

# Neural Field Turing Machine: A Differentiable Spatial Computer

Akash Malhotra, Nacéra Seghouani

akash.malhotra@lisn.fr, nacera.seghouani@lisn.fr

September 4, 2025

## Abstract

We introduce the Neural Field Turing Machine (NFTM), a differentiable architecture that unifies symbolic computation, physical simulation, and perceptual inference within continuous spatial fields. NFTM combines a neural controller, continuous memory field, and movable read/write heads that perform local updates. At each timestep, the controller reads local patches, computes updates via learned rules, and writes them back while updating head positions. This design achieves linear  $O(N)$  scaling through fixed-radius neighborhoods while maintaining Turing completeness under bounded error. We demonstrate three example instantiations of NFTM: cellular automata simulation (Rule 110), physics-informed PDE solvers (2D heat equation), and iterative image refinement (CIFAR-10 inpainting). These instantiations learn local update rules that compose into global dynamics, exhibit stable long-horizon rollouts, and generalize beyond training horizons. NFTM provides a unified computational substrate bridging discrete algorithms and continuous field dynamics within a single differentiable framework.

## 1 Introduction

Many of the hardest problems in computer vision, robotics, and physics involve reasoning over *spatially continuous fields*. From fluid dynamics and weather forecasting to scene understanding and robot control, the world presents itself not as a sequence of tokens but as multi-dimensional spatial fields that evolve over time. Despite their successes, current machine learning architectures are poorly matched to this setting. Transformers excel at sequence modeling but treat space as discrete tokens and suffer from quadratic attention cost. Diffusion models perform powerful iterative refinements but rely on stochastic noise injection rather than controllable update rules. Physics-informed approaches such as PINNs [28] and neural operators [20, 21] approximate PDE dynamics but do not generalize beyond physics domains. Neural Cellular Automata (NCA) [23] show the potential of local differentiable updates, yet lack an explicit controller or the flexibility to move beyond fixed-grid evolution. In short, existing models either sacrifice spatial continuity, locality, or algorithmic generality.

We propose the *Neural Field Turing Machine (NFTM)*, a neural architecture that embeds computation directly into a continuous field with differentiable read/write access. NFTM combines three ingredients: a neural controller, a spatial memory field, and movable read/write heads. At each timestep, the controller reads local patches of the field, computes updates via learned transition rules, and writes them back, simultaneously updating head positions. This formulation unifies symbolic and continuous computation: unlike Transformers, NFTM scales linearly with the number of discretized field sites  $N$  through fixed-radius neighborhoods; unlike NCAs, NFTM has explicit controller dynamics and Turing completeness under bounded error; unlike diffusion models, NFTM produces deterministic rollouts at inference given fixed parameters and initial conditions; and unlike PINNs, NFTM is a general-purpose framework that extends naturally beyond PDEs.

The spatial nature of NFTM is central. Because it operates directly on continuous fields, NFTM is well-suited to problems where locality, geometry, and iterative simulation matter. Research in cognitive

science suggests that aspects of human cognition, such as intuitive physics [1] and mental imagery [31], involve running internal simulations to predict outcomes. NFTM echoes this paradigm by providing a differentiable substrate for simulation-based reasoning, bridging symbolic cellular automata, physical PDE solvers, and perceptual iterative inference within a single model.

Our work makes the following contributions:

- We formally define the Neural Field Turing Machine (NFTM), a neural controller operating over continuous fields with explicit read/write heads, and prove its Turing completeness under bounded error.
- We demonstrate NFTM across three domains: symbolic (cellular automata), physical (heat equation with global and variable diffusion coefficients), and perceptual (image inpainting).
- We show that NFTM subsumes Neural Cellular Automata (NCA) as a special case, and provide an alternative to PINNs and iterative perceptual refinement models, positioning NFTM as a unifying framework for discrete and continuous computation.

Code and experiments are available at <https://github.com/akashjorss/NFTM>.

NFTM thus provides a blueprint for learned, differentiable, continuous-memory computers, a step toward architectures that unify algorithmic reasoning with the spatial and iterative structure of the physical world.

## 2 Related Work

NFTM builds on a broad body of work spanning implicit neural representations, neural cellular automata, memory-augmented models, physics-informed networks, neural operators, graph neural networks, and diffusion models. We highlight the most relevant connections below.

**Implicit Neural Representations and Neural Fields.** Coordinate-based models such as SIREN [32] and Fourier feature networks [34] have demonstrated that neural networks can represent signals in continuous domains. Neural Radiance Fields (NeRF) [22] and their extensions such as D-NeRF [27], Nerfies [25], and Control-NeRF [17] further show how neural fields can encode dynamic 3D structure. While these models learn static mappings from coordinates to values, NFTM instead operates through *iterative updates* with a neural controller, enabling both discrete and continuous computation.

**Neural Cellular Automata.** Neural Cellular Automata (NCA) [23, 5, 24] have shown that local differentiable update rules can generate complex emergent patterns, from self-organizing textures to universal computation. However, NCAs operate only over discrete grids with fixed neighborhoods. NFTMs subsume NCA dynamics as a special case ( $\Delta h_t = 0$ , fixed  $A_t$ ), but add explicit read/write heads and controller logic, granting both locality and Turing-complete expressivity.

