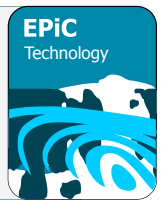
EPiC Series in Technology

Volume 3, 2026, Pages 113–120

Proceedings of AI for Accelerated Research Symposium



Diffusion with Attention for Inverse Optimization

John Lins and Wei Liu

Lawrence Livermore National Laboratory
johnlins@engineering.ucsb.edu, liu56@llnl.gov

Abstract

Traditional methods for inverse optimization of non-linear PDE systems face challenges such as getting trapped in local minima. An AI-based approach can be an alternative to drive optimization through a stable diffusion learning process with a more global context. The concept of using diffusion for generating new control sequences and control functions has already been lightly explored in the literature, however, this architecture has primarily been tested on smooth control functions in PDE simulations. In this paper, we test the limits of attention-based diffusion in inverting a 2D heterogeneous control function coupled in an advection–diffusion PDE system. Recent work has noted that methods such as supervised learning and reinforcement learning have proven somewhat effective; however, they often produce non-physical dynamics or fail to remain optimal long-term. This paper tests a UNet-based diffusion model that uses attention to solve inverse optimization problems. These encouraging results suggest that attention could potentially be an effective mechanism for inverse optimization.

1 Introduction

Control parameters $W(t, x, y)$ are crucial numerical parameters that evolve with time to steer a physical state $U(t, x, y)$ towards a desired final state U_T. In physics, these parameters often represent physical values such as force. Typically, these states can be constructed by iteratively passing control parameters into a PDE system to get the corresponding state sequence. In most inverse optimization problems, we already have knowledge of the PDE and the final state U_T, but we want to know what control steps must be taken to go from the starting state U_0 to the final state. Inverse optimization has many applications, such as finding the optimal energy deposition function for inertial confinement fusion [2] or minimizing instability from multi-material interaction [1].

In this work we explore diffusion AI as a powerful tool to perform a more global and contextualized optimization of the entire $[U, W]$ trajectory. Similarly to how diffusion is used for image generation, we can use it to iteratively subtract noise ϵ from a random W to generate the predicted control $\hat{W}$. The attention mechanism allows us to create contextualized representations of the state across both space and time, similar to how LLMs use it to contextualize words in a sequence; previous work [5] has tested a similar method on smooth hydrodynamic simulations such as Burgers, which may be more forgiving to noise. We build on top of this work to determine if attention is powerful enough to also learn highly heterogeneous 2D control functions with

J. Masnec, H.R. Karimian, P. Kordjamshidi and Y. Li (eds.), AIAS 2025 (EPiC Series in Technology, vol. 3), pp. 113–120

sharp boundaries. Having sharp boundaries in the final state yields high sensitivity to check the accuracy of the inversion. The PDE simulation that we have decided to use requires a control sequence to form English letters, which include sharp edges and corners; this allows us to more conclusively determine the effectiveness of this model.

2 Background

2.1 Problem Setup

We first designed two PDEs which are used to evolve a state towards visualizing a given letter sequence. These PDEs are used for both generating the training data, and are later used for reconstructing the state from the generated control.

2.1.1 Scalar Control PDE

We first tested the model using this PDE where the control $W(t) \in \mathbb{R}^{64\times 64}$ is a scalar for each point (x, y, t), which represents the magnitude of the restoring force.

$$W(x, y, t) = \lambda \left(U - U_T\right)$$

$$\frac{\partial U(x, y, t)}{\partial t} = D\,\nabla^2 U \;-\; \gamma \nabla \cdot \left(U\,\nabla U_T\right) \;-\; W$$

D is the diffusion coefficient, not to be confused with AI diffusion, that scales the Laplacian term in the PDE. Other hyperparameters include $\gamma = 5$, which represents the advection strength, and $\lambda = 0.5$, which scales the restoring force.

