Will Neural Operators Replace Numerical Solvers?

Aman Chourasia[§]
[§]*Indian Institute of Technology Patna*
Email: contact@amanchourasia.in

Neural operators are being studied as data-driven methods for approximating solution operators of partial differential equations (PDEs) and reducing the cost of repeated simulation. Recent work has examined data-driven PDE solution, physics informed learning, and neural operator methods for scientific computing problems [1–4]. Their computational value, however, cannot be determined from inference time alone because a learned model may require numerical training data generation, model training, and other computational resources before repeated predictions can be performed. This study compares a Fourier neural operator with conventional numerical solvers for two one dimensional PDEs, namely the viscous Burgers equation and the periodic heat equation, using a common CPU based computational framework. The comparison measures numerical solver time, neural operator training and inference time, fixed preparation cost, peak process memory, relative L^2 error, and cumulative computational cost as a function of the number of requested simulations. For the Burgers equation, the baseline configuration produced a mean numerical solver time of 0.012437 *s per simulation and a mean neural operator inference time of* 0.001221 *s, with an estimated break even point of approximately* 2.98×10^3 *simulations. A larger training configuration reduced the measured test relative L^2 error from approximately* 9.54×10^{-3} *to* 4.92×10^{-3}*, but increased the estimated break-even point to approximately* 8.00×10^3 *simulations. For the heat equation, the numerical solver required* 0.004105 *s per simulation and the neural operator required* 0.000738 *s per inference, while the estimated break even point was approximately* 2.75×10^4 *simulations. The numerical heat equation solution agreed with the analytical solution to approximately* 2.39×10^{-12} *relative L^2 error, whereas the neural operator produced approximately* 3.88×10^{-3} *relative error. These results show that lower neural operator inference cost does not necessarily imply lower total computational cost and that the relative advantage depends on the PDE, preparation cost, accuracy, and number of required simulations. The study therefore evaluates neural operators as workload dependent computational alternatives to numerical solvers rather than as universal replacements.*

1. Introduction

Partial differential equations (PDEs) provide a mathematical basis for describing physical systems across science and engineering, and physics-based computational models use governing equations to support prediction, simulation, and inference for systems that cannot be fully characterized through observation alone [5]. As scientific studies consider larger parameter spaces and increasingly complex physical systems, the computational cost of obtaining solutions can become an important limitation, particularly when high-fidelity models must be evaluated repeatedly. This has motivated the development of data-driven approaches for PDE solution and parameter estimation, including methods that learn solution mappings from previously generated data and methods that incorporate physical information into the learning process [1,2].

Conventional numerical solvers compute solutions for specified mathematical problems through discretization and numerical solution procedures, maintaining an explicit connection between the governing equations and the computed state [5]. When a new problem instance requires different parameters or other prescribed conditions, the numerical solution procedure is generally performed again, so the computational cost increases with the number of required evaluations. This characteristic becomes especially relevant in parameter studies, repeated prediction, optimization, and other many query settings in which a single physical model may need to be evaluated many times. Research in scientific machine learning has therefore investigated whether information obtained from previous computations can be reused to reduce the cost of subsequent PDE evaluations and repeated simulations [1].

Neural operators provide one approach to this problem by learning mappings between functions rather than constructing a separate approximation for every individual problem instance. Recent studies have applied neural operator models to infinite dimensional nonlinear systems and to PDE solution operators that depend on geometry [3,4]. The many query setting is particularly important because Yin et al. consider the computational cost of solving PDEs repeatedly across different geometries and develop a learned solution operator for this purpose [4]. These results demonstrate the potential of operator learning as a means of reusing information across related simulations and reducing repeated computational effort, but they do not establish that a learned approach is universally preferable to direct numerical computation.

The central issue is therefore broader than whether a neural operator can produce an individual prediction faster than a numerical solver. A learned approach introduces an initial computational investment for obtaining training data and optimizing the model, while a conventional solver performs the required numerical computation directly for each problem instance. Kadeethum et al. explicitly distinguish this start up cost from the lower marginal cost of additional learned model evaluations and note that the expected number of model enquiries should be considered before adopting the approach [1]. The relative advantage can consequently depend on the workload, the required accuracy, and the conditions under which the learned model is evaluated in practice.

A second issue concerns the validity of the learned approximation beyond the conditions represented during training. Kadeethum et al. report acceptable behavior when validation data share relevant underlying features with the training data, while performance can deteriorate when those fea-

tures are not shared [1]. Physics informed learning provides another direction by incorporating governing equations into the learning process, and Kharazmi et al. demonstrate this approach for parameter identification and prediction in differential equation models [2]. More broadly, Brunton and Kutz identify solution operator learning and improvements to traditional numerical algorithms as important directions in machine learning for PDEs, while also identifying unresolved challenges in the field and opportunities for investigation [6].

These observations motivate the central question of this study, which is to determine under what computational conditions a neural operator provides a meaningful advantage over a conventional numerical solver for repeated PDE simulation. Rather than treating inference speed as the sole measure of performance, the study compares the complete computational pathways of the competing approaches and examines how their relative behavior changes with simulation count, accuracy requirements, and evaluation conditions. The resulting analysis is intended to distinguish situations in which neural operators provide a practical computational benefit from situations in which conventional numerical methods remain the more appropriate choice, while also considering cases in which the two approaches may be used together.

