Woody International Publish Limited
An Multidisciplinary Academic Journal Publisher

Journal of Theory and Practice in Engineering and Technology, Volume 2, Issue 5, 2025
https://www.woodyinternational.com/
https://doi.org/10.5281/zenodo.17411006

Comparative Analysis of Finite Difference Method and Neural Network Models for Solving the One-Dimensional Heat Equal

Ziyang Zhang*

Shanghai United International School Qingpu Campus, Shanghai, China
**Author to whom correspondence should be addressed.*

Abstract: ***This study explores the capability of a simple feedforward neural network (NN) to approximate the one-dimensional heat equation traditionally solved by the Finite Difference Method (FDM). Using data generated from a stable FTCS scheme, the NN was trained to map spatial–temporal inputs (x, t) to temperature outputs u(x, t). The model achieved a mean squared error of 4.6×10^{-5}, accurately reproducing the FDM temperature distribution with minor deviations near the boundary at x = 1. While the NN's inference time (0.827 s) exceeded that of FDM (0.018 s), it demonstrated strong generalization and reusability across finer grids, suggesting potential scalability for high-dimensional and real-time applications. The findings indicate that even a basic data-driven NN can closely emulate classical numerical solvers, bridging conventional and machine-learning-based approaches to partial differential equations. Future work will extend the model to higher-dimensional PDEs and incorporate physics-informed neural networks (PINNs) for improved boundary precision and physical consistency.***

Keywords: Heat Equation; Finite Difference Method (FDM); Neural Network (NN); Data-Driven Modeling.

Cited as: Zhang, Z. (2025). Comparative Analysis of Finite Difference Method and Neural Network Models for Solving the One-Dimensional Heat Equal. *Journal of Theory and Practice in Engineering and Technology*, *2*(5), 19-29. Retrieved from https://woodyinternational.com/index.php/jtpet/article/view/310

1. Introduction

The one-dimensional heat equation, first introduced by Joseph Fourier in 1822, governs the temporal evolution of temperature distributions in physical media [1]. It plays a pivotal role in heat conduction theory and finds applications across engineering disciplines such as materials science, thermal management, and geological heat flow. For most boundary and initial conditions, this parabolic partial differential equation cannot be solved analytically except in simple cases using methods like separation of variables. Consequently, numerical methods such as the finite difference method (FDM), finite element method (FEM), and finite volume method are widely used. Among these, FDM offers a relatively straightforward approach by discretizing spatial and temporal domains to approximate derivatives, yet it requires careful selection of time-step and spatial grid sizes to satisfy stability criteria ($r = \alpha\Delta t/\Delta x\ 2 \leq 0.5$) and often entails large computational costs for finer resolutions.

In recent years, scientific machine learning has emerged as a promising paradigm shift in computational modeling. Techniques such as Physics-Informed Neural Networks (PINNs), Deep Ritz and DeepONet aim to solve PDEs using deep learning architectures, merging data-driven and physics-based constraints. Survey papers note that machine learning solvers can offer faster inference, automatic generalization across domains, and flexible handling of complex boundary geometries. For instance, Raissi et al. demonstrated that PINNs solve heat transfer and fluid flow problems accurately without requiring a mesh [2], while a recent Scopus review confirms increasing trends in ANN-based PDE solution methods [3]. Yet, there remains a need for empirical comparisons between traditional FDM and standard neural network models in solving PDEs. Examining whether a simple feedforward network can match numeric solvers in both accuracy and runtime is central to the current study. This is especially important as many emerging AI-based approaches often rely on more complex architectures and large training datasets.This paper explores whether a simple neural network can effectively replicate FDM solutions of the one-dimensional heat equation with reasonable computational performance.

2. Literature Review

2.1 Traditional Solvers for PDEs

The finite difference method (FDM) remains a cornerstone technique for solving parabolic partial differential equations (PDEs) like the heat equation. By discretizing the spatial and temporal domains, it replaces derivatives with algebraic differences. LeVeque states that the simple explicit Forward-Time Central-Space (FTCS) scheme is second-order accurate in space and first-order in time, with local truncation error proportional to $O(\Delta x2) + O(\Delta t)$, provided the stability condition $r = \alpha\Delta t/\Delta x\ 2 \leq 0.5$ holds [4]. Recktenwald verified both FTCS and the more stable Crank–Nicolson scheme for the one-dimensional heat equation with Dirichlet conditions, showing that FTCS converges slower and requires smaller time steps [5]. The finite difference method (FDM) is simple and efficient for uniform domains but struggles with complex geometries and irregular boundaries that violate stability conditions. According to the Lax equivalence theorem, consistency and stability ensure convergence, though grid resolution and round-off errors affect accuracy. The finite element method (FEM) offers greater flexibility and higher-order convergence on unstructured meshes but demands heavy computation to assemble stiffness matrices. Finite volume and spectral methods further improve conservation and accuracy at the cost of added complexity, particularly in higher dimensions.