**Neural Turing Machines and Memory-Augmented Models.** Neural Turing Machines [11] and Differentiable Neural Computers [12] augment RNNs with differentiable memory, enabling algorithmic tasks such as copying and sorting. However, their memory is organized as a discrete matrix and indexed by soft attention, making them poorly suited to spatially continuous domains. NFTMs generalize this idea by embedding the memory in a *continuous field*, providing differentiable read/write while retaining access to geometry and physics.

**Physics-Informed Neural Networks and Neural Operators.** Physics-Informed Neural Networks (PINNs) [28] solve PDEs by embedding the residuals of governing equations directly into the loss. Neural ODEs [6], Hamiltonian Neural Networks [13], and latent SDEs [19] extend this idea to learning continuous-time dynamics under expressive priors. Neural operators such as DeepONet [21] and Fourier Neural Operators

(FNO) [20] learn mappings between infinite-dimensional function spaces. NFTMs differ by implementing forward Euler-style updates on a spatial field, keeping computation local and explicit like classical numerical solvers, but fully differentiable and trainable end-to-end.

**Graph Neural Networks.** Graph Neural Networks (GNNs) apply iterative message-passing over local neighborhoods and have been effectively used in physical simulation and PDE modeling, notably via the Graph Network-based Simulators (GNS) framework [29]. NFTMs generalize this concept from discrete graphs to continuous spatial fields, where read/write heads move and query regions of space rather than fixed graph nodes.

**Transformers.** Transformers [35] implement global attention and have been shown to be Turing complete under appropriate conditions [26]. However, their  $\mathcal{O}(N^2)$  attention cost and lack of locality bias make them inefficient for large spatial grids. NFTMs instead employ fixed-radius, local neighborhood updates, achieving  $\mathcal{O}(N)$  per-step computational cost while retaining full differentiability.

**Diffusion Models.** Diffusion models [14, 33] refine noisy data through iterative denoising, achieving state-of-the-art generative performance. While similar in spirit to NFTMs as iterative solvers, diffusion models rely on stochastic noise injection and score matching, whereas NFTMs employ a deterministic controller that learns local transition rules. This makes NFTMs closer to algorithmic and physical solvers, with diffusion-like refinement emerging as a special case.

**Analog Computation Models.** NFTM also connects to analog models of computation such as the General Purpose Analog Computer (GPAC) [30] and the Blum–Shub–Smale (BSS) model [3]. Unlike these hand-designed continuous machines, NFTMs are learned from data via gradient descent and integrate seamlessly with modern architectures such as neural fields and differentiable renderers. Thus, NFTM unifies the expressiveness of analog systems with the practicality of trainable deep learning models.

### 3 Neural Field Turing Machine: Definition and Universality

A *Neural Field Turing Machine (NFTM)* is a computational model that unifies continuous neural fields with the differentiable read/write mechanisms introduced in Neural Turing Machines. Unlike NTMs, which address a discrete set of memory slots via global attention, NFTMs embed computation directly into a spatial domain, where updates are defined over local neighborhoods and executed by movable read/write heads (see Fig. 1). This makes NFTM closer in spirit to convolutional or cellular update rules, while retaining the algorithmic flexibility of a controller.

Table 1: Comparison of NFTM with related models. NFTM combines differentiable read/write from NTMs with the spatial locality of NCAs, while offering an alternative to PINNs for PDE learning.

Aspect	NTM	NFTM	NCA	PINNs
Memory	Discrete slots	Spatial field	Grid cells	MLP over coords
Access	Global attention	Local heads	Fixed local	PDE residuals
Update	Seq. controller	Euler-style field	Local rule	Residual minimization
Domain bias	Algorithmic	Symbolic, physical, perceptual	Pattern growth	PDE solving
Determinism	Yes	Yes	Yes	Yes
Per-step cost	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Depends on collocation

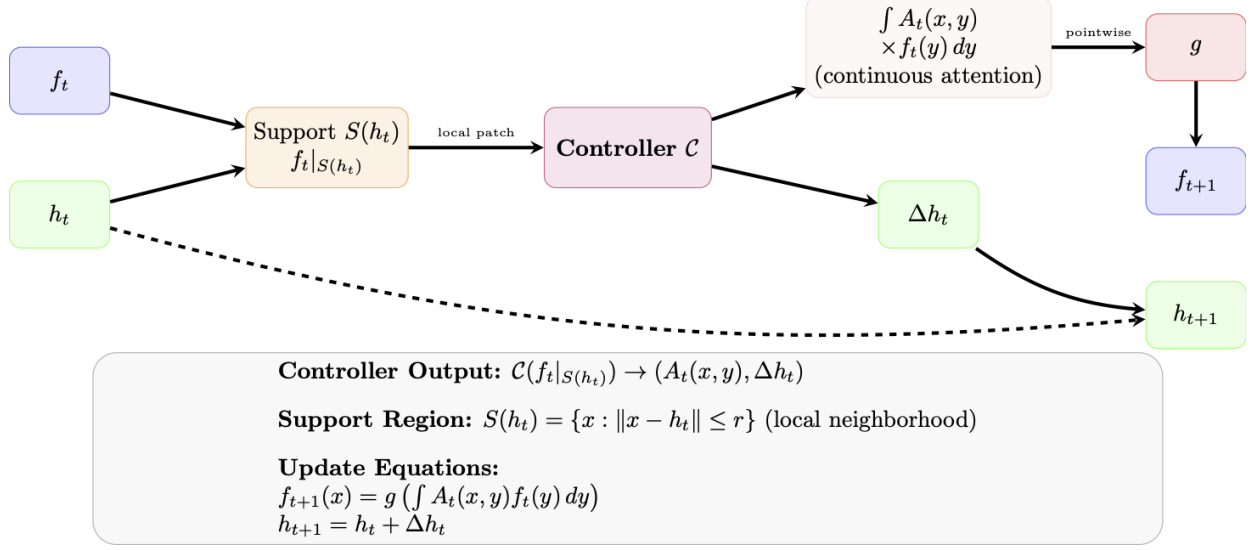


Figure 1: Neural Field Turing Machine (NFTM).