2. Background and Motivation

2.1. Background

PDE simulation has historically relied on numerical approximations because many physically important equations cannot be solved analytically for realistic domains, coefficients, boundary conditions, or nonlinearities. Finite-difference, finite-element, and spectral techniques have enabled the study of complex systems that would otherwise remain inaccessible, while machine learning has introduced an additional approach based on information obtained from simulation and experiment [6]. Physics-based modeling remains important because governing equations encode physical structure, causality, and generalizable relationships that can extend beyond the data used to construct a model [5].

The emergence of data-driven PDE methods does not replace this numerical framework but changes how computational information may be reused. Kadeethum et al. developed a data-driven framework for learning forward and inverse PDE solution operators from numerical training data, demonstrating that a learned representation can provide substantially lower marginal computational cost after the training stage [1]. Physics-informed approaches provide a related but distinct strategy in which known differential equations contribute directly to the learning problem, as demonstrated by Kharazmi et al. for parameter identification and prediction in epidemiological models [2]. These developments illustrate that learning-based PDE methods can differ substantially in how they obtain information, impose physical structure, and distribute computational effort.

Neural operators occupy a particular position within this landscape because they seek to approximate solution operators between function spaces. Brunton and Kutz describe solution operator learning as one of the major directions through which machine learning is being applied to PDEs, while Yin et al. develop a scalable framework for learning geometry dependent solution operators [4,6]. Pickering et al. similarly demonstrate the use of deep neural operators for approximating infinite dimensional nonlinear operators in an active learning framework [3]. Together, these studies establish the methodological basis for treating neural operators as reusable approximations of families of computational problems rather than models tied to one isolated numerical solution.

Scientific machine learning extends beyond PDE solution operators to a wider range of learned representations for physical and computational systems. Data-driven approaches have been used to infer dynamical mechanisms from incomplete and noisy observations, identify hidden physical state variables, and construct reduced thermodynamic descriptions from microscopic trajectories [7–9]. Other work has investigated the automated discovery of computational algorithms from data and the implementation of learning operators using specialized optical hardware, illustrating broader efforts to reduce computational effort or discover reusable computational representations [10,11]. These studies do not directly establish the performance of neural operators for PDE simulation, but they place operator learning within a broader movement toward data-driven representations of scientific computation.

The same broader development is visible in applications that require repeated computational prediction and integration between physical and computational systems. Research on predictive digital twins has examined computational models that are updated and evaluated throughout the life of an asset, while later work has considered the computational-science requirements, engineering applications, and industrial challenges associated with deploying digital twins at scale [12–15]. Digital-twin methods have also been extended to chemical science through coupled theory, experiment, forward modeling, and inverse inference [16]. These applications reinforce the importance of computational cost, repeated evaluation, model updating, and the relationship between learned representations and established computational models, although they address application domains different from the PDE benchmark considered here.

Learned scientific models have also been developed for atomistic simulation, where the computational objective is to obtain accurate predictions at lower cost than conventional high fidelity calculations. Deep learning interatomic potentials have been used to predict crystallization products across diverse inorganic materials, while chemistry motivated many body representations have been developed to improve the computational efficiency and transferability of machine learning potentials [17,18]. These studies are not direct comparisons between neural operators and PDE solvers, and therefore are not used as evidence for the break even results of the present work. Instead, they provide related examples in which the value of a learned scientific model depends on the balance between model construction cost, prediction cost, accuracy, and the computational workload of the target application. This broader perspective supports evaluating learned scientific models according to their intended computational use rather than prediction speed alone.

2.2. Motivation

The computational structure of the learned approach is fundamentally different from that of a direct numerical solve. A numerical method incurs computational work when a solution is requested, whereas a learned method divides its cost between preparation and subsequent evaluation. The preparation stage can include the generation of numerical training solutions, construction and storage of datasets, preprocessing, parameter optimization, and hardware utilization, while subsequent evaluations may be comparatively inexpensive. Kadeethum et al. provide direct evidence for this distinction and emphasize that the number of expected enquiries is an important consideration because training-data creation represents a major part of the overall preparation cost [1]. The appropriate comparison is therefore a workload-level comparison in which the initial and repeated costs are considered together.

This leads naturally to a break-even perspective. For a small number of requested solutions, the preparation cost of a learned model may represent a substantial fraction of the total computation, whereas for a sufficiently large number of related evaluations, a lower marginal prediction cost may become increasingly important. The same general issue appears in computational workflows in which expensive high-fidelity models must be evaluated repeatedly and computational resources become a limiting factor [12]. The purpose of the present study is therefore not to assign a universal advantage to one class of method, but to determine how the balance changes as the computational workload changes.

Accuracy introduces a second dimension to this comparison. A learned model must represent the relevant solution behavior from finite training information, and its performance can depend on whether evaluation cases remain within the characteristics represented during training. Kadeethum et al. explicitly observe degradation for validation data that do not share underlying features with the training data [1]. Physics-informed learning addresses part of this issue by incorporating governing equations into the learning formulation, while broader PDE machine-learning research continues to investigate representations and numerical strategies that improve robustness and generalization [2,6]. Consequently, a meaningful comparison must consider not only computational savings but also whether the resulting approximation remains suitable for the intended range of problem conditions.