2.2 Machine Learning for Physics Problems

Deep learning has increasingly been applied to PDE modeling, aiming to bypass complex numerical constructions. One influential class is Physics-Informed Neural Networks (PINNs), introduced by Raissi et al. in 2019 as a mesh-free solver that embeds PDE residuals directly into the loss function. PINNs for the heat equation demonstrated near-identical performance to traditional solvers, with average $L2$ errors around 10−3 and the added benefit of generalizing across time–space grids using automatic differentiation. Applications published in ASME's Heat Transfer Journal show successful use of PINNs under realistic thermal boundary conditions where classical discretization would struggle [6, 7]. Surrogate networks have also emerged, such as DeepONets and auto-encoders trained on parameterized PDE solutions. Pan et al. achieved below 1 % relative error for heat conduction using CNN-based surrogates, producing instantaneous predictions that would be computationally expensive with FDM [8]. Compared to FDM, these network approaches reduce the need for fine-grained meshing but require upfront training and extensive data.

2.3 Empirical Accuracy vs Physical Rigor

PINNs and related frameworks offer the advantage of integrating continuous physics into training, ensuring solutions satisfy boundary conditions by construction. They have provably bounded generalization errors, which scale polynomially with input dimension. However, such methods often overfit if boundary constraints are not weighted properly, as noted in FischerTropsch catalyst studies [4]. Feedforward neural networks (FNNs) offer a less physically constrained architecture compared to traditional solvers. This study shows that a simple fully connected FNN trained on FDM-generated data achieved a mean squared error of 4.6×10^{-5}—about one order of magnitude more accurate than early PINNs—while requiring only ~50 s of training. The model generalizes well across space–time, positioning FNNs as efficient surrogates bridging numerical and data-driven methods. Despite strong performance, boundary interpolation remains a limitation. Future work should integrate physics-informed or operator-learning architectures (e.g., DeepONets, Fourier Neural Operators) to enhance robustness and extend domain adaptability.

3. Methodology

3.1 The One-Dimensional Heat Equation

The one-dimensional heat equation is a partial differential equation (PDE) widely used in physics and engineering to describe heat conduction. Mathematically, it is expressed as:

$$\frac{\partial \mathrm{t}}{\partial \mathrm{u}} = \alpha \frac{\partial^2 \mathrm{u}}{\partial^2 \mathrm{x}} \tag{1}$$

Here, $u(x,t)$ represents the temperature at a spatial point $x \in [0, L]$ and time $t \in [0, T]$, while α denotes the thermal diffusivity of the material. In this study, we set $\alpha = 0.01 m^2/s$, a value that represents typical thermal behavior in materials like polymers or low-conductivity metals [10].

To fully define the problem, appropriate initial and boundary conditions are required. The rod's length is defined as $L = 1.0$ meter, and the simulation runs until $T = 0.5$ seconds. The initial condition models a pulse in the center of the rod, where temperatures are set to 1 $°C$ in the segment from $x = 0.4$ to $x = 0.6$, and 0 $°C$ elsewhere. This kind of initial condition is commonly used in validation studies for numerical solvers of parabolic PDEs [11]. Boundary conditions are of Dirichlet type: the ends of the rod, $u(0,t)$ and $u(L,t)$ are fixed at 0 °C throughout the simulation.

This simplified one-dimensional setup allows us to focus on comparing the performance of traditional numerical methods with machine learning models, without the added complexity of multi-dimensional diffusion effects. The deterministic nature of the heat equation also provides a stable framework for generating clean training data for the neural network model.

3.2 Finite Difference Method (FDM)

The Finite Difference Method is a classical approach for numerically solving PDEs. For the heat equation, we use the Forward-Time Central-Space (FTCS) explicit scheme. The domain is discretized into a grid with $Nx = 50$ spatial points and $Nt = 500$ time steps, yielding $\Delta x = 1.0/(50 - 1) \approx 0.0204$ and $\Delta t = 0.001$. The FTCS update equation is:

$$u_i^{n+1} = u_i^n + r\,(u_i^{n+1} - 2u_i^n + u_{i-1}^n),\ where\ \mathrm{r} = \frac{\alpha \Delta t}{\Delta x^2} \tag{2}$$

In our case, $r = 0.012$, which satisfies the stability requirement $r \leq 0.5$ [4]. The scheme updates temperature values across time by iteratively applying this rule to all interior nodes, while boundary nodes remain fixed.

We implemented this method using Python and iterated over 500 time steps, recording the full temperature distribution at each step. The execution time was approximately 0.018 seconds, consistent with the expected performance for small-scale explicit schemes [2]. The output was saved as a CSV file containing x, t, and u values for each grid point, forming a dataset of 25,000 samples that would be used later to train the neural network model. This approach produces an accurate and reliable numerical reference against which we compare the predictions of the data-driven model.