The system maintains a spatial field  $f_t$  at time  $t$ , which evolves over discrete timesteps. Read/write heads  $h_t$  specify positions within this field, and each update is computed over a local support region centered at  $h_t$ . In the simplest case this is a ball of radius  $r$ ,

$$S(h_t) = \{x \in \mathcal{X} : \|x - h_t\| \leq r\}, \quad (1)$$

but more generally  $S(h_t)$  can be defined by an arbitrary locality kernel  $K(x, h_t)$ , allowing the region to take non-spherical shapes such as boxes, anisotropic stencils, or ray frustums as in neural rendering. The controller extracts features  $f_t[S(h_t)]$  from this region, which serve as the context for computation.

A controller  $C$  operates on the local features  $f_t[S(h_t)]$  to produce two outputs:

$$C(f_t[S(h_t)]) \mapsto (A_t(x, y), \Delta h_t). \quad (2)$$

Here,  $A_t(x, y)$  defines a spatial attention field, while  $\Delta h_t$  specifies an update to the latent state.

The update equations governing NFTM dynamics are

$$f_{t+1}(x) = g\left(\int A_t(x, y) f_t(y) dy\right), \quad (3)$$

$$h_{t+1} = h_t + \Delta h_t, \quad (4)$$

where  $g$  is a pointwise nonlinearity applied to the aggregated features.

**Proposition 1** (Turing Completeness of NFTM). *NFTMs are Turing complete under bounded error.*

The claim follows by reduction to *Rule 110*, a one-dimensional cellular automaton proven to be Turing complete [8]. By discretizing the continuous state  $f_t$  into binary values and constraining the support region to a fixed local neighborhood (radius  $r = 1$ ), the NFTM controller can replicate the local update rules of Rule 110. Since Rule 110 can emulate any Turing machine, the NFTM inherits this universality. The bounded error condition arises because NFTMs operate in continuous domains, but quantization ensures symbolic dynamics can be recovered with arbitrarily small error.

Thus, the NFTM provides a bridge between Turing-complete symbolic systems and field-based neural dynamics, offering a framework for studying computation in continuous domains with explicit locality and bounded observation.

## 4 Methodology

Following the formal definition of NFTM in Section 3, we now show how the model can be instantiated in three domains of increasing complexity: (i) symbolic cellular automata, (ii) physics-informed PDE solvers, and (iii) image inpainting. Across these tasks, the unifying principle is that the controller  $\mathcal{C}$  applies local rules at head positions, and emergent global behavior arises through iterative rollout.

### 4.1 Cellular Automata Simulation

We first demonstrate how NFTM can simulate classical cellular automata. The controller MLP learns the rule’s truth table and acts on a grid, taking as input a cell and its neighborhood, and outputting the cell state at  $t + 1$ :

$$f_{t+1}(i) = \mathcal{C}(\text{read\_neighborhood}(f_t, i)) \quad (5)$$

Here, the number of heads equals the number of cells, and the field is discretized so that head positions correspond to cell indices. This instantiation is equivalent to Neural Cellular Automata (NCA), making NCA a strict subset of NFTM where the field is discretized, heads are fixed, and updates are Boolean.

To maintain differentiability while ensuring discrete states, we use Straight-Through Estimator (STE) binarization:

$$\text{STE}(x) = \begin{cases} \lfloor x + 0.5 \rfloor & \text{(forward)} \\ x & \text{(backward gradient)} \end{cases} \quad (6)$$

With this setup, the controller successfully learns Rule 110 and Conway’s Game of Life, both of which are Turing complete.

---

#### Algorithm 1 NFTM Cellular Automata Simulation

---

```

1: Input: Initial field  $f_0$ , timesteps  $T$ , controller  $\mathcal{C}$ 
2:  $f \leftarrow f_0$ , fields  $\leftarrow [f_0]$ 
3: for  $t = 1$  to  $T - 1$  do
4:    $f_{\text{read}} \leftarrow \text{STE}(f)$ 
5:   neighborhoods  $\leftarrow \text{read\_neighborhoods}(f_{\text{read}})$ 
6:    $f \leftarrow \text{STE}(\sigma(\mathcal{C}(\text{neighborhoods})))$ 
7:   fields.append( $f$ )
8: end for
9: return fields

```

---

### 4.2 Physics-Informed Neural PDE Solvers with NFTM

Beyond symbolic systems, NFTM can act as a neural solver for partial differential equations (PDEs) by combining explicit physics with learnable parameters. This addresses three challenges in physics-informed machine learning: (i) spatial parameter estimation, (ii) stable autoregressive training, and (iii) uncertainty quantification.