The evaluation framework adopted in this study follows from these computational differences. The analysis separates costs that occur before deployment from costs incurred during repeated use and considers the numerical and learned approaches under comparable problem specifications. It also distinguishes computational cost from predictive quality so that a faster method is not treated as preferable when its error or generalization is inadequate for the intended application. Hybrid formulations are included because the broader scientific-machine-learning literature considers combinations of learned and physics-based components rather than requiring one approach to replace the other [5,6].

The resulting framework is designed to answer a practical computational question. Given a PDE family, a required accuracy, and a specified number of simulations, which computational strategy provides the most appropriate balance of cost and solution quality? This formulation makes the simulation count, preparation burden, repeated use cost, and predictive validity explicit parts of the comparison, providing the basis for the quantitative evaluation developed in the following sections. It also allows the computational tradeoffs to be assessed consistently across the evaluated cases.

3. Methods

3.1. Research Design

This study uses a controlled comparative framework to evaluate a Fourier neural operator (FNO) and conventional numerical solvers for repeated partial differential equation (PDE) simulation. Two one dimensional PDEs are considered, namely the viscous Burgers equation and the periodic heat equation. The comparison separates the computational cost required to construct and train the neural operator from the cost of subsequent inference and compares these quantities directly with the repeated cost of direct solution across tested workloads.

For each PDE, numerical solutions are generated for prescribed initial conditions using a pseudo spectral numerical solver with fourth order Runge–Kutta time integration. A subset of these numerical solutions is used as training data for the FNO, while separate validation and test sets are retained for model selection and final evaluation. The numerical solver is also evaluated independently on held out test cases so that the marginal cost of direct numerical solution can be compared with the marginal cost of neural operator inference.

3.2. PDE Test Problems

The first problem is the 1D viscous Burgers equation,

$$u_t + uu_x = \nu u_{xx}, \tag{1}$$

defined on the periodic domain $x \in [0, 1)$. The viscosity is set to $\nu = 0.01$. The baseline experiment uses 64 spatial grid points, a time step of 0.001, and 100 time steps. A second Burgers configuration uses 128 spatial grid points while retaining the same viscosity and temporal discretization.

The second problem is the 1D periodic heat equation,

$$u_t = \alpha u_{xx}, \tag{2}$$

with $\alpha = 0.01$ and final time $t = 0.1$. The primary heat equation experiment uses 64 spatial grid points and a time step of 0.001, corresponding to 100 time steps. Because the periodic heat equation admits an analytical Fourier mode solution, the numerical solution and neural operator prediction can also be evaluated against an independent analytical reference.

3.3. Numerical Solver

The numerical reference solutions are obtained using a pseudo-spectral method. Spatial derivatives are evaluated in Fourier space using the discrete Fourier transform, while temporal integration is performed using the classical fourth-order Runge–Kutta method. For the Burgers equation, the nonlinear term is evaluated in physical space and Fourier components above the selected de-aliasing cutoff are removed before transforming the nonlinear contribution back to physical space.

The solver advances each initial condition through the prescribed number of time steps. Solver time is measured independently for each problem instance using wall clock measurements. The mean numerical solver time is computed from the held out test cases and is subsequently used in the cumulative cost analysis. This measurement isolates the repeated computational cost of direct numerical solution and enables a consistent comparison with neural operator inference.

For the heat equation, the numerical solution is additionally compared with the analytical Fourier solution. This provides an independent verification of the numerical discretization and prevents the numerical solution from being treated as an exact reference without verification.

3.4. Neural Operator Model

The learned model is a one-dimensional Fourier neural operator. The model receives the initial spatial field as input and predicts the corresponding final solution field. A spatial coordinate channel is included with the input representation.

The FNO contains an initial linear lifting layer followed by four spectral convolution blocks with pointwise convolutional terms. The baseline and larger experiments use 12 Fourier modes and a channel width of 32. The lifted representation is processed through the spectral layers and then projected back to the scalar solution field through two fully connected layers with a GELU activation between them.

The same model configuration is used for the baseline Burgers, larger Burgers, and heat equation experiments unless otherwise stated. The neural operator is trained only using the designated training set, while the validation set is used to monitor predictive performance during training. The test set is not used for model selection or parameter tuning, preserving its role as an independent evaluation set and limiting bias.

3.5. Training Data and Dataset Splits

Training data are generated from the numerical solver by applying independently generated smooth periodic initial conditions to the selected PDE. The baseline Burgers experiment uses 256 training samples, 64 validation samples, and 64 test samples. The larger Burgers experiment and the heat equation experiment use 512 training samples, 128 validation samples, and 128 test samples. These datasets are generated separately to maintain a clear evaluation protocol.

The same random seed is used for each benchmark configuration to maintain reproducibility. The training, validation, and test data are generated as separate datasets so that the final evaluation is performed on cases not used for parameter optimization. This separation prevents information from the test set from influencing model selection and provides a consistent basis for comparing predictive accuracy across the evaluated configurations. It also supports a transparent assessment of model performance across configurations.