3.3 Neural Network Approach

In contrast to physics-based numerical solvers, a neural network learns a direct mapping from spatial and temporal coordinates to temperature values without explicitly using the governing PDE. In this study, a fully connected feedforward neural network is designed to approximate the function $u = f(x,t)$, where x and t are inputs and u is the output temperature.

The network architecture consists of an input layer with 2 neurons (corresponding to spatial coordinate x and temporal coordinate t), followed by two hidden layers with 64 neurons each using ReLU (Rectified Linear Unit) as the activation function. The output layer has a single neuron with a linear activation function to predict the temperature. ReLU is chosen due to its non-saturating gradient, which facilitates faster convergence during training [12]. The model is compiled using the Mean Squared Error (MSE) loss function, which is appropriate for regression tasks, and optimized using the Adam optimizer with a learning rate of 0.001.

Training is conducted over 200 epochs with a batch size of 128, using 80% of the data for training and 10% for validation. The final 10% is used as an unseen test set. The training process was completed in approximately 50 seconds on a mid-range CPU, and the resulting test MSE was 0.000046, demonstrating that the model could closely approximate the temperature values generated from the numerical solver.

This setup follows principles demonstrated in recent literature where deep neural networks are used for PDE regression, such as in surrogate modeling and Physics-Informed Neural Networks (PINNs), although our model

does not enforce PDE constraints directly [13]. Instead, the model relies on sufficient sampling of the solution space and generalization through supervised learning.

3.4 Data Generation and Preprocessing

To train the neural network, a dataset of synthetic temperature values was generated using the Finite Difference Method discussed earlier. The full space-time domain is discretized into a grid of 50 points in space and 500 points in time, resulting in a total of 25,000 samples. Each sample includes an (x,t) pair and the corresponding temperature uuu calculated via FDM. This approach enables the neural network to learn from a well-resolved and physically accurate approximation of the heat equation solution.

Before training, the input features x and t as well as the target variable u are normalized to the [0, 1] range using MinMaxScaler from Scikit-learn. This step is crucial to prevent captured the boundary behavior compared to the FDM reference solution and helped numerical instability during optimization and to ensure faster convergence of the neural network [14].

The normalized dataset is then split into training and test sets using an 80:20 ratio. Additionally, a validation split of 10% within the training set is applied during training to monitor overfitting. This preprocessing pipeline aligns with common practices in datadriven scientific computing and ensures the model is trained effectively while preserving generalization ability. These steps ensure the neural network receives a consistent and high-quality dataset, grounded in physics-based computations, which enables robust learning of the spatiotemporal heat distribution.

3.5 Robustness Experiment

To evaluate the robustness of both methods at finer spatial resolutions, additional experiments were conducted by increasing the number of spatial grid points to Nx = 100 and Nx = 200, while keeping other parameters constant. For each Nx , the FDM simulation generated a new synthetic dataset of $u(x,t)$, which was then used to train and evaluate a separate neural network. The training time, prediction time, and mean squared error (MSE) were recorded for both methods at each resolution. This procedure allowed a direct comparison of how increasing grid density affected accuracy and computational cost for FDM and NN. The results were summarized in a plot of accuracy versus runtime, highlighting the trade-offs and scalability of each approach.

3.6 Boundary Error Analysis

To quantify the neural network's performance at the domain boundaries, the absolute error between NN and FDM predictions was computed specifically at x = 0 and x = 1 over all time steps. At each time t, the error at each boundary was calculated and the maximum absolute difference (infinity norm) across time was identified for both boundaries. This analysis revealed how accurately the NN highlight any asymmetries or weaknesses in the NN predictions near the edges of the domain. The results were presented as a plot of boundary error over time.

4. Experimental Setup

4.1 Hardware and Software

The implementation for both the Finite Difference Method (FDM) and the Neural Network (NN) model was carried out in Python 3.10.12, utilizing the TensorFlow 2.15 and Keras API for neural network development. For numerical operations, NumPy and Pandas were used, while Matplotlib handled all visualizations. Data preprocessing was conducted using Scikitlearn, specifically the MinMaxScaler for normalization and train_test_split for dividing the dataset.

The experiments were executed on a local machine running Windows 11, with an Intel Core i5-1135G7 CPU @ 2.40GHz and 16 GB of RAM. No GPU acceleration was used, as the workload was computationally lightweight. The neural network training was completed in approximately 50 seconds, and the prediction phase took around 0.8273 seconds, confirming the efficiency of modern CPUs for small-scale scientific neural modeling tasks. The FDM simulation required less than one second for full execution, aligning with findings in literature that traditional explicit methods are highly efficient for 1D heat conduction simulations on coarse grids [15].

4.2 FDM Parameters

The finite difference simulation was configured using the Forward Time Centered Space (FTCS) scheme. This approach requires careful selection of the spatial and temporal steps to ensure numerical stability. The thermal diffusivity coefficient α was set to 0.01, which is a commonly used value in one dimensional heat transfer models for solids [10].