**Heteroscedastic Loss.** To recover spatially varying coefficients  $\alpha(x, y)$  we use a heteroscedastic Gaussian likelihood on the local PDE update  $\delta$ :

$$\mathcal{L}_{\text{NLL}} = \frac{1}{2} \frac{(\delta_{\text{gt}} - \alpha L_{\text{phys}}(u))^2}{\sigma^2} + \frac{\beta}{2} \log \sigma^2, \quad (7)$$

where  $L_{\text{phys}}$  is the discretized Laplacian and  $\sigma^2$  models aleatoric uncertainty. This encourages  $\alpha$  to explain systematic variation while  $\sigma$  captures residual noise. We optionally reweight by  $|L_{\text{phys}}|^\gamma$  to emphasize high-curvature regions, and add weak Gaussian priors on  $\alpha$  and  $\sigma$  to avoid pathological scales. Positional encodings  $[u, x, y]$  break translation equivariance, enabling the recovery of spatially varying  $\alpha(x, y)$ .

**Two-Phase Training.** Autoregressive models face exposure bias: they are trained on ground truth but tested on their own rollouts. We mitigate this via a two-phase scheme. In Phase A (teacher forcing), the controller learns the local operator  $\delta = \alpha \nabla^2 u$  using ground truth transitions, which yields stable gradients and rapid parameter identification. In Phase B (rollout training), the NFTM is unrolled from  $f_0$  for  $T$  steps under its own predictions and optimized with point-wise MSE at head positions plus small regularizers. This enforces stability over long horizons and mitigates exposure bias, forming a curriculum that first learns local physics and then tunes for autoregressive dynamics.

**Head-Based Architecture.** NFTM’s heads collect values and neighbors via a 5-point stencil, apply learned operators, and scatter updates back. This mirrors finite-difference schemes: When the Laplacian is fixed, the learned  $\alpha(x, y)$  retains physical interpretability.

Experiments on 2D heat equations show that heteroscedastic loss recovers  $\alpha(x, y)$  with MAE below 0.02, and two-phase training maintains stability over 50+ steps with PSNR above 30 dB.

### 4.3 Image Inpainting with NFTM

NFTM can also tackle perceptual tasks such as image inpainting. Each image is treated as a field, and the controller iteratively refines it by predicting per-pixel corrections given the current image and mask. Training data are drawn from CIFAR-10, normalized to  $[-1, 1]$ , with 25–50% of pixels randomly removed using a mixture of dropout and square block masks. Corrupted pixels are filled with Gaussian noise, while known pixels are always clamped back to the ground truth.

The controller (a small convolutional stack) outputs a correction  $\Delta I_t$ , a gating map  $g \in [0, 1]$ , and (optionally) per-channel logvariance for heteroscedastic likelihoods. In this demo we use a homoscedastic Gaussian loss:

$$\mathcal{L}(I, I_{\text{gt}}) = \frac{1}{2} \left( \frac{(I - I_{\text{gt}})^2}{\sigma^2} + 2 \log \sigma \right) + \lambda_{\text{TV}} \text{TV}(I), \quad (8)$$

with  $\sigma$  fixed and a TV-L1 prior to encourage smoothness. We also include a small contractive penalty  $\|I_t - I_{t-1}\|^2$  to discourage oscillations.

To stabilize training, updates are “guarded” by an energy check resembling a line search:

$$I_{t+1} = \text{clamp}(I_t + \beta \cdot g \cdot \Delta I_t), \quad (9)$$

where  $\text{clamp}(\cdot)$  restores known pixels. If the proposed update increases the energy (data term + TV), the step is rejected and  $\beta$  is reduced, ensuring monotonic improvement.

Training follows a curriculum over rollout depth: at each epoch a rollout length  $t \in [1, K]$  is sampled, with  $K$  gradually increased. Step sizes  $\beta$  are annealed during training, and per-step correction magnitudes are clipped with a decay schedule to prevent divergence at long horizons. During evaluation, the model is rolled out for  $K_{\text{eval}}$  steps, and performance is measured by PSNR as a function of iteration.

This setup highlights NFTM’s behavior as an iterative perceptual solver: trained with short rollouts, it generalizes to longer horizons and continues refining images beyond the training depth, unlike fixed-depth feedforward inpainting networks.

## 4.4 Summary

Across these domains, NFTM consistently learns local rules that compose into complex global behaviors. In cellular automata, it discovers symbolic truth tables; in PDEs, it learns physics-consistent operators; in inpainting, it acts as a measurement-consistent solver. The design ensures GPU-friendly parallelism (linear in field size and horizon) while bridging discrete algorithmic reasoning and continuous physical simulation within a single framework.

## 5 Results

We evaluate NFTM across three toy demonstrations and one perceptual task, highlighting its ability to unify discrete symbolic rules, continuous PDE dynamics, and iterative perceptual inference within the same framework.

### 5.1 Cellular Automata

NFTM with a simple MLP controller successfully learns the local truth table of Rule 110 and reproduces rollouts faithfully. As shown in Figure 2, NFTM matches the ground-truth cellular automaton evolution over 100 steps when trained on ground-truth rollouts, and crucially continues to extrapolate correctly beyond the horizon of the training sequence. This indicates that the controller has learned the underlying local update rule rather than memorizing trajectories. While the task is ultimately trivial—Rule 110 reduces to a finite truth table—the experiment serves as a minimal validation that NFTM can internalize symbolic update dynamics and reproduce Turing-complete behavior.