3.6. Training Procedure

The FNO is trained on the CPU using the Adam optimization algorithm. The baseline Burgers configuration uses 30 training epochs, while the larger Burgers configuration and the heat-equation experiment use 50 epochs. The batch size is 16 and the initial learning rate is 2×10^{-3}. A stepwise learning-rate schedule is applied during training.

The model with the lowest validation relative L^2 error is retained for the final test evaluation. Training time is measured independently from numerical-data generation and neural-operator inference. No test-set information is used to modify the model after training.

3.7. Computational Environment

All computations are performed on the same Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz system. The benchmark environment reports 64 logical CPUs, with PyTorch configured to use 32 CPU threads. The software environment uses Python 3.11.9, PyTorch 2.3.1 with CPU support, and NumPy 1.26.4.

Peak process memory is recorded during the complete benchmark workflow. The reported timing measurements therefore correspond to the actual CPU environment used for the experiments rather than hardware independent estimates.

3.8. Evaluation Metrics

Computational performance is evaluated using wall clock time for numerical solution, neural operator training, and neural operator inference. The fixed neural operator cost includes the numerical generation of the training and validation data together with the FNO training time. Peak process resident memory is also recorded throughout the complete benchmark.

Solution quality is measured using the relative L^2 error between the predicted solution and the corresponding numerical reference. For the heat equation, the numerical solution and the FNO prediction are additionally compared with the analytical solution. The numerical solver error relative to the analytical solution provides an independent measure of the accuracy of the reference computation.

3.9. Cumulative Cost Model

The comparison treats the number of requested simulations as an explicit variable. For the neural-operator workflow, the cumulative computational cost is written as

$$C_{\mathrm{NO}}(N) = C_{\mathrm{data}} + C_{\mathrm{train}} + NC_{\mathrm{infer}}, \tag{3}$$

where C_{data} is the measured cost of generating the training and validation data, C_{train} is the FNO training cost, C_{infer} is the measured cost of one neural-operator inference, and N is the number of requested simulations.

For repeated direct numerical solution, the corresponding cumulative computational cost is

$$C_{\mathrm{NS}}(N) = NC_{\mathrm{solve}}, \tag{4}$$

where C_{solve} is the mean wall-clock time required for one numerical solution. The estimated break-even point is obtained from the intersection of the two cost expressions and is reported as the simulation count at which the cumulative neural operator cost becomes lower than the cumulative numerical solver cost for the evaluated workload.

The cost formulation is motivated by prior work that explicitly distinguishes the start-up cost of learned PDE models from the marginal cost of subsequent model evaluations [1].

3.10. Repeated Timing Measurements

The baseline Burgers configuration is executed four times using the same computational settings. These repeated measurements are used to estimate variability in wall clock timing and to obtain mean and standard deviation values for the numerical solver time, neural operator inference time, and fixed neural operator cost. This repeated evaluation also provides a more reliable basis for timing comparisons and reporting.

The larger Burgers and heat equation configurations are treated as separate experimental configurations because their training set sizes, validation set sizes, test set sizes, and training schedules differ from the baseline experiment. Their timing and accuracy values are therefore not pooled with the four baseline runs or combined in the reported averages.

3.11. Scope of the Evaluation

The evaluation is restricted to the two one-dimensional PDEs, the numerical solvers, FNO configurations, spatial resolutions, and CPU environment described above. The experiments are intended to measure the relationship between marginal prediction cost, fixed preparation cost, accuracy, and simulation count within these controlled settings. No broader performance claims are made beyond the tested configurations.

No claim is made that the measured break even points apply universally to other PDE classes, numerical solvers, neural operator architectures, spatial dimensions, hardware configurations, or broader computational workloads. These results should therefore be interpreted only within the experimental scope defined in this study and context of the manuscript.

4. Results

4.1. Computational Performance

The first four benchmark runs used the same one dimensional viscous Burgers equation, numerical discretization, neural operator architecture, training data size, validation data size, test data size, and training schedule. Across these repeated runs, the numerical solver required a mean of 0.012437 s per simulation with a standard deviation of 1.58×10^{-4} s. The corresponding mean neural operator inference time was 0.001221 s per simulation with a standard deviation of 6.01×10^{-4} s. Based on these mean measurements, a neural operator inference was approximately 10.2 times faster than an individual numerical solve. The repeated measurements also provide a basis for assessing timing variability under otherwise consistent computational conditions across the benchmark runs.

The lower marginal inference cost was accompanied by a substantially larger fixed computational cost. The mean fixed cost of the baseline neural-operator workflow was 33.346 s, with a standard deviation of 0.600 s. This fixed cost includes numerical generation of the training and validation data and subsequent neural-operator training. The results therefore show that lower inference time does not immediately translate into lower total computational cost for small workloads.

The second PDE experiment used the one-dimensional periodic heat equation with an analytical solution available for independent accuracy verification. The numerical solver required 0.004105 s per simulation, while the corresponding FNO inference required 0.000738 s per simulation. The neural operator was therefore approximately 5.6 times faster per inference for the heat equation. The fixed neural-operator cost was approximately 92.422 s, which was substantially larger than the cost of one numerical solve.