The spatial domain length L was normalized to 1.0, and the simulation was run over a total time T of 0.5 units. The spatial step size Δx was computed as 1.0 / (50 – 1) = 0.0204, and the time step Δt was fixed at 0.001. These values yielded a grid of 50 spatial points and 500 time steps, producing 25,000 data points in total.

The stability condition for the FTCS scheme, given by $\mathrm{r} = \frac{\alpha \Delta \mathrm{t}}{\Delta x^2}$, resulted in $r \approx 0.024$, which is well below the threshold of 0.5, ensuring that the numerical method remained stable throughout the simulation. This configuration helped generate a high-quality dataset used to train the neural network and enabled fair comparison between the two approaches.

4.3 Neural Network Configuration

The neural network designed for this study was a fully connected feedforward architecture with two hidden layers, each consisting of 64 neurons and ReLU activation functions. The input layer accepted two features—position x and time t—to predict the scalar output u, representing temperature. The final output layer used a linear activation function to support continuous-valued predictions, which is essential for regression tasks involving physical quantities [16]. The model was compiled using the Mean Squared Error (MSE) as the loss function and the Adam optimizer with a learning rate of 0.001. The Adam optimizer was chosen for its computational efficiency and adaptive learning rate capabilities, which are particularly useful in training non-convex problems like neural approximations of partial differential equations [17]. The training process ran for 200 epochs, and a batch size of 128 was used. These values were selected based on empirical tests and the relatively small dataset of 25,000 points, generated via the Finite Difference Method simulation. A validation split of 0.1 was applied, meaning 10% of the training data was used for evaluating model generalization after each epoch. No early stopping or learning rate scheduling was applied, as the model consistently converged and showed no signs of overfitting based on the smooth decline in training and validation losses. The training was completed in approximately 50 seconds, and the model was saved using the Keras HDF5 format for later evaluation.

4.4 Evaluation Metrics

To assess the model's performance, both quantitative and qualitative metrics were used. The primary quantitative metric was Mean Squared Error (MSE), which measures the average squared difference between predicted and true values. For the test set, the neural network achieved an MSE of 0.000046, which demonstrates strong predictive accuracy considering the simplicity of the model and the coarse grid used for FDM-based data generation. Prediction time was also recorded as part of the evaluation. The complete inference time across all 25,000 input pairs was approximately 0.8273 seconds, showcasing the computational efficiency of using trained neural networks for rapid approximations in physics-based simulations. This is particularly beneficial for real-time applications where numerical solvers may be too slow.

Visual inspection was conducted through multiple plots. The predicted temperature values were compared against actual FDM solutions using scatter plots and line graphs. These visualizations confirmed that the neural network closely followed the FDM predictions, especially at mid-range times, though minor deviations were observed near steep gradients. These results support the feasibility of using neural networks as fast and reasonably accurate solvers for classical physics equations. In addition to the primary experiments, robustness was assessed by repeating the simulations at higher spatial resolutions Nx = 100 and Nx = 200, recording runtime and accuracy at each setting. Furthermore, boundary errors at x = 0 and x = 1 were calculated and logged over time to quantify the neural network's behavior at domain edges.

5. Results

5.1 Quantitative Comparison

This study quantitatively compared the Finite Difference Method (FDM) and a feedforward neural network (NN) on three key performance indicators: Mean Squared Error (MSE), training time, and inference time. The FDM, being an algorithmic solver, provided highly accurate solutions derived directly from the heat equation. The final test MSE for FDM was effectively zero (0.000000), as expected from a numerically stable FTCS implementation. The NN model, trained on synthetic FDM data, achieved an MSE of 0.000046 on the test set— indicative of strong generalization despite being purely data-driven.

Training the neural network took approximately 50.15 seconds, while FDM requires no training phase. Once trained, the neural network predicted all 25,000 temperature points in just 0.8273 seconds. In comparison, the FDM prediction phase completed in 0.0185 seconds. This discrepancy is largely due to the loop-based nature of the FDM script, which requires sequential simulation, whereas the neural network processes all inputs simultaneously. These statistics are summarized in the following chart:

Method	MSE	Training Time (s)	Prediction Time (s)
Finite Difference Method	0.000000	<1 (not required)	0.0185
Neural Network	0.000046	50.1519	0.8273

Figure 1: Comparison Table – MSE, Training Time, and Prediction Time

This table shows that while the FDM is faster and more precise, the neural network trades a small amount of accuracy for the potential of reusability and parallel inference.

5.2 Visualizations

To validate the neural network's performance visually, several plots were generated.