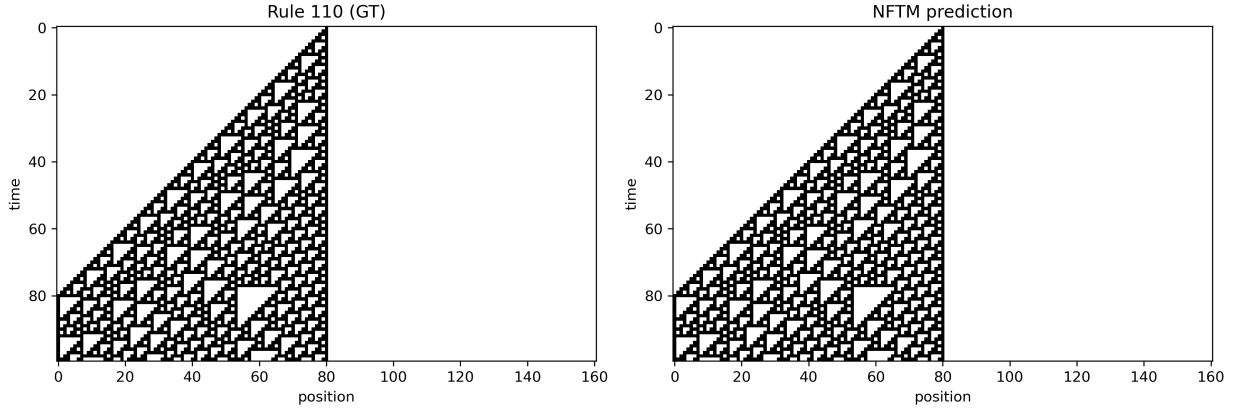


Figure 2: NFTM reproduces Rule 110 cellular automaton dynamics. Left: ground-truth evolution. Right: NFTM prediction. The match over 100 steps demonstrates NFTM’s ability to capture Turing-complete symbolic rules.

### 5.2 Heat Equation: Global Diffusion Coefficient

In the global diffusion experiment, NFTM accurately recovers the true diffusion coefficient  $\alpha$  using the heteroscedastic loss. As shown in Figure 3, the learned coefficients align closely with the true values across the tested range. For  $\alpha = 0.05$ , the model learns 0.067 (absolute error 0.018). For  $\alpha = 0.10, 0.15$ , and  $0.20$ , the learned values are 0.100, 0.150, and 0.200, respectively, with negligible errors ( $\leq 0.001$ ). This demonstrates that NFTM can reliably infer physical constants from spatio-temporal rollouts.

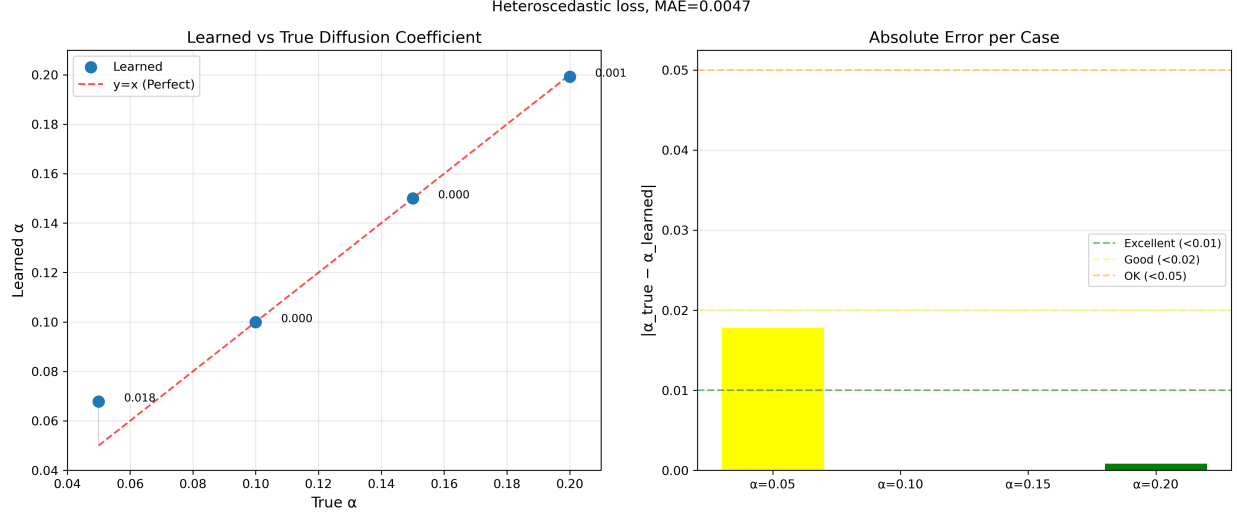


Figure 3: Learning global diffusion coefficient  $\alpha$  with heteroscedastic loss. Left: learned vs. true coefficients. Right: absolute error per case. NFTM recovers  $\alpha$  with mean absolute error below 0.01 in most cases.

### 5.3 Heat Equation: Variable Diffusion Coefficient

NFTM also handles spatially varying coefficients  $\alpha(x, y)$ . In this experiment,  $\alpha(x, y)$  was set to 0.05 everywhere except in a central square region where it was 0.15, smoothed by three successive  $3 \times 3$  average-pool operations. As shown in Figure 4, NFTM recovers this spatial profile: the predicted  $\alpha(x, y)$  closely matches the ground-truth map, with mean PSNR of 40.89 dB. Stepwise evolution shows that prediction error stabilizes around 0.03–0.04 absolute error magnitude after a few steps. These results highlight NFTM’s flexibility: a single architecture can recover both global scalars and spatially varying parameter fields.

