1-D TDE equation, with linear Q results with GELU activation function using a 4-layer FNO architecture.

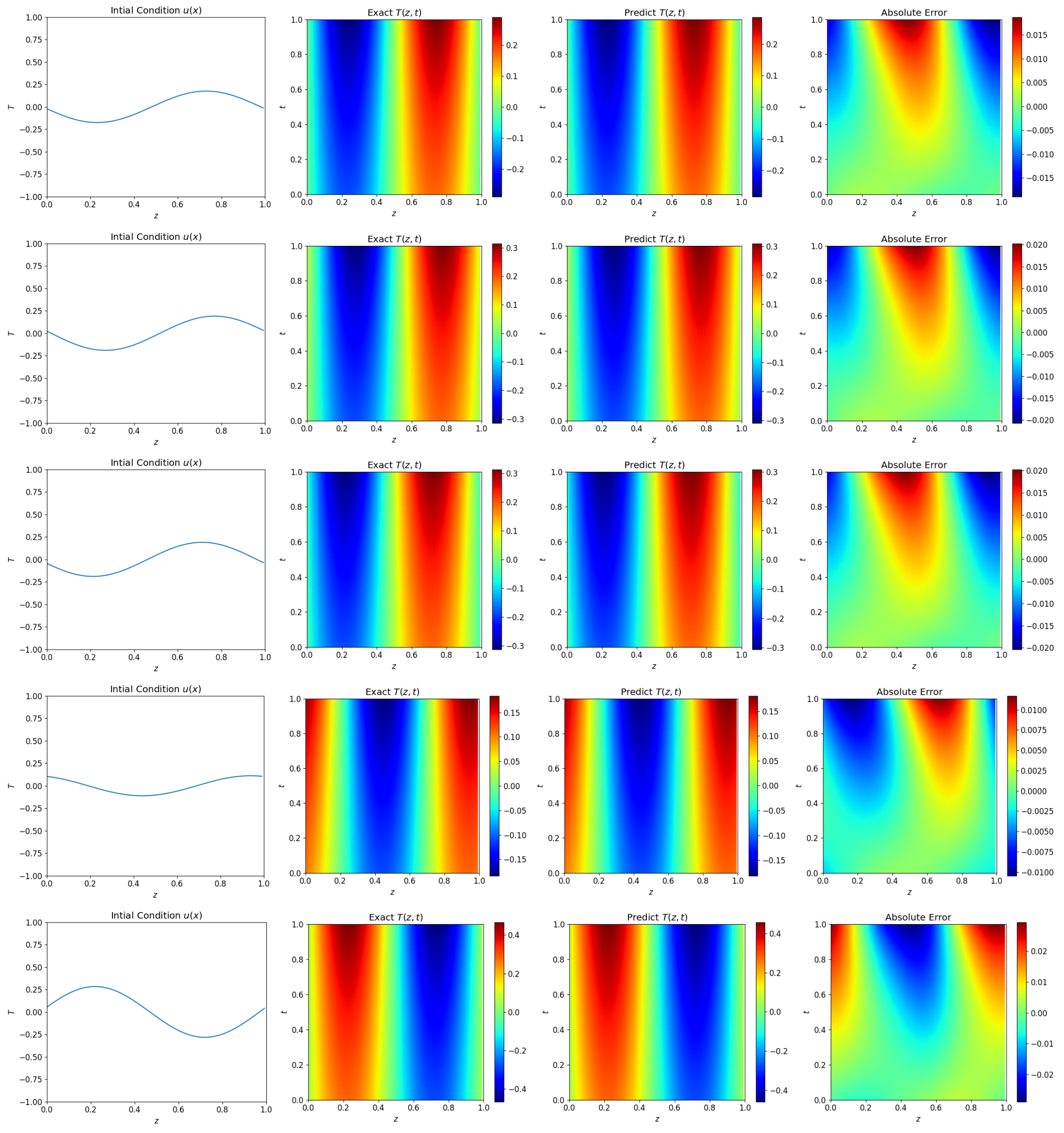

Figure 12: Additional 1-D TDE equation, with linear Q results with GELU activation function using a 4-layer HNO architecture.

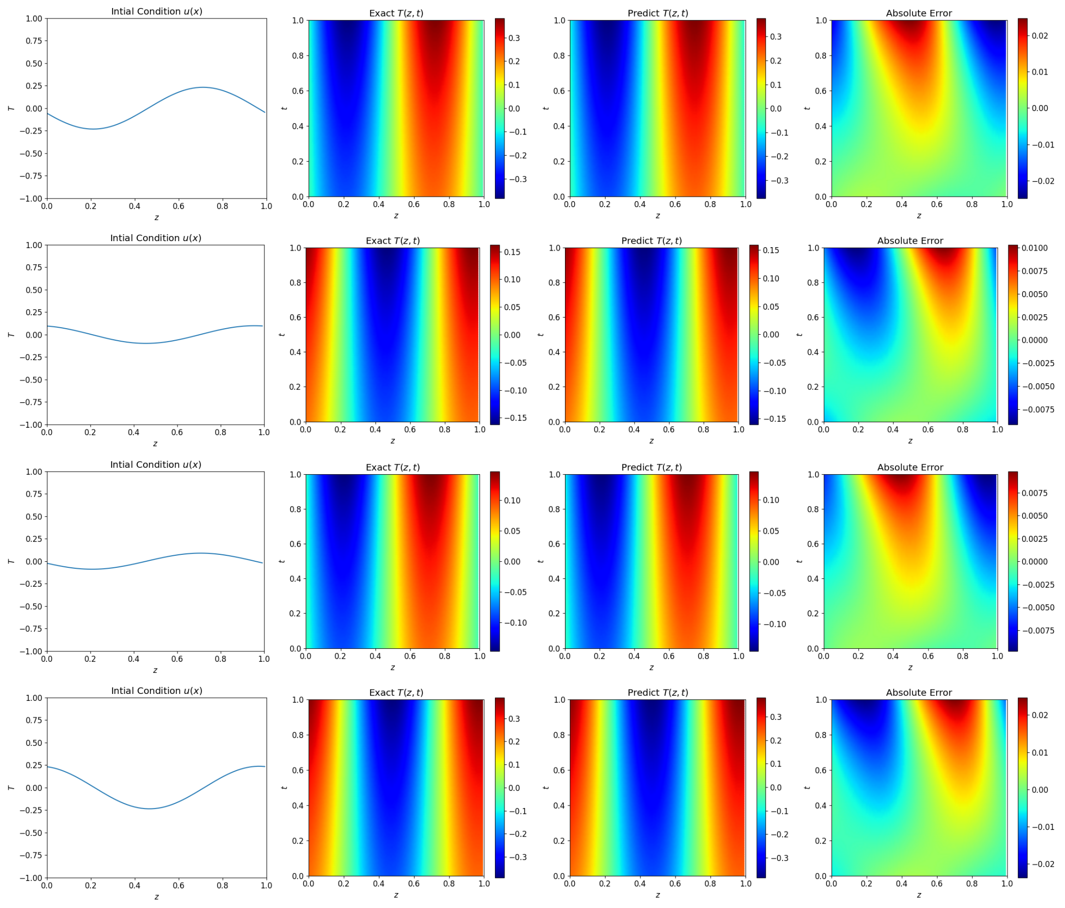

Figure 13: Additional 1-D TDE equation, with linear Q results with GELU activation function using a 4-layer HNO architecture.