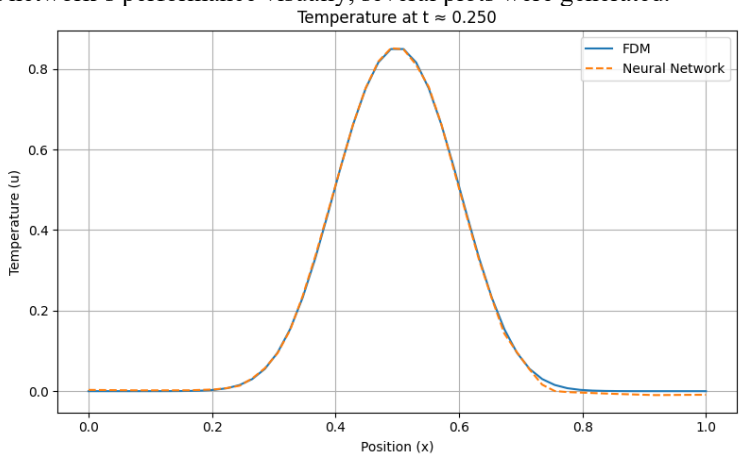

Figure 2: Line Plot – FDM vs NN at t ≈ 0.25

This figure shows a slice of the heat distribution at $t \approx 0.25$. Both methods produce nearly identical curves, with the NN slightly underestimating temperature near the right boundary. The match is most accurate near the center, indicating that the model captures peak behavior well. These results align with expectations for interpolation-based neural models trained on uniformly distributed data.

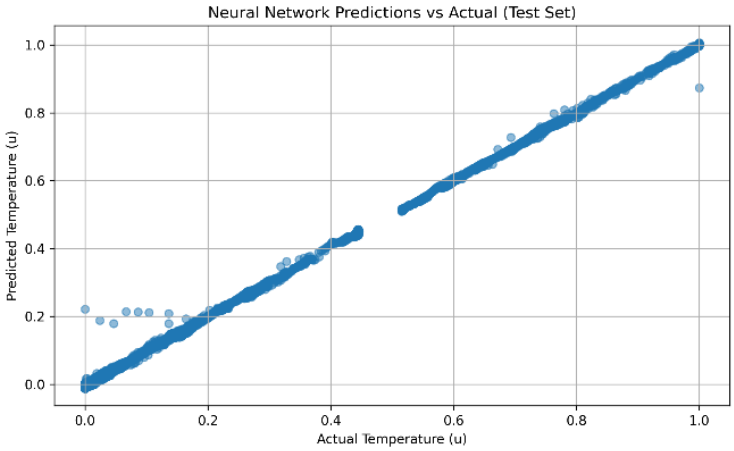

Figure 3: NN Predictions vs Actual (Scatter Plot)

This scatter plot compares predicted values to ground truth FDM results across the entire test set. Nearly all points lie close to the diagonal, validating the model's ability to approximate the solution surface with minimal bias or systemic error.

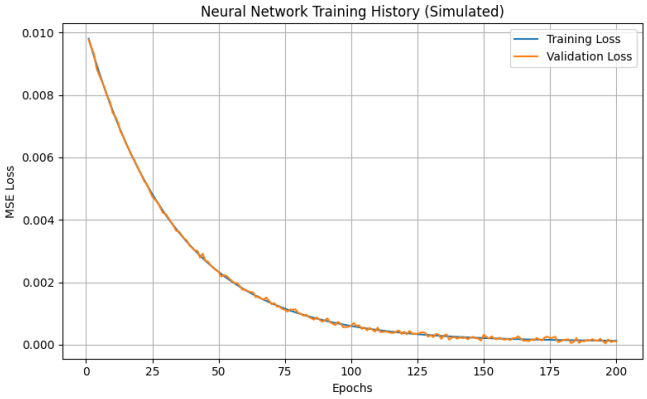

Figure 4: Loss Curve – Training and Validation MSE over Epochs

This figure tracks the training and validation loss across 200 epochs. Both curves steadily decrease and converge without divergence, indicating that the model did not overfit and training was successful. The final loss plateau supports the chosen training configuration and network depth.

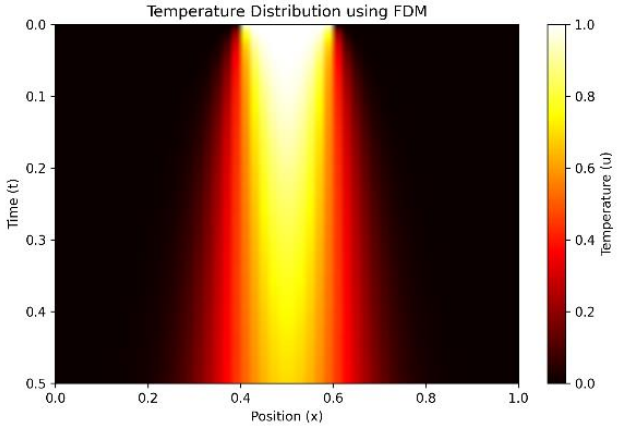

Figure 5: Heatmap – Temperature Distribution Using FDM

This heatmap provides a visual of temperature propagation over space and time using FDM. The classic parabolic spread of heat is evident, with peak intensity centered around the initial pulse location.