### 5.4 Image Inpainting

Finally, we apply NFTM to image inpainting on CIFAR-10. The model was trained for 20 rollout steps with guarded backtracking, and evaluated unguarded for 30 steps. As shown in Figure 5, PSNR improves monotonically throughout inference, rising from 15.2 dB at step 1 to 24.5 dB at step 30. Qualitative results (Figure 6) illustrate how missing regions are progressively reconstructed with increasing fidelity. Importantly, despite training with shorter rollouts, the controller generalizes to longer horizons, continuing to refine the image beyond the training horizon. This extrapolation ability underlines NFTM’s nature as an iterative solver rather than a fixed-depth feedforward model.

### 5.5 Discussion

Across experiments, NFTM demonstrates the ability to learn local update rules that compose into coherent global dynamics. In the heat equation tasks, NFTM adapts seamlessly between learning a single global diffusion coefficient and recovering a spatially varying coefficient field, reflecting its flexibility as a universal computational primitive. In perceptual inpainting, NFTM trained for 20 steps continues to improve quality when rolled out to 30 steps, showing that the controller learns a stable refinement process rather than memorizing a fixed sequence of updates. Together, these findings suggest that NFTM generalizes across problem types (symbolic, physical, perceptual) and across rollout horizons, reinforcing its role as a unifying framework for iterative computation.

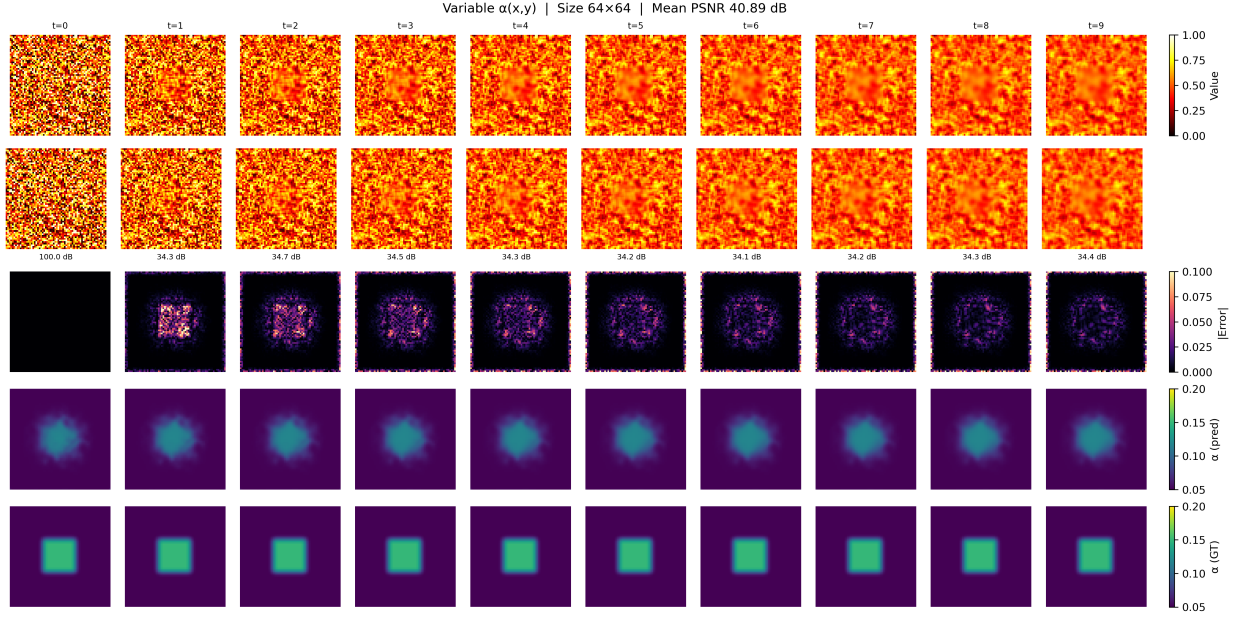


Figure 4: Learning spatially varying diffusion coefficient  $\alpha(x, y)$ . Top: predicted field over rollout steps. Middle: error maps. Bottom: predicted vs. ground-truth  $\alpha(x, y)$ . NFTM accurately reconstructs the spatial variation with mean PSNR 40.89 dB.

Our experiments are designed as illustrative demonstrations of NFTM’s versatility rather than competitive benchmarks. For each domain, NFTM subsumes or relates closely to natural baselines: NCAs for cellular automata, CNN/U-Net style solvers for PDEs, and diffusion-based refiners for inpainting. While we do not present head-to-head comparisons in this work, our goal is to highlight NFTM’s unifying nature as a differentiable spatial computer capable of spanning symbolic, physical, and perceptual domains. We view the present results as proof-of-concept validations, with systematic benchmarking against task-specific architectures left for future work.

## 6 Discussion and Conclusion

A key aspect of NFTM’s design is the configurable nature of the support region  $S(h_t)$ . The framework accommodates variable locality through attention-based selection, with support regions ranging from small local patches ( $\mathcal{O}(N)$  complexity) to the entire field ( $\mathcal{O}(N^2)$  complexity). NFTM can also employ multiple heads with different support regions simultaneously, enabling multi-scale computation where different heads operate at various spatial resolutions. This allows both fine-grained local processing and global long-range interactions within a single framework, while providing adaptive computation allocation based on spatial complexity.