4.2. Break-Even Analysis

The cumulative computational cost of the Burgers-equation baseline provides a different result from the single-simulation timing comparison. The estimated break-even simulation counts from the four repeated baseline runs were 3125, 2763, 2994, and 3036 simulations, respectively. The mean break-even point was approximately 2.98×10^3 simulations with a standard deviation of approximately 154 simulations. Thus, for workloads substantially below this range, repeated numerical solution remained less expensive despite the lower marginal inference cost of the neural operator.

At $N = 1000$ simulations, the numerical solver required approximately 12.44 s using the mean measured solver time, whereas the neural-operator workflow required approximately 34.57 s after inclusion of the fixed preparation cost. At $N = 10000$ simulations, the numerical solver required approximately 124.37 s, whereas the neural-operator workflow required approximately 45.56 s. The ordering therefore changed as the number of requested simulations increased, with the neural-operator approach becoming less expensive only after the initial computational investment had been amortized over a sufficiently large workload.

The larger Burgers-equation training configuration produced a different cost profile. Its fixed neural-operator cost was approximately 97.60 s, and its estimated break-even point was approximately 8.00×10^3 simulations. At $N = 10000$, the corresponding numerical-solver cost was approximately 128.84 s, whereas the neural-operator cost was approximately 104.42 s. Thus, the larger training configuration remained more expensive for smaller workloads but became less expensive at sufficiently large simulation counts.

The heat equation experiment produced a higher break even point. Its estimated break even simulation count was approximately 2.745×10^4 simulations. At $N = 10000$, the numerical solver required approximately 41.05 s, whereas the neural operator workflow required approximately 99.80 s. The numerical method therefore remained less expensive across the tested workload range for this PDE, despite the lower marginal inference cost of the neural operator.

4.3. Accuracy

The four Burgers baseline runs produced the same reported test relative L^2 error of approximately 9.54×10^{-3}, corresponding to approximately 0.95% relative error for the tested cases. The larger Burgers training configuration reduced the test relative L^2 error to approximately 4.92×10^{-3}, corresponding to approximately 0.49% relative error. The increase in training data and training duration therefore improved the measured predictive accuracy for the Burgers problem.

For the heat equation, the numerical solver could be compared directly against the analytical solution and produced a relative L^2 error of only 2.39×10^{-12}. The corresponding

Table 1: Measured computational and accuracy results for the repeated Burgers-equation baseline runs, the larger Burgers-equation training configuration, and the periodic heat-equation experiment. The four baseline Burgers runs use 256 training samples, 64 validation samples, 64 test samples, and 30 training epochs. The larger Burgers configuration and the heat-equation experiment use 512 training samples, 128 validation samples, 128 test samples, and 50 training epochs.

Configuration	PDE	Training	Validation	Test	Solver time (s)	FNO inference (s)	Fixed FNO cost (s)	Test relative L^2 error
Run 1	Burgers	256	64	64	0.012414	0.001565	33.901	9.538×10^{-3}
Run 2	Burgers	256	64	64	0.012518	0.000321	33.703	9.538×10^{-3}
Run 3	Burgers	256	64	64	0.012591	0.001495	33.226	9.538×10^{-3}
Run 4	Burgers	256	64	64	0.012226	0.001504	32.553	9.538×10^{-3}
Baseline mean ± s.d.	Burgers	256	64	64	0.012437 ± 0.000158	0.001221 ± 0.000601	33.346 ± 0.600	9.538×10^{-3}
Larger training configuration	Burgers	512	128	128	0.012884	0.000682	97.600	4.924×10^{-3}
Heat-equation configuration	Heat	512	128	128	0.004105	0.000738	92.422	3.883×10^{-3}

FNO test relative L^2 error against the analytical solution was 3.88×10^{-3}, or approximately 0.39%. The FNO therefore remained substantially less accurate than the numerical reference in this experiment, despite providing a lower marginal inference time under the tested conditions. This comparison provides an independent check of predictive accuracy.

The comparison across the two PDEs also shows that the measured neural operator error was of similar magnitude for the larger Burgers and heat equation configurations, while the computational break even behavior was substantially different. The heat equation required approximately 2.75×10^4 simulations to reach the estimated break even point, whereas the larger Burgers configuration required approximately 8.00×10^3 simulations. The results therefore indicate that predictive error and computational advantage cannot be inferred directly from one another in general across these configurations.

4.4. Memory and Hardware Use

The four Burgers baseline runs recorded peak process memory values between 0.271 and 0.298 GB, with a mean of approximately 0.279 GB and a standard deviation of approximately 0.013 GB. The larger Burgers configuration recorded approximately 0.303 GB, while the heat equation configuration recorded approximately 0.330 GB. These measurements indicate that the observed memory requirement increased with the larger experimental configurations, although the present experiments do not establish general memory scaling behavior for substantially larger datasets, spatial resolutions, or neural operators in practice across different systems.

4.5. Overall Results

Across both PDEs, the neural operator had a lower marginal inference cost than the corresponding numerical solver. The Burgers baseline showed an inference speed advantage of approximately 10.2 times, while the heat equation experiment showed an advantage of approximately 5.6 times. However, the cumulative computational results were different from the marginal timing results. The Burgers baseline reached a break even point near 3.0×10^3 simulations, the larger Burgers configuration near 8.0×10^3 simulations, and the heat equation configuration near 2.75×10^4 simulations, respectively.