2.1.2 Vector Control PDE

We then pushed the model further and designed a PDE where the control $W_x(t), W_y(t) \in \mathbb{R}^{64\times 64}$ is a vector (W_x, W_y) for each point (x, y, t), meaning that the restoring force has the freedom to move 360° within the plane.

$$\begin{aligned} w(x, y, t) &= U - U_T \\ W_x(x, y, t) &= \lambda\,[w(x, y, t) - w(x + \Delta x, y, t)] \\ W_y(x, y, t) &= \lambda\,[w(x, y, t) - w(x, y + \Delta y, t)] \\ \frac{\partial U(x, y, t)}{\partial t} &= D\,\nabla^2 U - \left(\frac{\partial W_x}{\partial x} + \frac{\partial W_y}{\partial y}\right) \end{aligned}$$

W_x and W_y are the approximate directional derivatives of the difference between the current state and the final state.

3 Methods

We used a UNet [4], which takes the combined state–control trajectory as input, and is trained to predict the noise to be subtracted per step k.

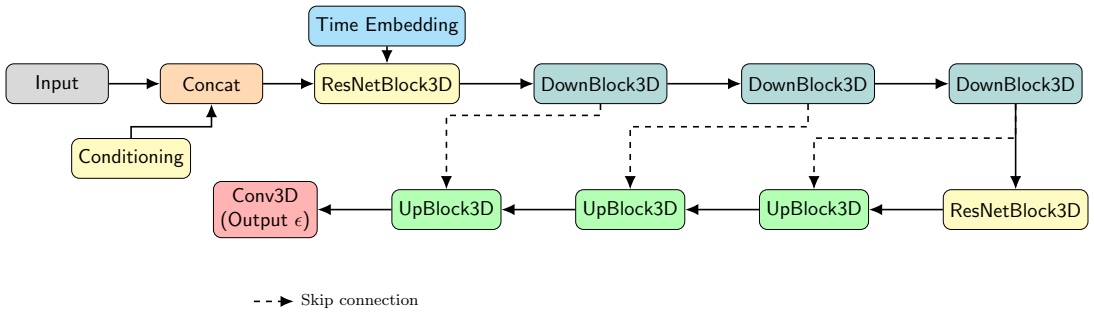


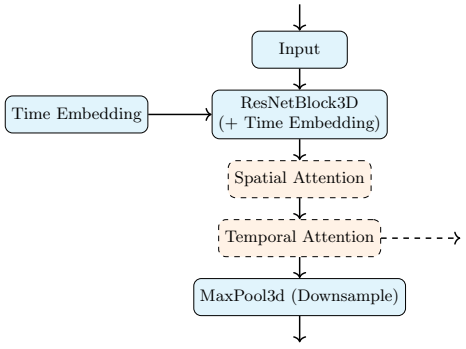
Figure 1: Compact architecture diagram of the UNet3D model.

3.1 Model

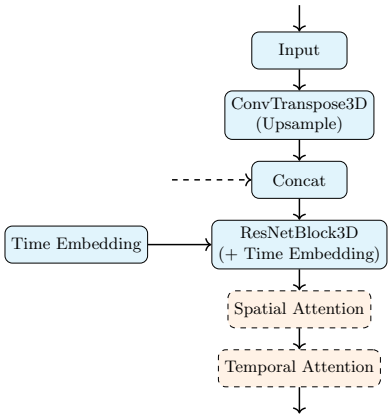
UpBlocks and DownBlocks work together as an encoder-decoder architecture to extract and recover features. Each block contains attention across both space and time for full contextual awareness.

$$[\mathbf{W}, \mathbf{U}] \leftarrow [\mathbf{W}, \mathbf{U}] - U\text{Net}([\mathbf{W}, \mathbf{U}], k, U_T)$$
$$[\mathbf{W}, \mathbf{U}] \leftarrow [\mathbf{W}, \mathbf{U}] - \epsilon$$

U and W are jointly denoised per k step by iteratively subtracting the predicted ϵ for 1000 diffusion steps.



(a) DownBlock3D



(b) UpBlock3D

Figure 2: DownBlocks and UpBlocks are the fundamental components of the UNet encoder-decoder structure. The powerful spatial and temporal attention mechanism allow the model to contextualize across space and all time steps.