NFTM’s local update paradigm also aligns with physical principles: even seemingly long-range forces like gravity and electromagnetism propagate locally through space-time. Iterative local updates can therefore simulate global dynamics by allowing information to propagate gradually. Moreover, the differentiable nature of NFTM opens avenues for enforcing conservation laws. Invertible neural network architectures [15, 16] can preserve information flow, while geometric deep learning demonstrates how symmetries can be encoded into neural architectures to enforce invariants via Noether’s theorem [4, 7, 9]. For example, translation equivariance in the controller could enforce momentum conservation, and rotation equivariance could preserve angular momentum. Embedding such symmetries into NFTM controllers may ensure physically

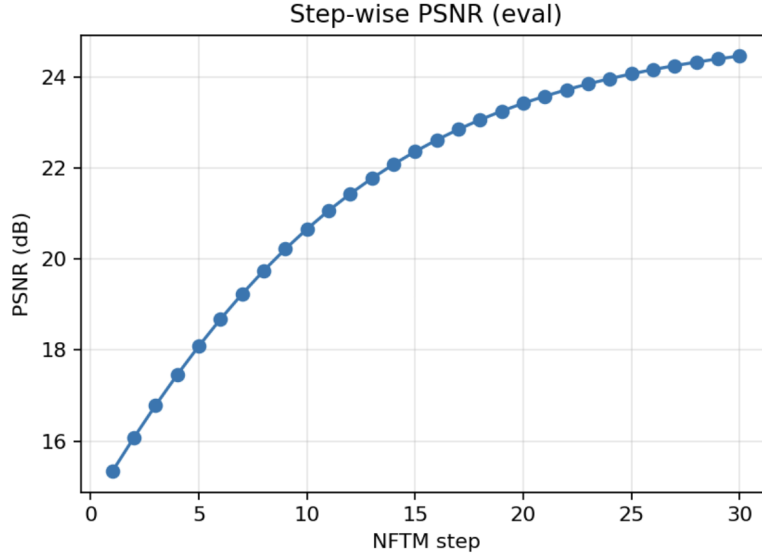


Figure 5: Step-wise average PSNR on CIFAR-10 inpainting task. NFTM exhibits monotonic improvement, generalizing from 20 training steps to 30 evaluation steps.

consistent dynamics that respect conservation of energy, momentum, and other invariants.

In terms of computational complexity, NFTM scales linearly with field size  $N$  for fixed-radius neighborhoods. For radius  $r$ , each head processes  $\mathcal{O}(r^2)$  elements in 2D (or  $\mathcal{O}(r^3)$  in 3D). Since  $r$  is constant, the per-head cost is  $\mathcal{O}(r^2 + C)$ , where  $C$  is controller cost, yielding  $\mathcal{O}(N)$  overall. This is directly comparable to convolutional networks, which cost  $\mathcal{O}(N \times k^2 \times \text{channels})$  for kernel size  $k$ , or finite-difference solvers that scale with stencil size [18]. The true trade-off is not asymptotic but constant factors: NFTM’s controller may be more expensive per head than a fixed convolution, but offers far greater flexibility in defining update rules. More significant limitations arise from error accumulation in autoregressive rollouts, a known challenge for sequential prediction models [2]. Our two-phase training reduces exposure bias but does not eliminate it entirely, and long-horizon stability remains sensitive to hyperparameters and discretization. While our experiments span symbolic, physical, and perceptual tasks, they remain at toy scale compared to real-world turbulence, coupled fields, or high-resolution visual reasoning.

Another distinctive property of NFTM is its ability to improve predictions when rolled out beyond its training horizon, effectively trading computation for accuracy. This test-time scaling resembles adaptive computation time [10], where complex inputs are processed with more steps while simple cases converge quickly. Our inpainting experiments confirm that NFTM can act as an iterative solver, refining outputs monotonically over extended rollouts. This property is particularly attractive for spatial reasoning in robotics or perception: a robot could “think longer” in challenging environments while making faster decisions in simple cases. Such adaptive scaling combines the representational power of modern controllers (e.g., U-Nets) with the iterative refinement capabilities of NFTM.

Looking forward, several extensions could enhance NFTM’s capabilities. Extending to 3D fields could impact scientific simulation and perception tasks involving volumetric data or point clouds. Adaptive time-stepping could improve multi-scale dynamics, while physics-informed constraints could enforce conservation directly. Hybrid designs that combine generative models with NFTMs might yield controllable refiners for creative or scientific applications. Most importantly, systematic evaluation on large-scale domains such as climate modeling, materials simulation, or robotics would clarify NFTM’s practical utility relative to established methods.



Figure 6: Qualitative image inpainting results on CIFAR-10. Left to right: ground truth, initialization, and successive NFTM refinement steps. NFTM progressively reconstructs occluded regions with higher fidelity.

In summary, we introduced the Neural Field Turing Machine (NFTM), a differentiable architecture that unifies symbolic, physical, and perceptual computation. NFTM combines a spatial field with a neural controller and read/write heads, proving Turing completeness under bounded error and demonstrating versatility across cellular automata, PDE solvers, and image inpainting. Its flexible support regions, capacity for multi-scale reasoning, and ability to trade compute for accuracy suggest NFTM as a step toward a differentiable spatial computer, a unifying framework for iterative field evolution across domains.