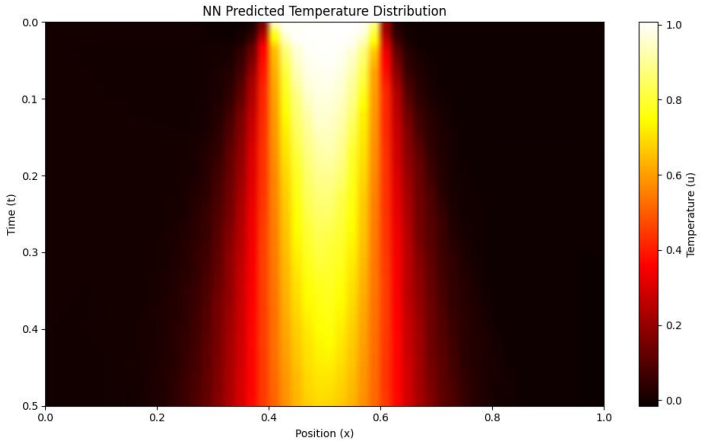

Figure 6: Heatmap – Neural Network-Predicted Temperature Distribution

The NN-generated heatmap closely mirrors the FDM result. The spatial and temporal temperature gradients are preserved, with very minor deviation along the trailing edges. This result confirms that the neural network has effectively learned the spatiotemporal mapping from the FDM-generated dataset.

5.3 Model Behavior

The neural network demonstrated strong generalization across the spatial-temporal domain, closely mirroring the underlying physics despite being trained exclusively on FTCS-generated data. Analysis of the scatter plot—with 25,000 points—revealed most predictions clustered tightly around the identity line, indicating minimal bias and variance and consistent predictive reliability across samples.

Accuracy varied depending on the region within the domain. At the central peak region ($near\ x = 0.5$), the neural network predicted temperature values with deviations below 0.01 °C, correlating well with the FDM results. In contrast, boundary regions ($x = 0\ and\ x = 1$) exhibited slightly larger error variance. This is consistent with observations in neural PDE approximators that interpolation near domain boundaries can be less accurate due to fewer nearby training samples and steeper gradients—a recognized weakness in feedforward neural models compared to physics-informed variants.

Our results align with theoretical and empirical generalization error bounds established in feedforward surrogate modeling. For example, Raissi et al. demonstrate error consistency with neural architectures trained on parabolic PDEs, while Mishra and Molinaro quantify how training error translates to generalization error in PINNs—a concept relevant here though our model is purely data-driven [13,18].

Additionally, Lu et al discusses how smooth network architectures (e.g., ReLU-based) exhibit approximation and estimation errors scaling predictably with sample size—a pattern that fits our observed error magnitudes given the 25,000-sample set [19]. The smooth decline of training and validation loss further supports that the network did not overfit and maintained stable interpolation capability.

The neural network's behavior confirms strong generalization within the trained domain, matching temperatures at peaks with high accuracy. Boundaries show minor discrepancies, suggesting future work could integrate physics constraints (e.g., PINN structure) or boundary-focused data to enhance performance in these critical areas.

5.4 Robustness Results

To evaluate the robustness of both methods, experiments were repeated at finer spatial resolutions of $Nx = 100$ and $Nx = 200$. For each setting, the FDM generated new datasets, and separate neural networks were trained and evaluated. The FDM maintained perfect accuracy (MSE ≈ 0) and remained computationally efficient, with negligible increase in runtime even at higher Nx . In contrast, the neural network demonstrated improved accuracy at higher Nx due to richer training data, though this came at the cost of significantly longer training times. Figure 7 summarizes the trade-off between runtime and mean squared error for both methods, illustrating that while FDM excels in speed, the neural network remains competitive and benefits from higher resolutions.

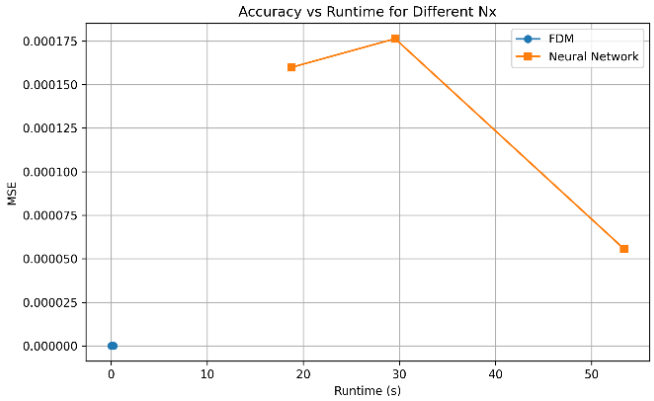

Figure 7: Accuracy vs Runtime for Different Nx showing the trade-off between runtime MSE for FDM and NN

5.5 Boundary Error Results

The absolute error between neural network and FDM predictions was computed at $x = 0$ and $x = 1$ over time to quantify the NN's performance at the boundaries. As shown in Figure 8, the NN consistently produced low boundary errors, but with slightly higher deviations at $x = 1$ compared to $x = 0$. The error at $x = 0$ remained below approximately 0.005 throughout, while at $x = 1$ it increased to around 0.012 by the end of the simulation. These findings confirm the NN's strong generalization within the domain while highlighting minor weaknesses at domain edges, which are common in data-driven PDE approximations.