These measurements demonstrate that the computational value of a neural operator depends on the interaction between fixed preparation cost, marginal inference cost, solution accuracy, and the number of simulations required. A lower inference time alone was not sufficient to make the learned approach less expensive over the tested workloads. In particular, the heat equation experiment remained numerically cheaper through 10000 simulations even though the neural operator provided an overall faster individual predictions.

Overall, the experiments provide evidence for a workload-dependent relationship between neural operators and numerical solvers rather than a universal computational replacement. The numerical solver remained favorable for small workloads and for the heat-equation experiment across the tested range, while the neural operator became favorable for sufficiently large Burgers workloads. These results establish the basis for the comparison discussed in the following section.

5. Discussion

5.1. Workload-Dependent Computational Advantage

The experiments show that the computational advantage of a neural operator depends on the number of simulations required rather than on inference time alone. For the baseline Burgers configuration, the mean numerical solver time was approximately 0.012437 s per simulation, while the mean FNO inference time was approximately 0.001221 s per simulation. For the heat equation, the corresponding values were approximately 0.004105 s and 0.000738 s. Thus, the neural operator had a lower marginal evaluation cost for both PDEs, but the lower marginal cost did not immediately produce a lower total computational cost for small workloads.

The difference resulted from the fixed computational investment required before neural-operator inference could be used repeatedly. The baseline Burgers configuration required approximately 33.346 s of fixed computation, the larger Burgers configuration required approximately 97.600 s, and the heat-equation configuration required approximately 92.422 s. The estimated break-even points were approximately 2.98×10^3, 8.00×10^3, and 2.75×10^4 simulations, respectively. These results show that a lower marginal prediction cost can become useful only after the initial data-generation and training cost has been distributed across a sufficiently large workload.

This behavior is consistent with the distinction between start up and marginal computational costs reported in prior data driven PDE work, where the cost of preparing and training a learned model is separated from the cost of subsequent model evaluations [1]. The present experiments show that the

resulting break even workload can vary substantially between PDEs even when the same general FNO architecture is used. The computational value of a neural operator is therefore workload dependent rather than an inherent consequence of its lower inference time in isolation.

5.2. PDE Dependence of the Cost Trade-off

The difference between the Burgers and heat-equation experiments provides a key result for the central question of this study. The neural-operator workflow became less expensive than repeated numerical solution at sufficiently large workloads for the Burgers equation, while the numerical solver remained less expensive through the tested workload of 10000 simulations for the heat equation. The estimated break-even point for the heat equation was approximately 27450 simulations, substantially larger than the corresponding break-even points observed for the Burgers configurations.

The difference can be understood from the measured marginal solver costs. A heat equation solution required approximately 0.004105 s, whereas a baseline Burgers solution required approximately 0.012437 s. Because the numerical solver was faster for the heat equation, more simulations were required before the lower FNO inference cost could recover the fixed preparation cost. The results show that the break even point is governed by the interaction between numerical solver cost, neural operator preparation cost, and neural operator inference cost rather than by inference speed alone.

This observation places a direct limit on broad claims that neural operators replace numerical solvers. A neural operator can provide a faster individual prediction while remaining more expensive over an entire scientific workload. The relevant computational question is therefore whether the savings from repeated inference are sufficient to recover the resources required to construct the learned model for the intended use.

5.3. Accuracy and Computational Cost

The accuracy results show that increasing the training configuration can improve the learned solution while increasing the computational investment required to obtain it. For the Burgers equation, the baseline configuration produced a test relative L^2 error of approximately 9.54×10^{-3}, while the larger configuration reduced this value to approximately 4.92×10^{-3}. At the same time, the fixed computational cost increased from approximately 33.346 s to approximately 97.600 s, and the estimated break-even point increased from approximately 2.98×10^3 to approximately 8.00×10^3 simulations.

The heat equation provides an independent accuracy check because its numerical solution can be compared with an analytical solution. The numerical solver produced a relative L^2 error of approximately 2.39×10^{-12} against the analytical reference, while the FNO produced approximately 3.88×10^{-3}. The neural operator therefore provided a faster marginal evaluation but a substantially larger error for this analytically verifiable test case. The comparison demonstrates that computational cost and numerical accuracy can favor different methods under the same problem conditions and workload needs.

These results indicate that model accuracy and computational cost should be evaluated jointly. Increasing training resources can improve prediction quality, but the associated cost may move the break-even point to a larger workload. A training configuration that is preferable for accuracy may therefore be less attractive for applications involving only a limited number of simulations in practice and under constrained computational budgets or available system resources.

5.4. Generalization and Applicability

The present experiments use held-out test cases generated from the same problem distributions used to construct the corresponding training datasets. They therefore provide evidence for predictive performance within the tested distributions, but they do not establish broad generalization to arbitrary changes in parameters, boundary conditions, geometries, or spatial resolutions under unseen conditions.

This limitation is relevant because previous work has shown that learned PDE models can depend on the relationship between training and evaluation data. Kadeethum et al. reported degradation when validation data did not share relevant underlying features with the training data [1]. Neural operator studies have also investigated the representation of infinite-dimensional operators and solution operators associated with geometry-dependent PDE problems [3,4]. These studies establish the importance of generalization, but the present experiments do not provide sufficient evidence to quantify out-of-distribution performance beyond the tested cases or under broader distribution shifts in practice.