## References

- [1] Peter W. Battaglia, Jessica B. Hamrick, and Joshua B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013. doi: 10.1073/pnas.1306572110.
- [2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1171–1179, 2015.
- [3] Lenore Blum, Michael Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. In Ciprian Foias and Georg Sell, editors, *Complexity of Computation*, pages 170–195. Springer, 1989.
- [4] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.
- [5] Gabriel Béna, Maxence Faldor, Dan F. M. Goodman, and Antoine Cully. A path to universal neural cellular automata. *arXiv preprint*, 2025. arXiv:2505.13058.

- [6] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.
- [7] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999. PMLR, 2016.
- [8] Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.
- [9] Marc Finzi, Max Welling, et al. Generalizing convolutional neural networks for equivariance to lie groups. In *International Conference on Machine Learning (ICML)*, 2020.
- [10] Alex Graves. Adaptive computation time for recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2016. arXiv:1603.08983.
- [11] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- [12] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016. doi: 10.1038/nature20101.
- [13] Samuel Greydanus, Misko Dzamba, and Jascha Yosinski. Hamiltonian neural networks. In *NeurIPS*, 2019.
- [14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. arXiv preprint arXiv:2006.11239.
- [15] Jörn-Henrik Jacobsen, Arnold W. M. Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL <https://openreview.net/forum?id=HJsjkMb0Z>.
- [16] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 10236–10245, 2018.
- [17] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022.
- [18] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2007. ISBN 978-0-898716-29-0.
- [19] Yulia Li et al. Latent ODEs for irregularly-sampled time series. *NeurIPS*, 2020.
- [20] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- [21] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deep operator networks (deeponets) for learning nonlinear operators. *arXiv preprint arXiv:1910.03193*, 2019.

- [22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [23] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023. URL <https://distill.pub/2020/growing-ca>.
- [24] Ehsan Pajouheshgar, Yitao Xu, Ali Abbasi, Alexander Mordvintsev, Wenzel Jakob, and Sabine Süsstrunk. Neural cellular automata: From cells to pixels. *arXiv preprint*, 2025. arXiv:2506.22899.
- [25] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.
- [26] Jorge Pérez, Pablo Barceló, and Javier Marinković. Attention is turing complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021.
- [27] Albert Pumarola, Antonio Agudo, Wolfgang Heidrich, and Guillermo Vidal. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.
- [28] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [29] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *PMLR*, page 869–877, 2020. Also available as arXiv preprint arXiv:2002.09405.
- [30] Claude E. Shannon. Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics*, 20:337–354, 1941.
- [31] Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171 (3972):701–703, 1971. doi: 10.1126/science.171.3972.701.
- [32] Vincent Sitzmann, P. Martel, Julien N. Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.
- [33] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. Outstanding Paper Award.
- [34] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low-dimensional domains. In *NeurIPS*, 2020.
- [35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

## A Complexity Analysis of NFTM

We analyze the computational complexity of the Neural Field Turing Machine (NFTM) under its three main instantiations: cellular automata simulation, physics-informed PDE solvers, and image inpainting. Across all settings, let  $N$  denote the number of spatial sites in the field,  $T$  the rollout horizon, and  $P$  the computational cost of a forward pass through the controller  $\mathcal{C}$ .

### A.1 General Complexity

At each timestep, the NFTM performs three operations:

1. Reading a local neighborhood at head positions.
2. Applying the controller  $\mathcal{C}$  to compute updates.
3. Writing updates back into the field.

For a neighborhood of fixed size  $k$ , the read/write cost is  $\mathcal{O}(kN)$ , while the controller adds  $\mathcal{O}(NP)$ . Thus, the total time complexity is

$$\mathcal{O}(N \cdot T \cdot (k + P)),$$

which is effectively linear in field size  $N$  and rollout length  $T$ . Memory usage is  $\mathcal{O}(N)$  for forward simulation, and up to  $\mathcal{O}(N \cdot T)$  if full trajectories are stored for backpropagation (as in teacher forcing). With truncated backpropagation, this can be reduced to  $\mathcal{O}(N \cdot w)$ , where  $w$  is the truncation window.

### A.2 Local Attention Variant

If NFTM employs local attention, each head aggregates values from a patch of radius  $r$ . The cost per timestep becomes

$$\mathcal{O}(N \cdot r^2 \cdot T),$$

since each head reads  $\mathcal{O}(r^2)$  neighbors. For small  $r$  (e.g., 1–3), this remains efficient and GPU-parallelizable.

### A.3 Comparison Across Instantiations

Table 2 summarizes complexity across the three experimental domains.

Table 2: Asymptotic complexity of NFTM instantiations.  $N$  = field size,  $T$  = rollout horizon,  $P$  = controller cost,  $r$  = patch radius.

Task	Complexity	Notes
Cellular Automata	$\mathcal{O}(N \cdot T \cdot P)$	MLP controller, STE binarization
PDE Solvers	$\mathcal{O}(N \cdot T \cdot (P + k))$	5-point stencil, physics operators
Inpainting	$\mathcal{O}(N \cdot T \cdot P)$	CNN controller, curriculum rollouts
NFTM w/ Local Attention	$\mathcal{O}(N \cdot r^2 \cdot T)$	Patch-based attention, $r$ small

### A.4 Discussion

In all cases, NFTM rollouts scale linearly with field size  $N$  and horizon  $T$ , comparable to classical convolutional neural networks and finite-difference solvers. The constant factors differ by task (e.g., neighborhood size for PDEs, kernel size for CNNs), but remain modest in practice. This ensures that NFTM is both expressive, capable of simulating symbolic and physical systems, and computationally practical for large-scale training on GPUs.