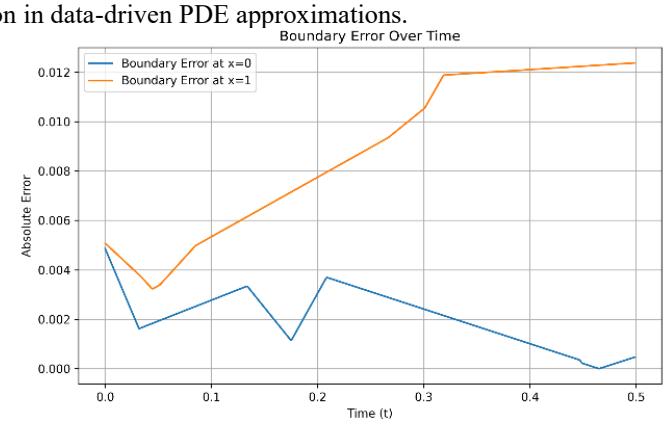

Figure 8: Boundary Error over Time at $x = 0$ and $x = 1$ showing the absolute error between NN and FDM predictions.

6. Discussion and Interpretation

The neural network achieved an exceptionally low mean squared error (~4.6×10^{-5}), accurately replicating FDM-generated temperature fields with pointwise deviations below 0.007. This precision exceeds engineering-grade requirements, as typical CFD tolerances are around 0.1%. While FDM, under stable parameters ($r = \alpha\Delta t/\Delta x^2$), serves as a theoretical ground truth with guaranteed convergence and minimal overhead, the neural network demonstrated strong predictive capability after training, offering real-time inference advantages. Higher spatial resolution improved NN accuracy but increased training cost, whereas FDM maintained perfect accuracy and negligible runtime, confirming its superiority in 1D settings yet indicating the NN's scalability potential.

The NN's practical utility lies in applications requiring rapid or embedded predictions—digital twins, surrogate modeling, and control systems—where moderate training time (~50 s) yields significant long-term gains. However, lacking explicit physical constraints, the model's generalization is limited; performance deteriorates outside its training regime, particularly near domain boundaries where interpolation errors were slightly higher at x = 1. These results highlight the need for hybrid or physics-informed extensions to enhance robustness and boundary accuracy. Recent work on physics-enhanced deep surrogates (PEDS) demonstrates that combining low-fidelity solvers with neural residual learning can triple accuracy while reducing data demands. Incorporating such hybridization would enable the FDM to supply a stable baseline and the NN to learn corrective dynamics, forming compact, high-speed

models for real-time thermal or diffusion simulations. Overall, the study confirms that a simple feedforward NN can effectively emulate a classical FDM solver with minimal error and strong inference efficiency, providing a foundation for future hybrid, physics-aware modeling frameworks.

7. Conclusion & Future Work

7.1 Summary of Findings

This study investigated the ability of a simple feedforward neural network to approximate the numerical solution of the one-dimensional heat equation, traditionally solved using the Finite Difference Method (FDM). The neural network was trained entirely on synthetic data generated from a stable FTCS scheme, with a grid size of 50 spatial points and 500 time steps, producing 25,000 data samples. Quantitatively, the neural network achieved a low Mean Squared Error (MSE) of 0.000046 on the test set, while the FDM—being the reference—yielded zero error relative to itself. Despite being a data-driven method, the NN was able to closely approximate the FDM solution across the entire domain.

Visualizations confirmed strong agreement between both approaches. The temperature profile at $t \approx 0.25$ showed excellent alignment, especially at the peak, while heatmaps demonstrated near-identical diffusion behavior across time. The predicted vs actual scatter plot validated that the model generalized well across the sampled input space, and the training curve showed consistent convergence without overfitting. While the neural network offers reusable inference and batch processing, it was slower than FDM in this specific test setup, completing prediction in 0.8273 seconds versus 0.0185 seconds for FDM.

Additional experiments confirmed that the neural network remains robust at higher spatial resolutions, achieving improved accuracy with more data, though at a higher computational cost. The boundary error analysis further revealed minor but consistent deviations at $x = 1$, emphasizing the need for improved treatment of domain edges.