ResNetBlock3D acts as the primary feature extractor. Since the model does not inherently understand time, we inject time embeddings, similar to positional embeddings used in LLMs. After the attention layers, we then use MaxPool3D to down-sample these rich feature maps. When we decode in the UpBlocks3D, we use ConvTranspose3D to revert back to higher resolution. To help preserve details, we concatenate with high-resolution skip connections from the encoder.

3.2 Data

We generated 100k state–control sequence pairs of training data (random 1 to 4 letter words) where each state is a 64×64 pixel grid. The scalar control used 100 time steps while the vector control used 200 time steps per sample.

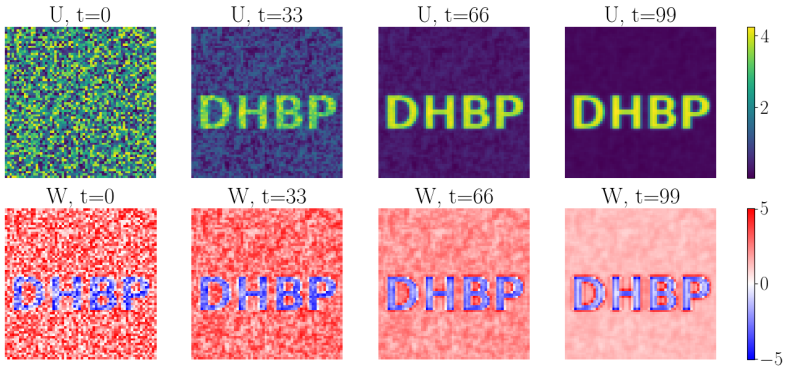


Figure 3: Example of a training datapoint generated for the scalar control function, where the control values only have magnitude.

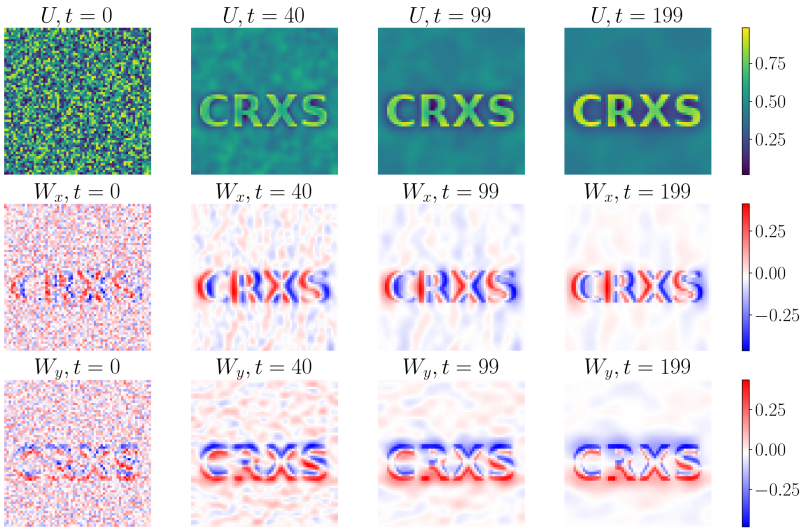


Figure 4: Example of a training datapoint generated for the vector control function, where the control values act as a 2D vector within the plane.