5.5. Why Neural Operators Do Not Replace Numerical Solvers

The combined results do not support a universal replacement of numerical solvers by neural operators. Conventional numerical methods remain attractive when relatively few solutions are required or when the cost of one numerical solve is sufficiently low that the training investment cannot be recovered. The heat equation experiment provides a direct example because the numerical solver remained less expensive through the tested workload despite the lower FNO inference cost.

Neural operators can nevertheless provide a computational advantage for sufficiently large repeated-query workloads. The baseline Burgers experiment provides such a case, with the neural-operator workflow becoming less expensive after approximately 3.0×10^3 simulations. The appropriate interpretation is therefore conditional substitution within selected workloads rather than universal replacement. This interpretation is consistent with broader scientific machine learning (SciML) research that considers learned solution operators alongside traditional numerical algorithms [5,6].

5.6. Limitations

The principal limitation of the study is the breadth of the computational evaluation. The experiments consider two one-dimensional PDEs, the numerical solver implementations used in the benchmark, a family of FNO configurations, and one Intel(R) Xeon(R) Gold 6226R CPU environment. The results therefore demonstrate workload-dependent behavior within the tested settings but do not establish how the measured break-even points will change for higher-dimensional PDEs, different numerical methods, different neural-operator

architectures, GPU execution, or substantially larger spatial resolutions.

The repeated Burgers baseline runs also use the same random seed and data generation procedure. These repetitions therefore characterize timing variability rather than independent variation in model training and accuracy. Additional training runs with independent seeds would be required for a fuller assessment of stochastic variation in learned model performance across training trials and configurations.

The heat equation provides an analytical reference for independent verification, whereas the Burgers experiments use numerical solutions as the reference targets. The two PDE experiments therefore do not provide identical forms of accuracy verification. This distinction should be retained when interpreting the numerical error values across the two problems and their respective evaluation settings separately.

Finally, the reported fixed neural-operator cost includes the computational stages implemented in the benchmark, including numerical generation of training and validation data and model training. Additional deployment costs such as long-term storage, data transfer, model maintenance, and future retraining are not included. These factors may become relevant in larger operational working environments.

5.7. Implications for Scientific Computing

The combined experiments indicate that neural operators should be evaluated as workload-dependent computational alternatives rather than as automatic replacements for numerical solvers. In both PDEs, the neural operator reduced the measured marginal cost of obtaining a prediction, but the total computational benefit depended strongly on the fixed preparation cost and the cost of the corresponding numerical solve.

For applications requiring many related simulations, the lower marginal inference cost may eventually compensate for the initial training investment. For applications requiring fewer simulations or involving relatively inexpensive numerical solves, the numerical method can remain more economical. The appropriate method therefore depends on the relationship between simulation count, numerical-solver cost, training cost, inference cost, and required accuracy.

The results also support a broader view in which machine-learning methods and numerical methods can coexist within scientific computing. Physics-based modeling remains important for representing governing relationships, while learned models can provide computationally efficient approximations for selected repeated-query problems [5,6]. Within the scope of the present experiments, the central conclusion is that neural operators offer a conditional computational advantage rather than a universal replacement for numerical solvers.

6. Conclusion

This study compared a Fourier neural operator with conventional numerical solvers for repeated simulation of two one-dimensional partial differential equations, namely the viscous Burgers equation and the periodic heat equation. The comparison treated the computational cost of data generation and model training separately from the marginal cost of subsequent neural-operator inference and compared these quantities with the repeated cost of direct numerical solution.

For both PDEs, the neural operator had a lower measured marginal inference cost than the numerical solver. The baseline Burgers configuration required approximately 0.001221 s per FNO inference compared with approximately 0.012437 s per numerical solve, while the heat equation required approximately 0.000738 s and 0.004105 s, respectively. These results demonstrate a clear marginal computational advantage for the learned model under the tested CPU configuration.

The cumulative cost comparison produced a different result overall. The baseline Burgers configuration reached an estimated break even point at approximately 2.98×10^3 simulations, while the larger Burgers configuration reached approximately 8.00×10^3 simulations. For the heat equation, the estimated break even point was approximately 2.75×10^4 simulations, and the numerical solver remained less expensive through the tested workload of 10000 simulations. The lower inference cost therefore did not by itself determine the lower cost computational strategy across the evaluated workloads.

The accuracy results further demonstrate the trade-off between predictive quality and computational preparation. Increasing the Burgers training configuration reduced the test relative L^2 error from approximately 9.54×10^{-3} to approximately 4.92×10^{-3}, but increased the fixed computational cost and shifted the break-even point to a larger workload. For the heat equation, the numerical solver agreed with the analytical solution to approximately 2.39×10^{-12} relative L^2 error, while the FNO produced approximately 3.88×10^{-3} relative error. The numerical method therefore retained a clear accuracy advantage for this analytically verifiable case.

The evidence does not support the conclusion that neural operators universally replace numerical solvers. Instead, the experiments support a conditional conclusion in which neural operators can become computationally advantageous when many related simulations are required and the lower marginal inference cost is sufficient to amortize the initial preparation cost. Conventional numerical solvers remain preferable for smaller workloads and for problems in which the direct numerical cost and required accuracy make neural operator preparation difficult to justify in practical applications.