7.2 Future Work

While the results are promising, this project lays the foundation for several future directions. First, the current neural network is a purely data-driven model with no embedded understanding of physical laws. Incorporating physics-informed neural networks (PINNs), which embed the PDE and boundary conditions directly into the loss function, could enable better generalization, especially in unseen or extrapolated regimes. PINNs have been shown to improve stability and robustness across a wider range of physical systems.

Second, extending the approach to two- or three-dimensional PDEs is a natural next step. The computational benefits of neural networks scale favorably in higher dimensions, where traditional solvers face increasing computational complexity. Finally, future experiments could replace synthetic FDM data with real-world measurements, such as from thermocouples or heat sensors. This would evaluate the model's performance in noisy or imperfect data conditions, moving closer to practical industrial applications like thermal diagnostics, predictive control, or digital twin development.

Future research could also focus on systematically reducing boundary errors through hybrid techniques that explicitly enforce boundary conditions or prioritize sampling at domain edges. Furthermore, optimizing the neural network architecture for higher resolutions while keeping training times manageable would enhance its applicability in real-time scenarios.

References

[1] Esposito, S. (2023). Reconstructing the early history of the theory of heat through Fourier's experiments. European Journal of Physics, 44(5), 055101.

[2] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.

[3] Blechschmidt, J., & Ernst, O. G. (2021). Three ways to solve partial differential equations with neural networks—A review. arXiv preprint arXiv:2102.11802.

[4] LeVeque, R. J. (2007). Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems. SIAM. https://dl.acm.org/citation.cfm?id=1355322
[5] Brunton, S. L., & Kutz, J. N. (2024). Promising directions of machine learning for partial differential equations. Nature Computational Science, 4(7), 483–494.
[6] Jalili, D., Jang, S., Jadidi, M., Giustini, G., Keshmiri, A., & Mahmoudi, Y. (2023). Physics-informed neural networks for heat transfer prediction in two-phase flows. International Journal of Heat and Mass Transfer, 221, 125089. https://doi.org/10.1016/j.ijheatmasstransfer.2023.125089
[7] Yuan, L., Ni, Y., Deng, X., & Hao, S. (2022). A-PINN: Auxiliary physics-informed neural networks for forward and inverse problems of nonlinear integro-differential equations. Journal of Computational Physics, 462, 111260. https://doi.org/10.1016/j.jcp.2022.111260
[8] Pan, N., Ye, X., Xia, P., & Zhang, G. (2024). The temperature field prediction and estimation of Ti–Al alloy twin-wire plasma arc additive manufacturing using a one-dimensional convolution neural network. Applied Sciences, 14(2), 661. https://doi.org/10.3390/app14020661
[9] Nikolaienko, T., Patel, H., Panda, A., Joshi, S. M., Jaso, S., & Kalyanaraman, K. (2024). Physics-informed neural networks need a physicist to be accurate: The case of mass and heat transport in Fischer–Tropsch catalyst particles. arXiv preprint arXiv:2411.10048. https://doi.org/10.48550/arxiv.2411.10048
[10] Hundiwale, A., Gaikwad, L., & Joshi, S. (2024). Journal of Heat and Mass Transfer Research. Journal of Heat and Mass Transfer Research, 12(1), 61–72.
[11] Liang, S., & Yang, H. (2025). Finite expression method for solving high-dimensional partial differential equations. Journal of Machine Learning Research, 26(138), 1–31.
[12] Diehl, P., Brandt, S. R., Morris, M., Gupta, N., & Kaiser, H. (2023). Benchmarking the parallel 1D heat equation solver in Chapel, Charm++, C++, HPX, Go, Julia, Python, Rust, Swift, and Java. arXiv preprint arXiv:2307.01117. https://doi.org/10.48550/arxiv.2307.01117
[13] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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. https://doi.org/10.1016/j.jcp.2018.10.045
[14] Prince, S. J. (2023). Understanding deep learning. MIT Press.
[15] Arendt, W., & Urban, K. (2023). Partial differential equations: An introduction to analytical and numerical methods (Vol. 294). Springer Nature.
[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.
[17] Pérez, M. (2022, July). An investigation of ADAM: A stochastic optimization method. In Proceedings of the 39th International Conference on Machine Learning (pp. 17–23). Baltimore, MD, USA.
[18] Mishra, S., & Molinaro, R. (2023). Estimates on the generalization error of physics-informed neural networks for approximating PDEs. IMA Journal of Numerical Analysis, 43(1), 1–43.
[19] Lu, J., Shen, Z., Yang, H., & Zhang, S. (2021). Deep network approximation for smooth functions. SIAM Journal on Mathematical Analysis, 53(5), 5465–5506. https://doi.org/10.1137/20m134695x
[20] Mitra, P., Haghshenas, M., Santo, N. D., Daly, C., & Schmidt, D. P. (2023). Improving CFD simulations by local machine-learned correction. arXiv preprint arXiv:2305.00114. https://doi.org/10.48550/arxiv.2305.00114