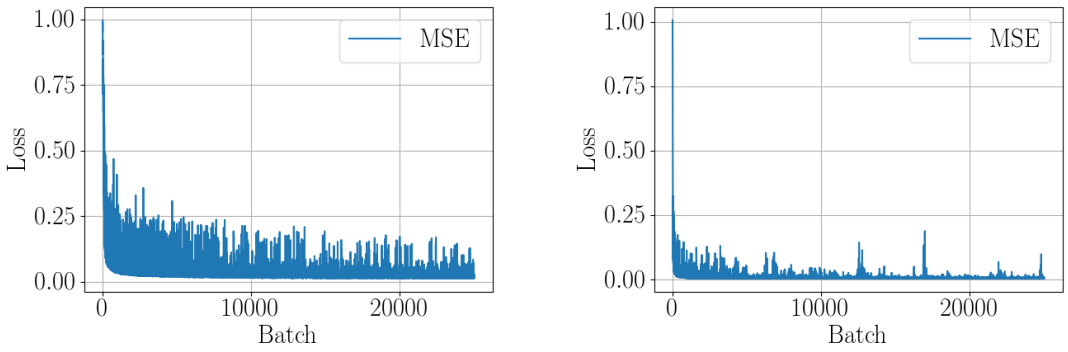
4 Results

After training the UNet to iteratively subtract noise ϵ from a random W sequence conditioned on fixed U_T, it learns how to steer the simulation closer to the final state. After performing inference with the diffusion model, we use the generated $\hat{W}$ by plugging it back into the PDE to reconstruct U.

4.1 Training

We trained for just one epoch, since we had a significant amount of data, and instead of tracking loss by epoch, we tracked it by batch (size=4). The learning rate was 10^{-5} and we used the Adam optimizer with 10^{-4} L2 weight decay. Training finished within 2 hours on 4 NVIDIA Tesla v100 GPUs.

4.1.1 Loss



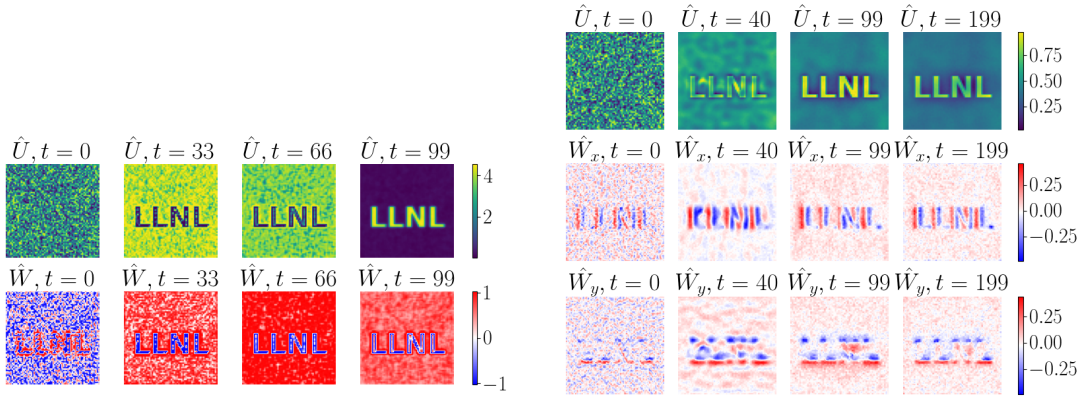
(a) Loss curve for the scalar W model.

(b) Loss curve for the vector (W_x, W_y) model.

Figure 5: Stable reduction in mean squared error (MSE) loss for both models.

Once trained, we then performed inference on a singular test sample "LLNL," which the model was never trained on. The model's UNet iteratively denoises a random sample to recover a given final state.

4.2 Inference



(a) Inference: Generated $\hat{U}$ and $\hat{W}$

(b) Inference: Generated $\hat{U}$ and $\hat{W}_x$, $\hat{W}_y$

Figure 6: Inference results for $\hat{U}$, $\hat{W}$, $\hat{W}_x$, and $\hat{W}_y$.

Important differences can be observed between the vector $(\hat{W}_x, \hat{W}_y)$ and scalar $\hat{W}$ inference results. Scalar $\hat{W}$ seems to drive values at each point independently. Alternatively, vector $(\hat{W}_x, \hat{W}_y)$ seems to drive values around within the 2D plane to uncover the letters; evidently, $\hat{W}_x$ tends to guide values left–right while $\hat{W}_y$ tends to do so up-down.

4.2.1 Simulation Validation

To more rigorously determine if the generated control sequence is correct, we cannot just rely on the inference results alone. We extracted $\hat{W}$ only and plugged that trajectory back into the original PDE without knowledge of the final state to see if it can reconstruct something that resembles U.