The study is limited to two one-dimensional PDEs, the numerical solver implementations used in the benchmark, a family of FNO configurations, and one CPU environment. Broader conclusions require additional PDE classes, spatial resolutions, numerical methods, hardware configurations, independent training realizations, and systematic tests of generalization. Within the scope of the present experiments, however, the central finding is clear: neural operators can reduce the marginal cost of repeated PDE prediction, but their overall computational value depends on workload size, preparation cost, inference cost, and required solution accuracy.

Author Contributions

A.C. conceived the study, performed the analysis, developed the computational methods, and wrote the manuscript.

Funding
No external funding was received for this work.

AI Statement
AI tools were used for minor assistance. The author has made every effort to review the work presented in this manuscript.

Competing Interests
The author declares no competing interests.

Data and Code Availability
The data used in this study are available from the author upon reasonable request. The code used for the numerical experiments is available from the author upon reasonable request.

References
[1] T. Kadeethum, D. O'Malley, J. N. Fuhg, Y. Choi, J. Lee, H. S. Viswanathan, and N. Bouklas, "A framework for data-driven solution and parameter estimation of PDEs using conditional generative adversarial networks," *Nature Computational Science*, vol. 1, pp. 819–829, 2021.

[2] E. Kharazmi, M. Cai, X. Zheng, Z. Zhang, G. Lin, and G. E. Karniadakis, "Identifiability and predictability of integer- and fractional-order epidemiological models using physics-informed neural networks," *Nature Computational Science*, vol. 1, pp. 744–753, 2021.

[3] E. Pickering, S. Guth, G. E. Karniadakis, and T. P. Sapsis, "Discovering and forecasting extreme events via active learning in neural operators," *Nature Computational Science*, vol. 2, pp. 823–833, 2022.

[4] M. Yin, N. Charon, R. Brody, L. Lu, N. Trayanova, and M. Maggioni, "A scalable framework for learning the geometry-dependent solution operators of partial differential equations," *Nature Computational Science*, vol. 4, pp. 928–940, 2024.

[5] K. E. Willcox, O. Ghattas, and P. Heimbach, "The imperative of physics-based modeling and inverse theory in computational science," *Nature Computational Science*, vol. 1, pp. 166–168, 2021.

[6] S. L. Brunton and J. N. Kutz, "Promising directions of machine learning for partial differential equations," *Nature Computational Science*, vol. 4, pp. 483–494, 2024.

[7] T.-T. Gao and G. Yan, "Autonomous inference of complex network dynamics from incomplete and noisy data," *Nature Computational Science*, vol. 2, pp. 160–168, 2022.

[8] B. Chen, K. Huang, S. Raghupathi, I. Chandratreya, Q. Du, and H. Lipson, "Automated discovery of fundamental variables hidden in experimental data," *Nature Computational Science*, vol. 2, pp. 433–442, 2022.

[9] X. Chen, B. W. Soh, Z.-E. Ooi, E. Vissol-Gaudin, H. Yu, K. S. Novoselov, K. Hippalgaonkar, and Q. Li, "Constructing custom thermodynamics using deep learning," *Nature Computational Science*, vol. 4, pp. 66–85, 2024.

[10] P. J. Blazek, K. Venkatesh, and M. M. Lin, "Automated discovery of algorithms from data," *Nature Computational Science*, vol. 4, pp. 110–118, 2024.

[11] U. Tegin, M. Yildirim, I. Oguz, C. Moser, and D. Psaltis, "Scalable optical learning operator," *Nature Computational Science*, vol. 1, pp. 542–549, 2021.

[12] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, "A probabilistic graphical model foundation for enabling predictive digital twins at scale," *Nature Computational Science*, vol. 1, pp. 337–347, 2021.

[13] K. Willcox and B. Segundo, "The role of computational science in digital twins," *Nature Computational Science*, vol. 4, pp. 147–149, 2024.

[14] A. Ferrari and K. Willcox, "Digital twins in mechanical and aerospace engineering," *Nature Computational Science*, vol. 4, pp. 178–183, 2024.

[15] F. Tao, H. Zhang, and C. Zhang, "Advancements and challenges of digital twins in industry," *Nature Computational Science*, vol. 4, pp. 169–177, 2024.

[16] J. Qian, A. Jana, S. Menon, A. E. Bogdan, R. Hamlyn, J. Mahl, and E. J. Crumlin, "Digital Twin for Chemical Science: a case study on water interactions on the Ag(111) surface," *Nature Computational Science*, vol. 5, pp. 793–800, 2025.

[17] M. Aykol, A. Merchant, S. Batzner, J. N. Wei, and E. D. Cubuk, "Predicting emergence of crystals from amorphous precursors with deep learning potentials," *Nature Computational Science*, vol. 5, pp. 105–111, 2025.

[18] Q. Yu, R. Ma, C. Qu, R. Conte, A. Nandi, P. Pandey, P. L. Houston, D. H. Zhang, and J. M. Bowman, "Extending atomic decomposition and many-body representation with a chemistry-motivated approach to machine learning potentials," *Nature Computational Science*, vol. 5, pp. 418–426, 2025.