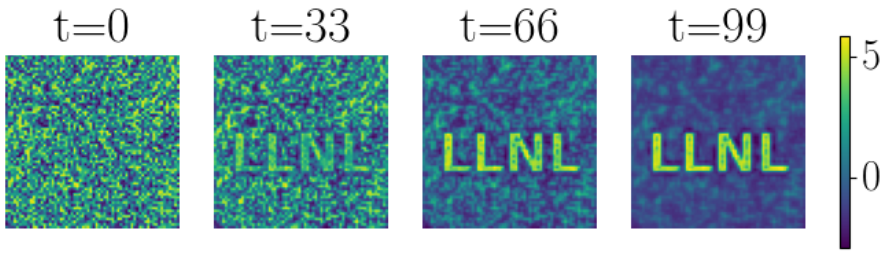


Figure 7: State sequence built using $\hat{W}$ shows a successful validation of the generated control.

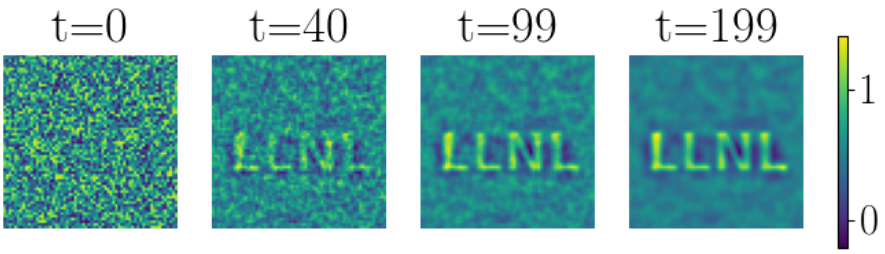


Figure 8: State sequence built using $(\hat{W}_x, \hat{W}_y)$ shows a successful validation of the generated control.

4.2.2 Quantitative Comparison

The percent error between the min-max normalization of the reconstructed final state and the ground truth is 27.1% for the scalar W case, and 6.7% for the vector (W_x, W_y) case; the error seems to be higher around the edges of the letters. This difference indicates that systems where the control has more degrees of freedom tend to minimize the state differences more aggressively.

5 Conclusion

Our findings demonstrate that the diffusion process utilizing UNets with attention mechanisms effectively learns the underlying physical system, enabling the generation of multi-dimensional control sequences. The model successfully captured the dynamics of the PDE without explicit guidance during inference and was able to reconstruct the desired final state, even in challenging inverse problems characterized by sharp features and high sensitivity. Numerical experiments indicate that phase errors in the reconstructed features are consistently lower than amplitude errors, with the latter exhibiting greater variability across different PDE systems and degrees of freedom. Future research will focus on further reducing reconstruction errors and investigating the applicability of this approach to a broader range of PDE problems. Overall, this methodology presents a promising alternative to traditional gradient-based optimization techniques, such as adjoint methods [3] and model predictive control [6], benefiting from the stability of the diffusion process when navigating complex, high-dimensional optimization landscapes.

6 Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. (LLNL-CONF-2009871).

References

[1] Robert W Anderson, Veselin A Dobrev, Tzanio V Kolev, Robert N Rieben, and Vladimir Z Tomov. High-order multi-material ale hydrodynamics. *SIAM Journal on Scientific Computing*, 40(1):B32–B58, 2018.

[2] MG Haines. Review of inertial confinement fusion. *Astrophysics and space science*, 256(1):125–139, 1997.

[3] Yvon Jarny. The adjoint method to compute the numerical solutions of inverse problems. *Inverse Engineering Handbook*, pages 103–218, 2003.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[5] Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Diffphycon: A generative approach to control complex physical systems. *Advances in Neural Information Processing Systems*, 37:4090–4147, 2024.

[6] Fawang Zhang, Jingliang Duan, Haoyuan Xu, Hao Chen, Hui Liu, Shida Nie, and Shengbo Eben Li. Inverse model predictive control: Learning optimal control cost functions for mpc. *IEEE Transactions on Industrial Informatics*, 2024.